NAME

bison − GNU Project parser generator (yacc replacement)

SYNOPSIS

bison [**−b** *file-prefix*] [**−−file-prefix=***file-prefix*] [**−d**] [**−−defines=***defines-file*] [**−g**] [**−−graph=***graph-file*] [**−k**] [**−−token-table**] [**−l**] [**−−no-lines**] [**−n**] [**−−no-parser**] [**−o** *outfile*] [**−−output-file=***outfile*] [**−p** *prefix*] [**−−name-prefix=***prefix*] [**−t**] [**−−debug**] [**−v**] [**−−verbose**] [**−V**] [**−−version**] [**−y**] [**−−yacc**] [**−h**] [**−−help**] [**−−fixed-output-files**] file
yacc [*similar options and operands*]

DESCRIPTION

Bison is a parser generator in the style of *yacc*(1). It should be upwardly compatible with input files designed for *yacc*.

Input files should follow the *yacc* convention of ending in **.y**. Unlike *yacc*, the generated files do not have fixed names, but instead use the prefix of the input file. Moreover, if you need to put *C++* code in the input file, you can end his name by a C++-like extension (.ypp or .y++), then bison will follow your extension to name the output file (.cpp or .c++). For instance, a grammar description file named **parse.yxx** would produce the generated parser in a file named **parse.tab.cxx**, instead of *yacc*'s **y.tab.c** or old *Bison* version's **parse.tab.c**.

This description of the options that can be given to *bison* is adapted from the node **Invocation** in the **bison.texinfo** manual, which should be taken as authoritative.

Bison supports both traditional single-letter options and mnemonic long option names. Long option names are indicated with **−−** instead of **−**. Abbreviations for option names are allowed as long as they are unique. When a long option takes an argument, like **−−file-prefix**, connect the option name and the argument with **=**.

OPTIONS

−b *file-prefix*
−−file-prefix=*file-prefix*

Specify a prefix to use for all *bison* output file names. The names are chosen as if the input file were named *file-prefix***.c**.

−d

Write an extra output file containing macro definitions for the token type names defined in the grammar and the semantic value type **YYSTYPE**, as well as a few **extern** variable declarations.

If the parser output file is named *name***.c** then this file is named *name***.h**.

This output file is essential if you wish to put the definition of **yylex** in a separate source file, because **yylex** needs to be able to refer to token type codes and the variable **yylval**.

−−defines=*defines-file*

The behavior of **−−defines** is the same than **−d** option. The only difference is that it has an optional argument which is the name of the output filename.

−g

Output a VCG definition of the LALR(1) grammar automaton computed by Bison. If the grammar file is **foo.y** , the VCG output file will be **foo.vcg**.

−−graph=*graph-file*

The behavior of **−−graph** is the same than **−g** option. The only difference is that it has an optional argument which is the name of the output graph filename.

−k
−−token-table

This switch causes the *name***.tab.c** output to include a list of token names in order by their token numbers; this is defined in the array *yytname*. Also generated are #defines for *YYNTOKENS*, *YYNNTS*, *YYNRULES*, and *YYNSTATES*.

−l
−−no-lines
Don't put any **#line** preprocessor commands in the parser file. Ordinarily *bison* puts them in the parser file so that the C compiler and debuggers will associate errors with your source file, the grammar file. This option causes them to associate errors with the parser file, treating it an independent source file in its own right.

−n
−−no-parser
Do not generate the parser code into the output; generate only declarations. The generated *name***.tab.c** file will have only constant declarations. In addition, a *name***.act** file is generated containing a switch statement body containing all the translated actions.

−o *outfile*
−−output-file=*outfile*
Specify the name *outfile* for the parser file.

The other output files' names are constructed from *outfile* as described under the **−v** and **−d** switches.

−p *prefix*
−−name-prefix=*prefix*
Rename the external symbols used in the parser so that they start with *prefix* instead of **yy**. The precise list of symbols renamed is **yyparse**, **yylex**, **yyerror**, **yylval**, **yychar**, and **yydebug**.

For example, if you use **−p c**, the names become **cparse**, **clex**, and so on.

−t
−−debug
In the parser file, define the macro **YYDEBUG** to 1 if it is not already defined, so that the debugging facilities are compiled.

−v
−−verbose
Write an extra output file containing verbose descriptions of the parser states and what is done for each type of look-ahead token in that state.

This file also describes all the conflicts, both those resolved by operator precedence and the unresolved ones.

The file's name is made by removing **.tab.c** or **.c** from the parser output file name, and adding **.output** instead.

Therefore, if the input file is **foo.y**, then the parser file is called **foo.tab.c** by default. As a consequence, the verbose output file is called **foo.output**.

−V
−−version
Print the version number of *bison* and exit.

−h
−−help Print a summary of the options to *bison* and exit.

−y
−−yacc
−−fixed-output-files
Equivalent to **−o y.tab.c**; the parser output file is called **y.tab.c**, and the other outputs are called **y.output** and **y.tab.h**. The purpose of this switch is to imitate *yacc*'s output file name conventions. Thus, the following shell script can substitute for *yacc* and is often installed as *yacc*:

bison −y "$@"

SEE ALSO

yacc(1)
The *Bison Reference Manual*, included as the file **bison.texinfo** in the *bison* source distribution.

DIAGNOSTICS

Self explanatory.

NAME

bison.yacc – GNU Project parser generator (yacc replacement)

DESCRIPTION

bison.yacc acts exacly as "bison -y" with all other parameters passed, that is the output file is called **y.tab.c** and the other outputs are called **y.output** and **y.tab.h**. This means **bison.yacc** cann be used as a substitute for *yacc*.

Please consult the bison documentation for further information.

SEE ALSO

bison(1)

AUTHOR

This manual page was written for the Debian GNU/Linux system by Robert Lemmen <robertle@semistable.com> (but may be used by others, of course)

NAME

yacc − yet another compiler-compiler

SYNOPSIS

yacc [**−vd**] grammar

DESCRIPTION

Yacc converts a context-free grammar into a set of tables for a simple automaton which executes an LR(1) parsing algorithm. The grammar may be ambiguous; specified precedence rules are used to break ambiguities.

The output file, *y.tab.c*, must be compiled by the C compiler to produce a program *yyparse*. This program must be loaded with the lexical analyzer program, *yylex*, as well as *main* and *yyerror*, an error handling routine. These routines must be supplied by the user; *Lex*(1) is useful for creating lexical analyzers usable by *yacc*.

If the **−v** flag is given, the file *y.output* is prepared, which contains a description of the parsing tables and a report on conflicts generated by ambiguities in the grammar.

If the **−d** flag is used, the file *y.tab.h* is generated with the *define* statements that associate the *yacc*-assigned 'token codes' with the user-declared 'token names'. This allows source files other than *y.tab.c* to access the token codes.

FILES

y.output	
y.tab.c	
y.tab.h	defines for token names
yacc.tmp, yacc.acts	temporary files
/usr/lib/yaccpar	parser prototype for C programs
/lib/liby.a	library with default 'main' and 'yyerror'

SEE ALSO

lex(1)
LR Parsing by A. V. Aho and S. C. Johnson, Computing Surveys, June, 1974.
YACC − Yet Another Compiler Compiler by S. C. Johnson.

DIAGNOSTICS

The number of reduce-reduce and shift-reduce conflicts is reported on the standard output; a more detailed report is found in the *y.output* file. Similarly, if some rules are not reachable from the start symbol, this is also reported.

BUGS

Because file names are fixed, at most one *yacc* process can be active in a given directory at a time.