Brief History of Automake

David MacKenzie
Tom Tromey
Alexandre Duret-Lutz

This manual describes (part of) the history of GNU Automake, a program that creates GNU standards-compliant Makefiles from template files.

Table of Contents

1 Timeline

1994-09-19 First CVS commit.

If we can trust the CVS repository, David J. MacKenzie (djm) started working on Automake (or AutoMake, as it was spelt then) this Monday.

The first version of the `automake` script looks as follows.

```
#!/bin/sh

status=0

for makefile
do
  if test ! -f ${makefile}.am; then
    echo "automake: ${makefile}.am: No such honkin' file"
    status=1
    continue
  fi

  exec 4> ${makefile}.in

done
```

From this you can already see that Automake will be about reading '`*.am`' file and producing '`*.in`' files. You cannot see anything else, but if you also know that David is the one who created Autoconf two years before you can guess the rest.

Several commits follow, and by the end of the day Automake is reported to work for GNU fileutils and GNU m4.

The modus operandi is the one that is still used today: variable assignments in '`Makefile.am`' files trigger injections of precanned '`Makefile`' fragments into the generated '`Makefile.in`'. The use of '`Makefile`' fragments was inspired by the 4.4BSD `make` and include files, however Automake aims to be portable and to conform to the GNU standards for '`Makefile`' variables and targets.

At this point, the most recent release of Autoconf is version 1.11, and David is preparing to release Autoconf 2.0 in late October. As a matter of fact, he will barely touch Automake after September.

1994-11-05 David MacKenzie's last commit.

At this point Automake is a 200 line portable shell script, plus 332 lines of '`Makefile`' fragments. In the '`README`', David states his ambivalence between "portable shell" and "more appropriate language":

> I wrote it keeping in mind the possibility of it becoming an Autoconf macro, so it would run at configure-time. That would slow configuration down a bit, but allow users to modify the Makefile.am without needing to fetch the AutoMake package. And, the Makefile.in files wouldn't need to be distributed. But all of AutoMake

> would. So I might reimplement AutoMake in Perl, m4, or some other more appropriate language.

Automake is described as "an experimental Makefile generator". There is no documentation. Adventurous users are referred to the examples and patches needed to use Automake with GNU m4 1.3, fileutils 3.9, time 1.6, and development versions of find and indent.

These examples seem to have been lost. However at the time of writing (10 years later in September, 2004) the FSF still distributes a package that uses this version of Automake: check out GNU termutils 2.0.

1995-11-12 Tom Tromey's first commit.

After one year of inactivity, Tom Tromey takes over the package. Tom was working on GNU cpio back then, and doing this just for fun, having trouble finding a project to contribute to. So while hacking he wanted to bring the '`Makefile.in`' up to GNU standards. This was hard, and one day he saw Automake on `ftp://alpha.gnu.org/`, grabbed it and tried it out.

Tom didn't talk to djm about it until later, just to make sure he didn't mind if he made a release. He did a bunch of early releases to the Gnits folks.

Gnits was (and still is) totally informal, just a few GNU friends who François Pinard knew, who were all interested in making a common infrastructure for GNU projects, and shared a similar outlook on how to do it. So they were able to make some progress. It came along with Autoconf and extensions thereof, and then Automake from David and Tom (who were both gnitsians). One of their ideas was to write a document paralleling the GNU standards, that was more strict in some ways and more detailed. They never finished the GNITS standards, but the ideas mostly made their way into Automake.

1995-11-23 Automake 0.20

Besides introducing automatic dependency tracking (see Chapter 2 [Dependency Tracking Evolution], page 13), this version also supplies a 9-page manual.

At this time `aclocal` and `AM_INIT_AUTOMAKE` did not exist, so many things had to be done by hand. For instance, here is what a configure.in (this is the former name of the '`configure.ac`' we use today) must contain in order to use Automake 0.20:

```
PACKAGE=cpio
VERSION=2.3.911
AC_DEFINE_UNQUOTED(PACKAGE, "$PACKAGE")
AC_DEFINE_UNQUOTED(VERSION, "$VERSION")
AC_SUBST(PACKAGE)
AC_SUBST(VERSION)
AC_ARG_PROGRAM
AC_PROG_INSTALL
```

(Today all of the above is achieved by `AC_INIT` and `AM_INIT_AUTOMAKE`.)

Here is how programs are specified in '`Makefile.am`':

```
PROGRAMS = hello
hello_SOURCES = hello.c
```

This looks pretty much like what we do today, except the `PROGRAMS` variable has no directory prefix specifying where '`hello`' should be installed: all programs are installed in '`$(bindir)`'. `LIBPROGRAMS` can be used to specify programs that must be built but not installed (it is called `noinst_PROGRAMS` nowadays).

Programs can be built conditionally using `AC_SUBST`itutions:

```
PROGRAMS = @progs@
AM_PROGRAMS = foo bar baz
```

(`AM_PROGRAMS` has since then been renamed to `EXTRA_PROGRAMS`.)

Similarly scripts, static libraries, and data can be built and installed using the `LIBRARIES`, `SCRIPTS`, and `DATA` variables. However `LIBRARIES` were treated a bit specially in that Automake did automatically supply the '`lib`' and '`.a`' prefixes. Therefore to build '`libcpio.a`', one had to write

```
LIBRARIES = cpio
cpio_SOURCES = ...
```

Extra files to distribute must be listed in `DIST_OTHER` (the ancestor of `EXTRA_DIST`). Also extra directories that are to be distributed should appear in `DIST_SUBDIRS`, but the manual describes this as a temporary ugly hack (today extra directories should also be listed in `EXTRA_DIST`, and `DIST_SUBDIRS` is used for another purpose, see Section "Conditional Subdirectories" in *GNU Automake*).

1995-11-26 Automake 0.21

In less time than it takes to cook a frozen pizza, Tom rewrites Automake using Perl. At this time Perl 5 is only one year old, and Perl 4.036 is in use at many sites. Supporting several Perl versions has been a source of problems through the whole history of Automake.

If you never used Perl 4, imagine Perl 5 without objects, without '`my`' variables (only dynamically scoped '`local`' variables), without function prototypes, with function calls that needs to be prefixed with '`&`', etc. Traces of this old style can still be found in today's `automake`.

1995-11-28 Automake 0.22

1995-11-29 Automake 0.23

Bug fixes.

1995-12-08 Automake 0.24

1995-12-10 Automake 0.25

Releases are raining. 0.24 introduces the uniform naming scheme we use today, i.e., `bin_PROGRAMS` instead of `PROGRAMS`, `noinst_LIBRARIES` instead of `LIBLIBRARIES`, etc. (However `EXTRA_PROGRAMS` does not exist yet, `AM_PROGRAMS` is still in use; and `TEXINFOS` and `MANS` still have no directory prefixes.) Adding support for prefixes like that was one of the major ideas in `automake`; it has lasted pretty well.

AutoMake is renamed to Automake (Tom seems to recall it was François Pinard's doing).

0.25 fixes a Perl 4 portability bug.

1995-12-18 Jim Meyering starts using Automake in GNU Textutils.
1995-12-31 François Pinard starts using Automake in GNU tar.
1996-01-03 Automake 0.26
1996-01-03 Automake 0.27

Of the many changes and suggestions sent by François Pinard and included in 0.26, perhaps the most important is the advice that to ease customization a user rule or variable definition should always override an Automake rule or definition.

Gordon Matzigkeit and Jim Meyering are two other early contributors that have been sending fixes.

0.27 fixes yet another Perl 4 portability bug.

1996-01-13 Automake 0.28

Automake starts scanning '`configure.in`' for `LIBOBJS` support. This is an important step because until this version Automake only knew about the '`Makefile.am`'s it processed. '`configure.in`' was Autoconf's world and the link between Autoconf and Automake had to be done by the '`Makefile.am`' author. For instance, if '`config.h`' was generated by '`configure`', it was the package maintainer's responsibility to define the `CONFIG_HEADER` variable in each '`Makefile.am`'.

Succeeding releases will rely more and more on scanning '`configure.in`' to better automate the Autoconf integration.

0.28 also introduces the `AUTOMAKE_OPTIONS` variable and the '`--gnu`' and '`--gnits`' options, the latter being stricter.

1996-02-07 Automake 0.29

Thanks to '`configure.in`' scanning, `CONFIG_HEADER` is gone, and rebuild rules for '`configure`'-generated file are automatically output.

`TEXINFOS` and `MANS` converted to the uniform naming scheme.

1996-02-24 Automake 0.30

The test suite is born. It contains 9 tests. From now on test cases will be added pretty regularly (see Chapter 3 [Releases], page 17), and this proved to be really helpful later on.

`EXTRA_PROGRAMS` finally replaces `AM_PROGRAMS`.

All the third-party Autoconf macros, written mostly by François Pinard (and later Jim Meyering), are distributed in Automake's hand-written '`aclocal.m4`' file. Package maintainers are expected to extract the necessary macros from this file. (In previous versions you had to copy and paste them from the manual...)

1996-03-11 Automake 0.31

The test suite in 0.30 was run via a long `check-local` rule. Upon Ulrich Drepper's suggestion, 0.31 makes it an Automake rule output whenever the `TESTS` variable is defined.

`DIST_OTHER` is renamed to `EXTRA_DIST`, and the `check_` prefix is introduced. The syntax is now the same as today.

1996-03-15 Gordon Matzigkeit starts writing libtool.

1996-04-27 Automake 0.32

`-hook` targets are introduced; an idea from Dieter Baron.

'`*.info`' files, which were output in the build directory are now built in the source directory, because they are distributed. It seems these files like to move back and forth as that will happen again in future versions.

1996-05-18 Automake 0.33

Gord Matzigkeit's main two contributions:

- very preliminary libtool support
- the distcheck rule

Although they were very basic at this point, these are probably among the top features for Automake today.

Jim Meyering also provides the infamous `jm_MAINTAINER_MODE`, since then renamed to `AM_MAINTAINER_MODE` and abandoned by its author (see Section "maintainer-mode" in *GNU Automake*).

1996-05-28 Automake 1.0

After only six months of heavy development, the `automake` script is 3134 lines long, plus 973 lines of '`Makefile`' fragments. The package has 30 pages of documentation, and 38 test cases. '`aclocal.m4`' contains 4 macros.

From now on and until version 1.4, new releases will occur at a rate of about one a year. 1.1 did not exist, actually 1.1b to 1.1p have been the name of beta releases for 1.2. This is the first time Automake uses suffix letters to designate beta releases, a habit that lasts.

1996-10-10 Kevin Dalley packages Automake 1.0 for Debian GNU/Linux.

1996-11-26 David J. MacKenzie releases Autoconf 2.12.

Between June and October, the Autoconf development is almost stalled. Roland McGrath has been working at the beginning of the year. David comes back in November to release 2.12, but he won't touch Autoconf anymore after this year, and Autoconf then really stagnates. The desolate Autoconf '`ChangeLog`' for 1997 lists only 7 commits.

1997-02-28 `automake@gnu.ai.mit.edu` list alive

The mailing list is announced as follows:

```
I've created the "automake" mailing list.  It is
"automake@gnu.ai.mit.edu".  Administrivia, as always, to
automake-request@gnu.ai.mit.edu.

The charter of this list is discussion of automake, autoconf, and
other configuration/portability tools (e.g., libtool).  It is expected
that discussion will range from pleas for help all the way up to
patches.

This list is archived on the FSF machines.  Offhand I don't know if
you can get the archive without an account there.

This list is open to anybody who wants to join.  Tell all your
friends!
```

```
-- Tom Tromey
```

Before that people were discussing Automake privately, on the Gnits mailing list (which is not public either), and less frequently on `gnu.misc.discuss`.

`gnu.ai.mit.edu` is now `gnu.org`, in case you never noticed. The archives of the early years of the `automake@gnu.org` list have been lost, so today it is almost impossible to find traces of discussions that occurred before 1999. This has been annoying more than once, as such discussions can be useful to understand the rationale behind a piece of uncommented code that was introduced back then.

1997-06-22 Automake 1.2

Automake developments continues, and more and more new Autoconf macros are required. Distributing them in '`aclocal.m4`' and requiring people to browse this file to extract the relevant macros becomes uncomfortable. Ideally, some of them should be contributed to Autoconf so that they can be used directly, however Autoconf is currently inactive. Automake 1.2 consequently introduces `aclocal` (`aclocal` was actually started on 1996-07-28), a tool that automatically constructs an '`aclocal.m4`' file from a repository of third-party macros. Because Autoconf has stalled, Automake also becomes a kind of repository for such third-party macros, even macros completely unrelated to Automake (for instance macros that fix broken Autoconf macros).

The 1.2 release contains 20 macros, including the `AM_INIT_AUTOMAKE` macro that simplifies the creation of '`configure.in`'.

Libtool is fully supported using `*_LTLIBRARIES`.

The missing script is introduced by François Pinard; it is meant to be a better solution than `AM_MAINTAINER_MODE` (see Section "maintainer-mode" in *GNU Automake*).

Conditionals support was implemented by Ian Lance Taylor. At the time, Tom and Ian were working on an internal project at Cygnus. They were using ILU, which is pretty similar to CORBA. They wanted to integrate ILU into their build, which was all '`configure`'-based, and Ian thought that adding conditionals to `automake` was simpler than doing all the work in '`configure`' (which was the standard at the time). So this was actually funded by Cygnus.

This very useful but tricky feature will take a lot of time to stabilize. (At the time this text is written, there are still primaries that have not been updated to support conditional definitions in Automake 1.9.)

The `automake` script has almost doubled: 6089 lines of Perl, plus 1294 lines of '`Makefile`' fragments.

1997-07-08 Gordon Matzigkeit releases Libtool 1.0.

1998-04-05 Automake 1.3

This is a small advance compared to 1.2. It adds support for assembly, and preliminary support for Java.

Perl 5.004_04 is out, but fixes to support Perl 4 are still regularly submitted whenever Automake breaks it.

1998-09-06 `sourceware.cygnus.com` is on-line.
Sourceware was setup by Jason Molenda to host open source projects.

1998-09-19 Automake CVS repository moved to `sourceware.cygnus.com`

1998-10-26 `sourceware.cygnus.com` announces it hosts Automake:
Automake is now hosted on `sourceware.cygnus.com`. It has a publicly accessible CVS repository. This CVS repository is a copy of the one Tom was using on his machine, which in turn is based on a copy of the CVS repository of David MacKenzie. This is why we still have to full source history. (Automake was on Sourceware until 2007-10-29, when it moved to a git repository on `savannah.gnu.org`, but the Sourceware host had been renamed to `sources.redhat.com`.)

The oldest file in the administrative directory of the CVS repository that was created on Sourceware is dated 1998-09-19, while the announcement that `automake` and `autoconf` had joined `sourceware` was made on 1998-10-26. They were among the first projects to be hosted there.

The heedful reader will have noticed Automake was exactly 4 years old on 1998-09-19.

1999-01-05 Ben Elliston releases Autoconf 2.13.

1999-01-14 Automake 1.4
This release adds support for Fortran 77 and for the `include` statement. Also, '`+=`' assignments are introduced, but it is still quite easy to fool Automake when mixing this with conditionals.

These two releases, Automake 1.4 and Autoconf 2.13 make a duo that will be used together for years.

`automake` is 7228 lines, plus 1591 lines of Makefile fragment, 20 macros (some 1.3 macros were finally contributed back to Autoconf), 197 test cases, and 51 pages of documentation.

1999-03-27 The `user-dep-branch` is created on the CVS repository.
This implements a new dependency tracking schemed that should be able to handle automatic dependency tracking using any compiler (not just gcc) and any make (not just GNU `make`). In addition, the new scheme should be more reliable than the old one, as dependencies are generated on the end user's machine. Alexandre Oliva creates depcomp for this purpose.

See Chapter 2 [Dependency Tracking Evolution], page 13, for more details about the evolution of automatic dependency tracking in Automake.

1999-11-21 The `user-dep-branch` is merged into the main trunk.
This was a huge problem since we also had patches going in on the trunk. The merge took a long time and was very painful.

2000-05-10
Since September 1999 and until 2003, Akim Demaille will be zealously revamping Autoconf.

> I think the next release should be called "3.0".
> Let's face it: you've basically rewritten autoconf.

> Every weekend there are 30 new patches.
> I don't see how we could call this "2.15" with a straight face.
> – Tom Tromey on autoconf@gnu.org

Actually Akim works like a submarine: he will pile up patches while he works off-line during the weekend, and flush them in batch when he resurfaces on Monday.

2001-01-24

On this Wednesday, Autoconf 2.49c, the last beta before Autoconf 2.50 is out, and Akim has to find something to do during his week-end :)

2001-01-28

Akim sends a batch of 14 patches to automake@gnu.org.

> Aiieeee! I was dreading the day that the Demaillator turned his sights on automake... and now it has arrived! – Tom Tromey

It's only the beginning: in two months he will send 192 patches. Then he would slow down so Tom can catch up and review all this. Initially Tom actually read all these patches, then he probably trustingly answered OK to most of them, and finally gave up and let Akim apply whatever he wanted. There was no way to keep up with that patch rate.

> Anyway the patch below won't apply since it predates Akim's sourcequake; I have yet to figure where the relevant passage has been moved :) – Alexandre Duret-Lutz

All these patches were sent to and discussed on automake@gnu.org, so subscribed users were literally drowning in technical mails. Eventually, the automake-patches@gnu.org mailing list was created in May.

Year after year, Automake had drifted away from its initial design: construct '`Makefile.in`' by assembling various '`Makefile`' fragments. In 1.4, lots of '`Makefile`' rules are being emitted at various places in the `automake` script itself; this does not help ensuring a consistent treatment of these rules (for instance making sure that user-defined rules override Automake's own rules). One of Akim's goal was moving all these hard-coded rules to separate '`Makefile`' fragments, so the logic could be centralized in a '`Makefile`' fragment processor.

Another significant contribution of Akim is the interface with the "trace" feature of Autoconf. The way to scan '`configure.in`' at this time was to read the file and grep the various macro of interest to Automake. Doing so could break in many unexpected ways; `automake` could miss some definition (for instance '`AC_SUBST([$1], [$2])`' where the arguments are known only when M4 is run), or conversely it could detect some macro that was not expanded (because it is called conditionally). In the CVS version of Autoconf, Akim had implemented the '`--trace`' option, which provides accurate information about where macros are actually called and with what arguments. Akim will equip Automake with a second '`configure.in`' scanner that uses this '`--trace`' interface. Since it was not sensible to drop the Autoconf 2.13 compatibility yet, this experimental scanner was only used when an environment variable was set, the traditional grep-scanner being still the default.

2001-04-25 Gary V. Vaughan releases Libtool 1.4

It has been more than two years since Automake 1.4, CVS Automake has suffered lot's of heavy changes and still is not ready for release. Libtool 1.4 had to be distributed with a patch against Automake 1.4.

2001-05-08 Automake 1.4-p1

2001-05-24 Automake 1.4-p2

Gary V. Vaughan, the principal Libtool maintainer, makes a "patch release" of Automake:

> The main purpose of this release is to have a stable automake which is compatible with the latest stable libtool.

The release also contains obvious fixes for bugs in Automake 1.4, some of which were reported almost monthly.

2001-05-21 Akim Demaille releases Autoconf 2.50

2001-06-07 Automake 1.4-p3

2001-06-10 Automake 1.4-p4

2001-07-15 Automake 1.4-p5

Gary continues his patch-release series. These also add support for some new Autoconf 2.50 idioms. Essentially, Autoconf now advocates '`configure.ac`' over '`configure.in`', and it introduces a new syntax for `AC_OUTPUT`ing files.

2001-08-23 Automake 1.5

A major and long-awaited release, that comes more than two years after 1.4. It brings many changes, among which:

- The new dependency tracking scheme that uses `depcomp`. Aside from the improvement on the dependency tracking itself (see Chapter 2 [Dependency Tracking Evolution], page 13), this also streamlines the use of `automake`-generated '`Makefile.in`'s as the '`Makefile.in`'s used during development are now the same as those used in distributions. Before that the '`Makefile.in`'s generated for maintainers required GNU `make` and GCC, they were different from the portable '`Makefile`' generated for distribution; this was causing some confusion.
- Support for per-target compilation flags.
- Support for reference to files in subdirectories in most '`Makefile.am`' variables.
- Introduction of the `dist_`, `nodist_`, and `nobase_` prefixes.
- Perl 4 support is finally dropped.

1.5 did break several packages that worked with 1.4. Enough so that Linux distributions could not easily install the new Automake version without breaking many of the packages for which they had to run `automake`.

Some of these breakages were effectively bugs that would eventually be fixed in the next release. However, a lot of damage was caused by some changes made deliberately to render Automake stricter on some setup we did consider bogus. For instance, '`make distcheck`' was improved to check that '`make uninstall`' did remove all the files '`make install`' installed, that '`make distclean`' did

not omit some file, and that a VPATH build would work even if the source directory was read-only. Similarly, Automake now rejects multiple definitions of the same variable (because that would mix very badly with conditionals), and '`+=`' assignments with no previous definition. Because these changes all occurred suddenly after 1.4 had been established for more than two years, it hurt users.

To make matter worse, meanwhile Autoconf (now at version 2.52) was facing similar troubles, for similar reasons.

2002-03-05 Automake 1.6

This release introduced versioned installation (see Section "API Versioning" in *GNU Automake*). This was mainly pushed by Havoc Pennington, taking the GNOME source tree as motive: due to incompatibilities between the autotools it's impossible for the GNOME packages to switch to Autoconf 2.53 and Automake 1.5 all at once, so they are currently stuck with Autoconf 2.13 and Automake 1.4.

The idea was to call this version '`automake-1.6`', call all its bug-fix versions identically, and switch to '`automake-1.7`' for the next release that adds new features or changes some rules. This scheme implies maintaining a bug-fix branch in addition to the development trunk, which means more work from the maintainer, but providing regular bug-fix releases proved to be really worthwhile.

Like 1.5, 1.6 also introduced a bunch of incompatibilities, intentional or not. Perhaps the more annoying was the dependence on the newly released Autoconf 2.53. Autoconf seemed to have stabilized enough since its explosive 2.50 release and included changes required to fix some bugs in Automake. In order to upgrade to Automake 1.6, people now had to upgrade Autoconf too; for some packages it was no picnic.

While versioned installation helped people to upgrade, it also unfortunately allowed people not to upgrade. At the time of writing, some Linux distributions are shipping packages for Automake 1.4, 1.5, 1.6, 1.7, 1.8, and 1.9. Most of these still install 1.4 by default. Some distribution also call 1.4 the "stable" version, and present "1.9" as the development version; this does not really makes sense since 1.9 is way more solid than 1.4. All this does not help the newcomer.

2002-04-11 Automake 1.6.1

1.6, and the upcoming 1.4-p6 release were the last release by Tom. This one and those following will be handled by Alexandre Duret-Lutz. Tom is still around, and will be there until about 1.7, but his interest into Automake is drifting away towards projects like `gcj`.

Alexandre has been using Automake since 2000, and started to contribute mostly on Akim's incitement (Akim and Alexandre have been working in the same room from 1999 to 2002). In 2001 and 2002 he had a lot of free time to enjoy hacking Automake.

2002-06-14 Automake 1.6.2

2002-07-28 Automake 1.6.3

2002-07-28 Automake 1.4-p6

Two releases on the same day. 1.6.3 is a bug-fix release.

Tom Tromey backported the versioned installation mechanism on the 1.4 branch, so that Automake 1.6.x and Automake 1.4-p6 could be installed side by side. Another request from the GNOME folks.

2002-09-25 Automake 1.7
: This release switches to the new '`configure.ac`' scanner Akim was experimenting in 1.5.

2002-10-16 Automake 1.7.1
2002-12-06 Automake 1.7.2
2003-02-20 Automake 1.7.3
2003-04-23 Automake 1.7.4
2003-05-18 Automake 1.7.5
2003-07-10 Automake 1.7.6
2003-09-07 Automake 1.7.7
2003-10-07 Automake 1.7.8
: Many bug-fix releases. 1.7 lasted because the development version (upcoming 1.8) was suffering some major internal revamping.

2003-10-26 Automake on screen
: Episode 49, 'Repercussions', in the third season of the 'Alias' TV show is first aired.

  Marshall, one of the characters, is working on a computer virus that he has to modify before it gets into the wrong hands or something like that. The screenshots you see do not show any program code, they show a '`Makefile.in`' generated by automake...

2003-11-09 Automake 1.7.9
2003-12-10 Automake 1.8
: The most striking update is probably that of `aclocal`.

  `aclocal` now uses `m4_include` in the produced '`aclocal.m4`' when the included macros are already distributed with the package (an idiom used in many packages), which reduces code duplication. Many people liked that, but in fact this change was really introduced to fix a bug in rebuild rules: '`Makefile.in`' must be rebuilt whenever a dependency of '`configure`' changes, but all the '`m4`' files included in '`aclocal.m4`' where unknown from `automake`. Now `automake` can just trace the `m4_include`s to discover the dependencies.

  `aclocal` also starts using the '`--trace`' Autoconf option in order to discover used macros more accurately. This will turn out to be very tricky (later releases will improve this) as people had devised many ways to cope with the limitation of previous `aclocal` versions, notably using handwritten `m4_include`s: `aclocal` must make sure not to redefine a rule that is already included by such statement.

  Automake also has seen its guts rewritten. Although this rewriting took a lot of efforts, it is only apparent to the users in that some constructions previously disallowed by the implementation now work nicely. Conditionals, Locations,

Variable and Rule definitions, Options: these items on which Automake works have been rewritten as separate Perl modules, and documented.

2004-01-11 Automake 1.8.1

2004-01-12 Automake 1.8.2

2004-03-07 Automake 1.8.3

2004-04-25 Automake 1.8.4

2004-05-16 Automake 1.8.5

2004-07-28 Automake 1.9

This release tries to simplify the compilation rules it outputs to reduce the size of the Makefile. The complaint initially come from the libgcj developers. Their '`Makefile.in`' generated with Automake 1.4 and custom build rules (1.4 did not support compiled Java) is 250KB. The one generated by 1.8 was over 9MB! 1.9 gets it down to 1.2MB.

Aside from this it contains mainly minor changes and bug-fixes.

2004-08-11 Automake 1.9.1

2004-09-19 Automake 1.9.2

Automake has ten years. This chapter of the manual was initially written for this occasion.

2007-10-29 Automake repository moves to `savannah.gnu.org`
and uses git as primary repository.

2 Evolution of Automatic Dependency Tracking

Over the years Automake has deployed three different dependency tracking methods. Each method, including the current one, has had flaws of various sorts. Here we lay out the different dependency tracking methods, their flaws, and their fixes. We conclude with recommendations for tool writers, and by indicating future directions for dependency tracking work in Automake.

2.1 First Take on Dependency Tracking

Description

Our first attempt at automatic dependency tracking was based on the method recommended by GNU `make`. (see Section "Generating Prerequisites Automatically" in *The GNU make Manual*)

This version worked by precomputing dependencies ahead of time. For each source file, it had a special '`.P`' file that held the dependencies. There was a rule to generate a '`.P`' file by invoking the compiler appropriately. All such '`.P`' files were included by the '`Makefile`', thus implicitly becoming dependencies of '`Makefile`'.

Bugs

This approach had several critical bugs.

- The code to generate the '`.P`' file relied on `gcc`. (A limitation, not technically a bug.)
- The dependency tracking mechanism itself relied on GNU `make`. (A limitation, not technically a bug.)
- Because each '`.P`' file was a dependency of '`Makefile`', this meant that dependency tracking was done eagerly by `make`. For instance, '`make clean`' would cause all the dependency files to be updated, and then immediately removed. This eagerness also caused problems with some configurations; if a certain source file could not be compiled on a given architecture for some reason, dependency tracking would fail, aborting the entire build.
- As dependency tracking was done as a pre-pass, compile times were doubled–the compiler had to be run twice per source file.
- '`make dist`' re-ran `automake` to generate a '`Makefile`' that did not have automatic dependency tracking (and that was thus portable to any version of `make`). In order to do this portably, Automake had to scan the dependency files and remove any reference that was to a source file not in the distribution. This process was error-prone. Also, if '`make dist`' was run in an environment where some object file had a dependency on a source file that was only conditionally created, Automake would generate a '`Makefile`' that referred to a file that might not appear in the end user's build. A special, hacky mechanism was required to work around this.

Historical Note

The code generated by Automake is often inspired by the '`Makefile`' style of a particular author. In the case of the first implementation of dependency tracking, I believe the impetus and inspiration was Jim Meyering. (I could be mistaken. If you know otherwise feel free to correct me.)

2.2 Dependencies As Side Effects

Description

The next refinement of Automake's automatic dependency tracking scheme was to implement dependencies as side effects of the compilation. This was aimed at solving the most commonly reported problems with the first approach. In particular we were most concerned with eliminating the weird rebuilding effect associated with make clean.

In this approach, the '`.P`' files were included using the `-include` command, which let us create these files lazily. This avoided the '`make clean`' problem.

We only computed dependencies when a file was actually compiled. This avoided the performance penalty associated with scanning each file twice. It also let us avoid the other problems associated with the first, eager, implementation. For instance, dependencies would never be generated for a source file that was not compilable on a given architecture (because it in fact would never be compiled).

Bugs

- This approach also relied on the existence of `gcc` and GNU `make`. (A limitation, not technically a bug.)
- Dependency tracking was still done by the developer, so the problems from the first implementation relating to massaging of dependencies by '`make dist`' were still in effect.
- This implementation suffered from the "deleted header file" problem. Suppose a lazily-created '`.P`' file includes a dependency on a given header file, like this:

  ```
  maude.o: maude.c something.h
  ```

  Now suppose that you remove '`something.h`' and update '`maude.c`' so that this include is no longer needed. If you run `make`, you will get an error because there is no way to create '`something.h`'.

  We fixed this problem in a later release by further massaging the output of `gcc` to include a dummy dependency for each header file.

2.3 Dependencies for the User

Description

The bugs associated with '`make dist`', over time, became a real problem. Packages using Automake were being built on a large number of platforms, and were becoming increasingly complex. Broken dependencies were distributed in "portable" '`Makefile.in`'s, leading to user complaints. Also, the requirement for `gcc` and GNU `make` was a constant source of bug reports. The next implementation of dependency tracking aimed to remove these problems.

We realized that the only truly reliable way to automatically track dependencies was to do it when the package itself was built. This meant discovering a method portable to any version of make and any compiler. Also, we wanted to preserve what we saw as the best point of the second implementation: dependency computation as a side effect of compilation.

In the end we found that most modern make implementations support some form of include directive. Also, we wrote a wrapper script that let us abstract away differences

between dependency tracking methods for compilers. For instance, some compilers cannot generate dependencies as a side effect of compilation. In this case we simply have the script run the compiler twice. Currently our wrapper script (`depcomp`) knows about twelve different compilers (including a `"compiler"` that simply invokes `makedepend` and then the real compiler, which is assumed to be a standard Unix-like C compiler with no way to do dependency tracking).

Bugs

- Running a wrapper script for each compilation slows down the build.
- Many users don't really care about precise dependencies.
- This implementation, like every other automatic dependency tracking scheme in common use today (indeed, every one we've ever heard of), suffers from the "duplicated new header" bug.

  This bug occurs because dependency tracking tools, such as the compiler, only generate dependencies on the successful opening of a file, and not on every probe.

  Suppose for instance that the compiler searches three directories for a given header, and that the header is found in the third directory. If the programmer erroneously adds a header file with the same name to the first directory, then a clean rebuild from scratch could fail (suppose the new header file is buggy), whereas an incremental rebuild will succeed.

  What has happened here is that people have a misunderstanding of what a dependency is. Tool writers think a dependency encodes information about which files were read by the compiler. However, a dependency must actually encode information about what the compiler tried to do.

  This problem is not serious in practice. Programmers typically do not use the same name for a header file twice in a given project. (At least, not in C or C++. This problem may be more troublesome in Java.) This problem is easy to fix, by modifying dependency generators to record every probe, instead of every successful open.
- Since Automake generates dependencies as a side effect of compilation, there is a bootstrapping problem when header files are generated by running a program. The problem is that, the first time the build is done, there is no way by default to know that the headers are required, so make might try to run a compilation for which the headers have not yet been built.

  This was also a problem in the previous dependency tracking implementation.

  The current fix is to use `BUILT_SOURCES` to list built headers (see Section "Sources" in *GNU Automake*). This causes them to be built before any other build rules are run. This is unsatisfactory as a general solution, however in practice it seems sufficient for most actual programs.

This code is used since Automake 1.5.

In GCC 3.0, we managed to convince the maintainers to add special command-line options to help Automake more efficiently do its job. We hoped this would let us avoid the use of a wrapper script when Automake's automatic dependency tracking was used with `gcc`.

Unfortunately, this code doesn't quite do what we want. In particular, it removes the dependency file if the compilation fails; we'd prefer that it instead only touch the file in any way if the compilation succeeds.

Nevertheless, since Automake 1.7, when a recent `gcc` is detected at `configure` time, we inline the dependency-generation code and do not use the `depcomp` wrapper script. This makes compilations faster for those using this compiler (probably our primary user base). The counterpart is that because we have to encode two compilation rules in '`Makefile`' (with or without `depcomp`), the produced '`Makefile`'s are larger.

2.4 Techniques for Computing Dependencies

There are actually several ways for a build tool like Automake to cause tools to generate dependencies.

`makedepend`
: This was a commonly-used method in the past. The idea is to run a special program over the source and have it generate dependency information. Traditional implementations of `makedepend` are not completely precise; ordinarily they were conservative and discovered too many dependencies.

The tool
: An obvious way to generate dependencies is to simply write the tool so that it can generate the information needed by the build tool. This is also the most portable method. Many compilers have an option to generate dependencies. Unfortunately, not all tools provide such an option.

The file system
: It is possible to write a special file system that tracks opens, reads, writes, etc, and then feed this information back to the build tool. `clearmake` does this. This is a very powerful technique, as it doesn't require cooperation from the tool. Unfortunately it is also very difficult to implement and also not practical in the general case.

`LD_PRELOAD`
: Rather than use the file system, one could write a special library to intercept `open` and other syscalls. This technique is also quite powerful, but unfortunately it is not portable enough for use in `automake`.

2.4.1 Recommendations for Tool Writers

We think that every compilation tool ought to be able to generate dependencies as a side effect of compilation. Furthermore, at least while `make`-based tools are nearly universally in use (at least in the free software community), the tool itself should generate dummy dependencies for header files, to avoid the deleted header file bug. Finally, the tool should generate a dependency for each probe, instead of each successful file open, in order to avoid the duplicated new header bug.

2.4.2 Future Directions for Dependencies

Currently, only languages and compilers understood by Automake can have dependency tracking enabled. We would like to see if it is practical (and worthwhile) to let this support be extended by the user to languages unknown to Automake.

3 Release Statistics

The following table (inspired by '`perlhist(1)`') quantifies the evolution of Automake using these metrics:

Date, Rel — The date and version of the release.

am — The number of lines of the `automake` script.

acl — The number of lines of the `aclocal` script.

pm — The number of lines of the `Perl` supporting modules.

'`*.am`' — The number of lines of the '`Makefile`' fragments. The number in parentheses is the number of files.

m4 — The number of lines (and files) of Autoconf macros.

doc — The number of pages of the documentation (the Postscript version).

t — The number of test cases in the test suite. Of those, the number in parentheses is the number of generated test cases.

Date	**Rel**	**am**	**acl**	**pm**	**'*.am'**	**m4**	**doc**	**t**
1994-09-19	CVS	141			299 (24)			
1994-11-05	CVS	208			332 (28)			
1995-11-23	0.20	533			458 (35)		9	
1995-11-26	0.21	613			480 (36)		11	
1995-11-28	0.22	1116			539 (38)		12	
1995-11-29	0.23	1240			541 (38)		12	
1995-12-08	0.24	1462			504 (33)		14	
1995-12-10	0.25	1513			511 (37)		15	
1996-01-03	0.26	1706			438 (36)		16	
1996-01-03	0.27	1706			438 (36)		16	
1996-01-13	0.28	1964			934 (33)		16	
1996-02-07	0.29	2299			936 (33)		17	
1996-02-24	0.30	2544			919 (32)	85 (1)	20	9
1996-03-11	0.31	2877			919 (32)	85 (1)	29	17
1996-04-27	0.32	3058			921 (31)	85 (1)	30	26
1996-05-18	0.33	3110			926 (31)	105 (1)	30	35
1996-05-28	1.0	3134			973 (32)	105 (1)	30	38
1997-06-22	1.2	6089	385		1294 (36)	592 (20)	37	126
1998-04-05	1.3	6415	422		1470 (39)	741 (23)	39	156
1999-01-14	1.4	7240	426		1591 (40)	734 (20)	51	197
2001-05-08	1.4-p1	7251	426		1591 (40)	734 (20)	51	197
2001-05-24	1.4-p2	7268	439		1591 (40)	734 (20)	49	197
2001-06-07	1.4-p3	7312	439		1591 (40)	734 (20)	49	197
2001-06-10	1.4-p4	7321	439		1591 (40)	734 (20)	49	198
2001-07-15	1.4-p5	7228	426		1596 (40)	734 (20)	51	198
2001-08-23	1.5	8016	475	600	2654 (39)	1166 (29)	63	327
2002-03-05	1.6	8465	475	1136	2732 (39)	1603 (27)	66	365

2002-04-11	1.6.1	8544	475	1136	2741 (39)	1603 (27)	66	372
2002-06-14	1.6.2	8575	475	1136	2800 (39)	1609 (27)	67	386
2002-07-28	1.6.3	8600	475	1153	2809 (39)	1609 (27)	67	391
2002-07-28	1.4-p6	7332	455		1596 (40)	735 (20)	49	197
2002-09-25	1.7	9189	471	1790	2965 (39)	1606 (28)	73	430
2002-10-16	1.7.1	9229	475	1790	2977 (39)	1606 (28)	73	437
2002-12-06	1.7.2	9334	475	1790	2988 (39)	1606 (28)	77	445
2003-02-20	1.7.3	9389	475	1790	3023 (39)	1651 (29)	84	448
2003-04-23	1.7.4	9429	475	1790	3031 (39)	1644 (29)	85	458
2003-05-18	1.7.5	9429	475	1790	3033 (39)	1645 (29)	85	459
2003-07-10	1.7.6	9442	475	1790	3033 (39)	1660 (29)	85	461
2003-09-07	1.7.7	9443	475	1790	3041 (39)	1660 (29)	90	467
2003-10-07	1.7.8	9444	475	1790	3041 (39)	1660 (29)	90	468
2003-11-09	1.7.9	9444	475	1790	3048 (39)	1660 (29)	90	468
2003-12-10	1.8	7171	585	7730	3236 (39)	1666 (31)	104	521
2004-01-11	1.8.1	7217	663	7726	3287 (39)	1686 (31)	104	525
2004-01-12	1.8.2	7217	663	7726	3288 (39)	1686 (31)	104	526
2004-03-07	1.8.3	7214	686	7735	3303 (39)	1695 (31)	111	530
2004-04-25	1.8.4	7214	686	7736	3310 (39)	1701 (31)	112	531
2004-05-16	1.8.5	7240	686	7736	3299 (39)	1701 (31)	112	533
2004-07-28	1.9	7508	715	7794	3352 (40)	1812 (32)	115	551
2004-08-11	1.9.1	7512	715	7794	3354 (40)	1812 (32)	115	552
2004-09-19	1.9.2	7512	715	7794	3354 (40)	1812 (32)	132	554
2004-11-01	1.9.3	7507	718	7804	3354 (40)	1812 (32)	134	556
2004-12-18	1.9.4	7508	718	7856	3361 (40)	1811 (32)	140	560
2005-02-13	1.9.5	7523	719	7859	3373 (40)	1453 (32)	142	562
2005-07-10	1.9.6	7539	699	7867	3400 (40)	1453 (32)	144	570
2006-10-15	1.10	7859	1072	8024	3512 (40)	1496 (34)	172	604
2008-01-19	1.10.1	7870	1089	8025	3520 (40)	1499 (34)	173	617
2008-11-23	1.10.2	7882	1089	8027	3540 (40)	1509 (34)	176	628
2009-05-17	1.11	8721	1092	8289	4164 (42)	1714 (37)	181	732 (20)

Appendix A Copying This Manual

A.1 GNU Free Documentation License

Version 1.3, 3 November 2008

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.

The "publisher" means any person or entity that distributes copies of the Document to the public.

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 under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

   Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

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. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

    "Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

    "CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

    "Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

    An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

    The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

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 1.3
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.