

Using Objective-C with GNUstep

An overview of the Objective-C language and its use with the GNUstep development framework.

Christopher Armstrong

© 2005 Christopher Armstrong

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2, as 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".

Table of Contents

1	Introduction	2
2	Declaring and using object instances	3
2.1	Declaring a reference to an object.....	3
2.2	Sending a message vs calling a method.....	3
2.3	Syntax	3
2.4	Creating objects.....	4
2.5	Using objects	4
3	Declaring and implementing classes	6
3.1	Interfaces	6
3.2	Declaring instance variables	6
3.3	Method definition	7
4	Memory Management	9
Appendix A	Generic makefile	10
A.1	Building a MakeFile.....	10
A.2	Compilation	10
Appendix B	GNU Free Documentation License.....	11
B.1	ADDENDUM: How to use this License for your documents.....	17

1 Introduction

Objective-C is powerful, dynamic and compiled object-oriented language. GNUstep uses Objective-C to provide a framework for developing applications more quickly than other environments with less code. It takes advantage of the OpenStep specification to provide this flexibility.

The language is a "strict superset" of C. This means you can take advantage of any existing C code or libraries without changing them, and make use of Objective-C's flexible and powerful object-orientation to design and build your software. Objective-C derives some of its syntax from Smalltalk, and hence much of its semantics.

This manual assumes familiarity with both the C language and object-oriented syntax and concepts. If you have ever programmed in C++ or Java, then Objective-C is relatively easy to learn. Like Java, it uses a single-inheritance model and allows the use of protocols. Some language features include:

- Single-inheritance structure
- Protocols/Interfaces
- Dynamic object typing with optional static typing
- Splitting class declarations and implementations over multiple source files or even multiple libraries.
- Full runtime reflection and introspection

We will explain these in some detail in this manual, including how to leverage these features with GNUstep.

2 Declaring and using object instances

Object instances in Objective-C are declared similarly to primitive types in C. They are always pointers, as Objective-C creates and destroys objects dynamically. We describe how to declare, instantiate and use object instances below.

2.1 Declaring a reference to an object

In Objective-C, all classes should take a base class. GNUstep uses NSObject as it's base class, as it provides some useful methods for runtime introspection and memory management.

We declare an instance of an object as follows:

```
id objectName;
```

And with an example:

```
id myObject;
```

`id` is a special object type that refers to any class. It also prevents the compiler from checking whether your object will respond to the messages you later send it. It is strongly recommended that you statically type your instance declarations more strongly, in which case you use the class name in place of "id". For example:

```
NSArray* myArray;
```

Note the lowercase "m" is convention, and is not required. This declares an instance of a NSArray object. It also makes the compiler check that "myArray" can respond to any messages sent to it. The compiler will throw a warning if you try to send it a message it doesn't understand. Also take note that the object is declared as a pointer to an NSArray, as opposed to being declared statically. In Objective-C, all objects are created and destroyed on the heap, and cannot be used in a static context (i.e. on the stack).

2.2 Sending a message vs calling a method

It is important that we first make a small distinction between Objective-C and other object-oriented languages. In languages like C++, we use the term "call a method" to describe how methods are invoked, instead of "sending a message", although what is said is usually a matter of taste. With Objective-C, we distinguish between the two, so that we use the concept of "sending a message" to reinforce the dynamic nature of Objective-C objects.

Objective-C does what is known as *late-binding* or *runtime-binding*, where it looks up the name of the method when the programme is running, as opposed to when the programme is compiled (by the way, this introduces little speed penalty, as the process is heavily optimised). This means it is possible to extend and subclass by loading external modules, but it reinforces the idea that a message is sent to the object, as opposed to calling a method, which tends to imply more static typing of object instances. This can be pedantic at times, but it does help to explain a few things.

For example, it is possible to send messages to objects that don't respond to them; this should by default induce a runtime error, but it also allows you to use proxy objects or handle this situation more elegantly. We will come back to this later.

2.3 Syntax

The syntax for sending a simple message in Objective-C is as follows:

```
[object message]
```

This may seem just a bit weird if you come from a C++/Java/C# background, but it often helps to improve the readability of code. The above example describes a message with no parameters, just the object and the message (or method).

A more complex message would take the form:

```
[object message: parameter1
 parameter2name: parameter2
 parameter3name: parameter3];
```

This describes a message that takes three parameters: `parameter1`, `parameter2` and `parameter3`. `parameter2name` and `parameter3name` are examples of the naming of parameters that Objective-C allows. Note that the message call has been split over three lines to aid readability; it is optional how you arrange the whitespace for a message call. You need to insert the parameter names when sending a message in Objective-C, as they form part of the message name. In the case, the "whole" message name (known within the language as a "selector") is `message:parameter2name:parameter3name:`.

This can be used as a string to dynamically send a message (will we cover this later as well). Although they are optional for the designer, you are highly encouraged to use these when designing your methods and classes.

Now here's a real example of a message. In the following code, we take an existing `NSArray` object and extract the item at the fifth index (item no. 4, as we count from zero for array indices like in C).

```
NSArray* myArray;
/* Initializer code in here*/
id myObject = [myArray objectAtIndex: 4];
```

Note that the space between the colon and the "4" is optional (this is somewhat flexible, like the space between commas when calling C functions).

2.4 Creating objects

In this section, we will show you how to declare and instantiate an object. Take a look at the following code:

```
NSNumber* aNumber = [NSNumber alloc];
aNumber = [aNumber initWithInt: 6];
NSLog(Öreated a number: %; aNumber);
```

Instantiating objects requires two calls: one to `alloc` and one to an `init` method. `alloc` creates the object and allocates free space for it. It is sent to the class. `init` calls the constructor for the object; it is sent to the newly allocated object returned from `alloc`. Note that a class may have multiple constructors, depending on what input parameters they take. All classes support the `init` method, but sometimes it's more appropriate to call another constructor (always check the class documentation first).

You can also nest method calls, and it's typical to nest `alloc/init`, such as follows:

```
NSNumber * aNumber = [[NSNumber alloc] init];
```

2.5 Using objects

Taking what we have learned above, we will now describe a small program that creates and declares objects, and sends messages to them. If you want to compile what's below, you will need `GNUstep-base` installed, as well as a `makefile` to compile it with. Please refer to the `GNUstep-make` manual for details on constructing a `makefile`. Alternatively, a generic one is provided in this manual (see [\[Generic makefile\]](#), page 9).

This example manipulates the `NSMutableString` and `NSDate` classes in the Foundation library (`gnustep-base`). Although this is certainly not typical usage, it provides some idea of the type of message passing that goes on with `GNUstep` programs.

```
#include <Foundation/Foundation.h>

int main(int argc, char** argv)
{
    CREATE_AUTORELEASE_POOL(pool);
    NSDate* currentDate = [NSDate date];
    NSLog(@"The current date is described as: %@",
          [currentDate description] );

    NSMutableString* myString = [[NSMutableString alloc]
                                  initWithCapacity: 255] ;
    [myString setString: [currentDate description]] ;
    [myString appendString: @" (Something to append with) " ] ;

    NSLog(myString);
    RELEASE(pool);
    return 0;
}
```

For starters, note that static strings that are flattened automatically to NSString types are declared by placing an "at" symbol before the double quotation marks. Also note that the CREATE_AUTORELEASE_POOL line is used to handle garbage collection of the objects we create during the lifetime of the program (see [\[Memory Management\]](#), page 8)

3 Declaring and implementing classes

3.1 Interfaces

Objective-C splits class definition into two separate parts: an *interface* and an *implementation*. The *interface* declares the instance variables and class and instance methods that the class will contain, as well as what class it inherits and the formal protocols it implements. The *implementation* contains the class and instance method code.

An interface is declared using the `@interface` token. A simple interface declaration takes the form:

```
@interface class_name : superclass_name
{
    /* instance variable declarations */
}
/* class and instance method prototypes */
@end
```

where *class_name* refers to the name of the class you're declaring, and *superclass_name* is the class that you're inheriting. Note that the above is not syntactically complete, for reasons of simplicity.

Instance variable declarations take place inside the braces (as above), and instance and class methods are declared below the braces, before the `@end` token that indicates the end of an interface declaration. The instance variable block is optional.

An example of a class declaration is:

```
@interface Rectangle : NSObject
{
    @private
    int width;
    int height;
    NSView drawView;
}
- (id) init;
- (id) initWithWidth:(int)width height:(int) height;
- (int) width;
- (int) height;
+ (Rectangle*) rectangleWithSize:(NSSize)size;
@end
```

Don't try to understand the method and variable declarations yet! We are just trying to give you a feel for a real interface declaration. The syntax and meaning of those will be explained in the sections below. `NSView` is an `AppKit` class, by the way, used for displaying GUI objects.

The above example declares a class called `Rectangle`, inheriting from `NSObject`. It has three instance variable declarations: `width`, `height` and `drawView`. As you can see, instance variable declarations take the same form as in C structs (in fact, objects are implemented internally using C structs, which allows for surprising things to be done).

The class then defines four instance methods and one class method. They are distinguished by the `+` or `-` at the beginning of the method prototype.

3.2 Declaring instance variables

Instance variables are declared within the braces of the *interface* declaration. They take the form:

```
Type variableName;
```

Note that it is convention to use a lowercase for the first letter of the variable name. Don't forget the semicolon as well. *Type* refers to the instance variable's type, whether it be a primitive C type (such as *int*, *float*, *char**, a struct, etc) or another Objective-C object.

Instance variables may also take privacy modifiers, as so to enforce encapsulation of data. These are:

@private The variable(s) will be only accessible by instance methods of that class, not including subclasses.

@protected The variable(s) will be only accessible by instance methods of that classes, and of it's subclasses.

@public The variable(s) can be accessed anywhere. This means by instance methods of your class, subclasses, other classes and plain C functions.

These modifiers are placed on a line by themselves above the instance variables they will affect. For example:

```
@interface Rectangle
{
@private
    int width;
    int height;
@public
    NSColor color;
@protected
    NSView drawView;
}
...
@end
```

In this example, `drawView` is protected, `color` is public, and `width` and `height` are both private.

3.3 Method definition

In the *interface* you may have either be class or instance methods. All methods are considered public (there is no private or protected methods). Instance method definition takes the form:

```
- (return_type) method_name:(parameter1_type) parameter1_variable
  parameter2_name:(parameter2_type) parameter2_variable ... ;
```

In the first parentheses is the *return type*, which can be void if necessary. The return type and it's brackets is actually optional: the compiler will default your method to returning *id* if a return type isn't specified.

Next follows the *method name*. It follows similar rules to the naming of methods in C. Parameters then follow, with the type placed in parentheses after each colon followed by the variable name. Each parameter may optionally be named (except the first), with the name placed before the colon. If not named, the colons are still necessary to distinguish the parameter name from the parameter type and variable.

Class methods follow the same rules as above, but instead of the leading minus ('-') sign, you replace it with a plus ('+').

A few example method declarations may include:

```
- (id) objectAtIndex:(int) index;  
- (void) invokeWithReceiver:(id) receiver delegate:(Delegate*) theDelegate;  
+ initialize;  
+ (NSArray*) arrayWithArray:(NSArray*) array;
```

4 Memory Management

Memory management in GNUstep is performed using what's known as *reference counting*. This is where a number is kept with each object, noting how many references to it are being maintained.

You call a method each time you want to maintain a reference to an object (*retain*) or let go of a reference to an object (*release*). GNUstep provides a few macros for this purpose, beyond the API's in NSObject, so that you can easily move your programme to a garbage collector (still being worked into GNUstep).

Appendix A Generic makefile

To compile some of the example programmes in this manual with GNUstep, here are some instructions for building a makefile that can compile the simple programmes in the file. You will need gnustep-make and gnustep-base configured and installed.

A.1 Building a MakeFile

Begin by creating a directory to store this programme in. Here we call it ‘TestProgram’. In this directory, create a blank text file called ‘GNUmakefile’ (the case is important). Place the following text in it:

```
include $(GNUSTEP_MAKEFILES)/common.make

TOOL_NAME = TestProgram

TestProgram_OBJC_FILES = main.m

include $(GNUSTEP_MAKEFILES)/tool.make
```

The first line includes a number of macros used to setup the program for compiling. `TOOL_NAME` declares the name of the program we will be creating. A *tool* is a type of program in GNUstep that is command line and links against the gnustep-base library (anything using *Foundation*).

The `OBJC_FILES` line is preceded by the name of the program, as declared in the `TOOL_NAME` line. A list of files to be compiled is placed after the equals sign, separated by a *space*.

Lastly, we include the makefile for building tools.

A.2 Compilation

First, load up a shell. If on a unix based system, this could be xterm, or if running on windows, a mingw or cygwin shell. Make sure GNUstep’s startup script is sourced (replace ‘`/usr/lib/GNUstep`’ with the directory containing your GNUstep installation):

```
. /usr/lib/GNUstep/System/Library/Makefiles/GNUstep.sh
```

and then change into your program’s directory and type `make` to compile:

```
cd TestProgram
make
```

If you’re successful, an executable will be generated in the ‘`shared_obj`’ subdirectory.

For more details on program compilation, see the gnustep-make manual.

Appendix B GNU Free Documentation License

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 noncommercially. Secondly, 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 in 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 Warranty 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.

B.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.