Bison Quick Reference

Starting Bison

To use Bison, type: `bison` *filename*

Options can be used as: `bison` *options filename*

Command Line Options

Display usage information.	`-h`
Display version information.	`-V`
Generate token and `YYSTYPE` definitions.	`-d`
Prepend a prefix to external symbols.	`-p` *prefix*
Don't put `#line` directives in the parser.	`-l`
Specify the output file.	`-o` *filename*
Debug or *trace* mode.	`-t`
Verbose description of the parser.	`-v`
Emulate `yacc` (generate `y.tab.*` files).	`-y`

Note: The token and `YYSTYPE` definitions are generated to a file called `y.tab.h` if the `-y` option is used, otherwise it will have the format *name*`.tab.h`, where *name* is the leading part of the parser definition filename.

Definitions

Declare a terminal symbol.	`%token <`*t*`>` *n*
Declare a terminal symbol, and define its association.	*association* `<`*t*`>` *n*
Generate a reentrant (pure) parser.	`%pure_parser`
Define the union of all data types used in the parser.	`%union{`*field list*`}`
Tell `bison` where to start parsing.	`%start` *m*
Tell `bison` the data type of symbols.	`%type <`*t*`>` *s1*...*sn*

In the above, *t* is a *type* defined in the `%union` definition, *n* is a *terminal* symbol name, *m* is a *non-terminal* symbol name, and *association* can be one of `%left`, `%right`, or `%nonassoc`.

The `<`*t*`>` after `%token, %left, %right` and `%nonassoc` is optional. Additionally, precedence may be overridden in the rules with `%prec` commands.

Parser Definition Files

The general form for a parser definition is:

```
{%
   /* Initial C code.  */
%}
  Token and type definitions
%%
   Rule definition 1
          ⋮
   Rule definition  n
%%
  /* Other C code.  */
```

Rule definitions

Rules take the form:

```
non-terminal :  statement 1
             |  statement 2
                   ⋮
             |  statement n
             ;
```

Where *statements* can be either empty, or contain a mixture of C code (enclosed in `{...}`), and the symbols that make up the non-terminal. For example:

```
expression : number '+' number { $$ = $1 + $3 }
           | number '−' number { $$ = $1 − $3 }
           | number '/' number { $$ = $1 / $3 }
           | number '*' number { $$ = $1 * $3 }
           ;
```

For altering the precedence of a symbol use:

```
%prec name
```

For example:

```
foo : gnu bar gnu      %prec bar
    ;
```

Grammar Variables and Symbols

Recognize an error & continue parsing.	`error`
Access data associated with a symbol.	`$$, $0`...`$`n
Access a field of the data union.	`$$.`t`, $0.`t`...$`n`.`t
Access symbol's location.	`@$, @0`...`@`n
Access data's line location.	`@`x`.`*line_spec*
Access data's column location.	`@`x`.`*column_spec*

Where t is a type defined in the `%union`, n is a number, x is a number or \$, *line_spec* one of `first_line` and `last_line`, and *column_spec* is specified as either `first_column` or `last_column`.

Variables and Types

Current look ahead token.	`yychar`
Debug mode flag.	`yydebug`
Data associated with the current token.	`yylval`
Source position of current token.	`yylloc`
Number of errors encountered.	`yynerrs`
Position information type.	`YYLTYPE`
Data type associated with symbols.	`YYSTYPE`

Functions

User defined error handler.	`int yyerror(char *)`
User defined lexical analyzer.	`int yylex()`
The grammar parser.	`int yyparse()`

Macros

Quit parsing immediately. Return 1.	`YYABORT`
Quit parsing immediately. Return 0.	`YYACCEPT`
Pretend a syntax error occurred.	`YYERROR`
Value in `yychar` if no look-ahead token.	`YYEMPTY`
Clear previous look ahead token.	`yyclearin`
Recover normally from an error.	`yyerrok`

Copyright © 2001 Free Software Foundation, Inc.

November 2001, Bison Refcard Version 0.5
designed by Brendan Kehoe and Gavin Nicol
for Bison 1.31

Permission is granted to make and distribute copies of this card provided the copyright notice and this permission notice are preserved on all copies.

For information, write to the:
Free Software Foundation, Inc.
59 Temple Place - Suite 330
Boston, MA 02111-1307 USA