TEX←Scm 3 Manual (Draft) *
file: texscm-manual

Keith Wright

September 29, 2025 at 14:39 EST

0.1 Preface

It might be said that the first draft of this program was written around 1995 when I was paid to write Pascal programs. I admired the typeset programs [?, ?] that Knuth wrote under the slogan "Literate Programming" [?, ?], but I was paid to write working programs, not beautiful programs. Still, I wrote a very simple program that read a Pascal program and produced a TEX input that listed the program with keywords in bold-face and comments typeset by TEX.

It is simple to do that with Pascal, because there is a small set of keywords fixed by the language definition. It is far more complicated to do it with Scheme, because macro definitions create new keywords. There is no code left from that so-called "first draft", but there is a design principle that I am following:

(1) The source code is a Scheme program. No preprocessing is needed before compiling and running the program.

A wise grad student once said to me "If you don't touch it, you can't mess it up." But this program is intended to facilitate a dialog. The computer must be allowed to have its say, but must not be allowed to mess it up.

(2) Despite talk of adding remarks to the program, the original (Scheme) source code is not modified. Instead, it is copied to the reply with changes made and remarks added there.

Despite the name "dialog", Plato's dialogues include more than two participants. Of course there may be more than one human programmer working on the program, but there also might be more than one computer program. "The computer" can refer to TEX←Scm or the underlying Scheme system. Some remarks are made by Scheme and recorded by TEX←Scm.

0.2 Introduction

Not a compiler, not a pretty printer, TEX←Scm is a new kind of program. It is a Dialog Moderator. It converts a Scheme program into a TEX document which contains the text of the program shown with different fonts for variables, keywords, and symbols, the defining occurrence of an identifier is underlined[1] . The comments are interpreted as TEX commands. That is, it is just Ascii text which the LATEX program can process into a nicely typeset document.

In addition to the program itself, the output of (parts of) the program is shown, in the same way as the text itself. The result looks like a transcript of a REPL session, but with better typography.

A line that begins with ;# is a remark. Any Scheme system will treat this as a comment. It will not affect the meaning of the program itself, which remains syntactically correct Scheme (assuming it ever was).

Both comments and remarks combine into blocks if they occur on subsequent lines with the same indentation as the first. In the case of comments, this just means that the entire block of consecutive

[1] At least it should be. Implementation not started.

comments is sent to LaTeX as a unit, possibly with the result that all of them are combined into a single paragraph in which line breaks are inserted and justification done by LaTeX.

In the case of remarks, the TeX←Scm program might make alterations. Since the intention is that reply will replace the original program it must be equivalent, at minimum, in the sense that the changes are idempotent, so running TeX←Scm on the reply generated should result in a identical copy.

On the other hand, the point of running TeX←Scm at all is that it acts as a REPL, displaying the result of evaluating a block of code as a reply remark. If the correct remark is already there it is not repeated. An incorrect reply remark will be corrected, not by removing the incorrect reply, but by adding a further remark which complains and displays the computed result.

In the case of a remark extended across several lines, each subsequent line must begin with `;#` , indented to the same column as the first `;#` .

After the `;#` in that first line of the block there is an identifier which determines the meaning of all the rest. The most important of these identifiers is `=>`. There is no whitespace allowed here, so that `;#=>` appears as a unit.

- `;#`
  A remark is contained in the remainder of the line, ending at the end of the line, like any comment. The Scheme program continues on the next line as if nothing happened.

- `;#=>` ⟨datum⟩
  Evaluate the preceding ⟨expression⟩ (i.e. part of the Scheme program). The given ⟨datum⟩ is the proposed result. If the ⟨datum⟩ comprises several lines, each line must begin with `;#` followed by white space. If the result computed by evaluating the preceding expression is *equal?* to the proposed result then nothing happens, the `;#=>` ⟨datum⟩ remains in the program text. If the ⟨datum⟩ computed by evaluation differs from the proposed result then the proposed result still remains as is, but it is followed by remarks inserted by TeX←Scm. The newly inserted remarks consists of a warning message followed by the computed value of the preceding ⟨expression⟩.

  For example:
  `(* 6 7)`
  `;#=>` 42
  remains unchanged in the Scheme file and prints as
  (* 6 7)
  ⇒ 42

  On the other hand:
  `(* 6 7)`
  `;#=>` 48
  is updated to something like:
  `(* 6 7)`
  `;#=>` 48
  `;#=>?` {\bf Warning! } proposed result differs from computed:
  `;#=>!` 42
  and prints as
  (* 6 7)
  ⇒ 42
  ⇒**? Warning!** proposed result differs from computed:
  ⇒**!** 48
  The (human) programmer should understand the reason for the discrepency, delete the incorrect ⟨datum⟩, and make the correct one the the new proposed result. Atmospherics in the proposed result may be edited while doing this.

  One might want to write:
  `(* 6 7);#=>`

all on one line. This could conflict with a definition of "block" that requires blocks to consist of whole lines, or we could says that the trailing remark is atmosphere in the data block and that responding to it is part of evaluating the data.

- `;#->`
  The result of evaluating the preceding ⟨exression⟩ should be not a general datum, but a (list of) string(s), which should be sent to LaTeX as is.

  `;# ;` ⟨character⟩*
  Following lines contain those strings as the body of one-semicolon comments. Those strings are sent directly to LaTeX.

- `;#:=` ⟨symbol⟩ ⟨value⟩
  The ⟨symbol⟩ must be one of a list which is meaningful to the dialog manager. For example:
  `;#:=` *title* "`\\TeXScm`"
  `;#:=` *author* "`Keith Wright`"
  `;#:=` *cmnt-width* "`8cm`"
  `;#:=` *cmnt-width* (*auto*)
  Let's call these "editorial remarks" because they do not pertain to the results of the program, but to the way they are presented.

Inside a comment, Scheme code is marked `;#'` . This is formatted for printing as Scheme, but of course it is not evaluated. If I write inside a comment:
`This is a semisharp-quote ;#'(define pi 3.14159) faked for demo`
I should see "This is a semisharp-quote #'(**define** *pi* 3.14159) faked for demo". Note `#'`⟨datum⟩ is the R6RS abbreviation for (`syntax` ⟨datum⟩). See R6RS [**?**, §4.3.5].

References

[1] David Herman & Mitchell Wand, *A Theory of Hygienic Macros*, ESOP 2008 & Springer-Verlag LNCS 4960 pp.48–62 (2008)

[2] Aaron Hsu, *ChezWEB User's Guide*, gopher://gopher.sacrideo.us/1chezweb

[3] Richard Kelsey, et. al. *Revised*^{5} *Report on the Algorithmic Language Scheme*,

[4] Donald E. Knuth, *Literate Programming*, Computer Journal 27(1):97–111, 1984.

[5] Donald E. Knuth, *Literate Programming*, CSLI, 1992. CSLI Lecture notes no. 27.

[6] Donald E. Knuth, *TeX: The Program (Volume B of Computers and Typesetting)*, Addison-Wesley, 1986, ISBN 0-201-13438-1

[7] Donald E. Knuth, *Metafont: The Program (Volume D of Computers and Typesetting)*, Addison-Wesley, 1986, ISBN 0-201-13438-1

[8] Helmut Kopka and Patrick W. Daly, *A Guide to LaTeX, third edition*, Addison Wesley (1999)

[9] B. O. Peirce, *A Short Table of Integrals*, Ginn and Company (1929)

[10] Plato, *The Collected Dialogs*, Bollingen Series LXXI, Princeton University Press

[11] Norman Ramsey, *Literate programming: Weaving a language-independent web*, Communications of the ACM, 32(9):1051–1055, 1989.

[12] Norman Ramsey *The noweb Hacker's Guide*, Princeton University, 1992. Revised 08/1994.

[13] John D. Ramsdell *SchemeWEB – WEB for Lisp. Simple support for literate programming in Lisp.* The MITRE Corporation, 1994

[14] `https://small.r7rs.org/wiki/R7RSSmallErrata/`

[15] Alex Shinn, John Cowen, and Arthur A. Gleckler (eds.),

[16] Alex Shinn, John Cowen, and Arthur A. Gleckler (eds.), *Revised*^{7} *Report on the Algorithmic Language Scheme*,

[17] Michael Sperber, et. al. *Revised*^{6} *Report on the Algorithmic Language Scheme* Cambridge University Press ISBN: 9780521193993 (2010) or http://www.r6rs.org/

[18] Dorai Sitaram, *SLaTeX*, 1991, 1999
`http://www.ccs.neu.edu/~dorai/slatex/slatex.tar.gz`

[19] Michael Spivak, *Calculus on Manifolds*, Benjamin/Cummings Publishing (1965)

[20] Fransisci Vietæ, *Variorum de rebus mathematicis responsorum* (1593)

[21] Alfred North Whitehead, *Process and Reality*, Macmillan Company (1929)