A Short Introduction to Autoconf

Stephen A. Edwards

January 11, 1996

Autoconf [1] is a tool for producing standalone shell scripts that automatically configure software source code packages to adapt to many UNIX-like systems.

Autoconf generates an executable Bourne (`/bin/sh`) shell [2] script called `configure`. Typically, `configure` generates a customized `Makefile` from a template `Makefile.in`, customized to each package being compiled. Such a flow is shown in Figure 1.

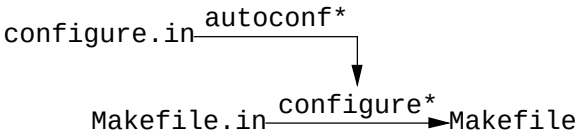

Figure 1: A typical `autoconf` flow. Names of executables have a trailing asterisk.

`Configure`, rather than trying to determine exactly what system it is on, tries instead to determine *characteristics* of the system that are needed for correct compilation. Thus, a `configure` script can often work on systems for which it wasn't specifically designed. For example, instead of trying to decide whether it is running on a Solaris 5.4 machine and concluding that ANSI header files are present, `configure` checks for `<stdlib.h>` and other ANSI header files directly. Another advantage to this approach is the ability to look for "extra" programs, such as `bison`, which are not usually part of commercial operating systems.

1 The `Makefile.in` file

`Configure` generates `Makefile` from `Makefile.in` by substituting all at-sign-enclosed words (e.g., `@srcdir@`) for strings in the substituion list defined by `configure.in`. Typical strings include

`@srcdir@`	Location of the source code, often `.`
`@prefix@`	Directory in which to install architecture-independent files, by default `/usr/local/`
`@CC@`	The name of the C compiler
`@DEFS@`	A list of `-D` flags for the C compiler, e.g., `-DHAVE STDC_HEADERS=1 -DHAVE_DIRENT_H=1`
`@CFLAGS@`	Additional flags for the C compiler, e.g., `-g`)

2 The `configure.in` file

Autoconf processes `configure.in` by running it through the `m4` [3] macro preprocessor to produce the `configure` script. Thus, the `configure.in` file is a series of `m4` macro calls with shell commands interspersed.

All autoconf macros start with `AC_`, and are all uppercase.

Here are some usefule autoconf macros:

- `AC_DEFINE(`*variable* [,*value*] `)`

  Define C preprocessor variable *variable*. If *value* is given, set *variable* to that variable. This definition shows up in the `@DEFS@` list in `Makefile.in`.

- `AC_SUBST(` *variable* `)`

  Substitute for the string *@variable@* in `Makefile.in`. The replacement string is the shell variable with the same name. For example, the following within `configure.in`

  ```
  AC_SUBST(vislibdir)
  vislibdir="/projects/vis/vis"
  ```

would replace `@vislibdir@` in `Makefile.in` with `/projects/vis/vis`. The first line is an `m4` macro call, the second is a Bourne shell command that sets a variable.

- `AC_ARG_WITH(`*package*, *help-string*, [, *action-if-given* [,*action-if-not-given*]]`))`

  Add a command-line argument to `configure` of the form `--with-`textitpackage`=`*arg*. Execute the shell commands *action-if-given* and *action-if-not-given* depending on whether the argument was provided. The shell variable `withval` is set to *arg* when specified.

  Here's a partial example. Note the use of the quote characters `[` and `]` to specify a multi-line comment.

```
AC_ARG_WITH(comp-mode,
[  --with-comp-mode=<mode>
   Specify a compilation mode:
   optimize or debug],
[comp_mode=$withval],
[comp_mode=optimize])

case "$comp_mode" in
  debug)
        CFLAGS=-g ;;
  optimize | * )
        CFLAGS=-O
        AC_DEFINE(NDEBUG) ;;
esac
```

- `AC_CHECK_HEADERS(`*header-file-list*`)`

  This macro checks for the presence of a list of named header files, defining C preprocessor macros. For example, `AC_CHECK_HEADERS(bsd/sgtty.h)` checks if `#include <bsd/sgtty.h>` finds a header. If so, `HAVE_BSD_SGTTY_H` is `#define`ed.

- `AC_EGREP_CPP`

  This macro runs the C preprocessor on a small input file, pipes the result through egrep, and runs one of two shell commands based on the results. Useful for checking whether something is defined in a header file.

There are many, many more autoconf macros, including ones to cache the result of expensive tests (for later runs of `configure`), to print explanatory messages, to check for generic or specific programs (such as the C compiler), to check for libraries and functions within, and many strange-case-specific macros (e.g., things that figure out which of the many "time" header files to include).

References

[1] The GNU project. `autoconf`. Available from `ftp://prep.ai.mit.edu/pub/gnu`. Documentation under the `emacs` info system (try `C-h i`). Also available as HTML at `http://www-cad.eecs.berkeley.edu/~sedwards/autoconf.html`.

[2] Steve Bourne. `/bin/sh`. The old standard shell. See, e.g., Brian W. Kernigan and Rob Pike. *The UNIX Programming Environment*. Prentice-Hall 1984.

[3] `m4`. One of the standard UNIX utilities. The GNU project has a version, available from `ftp://prep.ai.mit.edu/pub/gnu`.