Chapter 1 Introduction

A physicist, an engineer, and a computer scientist were discussing the nature of God. “Surely a Physicist,” said the physicist, “because early in the Creation, God made Light; and you know, Maxwell’s equations, the dual nature of electromagnetic waves, the relativistic consequences. . .” “An Engineer!,” said the engineer, “because before making Light, God split the Chaos into Land and Water; it takes a hell of an engineer to handle that big amount of mud, and orderly separation of solids from liquids. . .” The computer scientist shouted: “And the Chaos, where do you think it was coming from, hmm?”

—Anonymous

Autoconf is an extensible package of M4 macros that produce shell scripts to automatically configure software source code packages. The configuration script generated by Autoconf are executed by a shell interpreter and not by Autoconf. At execution time, Autoconf is not needed.

The generated shell script requires no manual user intervention when run. The generated script tests for the presence of each feature that the software packageneeds to compie successfully. (Before each check, a one-line message stating what is being checked is output, so the user doesn’t get too bored while waiting for the script to finish.) The generated script deals well with systems that are hybrids or customized from the more common Posix variants. The generated script replaces user files or scripts which list or check for individual Posix features or Posix variants.

Autoconf creates a configuration script from a macro template file that lists the system features needs for a successful compile. The input template file(s) can be used by Autoconf to generate scripts for any software package (the input drives the output). Changes in the compilation environment which change the required checks are supported for all generated scripts for all software packages by changing the template file and using Autoconf to generate new scripts. The template file is a single source serving multiple builds and becomes a single maintenance point for these builds.

Those who do not understand Autoconf are condemned to reinvent it[1], poorly. The primary goal of Autoconf is making the user’s life easier; making the maintainer’s life easier is only a secondary goal. Put another way, the primary goal is not to make the generation of ‘configure’ automatic for package maintainers (although patches along that front are welcome, since package maintainers form the user base of Autoconf); rather, the goal is to make ‘configure’ painless, portable, and predictable for the end user of each autoconfiscated package. And to this degree, Autoconf is highly successful at its goal — most complaints to the Autoconf list are about difficulties in writing Autoconf input, and not in the behavior of the resulting ‘configure’. Even packages that don’t use Autoconf will generally provide a ‘configure’ script, and the most common complaint about these alternative home-grown scripts is that they fail to meet one or more of the GNU Coding Standards (see Section “Configuration” in The GNU Coding Standards) that users have come to expect from Autoconf-generated ‘configure’ scripts.

1 The original quote by George Santana is “”*Those* who cannot *learn* from *history are doomed to repeat it* .

Autoconf is one member of a family of tools useful in making it possible to compile software packages in multiple environments. Autoconf should be used in concert with GNU build tools like Automake and Libtool. These other tools take on jobs like the creation of a portable, recursive makefile with all of the standard targets, linking of shared libraries, and so on. See Chapter 2 [The GNU Build System], page 3, for more information. Autoconf imposes some restrictions on the names of macros used with #if in C/C++ programs (see Section B.3 [Preprocessor Symbol Index], page 366).

Autoconf requires GNU M4 version 1.4.6 or later. Autoconf works better with GNU M4 version 1.4.14 or later, though this is not required.

See Section 18.5 [Autoconf 1], page 318, for information about upgrading Autoconf from version 1. See Chapter 21 [History], page 351, for the story of Autoconf's development. See Chapter 20 [FAQ], page 341, for answers to some common questions about Autoconf.

See the Autoconf web page for up-to-date information, details on the mailing lists, pointers to a list of known bugs, etc.

Mail suggestions to the Autoconf mailing list. Past suggestions are archived.

Mail bug reports to the Autoconf Bugs mailing list. Past bug reports are archived. If possible, first check that your bug is not already solved in current development versions, and that it has not been reported yet. Be sure to include all the needed information and a short 'configure.ac' that demonstrates the problem.

Autoconf's development tree is accessible via git from http://savannah.gnu.org/projects/autoconf/; see the Autoconf Summary for details, or view the actual repository. Anonymous CVS access is also available, see 'README' for more details. Patches relative to the current git version can be sent for review to the Autoconf Patches mailing list, with discussion on prior patches archived; and all commits are posted in the read-only Autoconf Commit mailing list, which is also archived.

Because of its mission, the Autoconf package itself includes only a set of often-used macro templates that have already demonstrated their usefulness. Nevertheless, if you wish to share your macros, or find existing ones, see the Autoconf Macro Archive, which is kindly run by
Peter Simons.

Chapter 2 Terminology

Throughout the remainder of this book the following terminology will be used.

Table 1: autoconf standard terminology

application	The source code for which autoconf generated scripts are being generated..
autoconf	A collection of shell command and M4 macros which describe the tests to be inserted into the generated script.
build system	The computer and operating system where the build tools are employed to generate a script.
compiler	The toolset is specific to the GNU gcc/g++ compilers and linkers. Other compilers and linkers can be used provided their command line interface conforms to the required by GNU.
configuration script	The shell script generated by the autoconf build tool.
developer	The role of the source code package creator.
host host system	The computer and operating system where the build tools are employed to generate a script.
maintainer	The role of the post-delivery source code package maintainer responsible for enhancements and bug fixes.
script	The shell script generated by the autoconf build tool.
source code	The C/C++ source code and headers which constitute the application.
source package	Both source code and application associated files (AUTHORS, ChangeLog, COPYING, INSTALL, NEWS, PACKAGING, README, THANKS, TODO)
source code package	Same as source package.
target target system	The computer and operating system where the generated script is executed.
user	The role of the application user using the autoconf script to compile the application.

The developer, maintainer, and user are roles. Any one person or group of people can be in any one or all of these roles at any given time. Each role has certain responsibilities and functions with respect to using the autoconf tool and GNU build suite or the generated scripts. The developer and maintainer use the tool and the user uses the script.

The host system and the target system may or may not be the same. The developer and maintainer are responsible for activities done on the host system. The user uses the generated scripts on the target system. The autoconf tool generates scripts which validates that the application can be compiled and executed on the target system.

2.1 Colophon

TBS

Chapter 3 The GNU Build Tool Suite

GNU has a suite of command line tools to support the maintainer in generating scripts and files for a user on a target system to use to compile and build an application. The tool suite is designed to develop target independent compilation systems to validate that a compilation and build will be successful and the perform the compilation and build. The tool set is specifically for the maintainer and not the user. It is unnecessary for the user to have any tool in the suite on the target system.

The maintainer is required to have a GNU compatible shell, to execute the command line tools, and to have M4, a GNU macro processor.

The user is required to have a GNU compatible to execute the autoconf generated script, and a GNU compliant make tool. The user is not required to have M4 or any of the tools in this tool suite.

The tools are optional The maintainer can elect to use or not use any tool. Each tool can have a unique set of input files or share input files with other tools. The shared files maybe be files generated by other tools, which establishes an runtime dependency. Each tool generates or modifies output files. All files, both input and output, are ASCII and can be generated by hand, as desired, or viewed.

One tool in this suite is autoconf. Autoconf generates a script to validate that the application can be compiled and built on the target system. Autoconf solves an important problem—reliable discovery of system-specific build and runtime information

The list of tools is:

Table 2: GNU Build Tool Suite

Tool	Main Purpose
autoconf	Generates a target script, 'configure', to validate compilation.
autoheader	Generates and application specific header file.
aclocal	Generates a host specific macro file.
autoscan	Generates a preliminay configure.ac (autoconf) input file.
automake	Supports generation of make files.
libtool	Supports the development of shared, portable libraries.
lilbtoolize	

The input and output files for each tool is:

Table 3: GNU Build Tool Input and Output Files

input	tool	output
configure.ac [aclocal.m4] [acsite.m4]	autoconf	configure
configure.ac aclocal.m4 acsite.m4	autoheader	config.h.in
[acinclude.m4] configure.ac [*.m4]	aclocal	aclocal.m4
application source files	autoscan	configure.scan
configure.ac Makefile.am	automake	Makefile.in
ltmain.sh	libtool	
libtool.m4 ltdl.m4 ltoptions.m4 lt~obsolete.m4 ltsugar.m4 ltversion.m4 *.m4	libtoolize	m4/*.m4

Note: '[' filename ']' means that the file is optional for the application.

The file descriptions are:

Table 4: GNU Build Tool File Descriptions

File	Description
aclocal	
acsite	
acinclude	
application source files	
configure	Target configuration script.
configure.ac	Input configuration file to autoconf.
configure.scan	Preliminary configure.ac file generated by autoscan.
libtool.m4	
ltdl.m4	
ltoptions.m4	
ltmain.sh	
ltsugar.m4	
ltsugar.m4	
lt~obsolete.m4	
*.m4	Maintainer developed m4 macros for configuration.

3.1 Maintainer Host Build

The purpose of the Maintainer Host Build is to construct a configure script for use in compiling and building the application on the target system. Programs in the GNU Build Tool Suite are executed in a specific order to accomplish the generation of the user configure script and, as necessary, the required make files (Makefile) which allow compiling and building of the application. The resultant configure is executed during the User Target Build.

The order of the GNU Build Tool Suite tools preserves the file dependencies used in the tools. The GNU Build Tool autoconf is the only required tool. The input and output files usage is described in Table 3: GNU Build Tool Input and Output Files and the files are described in Table 4: GNU Build Tool File Descriptions.

Square brackets, '[ ]', surround files which are optional. The generation of optional output files is controlled by command line arguments for the tool.

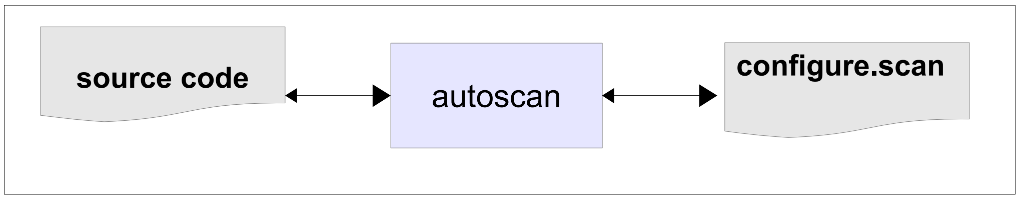

Figure 1: autoscan build

The autoscan build in Figure 1 is an optional operation. The input to the autoscan program are the C/C++ source and header files, and the program generates the configure.scan ASCII file. The configure.scan file contains macros and shell script fragments which define a preliminary version of configure.ac needed by autconf.

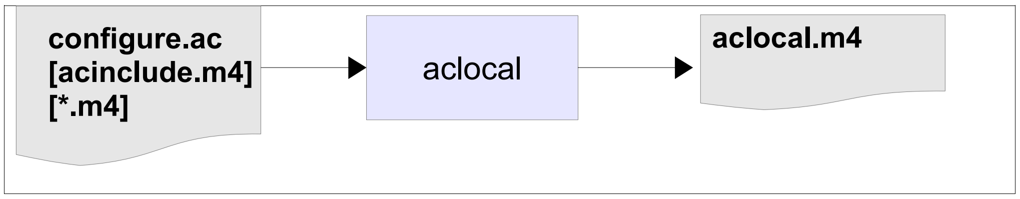

Figure 2: aclocal build

The aclocal program accepts the configure.ac file generated by autoscan and modified by the maintainer. The remaining optional input files, acinclude.m4 and *.m4, are created by the maintainer for use by aclocal. The generated file instruments the build process of autoconf on the host computer. This allows the flexibility for maintainers to use a variety of hosts in enhancing and fixing the application and subsequently building a configure script.

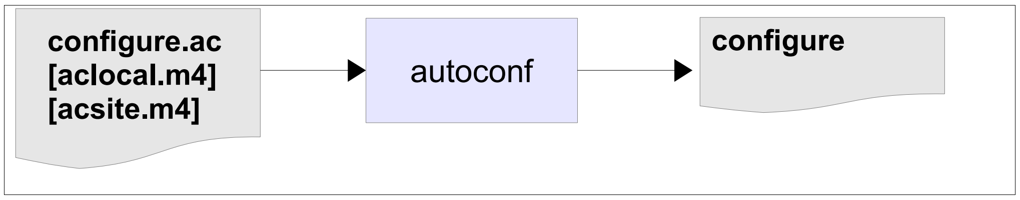

Figure 3: autoconf build

The autoconf program accepts the configure.ac, defining application specific checks to perform, and optionally accepts the aclocal.m4 and acsite.m4 intermixed macro and script files. The generated configure scripts performs a number of services for the user. Primarily, the script checks if various capabilities exist on the target machine and target compiler, and modifies Makefiles to support a successful, target specific, compile and build of the application.

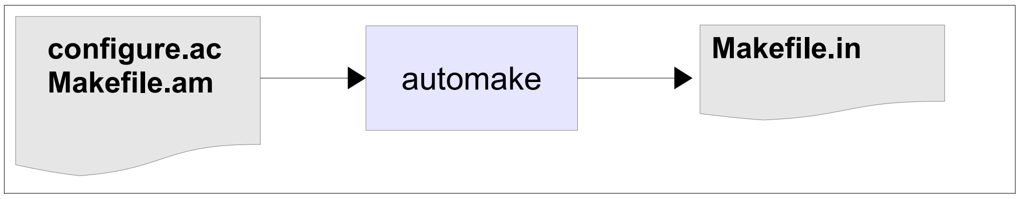

Figure 4: automake build

automake is a stand-alone program designed to support the creation of application specific Makefile(s) to compile and build the application. The tool is an aid. The maintainer is free to create application Makefiles using a text editor. **Note: the Makefile targets are missing from the original text. More description is needed.**

3.2 User Target Build

In order for the user to use the application, the user must compile and build the code. This process uses the generated configure script, see Figure 3: autoconf build, and maintainer created Makefile(s), see Figure 4: automake build and a GNU compatible make program. The minimum target build command line is given as:

```
> configure && make && make install
```

where the generated configure script checks the target environment (architecture and compiler options) and alters the Makefile(s) to allow a successful build and then make does the compile and build, and make install installs the resulting executables (or libraries) and associated ASCII files into their appropriate locations. The assumptions throughout are that the target environment is GNU and Unix compatible.