The MAKEPROG System

Joachim Schrod

Introduction

D. Knuth has introduced the concept of "Literate Programming" where programmers should explain to potential human readers of their programs what they want the computer to do. To support this for Pascal programs he has created the `WEB` *system of structured documentation*, an extension of Pascal. In `WEB` a program is splitted into sections, every section contains a documentation and a program part (both parts can be empty).* Such a `WEB` program is transformed by the `TANGLE` processor into a program source and the `WEAVE` processor produces an output which can be fed into TeX to get a fine looking document.

At the Technical University of Darmstadt we have used this concept—with CWEB—in many of our projects, e.g. for our portable `DVI` driver family. But for many programming languages a `WEB` system is not available and is often difficult to make; e.g., to prettyprint TeX macros is a non-trivial task due to TeX's dynamic lexical analysis. To stop this gap I have developed the MAKEPROG system. It makes it possible to document programs with TeX in a `WEB` like fashion, the program parts of the documentation file can be extracted to yield the program file. During the extraction process the documentation file can be altered with change files.

The MAKEPROG system consists of two parts: (1) the MAKEPROG processor which does the extraction and (2) the macro file `progdoc` which makes formatting facilities available. The MAKEPROG processor is derived from `TANGLE`, therefore there should be no difficulties for every site running `WEB` to install MAKEPROG. The macros in `progdoc` are implemented with MAKEPROG, the file `progdoc.doc` is the definitive source. `progdoc.tex` is only delivered to allow the printing of `progdoc.doc`.

The following deficiencies are known to me (some are not inherent but there wasn't the time to do it until now):

— MAKEPROG does not rearrange the code as `TANGLE` does. This is the most important feature which is lacking. The support of stepwise refinements is one of `WEB`'s main advantages. But for a few programming languages—e.g. TeX—this is not so problematical because their identifiers are dynamically bound at run time.

— MAKEPROG does not prettyprint the program part because it knows nothing about the host language and therefore nothing about the lexical and syntactical structure of the program. Instead it just prints the program part verbatim. But compared with wide spread verbatim setting macros they have the advantage that you can print your program part if you have embedded tabs in it. The macros will replace every tab with one to eight spaces according to the current column.

— MAKEPROG produces no index. This is impossible because it does not know what an identifier looks like. (This can even change inmidst a program, cf. TeX!)

— Because the documentation file is a TeX file and TeX does not recognize change files the complete documentation (with a change file) cannot be printed. `TIE` must be used to create a new master file which can be printed afterwards.

— The macros in `progdoc` do not produce a title page and a table of contents. This could be done easily.

— There is no LaTeX version of `progdoc` available. But this is simple, too—look at the comments in `progdoc.doc`.

— The page breaking is not very lucky. I still have to play with the penalties.

— MAKEPROG should insert the actual date as a comment line in front of the produced program file. The syntax of the comment line (start and end of a comment) must be specifiable.

* In fact there is a third part, the macro part, which is not important in this context.

How To Use MAKEPROG

The documentation is produced with the macro set `progdoc` built on Plain TEX. Therefore the documentation file must start with `\input progdoc`. Afterwards you can structure your document into sections and the sections into groups. The first section of a group starts with `\chap`, it corresponds to the starred section (‘`@*`’) of a `WEB` program. `\chap` has one parameter, the title of the section group. The parameter is ended by a dot. The dot is printed by the macro. Every other section starts with `\sect`. These macros produce a number in front of each section; this number is incremented with each new section.

Within a section one or more program part(s) can be specified with the macros `\beginprog` and `\endprog`. Both macros must start at the beginning of a line. If `\beginprog` does not start at the beginning of a line verbatim typesetting will be switched on but no extraction to the program file will result afterwards. After `\beginprog` the rest of the line is ignored. If `\endprog` does not start at the beginning of a line or if it is not followed by white space (blanks, tabs, or end of line) neither verbatim typesetting nor extracting will stop.

Outside of the program part—in the so called documentation part—you can use the vertical bar to print small texts verbatim, e.g. identifiers, macro names, etc. A vertical bar starts the verbatim mode, the next vertical bar stops it. You can use `\origvert` to get an original vertical bar. `\vbar` is the character with the ASCII representation of a vertical bar in the actual font.

After you have finished writing your document you can print it with TEX and you can run the MAKEPROG processor to extract all program parts into a program file. During the extraction MAKEPROG will recognize change files like `TANGLE` does.

Installation

The first step is to install the MAKEPROG processor. Because it is derived from `TANGLE` this should be rather easy. Just take your local `TANGLE` change file and you should have very few alterations (perhaps much to delete). You will need to put the MAKEPROG processor somewhere where your local command processor will find it—perhaps you will even need a command script around it. But this will be the same as it was with `TANGLE`. By the way, you can contact me if you need change files for MS-Pascal V3.11 or higher, for the HP-Pascal compiler on a HP-UX machine, or for the Pascal compiler on a PCS Cadmus System, contact me. Also you can get an object file for an Atari ST from me.

You can test your new program by running `progdoc.doc` through it. The output must be identical to `progdoc.tex`. Well, the main work is now done. You still have to put `progdoc.tex` in a directory where TEX will find it and then have much fun. (Don’t worry—be happy ...)

Errors and Remarks

If you have found an error or if you have some remarks or suggestions please contact me. My Bitnet address is `XITIJSCH@DDATHD21` but please note that you perhaps have to mail me again because our connection is very instable and incoming mails are often lost. *I will acknowledge every mail within a week.*

You can also reach me with the old fashioned mail:

Detig · Schrod TEXsys OHG
Joachim Schrod
Kranichweg 1

D-6074 Rödermark-Urberach
FR Germany
Phone: (+60 74) 16 17

Distribution

You can give the MAKEPROG system to everyone you want but it must be the complete system, i.e. at least the three files `makeprog.web`, `progdoc.doc`, and `progdocu.tex`. *It is explicitly forbidden to pass on the system without the documentation—if you distribute* `progdoc.tex` *without* `progdoc.doc` *your bad conscience will torture you until eternity.* Furthermore this restriction for distribution must be told to everyone who gets the MAKEPROG system from you.

Of course I do not make any warranties. (The usual blablah should follow here.)