Next: C++ Support, Previous: Program Variables, Up: Programs [Contents][Index]

8.8 Yacc and Lex support

Automake has somewhat idiosyncratic support for Yacc and Lex.

Automake assumes that the `.c` file generated by `yacc` (or `lex`) should be named using the basename of the input file. That is, for a yacc source file `foo.y`, Automake will cause the intermediate file to be named `foo.c` (as opposed to `y.tab.c`, which is more traditional).

The extension of a yacc source file is used to determine the extension of the resulting C or C++ source and header files. Note that header files are generated only when the `-d` Yacc option is used; see below for more information about this flag, and how to specify it. Files with the extension `.y` will thus be turned into `.c` sources and `.h` headers; likewise, `.yy` will become `.cc` and `.hh`, `.y++` will become `c++` and `h++`, `.yxx` will become `.cxx` and `.hxx`, and `.ypp` will become `.cpp` and `.hpp`.

Similarly, lex source files can be used to generate C or C++; the extensions `.l`, `.ll`, `.l++`, `.lxx`, and `.lpp` are recognized.

You should never explicitly mention the intermediate (C or C++) file in any `SOURCES` variable; only list the source file.

The intermediate files generated by `yacc` (or `lex`) will be included in any distribution that is made. That way the user doesn't need to have `yacc` or `lex`.

If a `yacc` source file is seen, then your `configure.ac` must define the variable `YACC`. This is most easily done by invoking the macro `AC_PROG_YACC` (see Particular Program Checks in *The Autoconf Manual*).

When `yacc` is invoked, it is passed `AM_YFLAGS` and `YFLAGS`. The latter is a user variable and the former is intended for the `Makefile.am` author.

`AM_YFLAGS` is usually used to pass the `-d` option to `yacc`. Automake knows what this means and will automatically adjust its rules to update and distribute the header file built by '`yacc -d`'[4]. What Automake cannot guess, though, is where this header will be used: it is up to you to ensure the header gets built before it is first used. Typically this is necessary in order for dependency tracking to work when the header is included by another file. The common solution is listing the header file in `BUILT_SOURCES` (see Sources) as follows.

```
BUILT_SOURCES = parser.h
AM_YFLAGS = -d
bin_PROGRAMS = foo
foo_SOURCES = … parser.y …
```

If a `lex` source file is seen, then your `configure.ac` must define the variable `LEX`. You can use `AC_PROG_LEX` to do this (see Particular Program Checks

in *The Autoconf Manual*), but using `AM_PROG_LEX` macro (see Macros) is recommended.

When `lex` is invoked, it is passed `AM_LFLAGS` and `LFLAGS`. The latter is a user variable and the former is intended for the `Makefile.am` author.

When `AM_MAINTAINER_MODE` (see maintainer-mode) is used, the rebuild rule for distributed Yacc and Lex sources are only used when `maintainer-mode` is enabled, or when the files have been erased.

When `lex` or `yacc` sources are used, `automake -a` automatically installs an auxiliary program called `ylwrap` in your package (see Auxiliary Programs). This program is used by the build rules to rename the output of these tools, and makes it possible to include multiple `yacc` (or `lex`) source files in a single directory. (This is necessary because yacc's output file name is fixed, and a parallel make could conceivably invoke more than one instance of `yacc` simultaneously.)

For `yacc`, simply managing locking is insufficient. The output of `yacc` always uses the same symbol names internally, so it isn't possible to link two `yacc` parsers into the same executable.

We recommend using the following renaming hack used in `gdb`:

```
#define yymaxdepth c_maxdepth
#define yyparse c_parse
#define yylex   c_lex
#define yyerror c_error
#define yylval  c_lval
#define yychar  c_char
#define yydebug c_debug
#define yypact  c_pact
#define yyr1    c_r1
#define yyr2    c_r2
#define yydef   c_def
#define yychk   c_chk
#define yypgo   c_pgo
#define yyact   c_act
#define yyexca  c_exca
#define yyerrflag c_errflag
#define yynerrs c_nerrs
#define yyps    c_ps
#define yypv    c_pv
#define yys     c_s
#define yy_yys  c_yys
#define yystate c_state
#define yytmp   c_tmp
#define yyv     c_v
#define yy_yyv  c_yyv
```

```
#define yyval   c_val
#define yylloc  c_lloc
#define yyreds  c_reds
#define yytoks  c_toks
#define yylhs   c_yylhs
#define yylen   c_yylen
#define yydefred c_yydefred
#define yydgoto  c_yydgoto
#define yysindex c_yysindex
#define yyrindex c_yyrindex
#define yygindex c_yygindex
#define yytable  c_yytable
#define yycheck  c_yycheck
#define yyname   c_yyname
#define yyrule   c_yyrule
```

For each define, replace the ‘`c_`’ prefix with whatever you like. These defines work for `bison`, `byacc`, and traditional `yacc`s. If you find a parser generator that uses a symbol not covered here, please report the new name so it can be added to the list.

Footnotes

(4)

Please note that `automake` recognizes `-d` in `AM_YFLAGS` only if it is not clustered with other options; for example, it won’t be recognized if `AM_YFLAGS` is `-dt`, but it will be if `AM_YFLAGS` is `-d -t` or `-t -d`.

Next: C++ Support, Previous: Program Variables, Up: Programs [Contents][Index]