Software & Tools

A GNU Emacs editing mode for METAFONT and METAPOST sources

Ulrik Vieth

Abstract

This article announces the release of `meta-mode.el`, a GNU Emacs editing mode for METAFONT and METAPOST source files. `meta-mode.el` provides a number of features commonly found in GNU Emacs editing modes for programming languages, such as automatic indenting of source code, syntactic highlighting, symbol completion for partially-typed keywords, motion commands to move to the beginning or end of the enclosing environment, or commands to comment out or reindent environments, regions, or buffers. An interface to running METAFONT or METAPOST in a shell buffer from within Emacs is currently under development and may be integrated into `meta-mode.el` later.

1 Introduction

The GNU Emacs[1] editor [1, 2] is one of the most widely used editors on Unix systems and some other platforms. As one of its most remarkable features it provides a vast number of specialized editing modes for a large variety of text formatting or programming languages. While support for editing TEX or LATEX files has been included in Emacs for many years, either as part of the standard GNU Emacs distribution or through the optional AUC-TEX package [3], no such mode-specific editing support existed so far for the somewhat esoteric programming languages of METAFONT and METAPOST.

When I started using METAPOST on a regular basis in early 1995, shortly after I had completed my first port of METAPOST, integrating it into the Web2C/Kpathsea distribution, I wasn't too much concerned about the lack of editing support since I was primarily interested in getting acquainted with METAPOST so that I could get some data plotted. During the course of time, however, especially after I started gaining a little experience with Emacs Lisp programming [4, 5], I became increasingly unhappy about being stuck with fundamental mode for editing METAPOST sources in Emacs.

[1] While this article only refers to GNU Emacs, most of it should be applicable to the XEmacs editor as well. Although `meta-mode.el` was developed exclusively under GNU Emacs, it was tried to make sure that everything will also work under XEmacs.

So it happened one day in January 1997 that I began asking myself (and also our local Emacs guru) what it would take to write a new major mode for editing METAFONT or METAPOST sources. After consulting the *GNU Emacs Lisp Manual* [5] the task turned out to be simpler than expected and pretty much straightforward. On the following weekend, when I had some spare time, I sat down to begin writing what was to become `meta-mode.el`. By coincidence, it happened to be February 1, 1997, exactly twenty years after the day on which genesis of TEX took place according to Don Knuth's own account.[2] I suppose I couldn't have chosen a better date to embark on this project nor a better way to celebrate this very special anniversary ...

2 Overview of `meta-mode.el`

2.1 Installation

From the technical point of view `meta-mode.el` is a contributed Emacs Lisp package that first needs to installed in a place where it can be found by Emacs, i.e. either in a personal or system-wide Emacs Lisp library directory listed in the `load-path` variable. To activate the features provided in `meta-mode.el` the package then needs to be loaded, which is most easily arranged for by adding a few lines of Lisp code like these

```
(autoload 'metafont-mode "meta-mode"
  "Major mode for editing Metafont sources" t)
(autoload 'metapost-mode "meta-mode"
  "Major mode for editing MetaPost sources" t)
(setq auto-mode-alist
      (append '(("\\.mf\\'" . metafont-mode)
                ("\\.mp\\'" . metapost-mode))
              auto-mode-alist))
```

to the personal or system-wide Emacs startup file to have `meta-mode.el` autoloaded at the first time a METAFONT or METAPOST source file is opened.

2.2 Initialization

Once `meta-mode.el` is loaded, the above code has the effect of invoking an Emacs Lisp function called `metafont-mode` or `metapost-mode` whenever a '`.mf`' or '`.mp`' file is loaded, which then proceeds to set up everything necessary when entering the new editing mode. Much of this initialization code is actually identical for both METAFONT and METAPOST mode as far as it concerns routine tasks needed for every Emacs editing mode, such as setting up a syntax table or installing a keymap and a pull-down menu for the mode-specific functions. However, it turned out to be necessary to have two separate initialization functions to be able to take care of subtle differences between the two modes, most notably, perhaps, when it comes to the list of known symbols for the completion function or the list of shell commands to generate proof sheets.

Following the usual Emacs conventions, both of these initialization functions provide hook variables `metafont-mode-hook` and `metapost-mode-hook` to allow adding extra setup or customization code to the individual modes if desired. In addition, there is also a `meta-common-mode-hook` that applies to both modes as well as a `meta-mode-load-hook` that is evaluated when `meta-mode.el` is first loaded.

2.3 Features

Once the general framework for a major mode is in place, adding more features and mode-specific functions becomes relatively easy since they can be conveniently added one by one as needed. The functionality currently implemented in `meta-mode.el` can be summarized in the following areas:

- automatic indenting of source code
- syntactic highlighting (a.k.a. fontification)
- completion for partially-typed keywords
- other miscellaneous editing functions

Additional functionality for running METAFONT or METAPOST and related commands for producing proof sheets in a shell buffer from within Emacs is currently under development and may be included into `meta-mode.el` later. At present, a preliminary test version is implemented in a separate Emacs Lisp package, tentatively called `meta-buf.el`, which may be integrated with `meta-mode.el` by making clever use of the various hook variables discussed above. For example, the load hook may be used to load `meta-buf.el` at the same time `meta-mode.el` is loaded while the common mode hook may be used to make the functions provided in `meta-buf.el` available in the keymap.

2.3.1 Indentation

The default keymap used in `meta-mode.el` maps the `TAB` key to a function that reindents the current line using an appropriate indent level computed automatically. Furthermore, the `RET` key also reindents the current line before jumping to the appropriate indent level on the next line. This allows you to blindly type arbitrary META[3] code, terminating each line with `RET` as you type, and get a nicely

[2] Donald E. Knuth, *The Errors of TEX*, reprinted as Chapter 10 of *Literate Programming*, p. 249.

[3] We will henceforth use the term "META" whenever we discuss features that are applicable to both METAFONT and METAPOST.

indented source file from which the grouping level and the control flow of conditionals and loops is immediately apparent.

At present, `meta-mode.el` recognizes all standard METAFONT language constructs, including **if**...**fi**, **for**...**endfor** and **def**...**enddef** blocks, as well as common variants like **forever**, **forsuffixes**, **vardef**, or even **mode_def**. In addition, it also recognizes standard macros introducing block structures such as **beginchar**...**endchar** in METAFONT as well as **beginfig**...**endfig** and **begingraph**...**endgraph** in METAPOST.

Furthermore, occurrences of **begingroup** and **endgroup** are also considered, although this might actually be the wrong thing to do if these are used unbalanced across different macro definitions. Users should therefore be aware that it may occasionally be necessary to adjust the indentation of their source files manually in some unusual cases.

Much of the Emacs Lisp code used in the indentation function in `meta-mode.el` was adopted from AUC-TEX's `latex.el`, which actually had a somewhat simpler job since it only had to look out for `\begin`...`\end` environments or lonely `\item`s while we have to handle a wider variety of METAFONT language constructs. Nevertheless, most of the AUC-TEX code could be put to a very good use. For example, the code that previously used to outdent `\item`s could be adapted to handle occurrences of **elseif** and **else** within **if**...**fi** blocks. It would have been possible to apply the same logic to **exitif** and **exitunless** in the middle of **forever**...**endfor** blocks, but this idea was rejected since it appeared too different from common coding style.

In any event, `meta-mode.el` allows easy customization of the kinds of METAFONT language constructs recognized by modifying the default regular expressions, either by using `M-x edit-options` or by writing a few lines of Lisp code to put in the personal `~/.emacs` startup file. Some familiarity with Emacs regular expressions will be unavoidable, however, to customize `meta-mode.el` at this level.

2.3.2 Fontification

Font Lock mode is a minor mode provided in GNU Emacs which allows modification of the appearance of a variety of major editing modes for different programming or text formatting languages. The basic idea is to have a number of differently colored text faces, which are used to highlight various language elements consistently throughout all editing modes, such as keywords, function or variable names, references to external filenames, etc. In addition, certain language elements are also highlighted on the basis of their syntactic properties, such as comment lines or quoted strings.

Since Font Lock mode is an optional package it needs to be loaded and activated first. With recent versions of GNU Emacs this has become very easy, as it is possible to turn on Font Lock mode as well as optional Font Lock support packages globally with just two lines of Lisp code:

```
(global-font-lock-mode t)
(setq font-lock-support-mode 'lazy-lock-mode)
```

Once Font Lock mode is globally activated like this, it will automatically apply to any new editing mode that supports it. In order to take advantage of fontification when writing a major mode such as `meta-mode.el`, it suffices to set up a few syntactic variables and put together a list of regular expressions that match the various language elements we wish to have highlighted.

While putting together a regular expression to match a list of keywords is fairly easy, writing good patterns to match macro definition headers presents quite a challenge since we have to cope with the rich variety of language constructs that are available in the METAFONT languages. For instance, we have to be aware that there are not only straightforward unary macro definitions introduced by **def** or **vardef**, in which the name of the function follows immediately after the definition keyword, but also binary operator macro definitions introduced by **primarydef**, **secondarydef**, or **tertiarydef**, in which the name of the function is embedded in between the parameter arguments. Furthermore, function or variable names don't necessarily have to consist of alphabetic characters or underscores; they might just as well consist of non-word symbols such as '`$`' or '`@`' or operator symbols such as '`**`' or '`&&`'.

If this isn't enough, another complication arises when it comes to parsing seemingly straightforward variable declarations that involve a list of comma-separated arguments of arbitrary length. To handle this case a simple regular expression isn't enough; instead, it is necessary to write a special-purpose utility function to match the arguments.

While all this has caused many headaches during the development of `meta-mode.el`, it appears that the Font Lock patterns currently implemented are good enough to handle most common cases, as can be verified by loading `plain.mf` or `plain.mp` into GNU Emacs and turning on fontification.

Finally, it should be noted that there was one more case that required special attention, namely TEX code embedded in between **btex**...**etex** or **verbatimtex**...**etex** in METAPOST sources.

From the point of view of syntactic highlighting it seemed best to treat this embedded TEX code just like a quoted string as it isn't interpreted in any way by METAPOST itself, but just passed on to `MakeMPX` for typesetting. However, to ensure proper parsing this interpretation also made it necessary to retain the meaning of escape character for the backslash, although this doesn't agree with its usual meaning of **relax** in the META languages.

2.3.3 Symbol Completion

Automatic completion of partially-typed keywords or filenames is a concept found throughout most parts of GNU Emacs as well as in some modern Unix shells. The basic idea is to save keystrokes by allowing one to type just the first few letters and perform completion on pressing `M-TAB`, resulting in partial completion and a display of all possible matches if no unique match is found.

An appreciable side-effect of symbol completion is that it provides a way of spell checking keywords in a programming language, which helps to avoid some of the most annoying compilation errors.

To implement symbol completion when writing a new major mode it takes two things: a completion function that does the actual job, and a list of known symbols that are offered for completion.

As for the completion function implemented in `meta-mode.el`, there is little to say. It was directly adopted from AUC-TEX's `latex.el`, but the framework was considerably simplified since it appeared unnecessary to support multiple completion lists for different kinds of symbols in META mode, whereas it did make sense to have them in LATEX mode.

As for the list of known symbols, there is a slightly more interesting story to tell: The idea was to have one comprehensive list of symbols for each of METAFONT and METAPOST which should include all primitives and macros defined in `plain.mf` or `plain.mp`, optionally augmented by the macros defined in standard packages, such as `graph.mp` or `boxes.mp` in the case of METAPOST. So what's the best method to get a complete list of primitives? The answer is simple: Use the source, Luke!

I eventually ended up with a little bit of Unix shell hackery along the lines of

```
$ grep '^primitive("[a-zA-Z]*"' {mf,mp}.web \
  | sed 's/primitive(\("[a-zA-Z]*"\).*/\1/' \
  | sort > {mf,mp}_prim.list
```

to extract the information about primitives directly from the `WEB` sources. Unfortunately, extracting the corresponding information from the macro definition headers in `plain.mf` and `plain.mp` didn't work out quite as well and required a little editing to fix up the extracted list, but this didn't matter too much since it had to be done only once anyway.

In any case, the resulting completion lists in `meta-mode.el` should be fairly comprehensive and might actually serve to give a good overview of what commands are available. Thus, if you ever wanted to know what tracing options exist, just type '`trac`' followed by `M-TAB` twice and see for youself. As this example illustrates, typing '`trac`' is sufficient to get a partial completion to '`tracing`', whereupon typing another one or two letters will be enough to resolve the remaining ambiguities.

In comparison to the completion in AUC-TEX it should be mentioned that `meta-mode.el` doesn't currently provide any context-sensitive completion, nor does it prompt the user to fill in the arguments where applicable. Instead, it just offers any known symbols for completion that match, regardless of whether they would make any sense in that context. Given the versatility of the completion function, it would certainly be possible to implement some of this by preparing a more involved completion list and some supporting functions if desired, but there are no such plans for the near future. After all, one might reasonably assume that users of METAFONT or METAPOST will be programmers who may be expected to know what they are doing, whereas authors of TEX documents don't necessarily have to be TEX macro programmers, and thus might require a little more help.

2.3.4 Miscellaneous Functions

As usual in Emacs editing modes for programming languages `meta-mode.el` also provides a small number of basic editing functions that are adapted to the mode-specific semantics. For instance, there are motion commands to move to the beginning or end of the previous or next "environment", or commands to apply the mode-specific indentation function or the standard Emacs comment-region function to each line in an "environment", a region, or a buffer.

As for what kinds of META language elements constitute an "environment", a somewhat different set of regular expressions is used than in the indentation function. Only the outermost block structures such as **beginchar** ... **endchar** in METAFONT or **beginfig** ... **endfig** in METAPOST are taken into account for this purpose, whereas conditionals and loops are disregarded. In addition, definition blocks such as **def** ... **enddef** and variants thereof are also considered as defining an "environment" for the convenience of editing more extensive macro packages. However, this may lead to problems if local macro definitions are nested inside **beginchar** ... **endchar**

blocks, in which case a command on an environment might be incorrectly applied to the inner block rather than the outer one. Unfortunately, there doesn't seem to be a general solution to this other than modifying the default regular expressions.

2.3.5 Keybindings

Most of the mode-specific editing functions provided in `meta-mode.el` are bound to fairly standard keybindings also used in Emacs editing modes for other programming languages. For example, `M-C-a` and `M-C-e` are bound to the motion commands applicable to environments while `M-a` and `M-e` are retained for the motion commands applicable to sentences, primarily for use in comment paragraphs. Likewise, `M-C-q` reindents an environment of META code while `M-q` is retained as the function to refill text in a comment paragraph.

A complete listing of mode-specific keybindings in `meta-mode.el` can be obtained using the Emacs help system. Furthermore, if Emacs is run under a windowing system such as X11, all mode-specific editing commands are also accessible from a pull-down menu entitled "Meta" that gets installed in the menubar when entering METAFONT or METAPOST mode. This menu lists all available mode-specific editing commands along with their corresponding keybindings.

3 Availability

In the past, preliminary versions of `meta-mode.el` have been made available by postings to the Usenet newsgroup `gnu.emacs.sources` and the METAFONT mailing list at `metafont@ens.fr`. As of version 1.0, `meta-mode.el` has been uploaded to CTAN where it has found a place in the `tex-archive/support/emacs-modes/` directory.

Shortly after releasing one of the early test versions, I was contacted by Richard Stallman about signing a copyright transfer agreement to the Free Software Foundation to allow the integration of `meta-mode.el` into the GNU Emacs distribution, which I have done now. Therefore, readers may look forward to finding `meta-mode.el` as the default METAFONT or METAPOST editing mode when the next version of GNU Emacs eventually arrives.

While the functionality provided in version 1.0 of `meta-mode.el` is pretty stable now, development of some additional features will continue. Most importantly, there are plans to implement an interface to allow running METAFONT or METAPOST in a shell buffer from within Emacs. In order to ensure stability, however, such development will be confined to add-on packages such as `meta-buf.el` that may be merged back into `meta-mode.el` from time to time, when the new features have proved stable.

4 Acknowledgements

A number of features implemented in `meta-mode.el` have been significantly influenced by features found in various Emacs editing modes for other programming or text formatting languages, among them, in particular, the AUC-TEX package, from which I drew much of the indentation and symbol completion functions. Emacs Lisp code was borrowed and adapted to the new purposes wherever possible, thereby sharing all the good ideas in the true free software tradition while at the same time avoiding to reinvent the wheel unnecessarily. Putting it all together and supplying the necessary knowledge about META language features to write appropriate font lock patterns and regular expressions, however, is the main ingredient for which I take responsibility entirely myself. I hope METAFONT and METAPOST users using any flavor of Emacs will enjoy it!

References

[1] Debra Cameron, Bill Rosenblatt, and Eric Raymond. *Learning GNU Emacs*. O'Reilly & Associates, Inc., September 1996.

[2] Richard Stallman. *GNU Emacs Manual*. Free Software Foundation, August 1996. 12th edition, for Emacs version 19.34.

[3] Kresten Krab Thorup. GNU Emacs as a front end to LaTeX. *TUGboat*, 13(3):304–308, October 1992.

[4] Robert J. Chassell. *Programming in Emacs Lisp — An Introduction*. Free Software Foundation, October 1995. edition 1.04.

[5] Bil Lewis, Daniel LaLiberte, Richard Stallman, and GNU Manual Group. *GNU Emacs Lisp Reference Manual*. Free Software Foundation, June 1995. edition 2.4, for Emacs Version 19.29.

◇ Ulrik Vieth
Heinrich-Heine-Universität
Düsseldorf
Institut für Theoretische Physik II
Universitätsstraße 1
D-40225 Düsseldorf
Germany
`vieth@thphy.uni-duesseldorf.de`
URL: `http://www.thphy.uni-duesseldorf.de/~vieth/`