Using the Error Token in Bison

By: Robert Heckendorn -- Computer Science Department -- University of Idaho

Here is various information on how Bison handles errors and how you can use the **error** token to control how errors are processed.

The Error Token

When an error occurs a **predefined** token **error** (token number 256) is generated. If your grammar handles the token **error** then you can program your grammar to have error recovery. For example:

```
stmts : /*empty*/
     | stmts '\n'
     | stmts exp '\n'
     | stmts error '\n'  <- error followed by a newline is a stmt
     ;
```

Error recovery is handled by messing with the state machine in order to keep it running. Specifically, Bison handles errors by this process in this order:

1. Discard terminals and nonterminals plus state off the parse stack until it finds a place where the error token is allowed in the current state.
2. The error token is shifted on.
3. Discard input tokens until an acceptable input token is found based on the parse stack including the error token.
4. To prevent cascades of errors only after 3 new tokens are read will the error messaging be turned back on. **yyerrok** statement turns on error messaging immediately indicating you have handled the error. In particular the tokens on input are not discarded (See infinite loop).

Remember that Bison is an LALR parser so the lookahead token is not removed from input until it is ready to be put on the stack or discarded by the error routine. The lookahead token is in **yychar**. If you want to throw it away then you use the **yyclearin** macro.

Examples of Bison Error Processing

A simple grammar with resynchronizing token: semicolon

Consider this grammar

```
%token YY ZZ
%%
slist : slist stmt ';' { printf("slist stmt\n"); }
      | stmt ';'       { printf("stmt\n"); }
      | error ';'      { printf("ERROR!!!\n");  yyerrok; }
      ;

stmt  : ZZ stmt
      | ZZ
      ;
%%
```

Which produces this state machine:

```
Rules:
    0 $accept: slist $end
    1 slist: slist stmt ';'
    2      | stmt ';'
    3      | error ';'                  <---- this is the addition
    4 stmt: ZZ stmt
    5     | ZZ

State Machine:
state 0
    0 $accept: . slist $end

    error  shift, and go to state 1     <---- jump to new state
    ZZ     shift, and go to state 2
    slist  go to state 3
    stmt   go to state 4

state 1
    3 slist: error . ';'                <---- new state to handle error

    ';'  shift, and go to state 5

state 2
    4 stmt: ZZ . stmt
    5     | ZZ .

    ZZ  shift, and go to state 2
    $default  reduce using rule 5 (stmt)  <--- Note that invalid token will do a reduce
    stmt  go to state 6

state 3
    0 $accept: slist . $end
    1 slist: slist . stmt ';'
    $end  shift, and go to state 7
    ZZ    shift, and go to state 2
    stmt  go to state 8

state 4
    2 slist: stmt . ';'

    ';'  shift, and go to state 9

state 5
    3 slist: error ';' .              <---- new state to handle error

    $default  reduce using rule 3 (slist)

state 6
    4 stmt: ZZ stmt .

    $default  reduce using rule 4 (stmt)

state 7
    0 $accept: slist $end .

    $default  accept

state 8
    1 slist: slist stmt . ';'

    ';'  shift, and go to state 10

state 9
    2 slist: stmt ';' .

    $default  reduce using rule 2 (slist)

state 10
    1 slist: slist stmt ';' .

    $default  reduce using rule 1 (slist)
```

Consider the input `zz zz yy zz zz ;` which has an error in the middle. We expect Bison to shift and reduce the initial `zz`'s and then arrive and the bad token `yy`. Then put on an **error** token until we get to the **;**. It effectively does that. Note: each version of Bison seems to generate different debug output but the actions are the same.

```
Starting parse
Entering state 0
Reading a token: zz zz yy zz zz ;
Next token is 258 (ZZ)
Shifting token 258 (ZZ), Entering state 2
state stack now 0 2
Reading a token: Next token is 258 (ZZ)
Shifting token 258 (ZZ), Entering state 2
state stack now 0 2 2
Reading a token: Next token is 257 (YY)
Reducing via rule 5 (line 34), ZZ  -> stmt  <-- use default on invalid token
state stack now 0 2
Entering state 6
state stack now 0 2 6
Reducing via rule 4 (line 33), ZZ stmt  -> stmt
state stack now 0
Entering state 4
state stack now 0 4
Next token is 257 (YY)
ERROR lineno(1):parse error, expecting `';''.  I got: yy
Error: state stack now 0                    <--- pop off until and error token on input works
Shifting error token, Entering state 1
state stack now 0 1
Next token is 257 (YY)
Discarding token 257 (YY).          <--- discard from input until putting a error will work
Error: state stack now 0
Shifting error token, Entering state 1
Error: state stack now 0 1
Reading a token: Next token is 258 (ZZ)  <--- doesn't accept "error ZZ"
Discarding token 258 (ZZ).
Error: state stack now 0
Shifting error token, Entering state 1
Error: state stack now 0 1
Reading a token: Next token is 258 (ZZ)  <--- doesn't accept "error ZZ"
Discarding token 258 (ZZ).
Error: state stack now 0
Shifting error token, Entering state 1
Error: state stack now 0 1
Reading a token: Next token is 59 (';')
Shifting token 59 (';'), Entering state 5 <---  state 1 + ';' works!
Reducing via rule 3 (line 30), error ';'  -> slist
ERROR!!!
state stack now 0
Entering state 3
```

Notice the matched production is the error production and the print statement is executed giving the line "error".

A simple grammar with and without yyerrok

What follows is a side by side comparison (using UNIX sdiff) of the above code run on this input:

```
zz ;
yy ;
yy ;
yy ;
yy ;
yy ;
yy ;
```

The results on the left are with **yyerrok** and the results on the right are without. Since the **yyerrok** is saying we know when we are at a semicolon we are in sync the extra errors are in that sense "correct". The lack of errors on the right are because the 3 token count for error reporting restarts with each error processed. Notice the same number of errors are processed on both sides.

```
Starting parse                                                          Starting parse
Entering state 0                                                        Entering state 0
Reading a token: Next token is 258 (ZZ)                                 Reading a token: Next token is 258 (ZZ)
Shifting token 258 (ZZ), Entering state 2                               Shifting token 258 (ZZ), Entering state 2
Reading a token: Next token is 59 (';')                                 Reading a token: Next token is 59 (';')
Reducing via rule 5 (line 34), ZZ  -> stmt                              Reducing via rule 5 (line 34), ZZ  -> stmt
state stack now 0                                                       state stack now 0
Entering state 4                                                        Entering state 4
Next token is 59 (';')                                                  Next token is 59 (';')
Shifting token 59 (';'), Entering state 8                               Shifting token 59 (';'), Entering state 8
Reducing via rule 2 (line 29), stmt ';'  -> slist                       Reducing via rule 2 (line 29), stmt ';'  -> slist
stmt                                                                    stmt
state stack now 0                                                       state stack now 0
Entering state 3                                                        Entering state 3
Reading a token: Next token is 257 (YY)                                 Reading a token: Next token is 257 (YY)
ERROR lineno(1):parse error, expecting `$' or `ZZ'.  I got: yy          ERROR lineno(1):parse error, expecting `$' or `ZZ'.  I got: yy
Error: state stack now 0                                                Error: state stack now 0
Shifting error token, Entering state 1                                  Shifting error token, Entering state 1
Next token is 257 (YY)                                                  Next token is 257 (YY)
Discarding token 257 (YY).                                              Discarding token 257 (YY).
Error: state stack now 0                                                Error: state stack now 0
Shifting error token, Entering state 1                                  Shifting error token, Entering state 1
Reading a token: Next token is 59 (';')                                 Reading a token: Next token is 59 (';')
Shifting token 59 (';'), Entering state 5                               Shifting token 59 (';'), Entering state 5
Reducing via rule 3 (line 30), error ';'  -> slist                      Reducing via rule 3 (line 30), error ';'  -> slist
ERROR!!!                                                                ERROR!!!
state stack now 0                                                       state stack now 0
Entering state 3                                                        Entering state 3
Reading a token: Next token is 257 (YY)                                 Reading a token: Next token is 257 (YY)
ERROR lineno(1):parse error, expecting `$' or `ZZ'.  I got: yy       <
Error: state stack now 0                                                Error: state stack now 0
Shifting error token, Entering state 1                                  Shifting error token, Entering state 1
Next token is 257 (YY)                                                  Next token is 257 (YY)
Discarding token 257 (YY).                                              Discarding token 257 (YY).
Error: state stack now 0                                                Error: state stack now 0
Shifting error token, Entering state 1                                  Shifting error token, Entering state 1
Reading a token: Next token is 59 (';')                                 Reading a token: Next token is 59 (';')
Shifting token 59 (';'), Entering state 5                               Shifting token 59 (';'), Entering state 5
Reducing via rule 3 (line 30), error ';'  -> slist                      Reducing via rule 3 (line 30), error ';'  -> slist
ERROR!!!                                                                ERROR!!!
state stack now 0                                                       state stack now 0
Entering state 3                                                        Entering state 3
Reading a token: Next token is 257 (YY)                                 Reading a token: Next token is 257 (YY)
ERROR lineno(1):parse error, expecting `$' or `ZZ'.  I got: yy       <
Error: state stack now 0                                                Error: state stack now 0
Shifting error token, Entering state 1                                  Shifting error token, Entering state 1
Next token is 257 (YY)                                                  Next token is 257 (YY)
Discarding token 257 (YY).                                              Discarding token 257 (YY).
Error: state stack now 0                                                Error: state stack now 0
Shifting error token, Entering state 1                                  Shifting error token, Entering state 1
Reading a token: Next token is 59 (';')                                 Reading a token: Next token is 59 (';')
Shifting token 59 (';'), Entering state 5                               Shifting token 59 (';'), Entering state 5
Reducing via rule 3 (line 30), error ';'  -> slist                      Reducing via rule 3 (line 30), error ';'  -> slist
ERROR!!!                                                                ERROR!!!
state stack now 0                                                       state stack now 0
Entering state 3                                                        Entering state 3
Reading a token: Next token is 257 (YY)                                 Reading a token: Next token is 257 (YY)
ERROR lineno(1):parse error, expecting `$' or `ZZ'.  I got: yy       <
Error: state stack now 0                                                Error: state stack now 0
Shifting error token, Entering state 1                                  Shifting error token, Entering state 1
Next token is 257 (YY)                                                  Next token is 257 (YY)
Discarding token 257 (YY).                                              Discarding token 257 (YY).
Error: state stack now 0                                                Error: state stack now 0
Shifting error token, Entering state 1                                  Shifting error token, Entering state 1
Reading a token: Next token is 59 (';')                                 Reading a token: Next token is 59 (';')
Shifting token 59 (';'), Entering state 5                               Shifting token 59 (';'), Entering state 5
Reducing via rule 3 (line 30), error ';'  -> slist                      Reducing via rule 3 (line 30), error ';'  -> slist
ERROR!!!                                                                ERROR!!!
state stack now 0                                                       state stack now 0
Entering state 3                                                        Entering state 3
Reading a token: Next token is 257 (YY)                                 Reading a token: Next token is 257 (YY)
ERROR lineno(1):parse error, expecting `$' or `ZZ'.  I got: yy       <
Error: state stack now 0                                                Error: state stack now 0
Shifting error token, Entering state 1                                  Shifting error token, Entering state 1
Next token is 257 (YY)                                                  Next token is 257 (YY)
Discarding token 257 (YY).                                              Discarding token 257 (YY).
Error: state stack now 0                                                Error: state stack now 0
```

```
Shifting error token, Entering state 1                                  Shifting error token, Entering state 1
Reading a token: Next token is 59 (';')                                 Reading a token: Next token is 59 (';')
Shifting token 59 (';'), Entering state 5                               Shifting token 59 (';'), Entering state 5
Reducing via rule 3 (line 30), error ';'  -> slist                      Reducing via rule 3 (line 30), error ';'  -> slist
ERROR!!!                                                                ERROR!!!
state stack now 0                                                       state stack now 0
Entering state 3                                                        Entering state 3
Reading a token: Next token is 257 (YY)                                 Reading a token: Next token is 257 (YY)
ERROR lineno(1):parse error, expecting `$' or `ZZ'.  I got: yy       <
Error: state stack now 0                                                Error: state stack now 0
Shifting error token, Entering state 1                                  Shifting error token, Entering state 1
Next token is 257 (YY)                                                  Next token is 257 (YY)
Discarding token 257 (YY).                                              Discarding token 257 (YY).
Error: state stack now 0                                                Error: state stack now 0
Shifting error token, Entering state 1                                  Shifting error token, Entering state 1
Reading a token: Next token is 59 (';')                                 Reading a token: Next token is 59 (';')
Shifting token 59 (';'), Entering state 5                               Shifting token 59 (';'), Entering state 5
Reducing via rule 3 (line 30), error ';'  -> slist                      Reducing via rule 3 (line 30), error ';'  -> slist
ERROR!!!                                                                ERROR!!!
state stack now 0                                                       state stack now 0
Entering state 3                                                        Entering state 3
Reading a token: Now at end of input.                                   Reading a token: Now at end of input.
Shifting token 0 ($), Entering state 10                                 Shifting token 0 ($), Entering state 10
Now at end of input.                                                    Now at end of input.
```

Notice that the **only difference** is the presence of reported errors.

The long wait

Consider this grammar:

```
%token YY ZZ
%%
slist : slist stmt ';' { printf("stmt\n"); }
      | stmt ';'       { printf("stmt\n"); }
      ;

stmt  :  ZZ
      | '(' stmt ')'
      | '(' error ')'
      ;
%%
```

and this input:

```
zz ;
( zz ) ;
( zz ;
zz ;
zz ;
zz );
zz ;
```

you get this result:

```
Starting parse
Entering state 0
Reading a token: Next token is 258 (ZZ)
Shifting token 258 (ZZ), Entering state 1
Reducing via rule 3 (line 32), ZZ  -> stmt
state stack now 0
Entering state 4
Reading a token: Next token is 59 (';')
Shifting token 59 (';'), Entering state 8
Reducing via rule 2 (line 29), stmt ';'  -> slist
stmt
state stack now 0
Entering state 3
Reading a token: Next token is 40 ('(')
Shifting token 40 ('('), Entering state 2
Reading a token: Next token is 258 (ZZ)
Shifting token 258 (ZZ), Entering state 1
Reducing via rule 3 (line 32), ZZ  -> stmt
state stack now 0 3 2
Entering state 6
Reading a token: Next token is 41 (')')
Shifting token 41 (')'), Entering state 10
Reducing via rule 4 (line 33), '(' stmt ')'  -> stmt
state stack now 0 3
Entering state 7
Reading a token: Next token is 59 (';')
Shifting token 59 (';'), Entering state 11
Reducing via rule 1 (line 28), slist stmt ';'  -> slist
stmt
state stack now 0
Entering state 3
Reading a token: Next token is 40 ('(')
Shifting token 40 ('('), Entering state 2
Reading a token: Next token is 258 (ZZ)
Shifting token 258 (ZZ), Entering state 1
Reducing via rule 3 (line 32), ZZ  -> stmt
state stack now 0 3 2
Entering state 6
Reading a token: Next token is 59 (';')
ERROR lineno(1):parse error, expecting `')''.  I got: ;
Error: state stack now 0 3 2
Shifting error token, Entering state 5 <--- SHIFT ERROR TOKEN AND WAIT FOR ')'
Next token is 59 (';')
Discarding token 59 (';').
Error: state stack now 0 3 2
Shifting error token, Entering state 5
Reading a token: Next token is 258 (ZZ)
Discarding token 258 (ZZ).
Error: state stack now 0 3 2
Shifting error token, Entering state 5
Reading a token: Next token is 59 (';')
Discarding token 59 (';').
Error: state stack now 0 3 2
Shifting error token, Entering state 5
Reading a token: Next token is 258 (ZZ)
Discarding token 258 (ZZ).
Error: state stack now 0 3 2
Shifting error token, Entering state 5
Reading a token: Next token is 59 (';')
Discarding token 59 (';').
Error: state stack now 0 3 2
Shifting error token, Entering state 5
Reading a token: Next token is 258 (ZZ)
Discarding token 258 (ZZ).
Error: state stack now 0 3 2
Shifting error token, Entering state 5
Reading a token: Next token is 41 (')')
Shifting token 41 (')'), Entering state 9
Reducing via rule 5 (line 34), '(' error ')'  -> stmt
state stack now 0 3
Entering state 7
Reading a token: Next token is 59 (';')
Shifting token 59 (';'), Entering state 11
Reducing via rule 1 (line 28), slist stmt ';'  -> slist
stmt
state stack now 0
Entering state 3
Reading a token: Next token is 258 (ZZ)
Shifting token 258 (ZZ), Entering state 1
Reducing via rule 3 (line 32), ZZ  -> stmt
state stack now 0 3
Entering state 7
Reading a token: Next token is 59 (';')
Shifting token 59 (';'), Entering state 11
Reducing via rule 1 (line 28), slist stmt ';'  -> slist
stmt
state stack now 0
Entering state 3
Reading a token: Now at end of input.
Shifting token 0 ($), Entering state 12
Now at end of input.
```

Notice how this grammar is forced to wait for the closing ')' throwing away useful statements.

Insufficient coverage

If the input is now:

```
zz );
zz ;
```

You aren't covered if there is an error not between parentheses

```
Starting parse
Entering state 0
Reading a token: Next token is 258 (ZZ)
Shifting token 258 (ZZ), Entering state 1
Reducing via rule 3 (line 32), ZZ  -> stmt
state stack now 0
Entering state 4
Reading a token: Next token is 41 (')')
ERROR lineno(1):parse error, expecting `';''.  I got: )
Error: state stack now 0
```

The error on the closing parenthesis is the last thing the parser does.

Resetting on a nonsynchronizing token

In this simple grammar we deal with a optional list of elements and resync when we see the next reliable element in the list (in this case `ZZ`.

```
%token YY ZZ
%%
seq   : /*empty*/
      | seq ZZ        { yyerrok; }
      | seq error
      ;
%%
```

The left code produces both errors on an input of `yy zz yy` while the right code syncs but does clear the error count.

```
Starting parse                                                          Starting parse
Entering state 0                                                        Entering state 0
Reducing via rule 1 (line 28),  -> seq                                  Reducing via rule 1 (line 28),  -> seq
state stack now 0                                                       state stack now 0
Entering state 1                                                        Entering state 1
Reading a token: yy zz yy                                               Reading a token: yy zz yy
Next token is 257 (YY)                                                  Next token is 257 (YY)
ERROR lineno(1):parse error, expecting `$' or `error' or `ZZ'.  I       ERROR lineno(1):parse error, expecting `$' or `error' or `ZZ'.  I
Shifting error token, Entering state 2                                  Shifting error token, Entering state 2
Reducing via rule 3 (line 30), seq error  -> seq                        Reducing via rule 3 (line 30), seq error  -> seq
state stack now 0                                                       state stack now 0
Entering state 1                                                        Entering state 1
Next token is 257 (YY)                                                  Next token is 257 (YY)
Discarding token 257 (YY).                                              Discarding token 257 (YY).
Shifting error token, Entering state 2                                  Shifting error token, Entering state 2
Reducing via rule 3 (line 30), seq error  -> seq                        Reducing via rule 3 (line 30), seq error  -> seq
state stack now 0                                                       state stack now 0
Entering state 1                                                        Entering state 1
Reading a token: Next token is 258 (ZZ)                                 Reading a token: Next token is 258 (ZZ)
Shifting token 258 (ZZ), Entering state 3                               Shifting token 258 (ZZ), Entering state 3
Reducing via rule 2 (line 29), seq ZZ  -> seq                           Reducing via rule 2 (line 29), seq ZZ  -> seq
state stack now 0                                                       state stack now 0
Entering state 1                                                        Entering state 1
Reading a token: Next token is 257 (YY)                                 Reading a token: Next token is 257 (YY)
ERROR lineno(1):parse error, expecting `$' or `error' or `ZZ'.  I    <
Shifting error token, Entering state 2                                  Shifting error token, Entering state 2
Reducing via rule 3 (line 30), seq error  -> seq                        Reducing via rule 3 (line 30), seq error  -> seq
state stack now 0                                                       state stack now 0
Entering state 1                                                        Entering state 1
Next token is 257 (YY)                                                  Next token is 257 (YY)
Discarding token 257 (YY).                                              Discarding token 257 (YY).
Shifting error token, Entering state 2                                  Shifting error token, Entering state 2
Reducing via rule 3 (line 30), seq error  -> seq                        Reducing via rule 3 (line 30), seq error  -> seq
state stack now 0                                                       state stack now 0
Entering state 1                                                        Entering state 1
```

Fixation on a lower level error

In this experiment we try to correct the problems we have seen where an error is not trapped or where we wait for a closing parenthesis. We are only partially successful. Here is the code:

```
%token YY ZZ
%%
slist : slist stmt ';' { printf("** stmt\n"); }
      | stmt ';'      { printf("** stmt\n"); }
      | error ';'     { printf("** error stmt\n");}
      ;

stmt  :  ZZ
      | '(' stmt ')'
      | '(' error ')'  { printf("** nested error\n"); }
      ;
%%
```

The input is:

```
zz ;
( zz ;
zz ;
) ;
zz );
zz ;
```

In the output we see that when it finds an `( zz ;` and commits to finding `( error )` . So again we wait for the ')' even though if we popped another token (namely '(') off the parse stack we could match the statement level error message. However we no longer die on **zz);**

```
Starting parse
Entering state 0
Reading a token: Next token is 258 (ZZ)
Shifting token 258 (ZZ), Entering state 2
Reducing via rule 4 (line 33), ZZ  -> stmt
state stack now 0
Entering state 5
Reading a token: Next token is 59 (';')
Shifting token 59 (';'), Entering state 10
Reducing via rule 2 (line 29), stmt ';'  -> slist
** stmt
state stack now 0
Entering state 4
Reading a token: Next token is 40 ('(')
Shifting token 40 ('('), Entering state 3
Reading a token: Next token is 258 (ZZ)
Shifting token 258 (ZZ), Entering state 2
Reducing via rule 4 (line 33), ZZ  -> stmt
state stack now 0 4 3
Entering state 8
Reading a token: Next token is 59 (';')
ERROR lineno(1):parse error, expecting `')''.  I got: ;
Error: state stack now 0 4 3
Shifting error token, Entering state 7
Next token is 59 (';')
Discarding token 59 (';').
Error: state stack now 0 4 3
Shifting error token, Entering state 7
Reading a token: Next token is 258 (ZZ)
Discarding token 258 (ZZ).
Error: state stack now 0 4 3
Shifting error token, Entering state 7
Reading a token: Next token is 59 (';')
Discarding token 59 (';').
Error: state stack now 0 4 3
Shifting error token, Entering state 7
Reading a token: Next token is 41 (')')
Shifting token 41 (')'), Entering state 11
Reducing via rule 6 (line 35), '(' error ')'  -> stmt
** nested error
state stack now 0 4
Entering state 9
Reading a token: Next token is 59 (';')
Shifting token 59 (';'), Entering state 13
Reducing via rule 1 (line 28), slist stmt ';'  -> slist
** stmt
state stack now 0
Entering state 4
Reading a token: Next token is 258 (ZZ)
Shifting token 258 (ZZ), Entering state 2
Reducing via rule 4 (line 33), ZZ  -> stmt
state stack now 0 4
Entering state 9
Reading a token: Next token is 41 (')')
ERROR lineno(1):parse error, expecting `';''.  I got: )
Error: state stack now 0 4
Error: state stack now 0
Shifting error token, Entering state 1
Next token is 41 (')')
Discarding token 41 (')').
Error: state stack now 0
Shifting error token, Entering state 1
Reading a token: Next token is 59 (';')
Shifting token 59 (';'), Entering state 6
Reducing via rule 3 (line 30), error ';'  -> slist
** error stmt
state stack now 0
Entering state 4
Reading a token: Next token is 258 (ZZ)
Shifting token 258 (ZZ), Entering state 2
Reducing via rule 4 (line 33), ZZ  -> stmt
state stack now 0 4
Entering state 9
Reading a token: Next token is 59 (';')
Shifting token 59 (';'), Entering state 13
Reducing via rule 1 (line 28), slist stmt ';'  -> slist
** stmt
state stack now 0
Entering state 4
Reading a token: Now at end of input.
Shifting token 0 ($), Entering state 14
Now at end of input.
```

Infinite Loop Example

In this case we have an infinite look because yyerrok will cause the Bison error scanning to prematurely leave the error processing loop.

The code

```
%token YY ZZ
%%
slist : slist stmt ';' { printf("** stmt\n"); }
      | stmt ';'       { printf("** stmt\n"); }
;

stmt  : ZZ
      | error          { yyerrok; }    <-- abort error process when error found
;
%%
```

The input is simply `yy`.

On the left with the call to **yyerrok** we find an **infinite loop**! On the right **without yyerrok** we find the input token is discarded and the parser moves on to the next statement. However, the error count is set and an error immediately following will be missed as in `yy ; yy ;` I find the response on the left to look very much like a bug to me. It should never let me build an infinite loop in the parser.

```
Starting parse                                                   Starting parse
Entering state 0                                                 Entering state 0
Reading a token: yy                                              Reading a token: yy
Next token is 257 (YY)                                           Next token is 257 (YY)
ERROR lineno(1):parse error, expecting `error' or `ZZ'.  I got: yy      ERROR lineno(1):parse error, expecting `error' or `ZZ'.  I got: y
Shifting error token, Entering state 1                           Shifting error token, Entering state 1
```

```
Reducing via rule 4 (line 33), error  -> stmt                           Reducing via rule 4 (line 33), error  -> stmt
state stack now 0                                                       state stack now 0
Entering state 4                                                        Entering state 4
Next token is 257 (YY)                                                  Next token is 257 (YY)
ERROR lineno(1):parse error, expecting `';''.  I got: yy             |  Discarding token 257 (YY).
Error: state stack now 0                                                Error: state stack now 0
Shifting error token, Entering state 1                                  Shifting error token, Entering state 1
Reducing via rule 4 (line 33), error  -> stmt                           Reducing via rule 4 (line 33), error  -> stmt
state stack now 0                                                       state stack now 0
Entering state 4                                                        Entering state 4
Next token is 257 (YY)                                               <
ERROR lineno(1):parse error, expecting `';''.  I got: yy             <
Error: state stack now 0                                             <
Shifting error token, Entering state 1                               <
Reducing via rule 4 (line 33), error  -> stmt                        <
state stack now 0                                                    <
Entering state 4                                                     <
Next token is 257 (YY)                                               <
ERROR lineno(1):parse error, expecting `';''.  I got: yy             <
Error: state stack now 0                                             <
Shifting error token, Entering state 1                               <
Reducing via rule 4 (line 33), error  -> stmt                        <
state stack now 0                                                    <
Entering state 4                                                     <
Next token is 257 (YY)                                               <
ERROR lineno(1):parse error, expecting `';''.  I got: yy             <
Error: state stack now 0                                             <
Shifting error token, Entering state 1                               <
Reducing via rule 4 (line 33), error  -> stmt                        <
state stack now 0                                                    <
     .                                                               <
     .                                                               <
     .                                                               <
                                                                     <
  forever!                                                           <
```

The fix is to put the **yyerrok** with the synchronizing token as in:

```
%token YY ZZ
%%
slist : slist stmt ';' { printf("** stmt\n"); }
      | stmt ';'       { yyerrok; printf("** stmt\n"); }
;

stmt  : ZZ
      | error
;
%%
```

References

- A good brief description of Bison error token usage can be found in the bison manual: The Bison Manual (2002 version of the book) for version 1.35.
- Also information can be found our book: Compiler Construction: Principles and Practice by Kenneth Louden, ISBN: 0534939724, Published by Brooks and Cole. Pages 247-250.

Robert Heckendorn Up One Level Last updated: Mar 22, 2006 22:27