Debugging with STKDB

The STk Source-Level Debugger

First Edition, for STKDB version 1.0
July 2003

Paul N. Hilfinger

(Send bugs and comments on STKDB to Hilfinger@cs.berkeley.edu.)
Debugging with STKDB
TEXinfo 2005-01-30.17

387 Soda Hall, M.C. 1776
University of California
Berkeley, CA 94720-1776

Summary of STKDB

Generally speaking, a "debugger" is a program that allows its users to run, control, and observe other programs in action for the purpose of diagnosing problems in these programs. STKDB does this for Scheme programs written for the STk system. It is deliberately designed to resemble the GNU debugger, GDB, which provides support for many languages, including Ada, C, C++, Java, Fortran, and Modula~2.

STKDB provides a number of features that are common to most modern debuggers for other languages as well. You can:

- Add (invisible) instrumentation to functions you have written so that you can track them.
- Place *breakpoints* on expressions of your program, which cause STKDB to stop the program when it evaluates those expressions.
- *Step* a program, evaluating one subexpression at a time and stopping after each one.
- Find out where a program has stopped, whether due to a breakpoint, an error, or an interrupt (i.e., a `C-c`).
- Find out how the program arrived at a point where it has stopped—that is, to find out what function call started the execution of the function body where the program stopped, and in turn how that function call came to be evaluatued.
- Examine the variables of a stopped program—in fact, you can evaluate any Scheme expression at the point where the program stopped.

STKDB is itself a Scheme program—a collection of function and variable definitions— that runs on top of STk. It is loaded into STk like any other Scheme program. Once loaded, it provides a function, `stkdb:debug-file`, which instruments and loads other files full of Scheme definitions. Ordinarily, the functions loaded in this way behave like ordinary Scheme functions. However, STKDB also provides a function, `stkdb`, that starts a special read-evaluate-print loop (in other words, it acts as STk does normally, reading what you type, evaluating it, and printing the result). This loop understands commands for performing special debugging functions, such as setting breakpoints, as well as the usual task of evaluating expressions.

While you can use this simple command-line interface to talk to STKDB, you will probably find it much more convenient to use Emacs, which has special Scheme debugging support that is integrated into the Scheme execution support already provided. Emacs provides

- Menu-based access to most STKDB commands.
- Fast key bindings ("accelerators") for frequent operations.
- Simplified setting of breakpoints.
- Text highlighting to indicate points in your program where execution has stopped.
- And, of course, access to this documentation.

In the discussion that follows, we will show both the text-based and Emacs-based commands.

1 Beginning a STKDB Session

STKDB uses the `slib` Scheme library; your STk system should first be configured to "know" the location of the `slib` library. STKDB can then load the library, if necessary. Once STk is properly configured, you can load STKDB into your STk session with the commands

```
(load "DIR/stkdb.scm")
(import stk-debugger)
```

where *DIR* is the directory in which `stkdb.scm` is installed.

This is all much simpler in Emacs. Assuming you have properly installed the STKDB Elisp file, `stkdb.el`, you need only issue the command *M-x load-file* ⟨RET⟩ *DIR/stkdb.el* ⟨RET⟩, or if *DIR* is on your Emacs load-path, then *M-x load-library* ⟨RET⟩ *stkdb* ⟨RET⟩ will also work. Once you've done this, Emacs will set up a buffer running Scheme and with STKDB properly loaded as soon as you use the "Debug File" menu command (or *C-c d*) in a Scheme source buffer (see ⟨undefined⟩ [Preparing Files], page ⟨undefined⟩).

Normally, you won't even bother to issue these `load` commands by hand, but will instead include the line

```
(load "stkdb")
```

in your `.emacs` file.

2 Preparing Files for Debugging

In order to control the evaluation of a set of functions you want to examine, STKDB needs to *instrument* them—that is, to add Scheme code to these functions before they are loaded. The instrumentation is normally invisible to you (there are STk functions that allow you to examine the internal definition of a function, but you'll usually just go back to the source file for that information instead, and STKDB doesn't need to modify any source files).

`stkdb:debug-file` *file* [Function]

`stkdb:debug-file` instruments and loads the Scheme source file *file*. The *file* argument is a string. Only functions that have been loaded in this fashion can be controlled by STKDB.

```
(stkdb:debug-file "problem1.scm")
```

The easy way to issue this command is from Emacs. There are several ways:

- *C-c d* while visiting a file containing Scheme code will create a Scheme buffer and start the Scheme process, if necessary, and then instrument and load the source file. This command also starts STKDB; you will see STKDB's prompt in the Scheme buffer.
- The "Debug File" item in the Debugging menu that appears when visiting a buffer of Scheme code does the same thing as *C-c d*.

3 Activating the Debugger

Once STKDB is loaded into a Scheme session running under Emacs (see ⟨undefined⟩ [Setting Up], page ⟨undefined⟩) and you have loaded one or more files containing Scheme functions you want to debug (see ⟨undefined⟩ [Preparing Files], page ⟨undefined⟩), the next step is to activate the debugger so that it controls the evaluation of test cases you type. The Scheme command `(stkdb)` puts your Scheme session into *debugging mode*; you tell that this mode is active because your prompt will either end in `[-]` or `[1]`, rather than the usual style of prompt (which ends in `>`).

In debugging mode, an assortment of commands for manipulating and examining the evaluation of your program become available. These are described in the sections that follow. In addition to these commands, ordinary expression evaluation is available, as at the usual STk prompt. Typically, you will enter debugging mode, type in a Scheme expression that exercises some piece of program text you are trying to debug, possibly after first setting up some breakpoints in that text. When something "interesting" happens, you will again get a debugging prompt (usually ending in `[1]`, which indicates that you are stopped in the middle of an expression), and can then enter debugging commands. The command `reset` (no parentheses needed) will exit from debugging mode, cancelling evaluation of any expression you might be in the middle of.

4 Stopping and Continuing

When in debugging mode, evaluation of an expression can be suspended and control given back to you (indicated by a `[1]` prompt, generally) under several circumstances:

- The program evaluates an expression that has a *breakpoint* set on it.
- You ask STKDB to execute a single evaluation *step*, and that step completes.
- Scheme detects a run-time error in your program (e.g., performing a `car` on the value nil).

4.1 Showing Where You've Stopped

Whenever STKDB stops your program during evaluation, it indicates where the stop has occurred. If you are not using Emacs, the information consists of a file name and line number, which is a bit clumsy to use. When run under Emacs, however, things are much clearer. Emacs will show you a buffer containing file in which evaluation has stopped, highlighting the precise expression at which you are stopped. This highlighting is color-coded to indicate the reason for the stop:

- Green highlighting indicates an expression that is about to be evaluated. It results from breakpoints (see ⟨undefined⟩ [Breakpoints], page ⟨undefined⟩), steps (see ⟨undefined⟩ [Stepping], page ⟨undefined⟩), and interrupts (*C-c C-c*).
- Light blue highlighting indicates an expression that has just finished being evaluated. The value just returned from that expression may be displayed in the Scheme execution buffer, depending on how you got there. This kind of stop can result from stepping or continuing a program with the `finish` command (see ⟨undefined⟩ [Continuing], page ⟨undefined⟩).
- Purple highlighting indicates an expression that is in the middle of being evaluated. This is how the `up` command (see ⟨undefined⟩ [Viewing Callers], page ⟨undefined⟩) indicates where a certain function was called.
- Red highlighting indicates an expression whose evaluation caused an error.

4.2 Setting and Clearing Breakpoints

A *breakpoint* is a point in program text at which to *break off* evaluation, returning control to the programmer. What makes them interesting is that, unlike errors, it is then possible to continue evaluation from the breakpoint.

In debugging mode (the `[-]` or `[1]` prompt), the command

```
break func
```

sets a breakpoint at the beginning of the code for function *func* (this is a command, not a Scheme function call; there are no parentheses). You can only set breakpoints in functions that have been instrumented first (see ⟨undefined⟩ [Preparing Files], page ⟨undefined⟩). You can use `br` or just `b` as shorthand for `break`. The command

```
break file:line
```

sets a breakpoint at line number *line* (numbering from 1) of file *file*.

As usual, it is much more convenient to use Emacs for setting breakpoints. When the cursor is positioned at the line on which you want to break in a Scheme source buffer, the Emacs command *C-x* ⟨SPC⟩, or the "Set Breakpoint" menu command in the Debugging menu, will set a breakpoint on the indicated line.

In response to these commands, STKDB will confirm with a *breakpoint number*, which you can use subsequently to refer to the breakpoint. For example,

```
stk[-] br sample1.scm:200
Breakpoint 1 at sample1.scm:200
stk[-] break replace-all
Breakpoint 2 at sample1.scm:20
```

To remove breakpoints, use the command `delete` in debugging mode. This command allows you to remove either specific breakpoints (by the numbers assigned by the `break` command) or to remove all breakpoints:

```
stk[-] delete 1 2
Removing breakpoint 1
Removing breakpoint 2
stk[-] delete
Remove all breakpoints? [yn] y
Removing all breakpoints.
```

In Emacs, put the cursor on the line whose breakpoint you wish to remove and use the "Clear Breakpoint" menu item, or use the "Clear All Breakpoints" menu item to delete all breakpoints. In addition, re-loading a file under STKDB (as with *C-c d*) removes all breakpoints on that file.

To see the current set of breakpoints set in your program, use the command

```
info break
```

in Scheme debugging mode.

4.3 Stopping Conditionally

Sometimes, you may have observed that there is a problem at some point in your program that only occurs under certain infrequent circumstances (such as a list being null). Placing a breakpoint at that point will force you to issue `continue` commands annoyingly many times before you get to the problem. One way around this is to make the breakpoint *conditional*.

The command

`condition` *bpnum expr*

where *bpnum* is the number of an existing breakpoint and *expr* is a Scheme expression, will cause STKDB to stop at the breakpoint only if *expr* evaluates to a true value. You may abbreviate `condition` as `cond`. STKDB evaluates *expr* in the frame of the breakpointed expression (see ⟨undefined⟩ [Viewing Values], page ⟨undefined⟩). To cancel the condition, use

`condition` *bpnum*

(that is, without the conditional expression). In Emacs, the "Condition Breakpoint" menu item will make the breakpoint at the cursor conditional, prompting for the condition in the minibuffer (simply typing ⟨RET⟩ in response will deconditionalize the indicated breakpoint).

For example, suppose that a certain function is supposed to return a list of symbols, but sometimes returns a list with a few scattered null lists as elements. The function constructs this return value using an expression

```
    (cons (car L) rest)
```

You place a breakpoint on this line, to which STKDB responds:

```
Breakpoint 4 at glorp.scm:44
```

In order to filter the responses so that you look only at interesting cases, use the command

```
cond 4 (null? (car L))
```

Your program will then stop if `L` starts with a null list (or if `L` is not a pair, so that evaluating `car` causes an error).

4.4 Stepping Through One Expression at a Time

Stepping a program means evaluating one subexpression at a time (just as a Scheme interpreter would), stopping after each. This only makes sense when you are in the midst of evaluating an expression in debugging mode and STKDB has stopped, presenting you with a *[1]* prompt. At that point, you have the following choices:

- The command `step` (which may be abbreviated `s`) continues evaluation until it reaches the next instrumented subexpression (that is, the next Scheme expression in a file that you have loaded with `stkdb:debug-file` or equivalent key sequence). In Emacs, simply use the ⟨F5⟩ key for this purpose.
- The command `next` (abbreviated `n`) continues evaluation until it completes the execution of the current subexpression and reaches the next one after that (or hits another breakpoint). In Emacs, this is the ⟨F6⟩ key for this purpose.

To see the distinction between these two, let's suppose that STKDB is stopped at the expression `(f (g 3))` below (in Emacs, that expression would be highlighted in green):

```
(define (g x) (+ x 3))
(define (f y) (* y 7))

(compute (f (g 3)) (h 1))
```

At this point, a `step` command moves to `(g 3)`. Another `step` command moves to `(+ x 3)` in the definition of `g`. A third returns us to `(g 3)`, but highlighted in blue (assuming we are using Emacs) to indicate that we have finished evaluating it. A fourth `step` takes us to `(* y 7)` in the definition of `f`. A fifth returns us to `(f (g 3))` (in blue), and a sixth takes us to `(h 1)`.

If, on the other hand, we go back to where we started and use a `next` instead of `step`, we go from `(f (g 3))` immediately to `(h 1)`, skipping the intermediate steps.

When using `step`, you can arrange to see the value of each subexpression you step through. Use the debugging-mode command `show values` (the opposite is `show novalues`), or turn on "Show All Returned Values" in the "Settings" submenu of the Debugging menu under Emacs.

4.5 Tail Recursion and Debugging

In a Scheme program such as this:

```
(define (factorial n p)
  (cond ((<= n 1) p)              ;; Breakpoint here
        (else (factorial (- n 1) (* n p))))) ;; Line 3
```

it *looks* as evaluation of `(factorial 15 1)` ought to give backtraces like this:

```
[0] foo.scm:2 (factorial)
[1] foo.scm:3 (factorial)
[2] foo.scm:3 (factorial)
[3] foo.scm:3 (factorial)
...
```

But in fact, the actual backtrace will always have one line! This is because the call to `follows` on line 4 is *tail recursive*. That is, `follows` returns the value of this recursive call directly, without examining it or performing any other operations with it; the recursive call is the very last action of `follows`.

The Scheme language actually requires that tail recursions such as this must be able to run indefinitely, just like a loop in other languages. They are not allowed to require increasing amounts of space just to keep track of the call chain. As a result, you'll normally see "truncated" backtraces like this in tail-recursive situations.

You'll also see that the `next`, `step`, and `finish` commands work confusingly when dealing with tail recursion. For example, if you were evaluating `(factorial 5 1)` and you are at the expression `(<= n 1)` at the point that `n` is 1, any of these commands will simply print 120 (the final answer). You will *not* stop again at the call on line~3 (highlighted blue), because all the intervening recursive calls that got you to the point where `n` is~1 will have been "forgotten."

This behavior can be confusing. If so, STKDB gives you a way to "cheat." In Emacs, simply turn on the the "Keep Tail Recursion" flag in the "Settings" submenu of the Debugging menu, and then reload the files you're interested in with *`C-c d`*, as usual. You must reload them *after* changing the value of this flag in order to have an effect. In this mode, STKDB will treat tail recursions like general recursions. Of course, your program will now "blow up" by exhausting memory in some cases where it wouldn't have before, so you can't expect to be able to do any really long evaluations.

4.6 Resuming Normal Execution

To allow evaluation to proceed to the next breakpoint (or error), use the command `continue` (abbreviated `cont` or `c`). In Emacs, this is the ⟨F8⟩ key.

It is sometimes useful to finish evaluation of the current function, and then stop again, showing the value computed. The `finish` command (abbreviated `f`) does this. In Emacs, this is the ⟨F7⟩ key. This command leaves us at the call whose body we were just executing (highlighted in blue under Emacs).

For example, suppose we have called `(printem foo)`, where:

```
(define (printem LL)
  (if (not (null? LL))
      (begin (print L) (printem (cdr LL)))))

(define (print L)
  (display "[")
  (let loop ((x L))
    (if (null? x)
```

```
(display "]")
(begin
  (display (car x))   ;; << STOPPED HERE
  (if (not (null? (cdr x))) (display ","))
  (loop (cdr x))))))
```

and our program is stopped in `print` on the indicated line. The `finish` command will continue the program until we finish printing the current list and return to `printem`; the expression `(print L)` will be highlighted in blue.

5 Examining the Evaluation State

While your program has stopped, there are basically two sorts of things you'll need to do:

- Figure you why evaluation got to this particular expression. In part, this usually means figuring out what function call elsewhere in your program caused evaluation of the function containing this expression.
- Observe the values of the local variables and parameters at the point in the program where you've stopped.

5.1 BackTraces and the Call Chain

In debugging mode, the command `bt` (or `where` or `backtrace`) prints a *backtrace* of the current state of evaluation. If you are running in an Emacs scheme buffer, the menu item "Backtrace" displays the output of a backtrace in a separate buffer.

A backtrace is an account of how the current expression (the one you're stopped at) came to be evaluated. To see what this involves, consider the following program:

```
;; This is file count.scm, line 1
(define (count-tips tree)
   (if (pair? tree)
       (count-kids-tips (cdr tree)) ;; line 4
       1))
(define (count-kids-tips kids)
   (if (null? kids) 0
       (+
          (count-tips (car kids))   ;; line 9 << BREAKPOINT HERE
          (count-kids-tips (cdr kids)))))
```

where we have set a breakpoint at the indicated location, are in the process of evaluating

```
(count-tips '(martin
               (marty (sally tommy matt) (heidi taylor))
               (donald peter (melinda jessica))
               (george paul ann (john dana))))
```

and have stopped a few times at the breakpoint, so that, let's say, we are at the point in the program where `kids` is `(tommy matt)`. Asking for a backtrace will get us this:

```
*[0] /tmp/count.scm:9 (count-kids-tips)
 [1] /tmp/count.scm:4 (count-tips)
 [2] /tmp/count.scm:9 (count-kids-tips)
 [3] /tmp/count.scm:4 (count-tips)
 [4] /tmp/count.scm:9 (count-kids-tips)
 [5] /tmp/count.scm:4 (count-tips)
```

Translation: we are now at line 9 in `count.scm`, which is in `count-kids-tips`; we're there because we were called from line 4, which is in `count-tips`; we got there by being called from line 9 in `count-kids-tips`, etc. If the debugger were perfect, there would be a line `[6]` that said that we were in `count-tips` because we got called from the `(count-tips '(martin...))` line that the user typed. Sorry; it's not perfect.

We say that each line in the backtrace denotes a *frame*, basically an instance of the evaluation of a function. Frame number~0 is known as the *innermost frame*. The asterisk marks the *current frame*, i.e., the one the debugger is examining at the moment. One can also look at the other frames; See ⟨undefined⟩ [Viewing Callers], page ⟨undefined⟩. The highlighted expression that Emacs shows you when you stop is the expression within the innermost frame that is currently being evaluated. We'll refer to this point (somewhat archaically) as the *program counter* of the frame.

In Emacs, a copy of the backtrace gets put in a separate buffer. You can arrange to have it reproduced automatically each time the program stops at a breakpoint or step by turning on "Auto-Display Backtrace" in the "Settings" submenu of the Debugging menu.

5.2 Looking at Other Frames

A backtrace (see ⟨undefined⟩ [Backtraces], page ⟨undefined⟩) gives you a rough idea of the overall program state. For more details, you can examine the stack frames in detail. In debugging mode, the command `up` will increment the current frame number by one, so that we are looking at what is called the *caller* of what used to be the current frame. In Emacs, this is ⟨F3⟩ or "View Caller" in the Debugging menu. Emacs will highlight the expression at the program counter of this new current frame—typically a function call—in purple, indicating that it is not the innermost frame. At this point, you can examine the variables in that frame (see ⟨undefined⟩ [Viewing Values], page ⟨undefined⟩).

The inverse operation to `up` is (of course) `down` (`d`), or in Emacs ⟨F4⟩ or "View Callee" in the Debugging menu. In addition, the command `frame` *n* (or `fr` *n*), immediately makes frame number *n* the current frame, so that `frame 0` returns immediately to the innermost frame.

Changing the current frame has no effect on the commands that step or continue the program. In effect, we always step or continue from the innermost frame.

5.3 Looking at Variables and Parameters

At any time, the environment associated with the current frame—that is, the values of all variables and parameters that are visible at that point in the program—is also available to the debugger. In debugging mode, you can print the value of any Scheme expression, *EXPR*, with the command `print` *EXPR* (and you may abbreviate `print` as `pr` or `p`). If the syntax of *EXPR* does not conflict with that of any debugging command (i.e, if you are not trying to evaluate a simple variable with a name like `next`, `c`, etc.), then can you leave off the `print`, just as you would in an ordinary interactive Scheme session.

The debugging-mode command `info locals` (Debugging menu item "See Local Variables") will print the "local variables" of the current frame. This is basically the set of all parameters and let-bound variables defined with the function associated with the current frame.

By combining printing with the use of the `up` and `down` commands (see ⟨undefined⟩ [Viewing Callers], page ⟨undefined⟩), you can look at most of the variables relevant to your program while it is stopped. Reusing an example from elsewhere, consider:

```
;; This is file count.scm, line 1
(define (count-tips tree)
```

```
   (if (pair? tree)
       (count-kids-tips (cdr tree)) ;; line 4
       1))
(define (count-kids-tips kids)
   (if (null? kids) 0
       (+
          (count-tips (car kids))   ;; line 9 << BREAKPOINT HERE
          (count-kids-tips (cdr kids)))))
```

where we are evaluating

```
(count-tips '(martin
               (marty (sally tommy matt) (heidi taylor))
               (donald peter (melinda jessica))
               (george paul ann (john dana))))
```

and are stopped at the indicated breakpoint with the following backtrace:

```
*[0] /tmp/count.scm:9 (count-kids-tips)
 [1] /tmp/count.scm:4 (count-tips)
 [2] /tmp/count.scm:9 (count-kids-tips)
 [3] /tmp/count.scm:4 (count-tips)
 [4] /tmp/count.scm:9 (count-kids-tips)
 [5] /tmp/count.scm:4 (count-tips)
```

At this point, we might type any of the following:

```
stk[1] print kids
(tommy matt)
stk[1] p kids
(tommy matt)
stk[1] kids
(tommy matt)
stk[1] pr (car kids)
tommy
stk[1] (car kids)
tommy
stk[1] info locals
  kids: (tommy matt)
```

Now suppose we go up to previous frames:

```
stk[1] up
stk[1] info locals
  tree: (sally tommy matt)
stk[1] (cadr tree)
tommy
stk[1] up
stk[1] info locals
  kids: ((sally tommy matt) (heidi taylor))
stk[1] (cadr kids)
(heidi taylor)
```

In Emacs mode, if there is a backtrace displayed, it will reflect the results of the `info locals` commands, like this:

```
 [0] /tmp/count.scm:9 (count-kids-tips)
    kids: (tommy matt)
 [1] /tmp/count.scm:4 (count-tips)
    tree: (sally tommy matt)
*[2] /tmp/count.scm:9 (count-kids-tips)
    kids: ((sally tommy matt) (heidi taylor))
 [3] /tmp/count.scm:4 (count-tips)
 [4] /tmp/count.scm:9 (count-kids-tips)
 [5] /tmp/count.scm:4 (count-tips)
```

In addition, you can arrange to have local variables automatically displayed either for the innermost (top) frame or for all frames whenever your program stops. To see just those for the topmost frame, choose the "Auto-Display Backtrace/Top Frame" item in the "Settings" submenu of the Debugging menu; to see locals in all frames, choose the "Auto-Display Backtrace/Locals" item.

Appendix A GNU GENERAL PUBLIC LICENSE

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

   These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:
   a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
   b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
   c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

   The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

   If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.
4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.
5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.
7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

   This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.
8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.
9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.
10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software

which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
one line to give the program's name and a brief idea of what it does.
Copyright (C) year  name of author

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place - Suite 330,
Boston, MA 02111-1307, USA.
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details
type 'show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type 'show c' for details.
```

The hypothetical commands '`show w`' and '`show c`' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than '`show w`' and '`show c`'; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the program
'Gnomovision' (which makes passes at compilers) written by James Hacker.

signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Appendix B GNU Free Documentation License

0. PREAMBLE

   The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

   This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

   We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

   This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you."

   A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

   A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

   The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque."

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

2. VERBATIM COPYING

   You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

   You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

   If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).
C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
H. Include an unaltered copy of this License.
I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the

Title Page. If there is no section entitled “History” in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the “History” section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. In any section entitled “Acknowledgements” or “Dedications”, preserve the section’s title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section entitled “Endorsements.” Such a section may not be included in the Modified Version.

N. Do not retitle any existing section as “Endorsements” or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

   You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications." You must delete all sections entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

    The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

    Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.1
or any later version published by the Free Software Foundation;
with the Invariant Sections being list their titles, with the
Front-Cover Texts being list, and with the Back-Cover Texts being list.
A copy of the license is included in the section entitled "GNU
Free Documentation License."
```

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being *list*"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Index

(Index is nonexistent)