For example, suppose that the following directive appears at the beginning of the file `foo.c`:

```
#line 10 "bar.c"
```

Let's say that the compiler detects an error on line 5 of `foo.c`. The error message will refer to line 13 of file `bar.c`, not line 5 of file `foo.c`. (Why line 13? The directive occupies line 1 of `foo.c`, so the renumbering of `foo.c` begins at line 2, which is treated as line 10 of `bar.c`.)

At first glance, the `#line` directive is mystifying. Why would we want error messages to refer to a different line and possibly a different file? Wouldn't this make programs harder to debug?

In fact, the `#line` directive isn't used very often by programmers. Instead, it's used primarily by programs that generate C code as output. The most famous example of such a program is `yacc` (Yet Another Compiler-Compiler), a UNIX utility that automatically generates part of a compiler. (The GNU version of `yacc` is named `bison`.) Before using `yacc`, the programmer prepares a file that contains information for `yacc` as well as fragments of C code. From this file, `yacc` generates a C program, `y.tab.c`, that incorporates the code supplied by the programmer. The programmer then compiles `y.tab.c` in the usual way. By inserting `#line` directives in `y.tab.c`, `yacc` tricks the compiler into believing that the code comes from the original file—the one written by the programmer. As a result, any error messages produced during the compilation of `y.tab.c` will refer to lines in the original file, not lines in `y.tab.c`. This makes debugging easier, because error messages refer to the file written by the programmer, not the (more complicated) file generated by `yacc`.

The #pragma Directive

The `#pragma` directive provides a way to request special behavior from the compiler. This directive is most useful for programs that are unusually large or that need to take advantage of the capabilities of a particular compiler.

The `#pragma` directive has the form

#pragma directive

```
#pragma tokens
```

where *tokens* are arbitrary tokens. `#pragma` directives can be very simple (a single token) or they can be much more elaborate:

```
#pragma data(heap_size => 1000, stack_size => 2000)
```

Not surprisingly, the set of commands that can appear in `#pragma` directives is different for each compiler; you'll have to consult the documentation for your compiler to see which commands it allows and what those commands do. Incidentally, the preprocessor must ignore any `#pragma` directive that contains an unrecognized command; it's not permitted to give an error message.