Psd for the Impatient

for version 1.1

Pertti Kellomäki, pk@cs.tut.fi
Tampere University of Technology
Software Systems Lab
Finland

August 19, 1997

1 Introduction

This is a quick introduction to psd, the Portabe Scheme Debugger. It assumes, that psd is already installed in your system. It also assumes that you are using GNU Emacs, because psd needs it for displaying source code.

2 How to Debug a Procedure?

Write your Scheme program as usual. Start up an inferior Scheme session with the command `M-x run-scheme`. If the mode line of the `*scheme*` buffer says "Inferior Scheme Psd: run", then psd is already ready for use. If not, you will need to give the command `M-x psd-mode` in the Scheme buffer. To debug a procedure:

1. Move the cursor on top of the procedure you want to debug.
2. Type `C-u ESC C-x`. The procedure has now been instrumented, and you can place breakpoints within it.
3. Put a breakpoint inside the procedure. Move the cursor to the line where you want to stop the execution and give the command `C-x SPC`. You can also set breakpoints by giving the name of a procedure where you want a breakpoint. Use the command `C-c b` for that.
4. Run your program. When execution reaches a line with a breakpoint, the debugger is invoked and you can examine and change variable bindings etc.
5. Once you have located and fixed the bug, simply load the corrected definition.

3 Debugger Commands

These are commands that you can give when the program has been stopped to a breakpoint, and the prompt says `psd>`.

val *sym* gives the value of *sym* in the current scope.

set! *sym val* sets the value of *sym* to *val* in the current scope.

c clears all breakpoints on current line

g continues evaluation until the next breakpoint

w shows the current context as file name and a list of procedure names. For example, if the context is `"/tmp/killme.scm:(encode encode-symbol)"`, it means, that the you are in file `/tmp/killme.scm`, inside a procedure called `encode-symbol`, which is inside the procedure `encode`.

s steps one step in the evaluation process. Each time an expression is about to be evaluated, psd displays it and waits for a command. When an expression has been evaluated, psd displays the result and waits for a command.

n continues evaluation until evaluation reaches a different line

r *expr* evaluates *expr* and returns its value as the return value of the current expression

A list is taken to be a procedure call that is to be evaluated. All the essential procedures in R4RS are visible to the evaluator. Any other command displays a list of available commands.

If the debugger does not seem to be doing the right things, try the Emacs command `M-x psd-reset`, which will clear all the breakpoints and reset the runtime system.

4 Command Summary

This is a short list of the available commands. The Emacs commands are:

`C-c b`	psd-break
`C-c d`	psd-debug-file
`C-c e`	scheme-or-psd-send-definition
`C-c C-e`	scheme-or-psd-send-definition-and-go
`C-x SPC`	psd-set-breakpoint
`ESC C-x`	scheme-or-psd-send-definition
`M-x psd-reset`	clear all breakpoints and reset the psd runtime

The debugger commands are:

`val` *sym*	give the value of *sym*
`set!` *sym val*	set the value of *sym* to *val*
`c`	remove breakpoints on this line
`g`	run until the next breakpoint
`w`	give the current context
`s`	step one step in the evaluation process
`n`	run until evaluation reaches another line
`r` *expr*	return *expr* as the value of current expression

A list is taken to be a procedure call to be evaluated. It can also be a `set!` form.

5 Useful Things for Your `.emacs`

If you want psd to be loaded automagically when you start up a Scheme session, put the expression

```
(setq inferior-scheme-mode-hook
      (cons '(lambda () (psd-mode 1))
            inferior-scheme-mode-hook))
```

in your `.emacs` file, or with Emacs 19, put instead

```
(add-hook 'inferior-scheme-mode-hook
          (function (lambda () (psd-mode 1))))
```

There is also a hook variable `psd-mode-hook` that you can use for customizing psd mode, for example to change key bindings.