The yyerrok function

Last Updated: 2021-04-08

When **yyparse()** discovers ungrammatical input, it calls **yyerror()**. It also sets a flag saying that it is now in an *error state*. **yyparse()** stays in this error state until it sees three consecutive tokens that make sense (that is, are not part of the error).

It is possible for **yyparse()** to leave the error state as soon as it finds one or two tokens that make sense; however, experience has shown that this is not enough to be sure that the error has really passed; one or two tokens being correct may just be a coincidence. If **yyparse()** leaves its error state quickly and then finds more erroneous input, it raises another error, calls **yyerror()** again to issue a new error message, and so on. In other words, it behaves as if it had found a brand new error, even though it is likely just a continuation of the old error. Waiting for three good tokens prevents a lot of error messages arising from a single error.

There are, however, times when you want **yyparse()** to leave the error state before it finds the three good tokens. To do this, invoke the macro **yyerrok**, as in:

```
yyerrok;
```

In effect, **yyerrok** says, ‘‘The old error is finished. If something else goes wrong, it is to be regarded as a new error.''
This should help you understand the rule:

```
program : program error '\n' { yyerrok; }
```

in the desk calculator program. Once **yyparse()** has found the newline that ends an erroneous input line, you want to leave the error state. Any errors on the line should be regarded as *closed*. If the next line also contains errors, you want to see a new error message produced.

Parent topic:

→ Error handling in yacc

Previous
Using yyerror()

Next
Other error handling facilities