Macros

What Constitutes a Well-Documented Macro?

Christina Thiele
Carleton University, Ottawa

Background

At the recent TEX Users Group annual meeting in Seattle, a number of us began to see that we had common problems with respect to using TEX as the means of typesetting materials for publication. We work in areas where people are designing macros, and then handing them on for general use. However, unless a macro is well-documented, it can be of little use to the ordinary user who is not a programmer, or who is not particularly interested in how TEX works. The main concern of the ordinary user is: Will this macro, or series of macros, do the job I need done with this particular piece of text?

On the second day of the conference, five of us* sat down to a working lunch to try to determine what would constitute a well-documented macro: what elements of description should it contain, how should it be presented, and in what order. We came up with what could provisionally be called "guidelines": the five basic elements outlined below. With this in hand, we then had a second meeting on Wednesday, with about 18 participants from the conference attending. Some more good ideas came out, which we then added to our first list. We now think it time to put all these ideas down on paper, and open the discussion to the general TUG community. For the confirmed TEX user, this may seem like a trivial exercise. However, if you bear with us, we hope that it will become clear that with a small, but solid corner block, this exercise should result in a number of benefits for novices and experts alike.

For example, it was suggested that a macro library be built up inside TUG, but unless the macros are well-documented, how can TUG take on the responsibility for building such a library? Would TUG want to be responsible for testing the macros, fixing any bugs or anomalies, helping people implement the macro? I think not; the onus should be on the supplier of the macro or macros to ensure the usability of what is being provided.

In that case, some sort of guidelines would be most helpful: the person writing the macro would at least have some headings or sections to complete the description. The user would be able to find out as much — or as little — as required in order to use the macro. A well-documented macro would not require the user to first disassemble the definition in order to see how to use it; the description would include this information.

Sometimes, neither the macro designer nor the user writes the documention, but a third party in the affair whose specific job it is to make the macro usable for various keyboarders. The person writing documentation may or may not be a professionally trained technical writer; the immediate task at hand is to write documentation on the local use of TEX and the local macros already designed. By establishing some sort of guidelines, the writing of macro documentation is rendered much easier; there are headings and topics to write towards, sections to fill in, comments to add, and so on. In short, guidelines make it easier to put something down on paper, beyond just the macro definition(s).

One reason for using macros is to make keyboarding work easier and the results more uniform. This becomes particularly important in a production environment, such as a journal operation, where a large volume of work is done on a regular basis. In most cases, production keyboarders are not computer experts; they are expert typists. Documented macros make it possible for them to work at top speed since a good macro description states clearly how the data should be keyed. For this and other reasons, I think you'll find that the sorts of things we consider part of a well-documented macro will respond to the actual needs of the many users and designers of macros.

The Initial Points from the Tuesday Meeting

1. What is the shape or format to be achieved? Give a visual and/or verbal description.
2. Give instructions on how to use the macro:
   i. the skeleton of the macro, without any text;
   ii. the skeleton with text filled in;
   iii. the output of that particular text and macro;
3. Supply the macro definition, heavily annotated with % comments.
4. How is the macro related to other macros? That is, if you give the macro to someone,

* 26 August, 1987: Mary Coventry (Univ. of Washington), Helen Gibson (The Wellcome Institute, London), Regina Girouard (AMS), Stephanie O'Hara (Univ. of Maryland), and myself.

make sure that it doesn't have some strings attached.

5. Where (i.e., at what points in the definition) can the definition be modified? Include some sort of naming convention, (especially if this is done in-house). And make sure to rename a redefined macro.

Additional Comments from the Wednesday Morning Meeting

a. "What goes in and what goes out?"

b. Make sure there are *lots* of examples, not just one sample input file with its output.

c. Provide a context for the macro: where would it be used?

d. make a list of potential error messages, and their likely source

e. Should the end-users see the technical side of the description?
Comment: If such macro descriptions are sent via the network, it's probably a fairly knowledgeable TEX user who's picking it up: they could then filter out the technical material before passing the description on to the end user.

f. Before making a macro part of an eventual TUG library of macros, it (the macro) should meet minimum specifications (i.e., something like these guidelines) before being included in the library. This would then keep the TUG office work to a minimum, and would also increase the degree of utility of even having such a library.

g. Include information on who wrote the macro, and how they can be reached (e-mail, telephone, mailing address).

Between the two meetings, Barbara Beeton took time to write down some of her ideas on all of this as well. The following are mainly concerned with the readability and stability of macros.

a. Use indentation to indicate "level"; that is, don't write a solid paragraph-like macro.

b. Within definitions, put a `%` after a `{` or `}` at the end of a line (but not after digits or control sequences) to help control unwanted side effects from carriage returns interpreted as spaces. If there is a good reason to do otherwise, e.g., to control `\obeylines`, *annotate* it.

c. Document anything that is even vaguely "inscrutable".

d. With respect to length of macros: Keep them shorter than about 60 lines (the length of a printed page on most line printers), and arrange them in the file so that every macro will be complete on one page in a printed copy.

e. If your system permits, put in page breaks to assist with the intent of the previous item; remember, the goal is "scrutable" printout.

f. Keep a thorough change log: date, perpetrator, etc. Put the date, etc., at top of file when a new version is released.

g. Put `\endinput` at the end of the actual macros in the file, to free anything that follows the `\endinput` from the usual TEX syntax rules. This is a natural place to put user documentation and mass commentary, so that it can all be in one file, while saving the (admittedly slight) time that TEX would require to read and ignore it if presented as comments within the body of the file.

So What Happens Now?

As mentioned above, a number of people came up with these ideas. The discussion should now go out to the general TUG community on what constitutes a well-documented macro. In addition to comments and criticisms, which we most certainly want to receive, we thought it might be worthwhile to invite TUG members to document an existing macro implementing the ideas presented above.

Please send all correspondence — suggestions, samples, criticisms — to either of the addresses below. Helen Gibson and Regina Girouard have agreed to share the task. All mail from Europe should be forwarded to Helen; all North American mail to Reggie. We will then sort through the material, and write up what we find in upcoming issues of the TUGboat.

Helen Gibson
c/o The Wellcome Institute
183 Euston Road
London NW1 2BP
Great Britain
01-387-4477

or

Regina Girouard
The American Mathematical Society
201 Charles Street
P. O. Box 6248
Providence, Rhode Island 02940 USA
(401) 272-9500 (ext. 224)