NAME

yacc – yet another compiler-compiler

SYNOPSIS

/usr/ccs/bin/yacc [**–dltVv**] [**–b** *file_prefix*] [**–Q** [**y** | **n**]] [**–P** *parser*]
[**–p** *sym_prefix*] *file*

AVAILABILITY

SUNWbtool

DESCRIPTION

The **yacc** command converts a context-free grammar into a set of tables for a simple automaton that executes an LALR(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 function **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; the **lex**(1) command is useful for creating lexical analyzers usable by **yacc**.

OPTIONS

The following options are supported:

–b *file_prefix*
Use *file_prefix* instead of **y** as the prefix for all output files. The code file **y.tab.c**, the header file **y.tab.h** (created when **–d** is specified), and the description file **y.output** (created when **–v** is specified), will be changed to *file_prefix***.tab.c**, *file_prefix***.tab.h**, and *file_prefix***.output**, respectively.

–d Generates the file **y.tab.h** with the **#define** statements that associate the **yacc** user-assigned "token codes" with the user-declared "token names." This association allows source files other than **y.tab.c** to access the token codes.

–l Specifies that the code produced in **y.tab.c** will not contain any **#line** constructs. This option should only be used after the grammar and the associated actions are fully debugged.

–P *parser* Allows you to specify the parser of your choice instead of **/usr/ccs/bin/yaccpar**. For example, you can specify:

example% yacc –P ˜/myparser parser.y

–p *sym_prefix*
Use *sym_prefix* instead of **yy** as the prefix for all external names produced by **yacc**. The names affected include the functions **yyparse( )**, **yylex( )** and **yyerror( )**, and the variables *yylval*, *yychar* and *yydebug*. (In the remainder of this section, the six symbols cited are referenced using their default names only as a notational convenience.) Local names may also be affected by the **–p** option; however, the **–p** option does not affect **#define** symbols generated by **yacc**.

-Q[y|n] The **–Qy** option puts the version stamping information in **y.tab.c**. This allows you to know what version of **yacc** built the file. The **–Qn** option (the default) writes no version information.

–t Compiles runtime debugging code by default. Runtime debugging code is always generated in **y.tab.c** under conditional compilation control. By default, this code is not included when **y.tab.c** is compiled. Whether or not the −**t** option is used, the runtime debugging code is under the control of **YYDEBUG** , a preprocessor symbol. If **YYDEBUG** has a non-zero value, then the debugging code is included. If its value is **0**, then the code will not be included. The size and execution time of a program produced without the runtime debugging code will be smaller and slightly faster.

−**V** Prints on the standard error output the version information for **yacc**.

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

OPERANDS

The following operand is required:

file A path name of a file containing instructions for which a parser is to be created.

EXAMPLES

Access to the **yacc** library is obtained with library search operands to **cc**. To use the **yacc** library **main**,

```
example% cc y.tab.c -ly
```

Both the **lex** library and the **yacc** library contain **main**. To access the **yacc main**,

```
example% cc y.tab.c lex.yy.c -ly -ll
```

This ensures that the **yacc** library is searched first, so that its **main** is used.

The historical **yacc** libraries have contained two simple functions that are normally coded by the application programmer. These library functions are similar to the following code:

```
#include <locale.h>
int main(void)
{
      extern int yyparse();

      setlocale(LC_ALL, "");

      /* If the following parser is one created by lex, the
         application must be careful to ensure that LC_CTYPE
         and LC_COLLATE are set to the POSIX locale.  */
      (void) yyparse();
      return (0);
}

#include <stdio.h>

int yyerror(const char *msg)
{
      (void) fprintf(stderr, "%s\n", msg);
      return (0);
```

}

ENVIRONMENT

See **environ**(5) for descriptions of the following environment variables that affect the execution of **yacc**: **LC_CTYPE**, **LC_MESSAGES**, and **NLSPATH**.

yacc can handle characters from EUC primary and supplementary codesets as one-token symbols. EUC codes may only be single character quoted terminal symbols. **yacc** expects **yylex( )** to return a wide character (**wchar_t**) value for these one-token symbols.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

FILES

y.output	state transitions of the generated parser
y.tab.c	source code of the generated parser
y.tab.h	header file for the generated parser
yacc.acts	temporary file
yacc.debug	temporary file
yacc.tmp	temporary file
yaccpar	parser prototype for C programs

SEE ALSO

cc(1B), **lex**(1), **environ**(5) and the **yacc** chapter in the *Programming Utilities Guide* manual.

DIAGNOSTICS

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

NOTES

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