Scheme–BRIEF interface

Larry Bartholdi & his gang

May 11, 1993

Contents

1 Introduction

It is well known that many powerful editors (like EMACS) are based on Scheme and cohabit with it; and it is well known to Scheme programmers that closing 10 parentheses or so with primitive command editing and no indentation is painful. Up to now, the only choice PCSCHEME users had was to endure the keyboard typing or to use the unreliable and poor editor EDWIN.

Rather than re-write yet another powerful editor, I have chosen to make good use of the macro capabilities of an already-existing product, BRIEF[1]. BRIEF has many of the options provided by EMACS, while being well suited to PC compatibles; it is programmable in a subset of the C language.

The problem with this approach is interfacing between BRIEF and PC-SCHEME. The solution chosen is to create a temporary file within BRIEF, that will be loaded as scheme code by PCSCHEME. The only two messages supported are "evaluate the buffer" and "evaluate the marked block", which are written as simple Scheme expressions by the BRIEFmacros.

2 Installation

You need the following files:

`SCHEME.CM` the BRIEF compiled code

[1] BRIEFis trademark of Borland International

`SCHEME.MNU` the help file

`SCHEME.CMP` the completion dictionary

`SCHEME.CB` (optional) the Brief source code.

Move `SCHEME.CM` to `\brief\macros` and `SCHEME.MNU` to `\brief\help`; add "`set BSCHEME=`*the path to SCHEME.CMP, including the trailing* `\`" to your `autoexec.bat` file; and you know what to do with `SCHEME.CB`...

You then have to add the following to your initials macro file:

```
(macro .s
    (
        (tabs 9 17)
        (if (== (inq_macro "scheme_indent") 0)
            (load_macro "scheme")
        )
    )
)
```

and (if you program in Scheme-WEB),

```
(macro .sw
    (
        (.s)
    )
)
```

Don't forget to recompile it!

3 Use of the Brief package

A few commands have been redefined, following where possible the Brief tradition. Interface to PCS, () matching, indentation, S-expression marking, and word completion from an arbirary dictionary have been added to the basics.

The commands whose meaning changed are:

⟨**Enter**⟩ Automatically indents the next line

⟨**Ctrl-h**⟩ Provides a short description of the new commands

")" Searches for the matching "("; if the match is visible, highlights it for a second; otherwise displays the line where the match came from on the status line.

⟨**Tab**⟩ If a block has been marked, indents every line in the block; if the cursor is in leading whitespace, indents the current line; otherwise completes the current word. The current word is considered to start at the closest parenthesis or whitespace backwards, whichever comes first (well, last). Continually depressing the ⟨Tab⟩ key searches for another completion to the current word. ⟨Esc⟩ aborts the completion process.

⟨Shift-Tab⟩ Positions the cursor to column 49 and inserts a `;␣` (start of comment). It is my hope this will make programs more readable...

⟨Ctrl-G⟩ Creates a menu of all the objects defined in the buffer. If procedures are defined by `(define (foo x y) ...)`, the procedure's name with its arguments are displayed. In any case selecting an entry positions the cursor on that object's declaration.

⟨Ctrl-A⟩ Marks the smallest S-expression (list) containing the cursor. An error occurs if the cursor is not inside a list.

⟨Ctrl-Z⟩ Marks the largest S-expression (list) containing the cursor. An error occurs if the cursor is not inside a list.

⟨Alt-F10⟩ Evaluates the entire buffer, passing control to PcScheme (or whoever called Brief).

⟨Ctrl-F10⟩ Evaluates the marked block, passing control to PcScheme (or whoever called Brief). If no block is marked, the largest S-expression is selected first.

⟨Ctrl-F1⟩ Toggles on or off the tabbing and autoindentation features. Use this when very desperate, esp. if you edit a very large file, or want to insert some non-scheme text (like in Scheme-WEB).

4 Bugs

- We proudly inherit all bugs in Brief(!).
- Brief is not shareware.
- There is no mechanism to handle source-file errors elegantly.
- The macros may be slow; this is because CBrief is an interpreted language.
- Some problems occur in Scheme-WEB files, when text editing should supersede Scheme macro capabilities.

5 The PcScheme side

A small procedure, `b`, is defined. It can be called with 0 or 1 arguments, the optional argument being faithfully passed to brief.

`B` calls Brief with all memory available and no screen-saving option. This is not necessary since Brief saves the screen itself.

`B` then loads the file handed back by Brief, and loads it.

```
(define (b . args)
  (let ((brief (dos-search-file "b.exe"))
        (transfer (string-append (dos-get-env "BSCHEME")
                                 "scheme.tmp"))
        (getarg (lambda (a)
          (cond ((string? a) (string-append " " a))
                ((integer? a)
                 (string-append " -m\"goto_line "
                                (integer->string a 10)
                                "\""))
                (else (error 'B "Invalid argument " a))))))
    (if (null? brief)
        (error 'B "Could not find B.EXE"))
    (dos-call brief
              (apply string-append (map getarg args))
              0
              1)
    (if (file-exists? transfer)
        (load transfer))
    *the-non-printing-object*))
```