The FFI Reference Manual

a Foreign Function Interface (version 0.2)
for MIT/GNU Scheme version 9.0.1
2011-09-19

by Matt Birkholz

This manual documents FFI 0.2.

Copyright © 1986, 1987, 1988, 1989, 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014 Massachusetts Institute of Technology

> Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover Texts and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License."

1 Introduction

This FFI provides Scheme syntax for calling C functions and accessing C data. The functions and data structures are declared in a case sensitive `.cdecl` file, which is used by a shim generator to produce callout and callback trampoline functions. The trampolines are compiled and linked to the C toolkit, producing a shared object that Scheme can dynamically load.

Synopsis

Examples of the new syntax:

```
(C-include "prhello")

(malloc (C-sizeof "GdkEvent"))
⇒ #[alien 42 0x081afc60]

(C->= #@42 "GdkEvent any type" 14)

(C-> #@42 "GdkEvent any type")
⇒ 14

(C-enum "GdkEventType" 14)
⇒ |GDK_MAP|

(C-enum "GDK_MAP")
⇒ 14

(C-sizeof "GdkColor")
⇒ 12

(C-offset "GdkColor blue")
⇒ 8

(C-array-loc #@43 "GdkColor" (C-enum "GTK_STATE_NORMAL"))
⇒ #[alien 44 0x081afc60] ; New alien.

(C-array-loc! #@43 "GdkColor" (C-enum "GTK_STATE_PRELIGHT"))
⇒ #[alien 43 0x081afc78] ; Modified alien.

(C-call "gtk_window_new" retval args ...)
⇒ #!unspecific

(C-callback "delete_event")
⇒ #[alien-function 44 Scm_delete_event]

(C-callback (lambda (window event) ...))
⇒ 13                        ; A fixnum registration ID.
```

Summary

A Scheme-like declaration of a toolkit's C functions, constants, and data types is given in a case sensitive `.cdecl` file. The C declarations look like this:

```
(extern (* GtkWidget)
        gtk_window_new
        (type GtkWindowType))

(typedef GtkWindowType
         (enum
          (GTK_WINDOW_TOPLEVEL)
          (GTK_WINDOW_POPUP)))
```

The `c-generate` procedure reads these declarations and writes three files: *`library`*`-types.bin` (a fasdump of the parsed declarations), *`library`*`-const.c` (a C program that prints C constants and struct offsets), and *`library`*`-shim.c` (trampoline functions adapting Scheme procedure application to C function call). The `-const.c` program generates a `-const.scm` file, which can be syntaxed to produce a `-const.bin` file.

```
(load-option 'FFI)
(c-generate "prhello" "#include <gtk/gtk.h>")
```

The `-types.bin` and `-const.bin` files together provide the information needed to expand `C-...` syntax, and are only needed at syntax time. The compiled `-shim.so` file is used at run time, dynamically loaded into the Scheme machine. Chapter 6 [Compiling and Linking], page 9, which describes these files in more detail, and shows how they might be built and installed.

```
(C-include "prhello")
```

The `C-include` syntax loads the `-types.bin` and `-const.bin` files *at syntax time*. It should appear at the top level of any file containing `C-...` syntax, or be evaluated in the syntax environment of such code.

The `C-call` syntax arranges to invoke a callout trampoline. Arguments to the trampoline can be integers, floats, strings or aliens (non-heap pointers, to C data structures, see Chapter 3 [Alien Data], page 6).

```
(let ((alien (make-alien '|GtkWidget|)))
  (C-call "gtk_window_new" alien type)
  (if (alien-null? alien) (error "could not open new window"))
  alien)
```

The `C-callback` syntax is used when registering a Scheme callback trampoline. The two forms of the syntax provide two arguments for the registration function: the callback trampoline's address, and a "user data" argument. When the toolkit calls the trampoline, it must provide the fixnum-sized user data as an argument.

```
(C-call "g_signal_connect" window "delete_event"
        (C-callback "delete_event")     ; e.g. &Scm_delete_event
        (C-callback                     ; e.g. 314
          (lambda (window event)
            (C-call "gtk_widget_destroy" window)
            0)))
```

The first use of `C-callback` (above) expands into a callback trampoline address — an alien function. The second use evaluates to a fixnum, which is associated with the given Scheme procedure.

The `C->` and `C->=` syntaxes peek and poke values into alien data structures. They take an alien and a constant string specifying the alien data type and the member to be accessed (if any).

```
(C-> alien "GdkRectangle y")
↦
(#[primitive c-peek-int] alien 4)

(C->= alien "GdkRectangle width" 0)
↦
(#[primitive c-poke-int] alien 8 0)
(C-> alien "GdkEvent any type")
↦
(#[primitive c-peek-int] alien 0)
(C-> alien "gfloat")
↦
(#[primitive c-peek-float] alien 0)
```

A three argument form of the syntax provides an alien to receive a peeked pointer. This avoids consing a new alien.

```
(C-> alien "GtkWidget style" alien)
```

The above syntax is understood to say "The data at this `alien` address is a GtkWidget. Load its `style` member (an alien address), into `alien` (clobbering `alien`'s old address)."

The `C-enum`, `C-sizeof` and `C-offset` syntaxes all transform into integer constants. The last two transform into a padded byte size and a byte offset respectively.

```
(C-enum "GTK_WINDOW_POPUP")
↦
1

(C-sizeof "GdkColor")
↦
12

(C-offset "GdkColor blue")
↦
8
```

The two element form of the `C-enum` syntax can be used to find the name of a constant given its runtime value. It expects the name of an enum type in a constant string. If the runtime (second) argument is not one of the constants declared by that type, the returned value is `#f`.

```
(C-enum "GdkEventType" (C-> #@42 "GdkEvent any type"))
⇒ |GDK_MAP|
```

The `c-array-loc` and `c-array-loc!` syntaxes compute the locations of C array elements. They can be used to advance a scan pointer or locate an element by its index. The examples in the synopsis might expand as shown here.

```
(C-array-loc #@43 "GdkColor" (C-enum "GTK_STATE_NORMAL"))
↦
(alien-byte-increment #@43 (* (C-sizeof "GdkColor")
                              (C-enum "GTK_STATE_NORMAL")))
↦
(alien-byte-increment #@43 (* 12 0))
⇒ #@44
```

```
(C-array-loc! #@43 "GdkColor" (C-enum "GTK_STATE_PRELIGHT"))
↦
(alien-byte-increment! #@43 (* (C-sizeof "GdkColor")
                               (C-enum "GTK_STATE_PRELIGHT")))
↦
(alien-byte-increment! #@43 (* 12 2))
⇒ #@43
```

A simple scan of characters in the wide string `alien` might look like this.

```
(let ((len (C-> alien "toolkit_string_type int_member"))
      (scan (C-> alien "toolkit_string_type array_member")))
  (let loop ((n 0))
    (if (< n len)
        (let ((wchar (C-> scan "wchar")))
          (process wchar)
          (C-array-loc! scan "wchar" 1)
          (loop (1+ n))))))
```

That is a quick look at the facilities. The next section describes the C declaration language, and the following sections examine the FFI's syntax and runtime facilities in detail. Final sections provide an example program and show how its dynamically loaded shim is built.

2 C Declarations

A shim between Scheme and a C toolkit is specified by a case sensitive `.cdecl` file containing Scheme-like declarations of all relevant toolkit types, constants, and functions. Callback functions to be passed to the toolkit are also specified here.

Each top-level form in the C declaration file must look like one of these:

```
(include "filename")
(typedef Name any)
(struct Name (Member type) ...)
(union Name (Member type) ...)
(enum Name (Member) ...)
(extern function-type Name (param1 arg-type) ...)
(callback callback-type Name (param1 callback-arg-type) ...)
```

The `include` expression includes another `.cdecl` file in the current `.cdecl` file. The string argument is interpreted relative to the current file's directory.

any can be a *type* or the word `void`.

arg-type can be any *type* *except* anonymous structs and unions.

function-type can be any *arg-type* or `void`.

callback-arg-type can be any *type* *except* struct and union types.

callback-type can be any *callback-arg-type* or `void`.

type can look like any of these:

```
Name
basics
(* any)
(enum Name)
(enum Name (Member) ...)
(struct Name)
(struct Name (Member type) ...)
(union Name)
(union Name (Member type) ...)
```

Name should be defined via a `typedef` form somewhere in the (included) file(s). It does not have to be defined before it is referenced. It does not have to be defined *at all* if it is only the target of a pointer type.

basics can be any of the words: `char`, `uchar`, `short`, `ushort`, `int`, `uint`, `long`, `ulong`, `float`, or `double` (all lowercase).

While the informal grammar above allows anonymous structs to be member types, they are useless outside a named type declaration. The peek and poke (`C->` and `C->=`) syntaxes require a type name (e.g. `"GdkEventAny"` or `"struct _GdkEventAny"`) before any member names.

```
(C-include "prhello")
```

The `C-include` syntax takes a library name and loads the corresponding `-types` and `-const` files at syntax time. This makes the C types and constants available to the other `C-...` syntax expanders. The form binds `c-includes` in the syntax environment *unless* it is already defined there. Thus a `(C-include "library")` form can be placed at the top of every file with `C-...` syntax, *or* loaded into the syntax-time environment of those files.

3 Alien Data

A C data structure is represented by an alien containing the data structure's memory address. "Peek" primitives are available to read pointers and the basic C types (e.g. ints, floats) at small (fixnum) offsets from an alien's address. They return to Scheme an alien address, integer or flonum as appropriate. "Poke" primitives do the reverse, storing pointers, integers or floats at fixnum offsets from alien addresses. Other procedures on aliens are `alien?`, `alien-null?`, `alien-null!`, `copy-alien`, `alien=?`, `alien-byte-increment`, and `c-peek-cstring`. Refer to `ffi.pkg` in The Source for a complete list.

The `C->` and `C->=` syntaxes apply the peek and poke primitives to constant offsets. They expect their first argument subform to be a constant string — space-separated words naming a C type and any member to be accessed. A member within a struct or union member is specified by appending its name. For example `"struct _GdkEvent any window"` would specify a peek at the `window` member of the `any` member of the `struct _GdkEvent` data at some alien address. Note that the final member's type must be a basic C type, pointer type, or enum type. Otherwise, an error is signaled at syntax time.

```
(C-> alien "struct _GdkEvent any window" window-alien)
↦
(#[primitive c-peek-pointer] alien 0 window-alien)
⇒ #[alien 44 (* GdkWindow) 0x081afc60]
```

Note that in the example above, the final member has a pointer type. In this case an extra alien argument can be provided to receive the peeked pointer. Otherwise a new alien is created and returned.

Malloc

The `malloc` procedure returns an alien that will automatically free the malloced memory when it is garbage collected. It can also be explicitly freed with the `free` procedure. The alien address can be incremented to scan the malloced memory, then freed (without returning it to the original, malloced address). A band restore marks all malloced aliens as though they have been freed.

```
(free (malloc '|GdkRectangle|))
```

4 Alien Functions

The `C-call` syntax produces code that applies `call-alien` to an alien function structure — a cache for the callout trampoline's entry address.

```
(C-call "gtk_button_new" (make-alien '(* |GtkWidget|)))
↦
(call-alien '#[alien-function gtk_button_new] (make-alien ...))
```

The alien function contains all the information needed to load the callout trampoline on demand (i.e. its name and library). Once the alien function has cached the entry address, `call-alien` can invoke the trampoline (via `#[primitive c-call]`). The trampoline gets its arguments off the Scheme stack, converts them to C values, calls the C function, conses a result, and returns it to Scheme.

A function returning a pointer type is treated specially. Its trampoline expects an extra (first) argument. If the argument is `#f`, the return value is ignored. If the argument is an alien, the function's return value clobbers the alien's address. This makes it easy to grab pointers to toolkit resources without dropping them, and to avoid unnecessary consing of aliens.

A function returning a struct or union type is treated similarly. Its trampoline expects an extra (first) argument. If the argument is `#f`, the return value is ignored. If the argument is an alien, the returned struct or union is copied to that address.

Struct and union type parameters of a function are treated similarly. The function's trampoline expects an alien argument for each such parameter and copies the struct or union from the argument address into a local variable. Callbacks currently cannot receive struct or union type arguments, though they *can* receive pointer type arguments (consing an alien for each).

The `alien-function` structures are fasdumpable. The caching mechanism invalidates the cache when a band is restored, or a fasdumped object is fasloaded. The alien function will lookup the trampoline entry point again on demand.

5 Callbacks

A callback declaration must include a parameter named "ID". The ID argument will be used to find the Scheme callback procedure. It must be the same "user data" value provided to the toolkit when the callback was registered. For example, a callback trampoline named `Scm_delete_event` might be declared like this:

```
(callback gint
          delete_event
          (window (* GtkWidget))
          (event (* GdkEventAny))
          (ID gpointer))
```

The callback might be registered with the toolkit like this:

```
(C-call "g_signal_connect" window "delete_event"
        (C-callback "delete_event")     ; e.g. &Scm_delete_event
        (C-callback                     ; e.g. 314
          (lambda (window event)
            (C-call "gtk_widget_destroy" window)
            0)))
```

The toolkit's registration function, `g_signal_connect`, would be declared like this:

```
(extern void
        g_signal_connect
        (object (* GtkObject))
        (name (* gchar))
        (CALLBACK GtkSignalFunc)
        (ID gpointer))
```

This function should have parameters named `CALLBACK` and `ID`. The callout trampoline will convert the callback argument from a Scheme alien function to an entry address. The `ID` argument will be converted to a C integer and then cast to its declared type (in this example, `gpointer`).

Note that the registered callback procedures are effectively pinned. They cannot be garbage collected. They are "on call" to handle callbacks from the toolkit until they are explicitly de-registered. A band restore automatically de-registers all callbacks.

The callback procedures are executed like an interrupt handler. They actually interrupt the thread executing the most recent callout, e.g. to `gtk_main`. The thread runs with thread switching disabled for the duration of the callback, and can callout to the toolkit, which can callback again. The (nested) callbacks and nested callouts all run in the same thread, and so will return in LIFO order as expected by the toolkit. Note that the runtime system will not balk at a callback procedure that calls `yield-thread`, waits for I/O, sleeps, or otherwise causes a thread switch. Presumably such a procedure has some other way of enforcing the LIFO ordering.

The `outf-error` procedure is provided for debugging purposes. It writes one or more argument strings (and `write`s any non-strings) to the Unix "stderr" channel, atomically, via a machine primitive, bypassing the runtime's I/O buffering and thread switching. Thus trace messages from multiple threads will appear on stderr intact and uninterrupted.

6 Compiling and Linking

The `c-generate` procedure takes a library name and an optional preamble. It reads the *`library`*`.cdecl` file and writes two `.c` files. The preamble is included at the top of both. It typically contains `#include` C pre-processor directives required by the C library, but could include additional shim code. Here is a short script that generates a shim for the example "Hello, World!" program.

```
(load-option 'FFI)
(c-generate "prhello" "#include <gtk/gtk.h>")
```

This script will produce three files:

`prhello-shim.c`
: This file contains the trampoline functions — one for each declared C extern or callback. It includes the `mit-scheme.h` header file, found in the `AUXDIR` directory — e.g. `/usr/local/lib/mit-scheme/`.

`prhello-const.c`
: This file contains a C program that creates `prhello-const.scm`. It is compiled and linked as normal for programs using the toolkit, and does not depend on the Scheme machine. It does not actually call any toolkit functions. It just collects information from the compiler about the declared C types and constants.

`prhello-types.bin`
: This file is a fasdumped `c-includes` structure containing all of the types, constants and functions declared in the `.cdecl` file.

The following Makefile rules describe the process of building and installing a shim for the example "Hello, World!" program.

```
install:  build
        echo '(install-shim "$(DESTDIR)" "prhello")' \
        | mit-scheme --batch-mode

clean:
        rm prhello-const* prhello-types* prhello-shim*

build:  prhello-shim.so prhello-types.bin prhello-const.bin

prhello-shim.so:  prhello-shim.o
        echo "(link-shim)" \
        | mit-scheme --batch-mode -- -o $@ $^ `pkg-config --libs gtk+-2.0`

prhello-shim.o:  prhello-shim.c
        echo '(compile-shim)' \
        | mit-scheme --batch-mode -- `pkg-config --cflags gtk+-2.0` -c $<

prhello-shim.c prhello-const.c prhello-types.bin:  prhello.cdecl
        echo '(generate-shim "prhello" "#include <gtk/gtk.h>")' \
        | mit-scheme --batch-mode
```

```
prhello-const.bin:  prhello-const.scm
        echo '(sf "prhello-const")' | mit-scheme --batch-mode

prhello-const.scm:  prhello-const
        ./prhello-const

prhello-const:  prhello-const.o
        $(CC) -o $@ $^ $(LDFLAGS) `pkg-config --libs gtk+-2.0`

prhello-const.o:  prhello-const.c
        $(CC) `pkg-config --cflags gtk+-2.0` $(CFLAGS) -o $@ -c $<
```

7 Hello World

This chapter includes the C declarations and Scheme code required to implement Havoc Pennington's Hello World example from GGAD. For an extra, Schemely treat, its `delete_event` callback is a Scheme procedure closed over a binding of `counter` that is used to implement some impertinent behavior.

```
#| -*-Scheme-*-

This is Havoc Pennington's Hello World example from GGAD, in the raw
FFI.  Note that no arrangements have been made to de-register the
callbacks.  |#

(declare (usual-integrations))

(C-include "prhello")

(define (hello)
  (C-call "gtk_init" 0 null-alien)
  (let ((window (let ((alien (make-alien '|GtkWidget|)))
                  (C-call "gtk_window_new" alien
                          (C-enum "GTK_WINDOW_TOPLEVEL"))
                  (if (alien-null?  alien) (error "Could not create window."))
                  alien))
        (button (let ((alien (make-alien '|GtkWidget|)))
                  (C-call "gtk_button_new" alien)
                  (if (alien-null?  alien) (error "Could not create button."))
                  alien))
        (label (let ((alien (make-alien '|GtkWidget|)))
                 (C-call "gtk_label_new" alien "Hello, World!")
                 (if (alien-null?  alien) (error "Could not create label."))
                 alien)))
    (C-call "gtk_container_add" button label)
    (C-call "gtk_container_add" window button)
    (C-call "gtk_window_set_title" window "Hello")
    (C-call "gtk_container_set_border_width" button 10)
    (let ((counter 0))
      (C-call "g_signal_connect" window "delete_event"
              (C-callback "delete_event")       ;trampoline
              (C-callback                       ;callback ID
               (lambda (w e)
                 (outf-error ";Delete me "(- 2 counter)" times.\n")
                 (set!  counter (1+ counter))
                 ;; Three or more is the charm.
                 (if (> counter 2)
                     (begin
                       (C-call "gtk_main_quit")
                       0)
```

```
                  1))))
      (C-call "g_signal_connect" button "clicked"
              (C-callback "clicked")    ;trampoline
              (C-callback                       ;callback ID
               (lambda (w)
                 (let ((gstring (make-alien '(* |gchar|))))
                   (C-call "gtk_label_get_text" gstring label)
                   (let ((text (c-peek-cstring gstring)))
                     (C-call "gtk_label_set_text" label
                             (list->string (reverse!  (string->list text))))))
                 unspecific))))
    (C-call "gtk_widget_show_all" window)
    (C-call "gtk_main")
    window))
```

Here are the C declarations.

```
#| -*-Scheme-*-

C declarations for prhello.scm.  |#

(typedef gint int)
(typedef guint uint)
(typedef gchar char)
(typedef gboolean gint)
(typedef gpointer (* mumble))

(extern void
        gtk_init
        (argc (* int))
        (argv (* (* (* char)))))

(extern (* GtkWidget)
        gtk_window_new
        (type GtkWindowType))

(typedef GtkWindowType
         (enum
          (GTK_WINDOW_TOPLEVEL)
          (GTK_WINDOW_POPUP)))

(extern (* GtkWidget)
        gtk_button_new)

(extern (* GtkWidget)
        gtk_label_new
        (str (* (const char))))
```

```
(extern void
        gtk_container_add
        (container (* GtkContainer))
        (widget    (* GtkWidget)))

(extern void
        gtk_window_set_title
        (window (* GtkWindow))
        (title  (* (const gchar))))

(extern void
        gtk_container_set_border_width
        (container (* GtkContainer))
        (border_width guint))

(extern void
        gtk_widget_show_all
        (widget (* GtkWidget)))

(extern void
        g_signal_connect
        (object (* GtkObject))
        (name (* gchar))
        (CALLBACK GtkSignalFunc)
        (ID gpointer))

(typedef GtkSignalFunc (* mumble))

(callback gboolean
          delete_event
          (window (* GtkWidget))
          (event (* GdkEventAny))
          (ID gpointer))

(callback void
          clicked
          (widget (* GtkWidget))
          (ID gpointer))

(extern void
        gtk_widget_destroy
        (widget (* GtkWidget)))

(extern (* (const gchar))
        gtk_label_get_text
        (label (* GtkLabel)))
```

```
(extern void
        gtk_label_set_text
        (label (* GtkLabel))
        (str (* (const char))))

(extern void gtk_main)
(extern void gtk_main_quit)
```

Appendix A GNU Free Documentation License

Version 1.2, November 2002

Copyright © 2000,2001,2002 Free Software Foundation, Inc.
51 Franklin St, Fifth Floor, Boston, MA 02110-1301, USA

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful 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 non-commercially. 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, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

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. (Thus, 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. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

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 Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document 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, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. 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, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include 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, PostScript or PDF 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.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

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 (or copies in media that commonly have printed covers) 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 computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. 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 fewer than five), unless they release you from this requirement.

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", Preserve 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. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, 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 to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

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, and that you preserve all their Warranty Disclaimers.

   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, is called

an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included an aggregate, this License does not apply to the other works in the aggregate which 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 half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket 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, and all the license notices in the Document, and any Warrany Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

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.

A.1 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 1.2
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

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.