minate. In C89, the value returned to the operating system is undefined. In C99, if `main` is declared to return an `int` (as in our examples), the program returns 0 to the operating system; otherwise, the program returns an unspecified value.

C99

Q: Does the compiler remove a comment entirely or replace it with blank space?

A: Some old C compilers deleted all the characters in each comment, making it possible to write

```
a/**/b = 0;
```

and have the compiler interpret it as

```
ab = 0;
```

According to the C standard, however, the compiler must replace each comment by a single space character, so this trick doesn't work. Instead, we'd end up with the following (illegal) statement:

```
a b = 0;
```

Q: How can I tell if my program has an unterminated comment?

A: If you're lucky, the program won't compile because the comment has rendered the program illegal. If the program does compile, there are several techniques that you can use. Stepping through the program line by line with a debugger will reveal if any lines are being skipped. Some IDEs display comments in a distinctive color to distinguish them from surrounding code. If you're using such an environment, you can easily spot unterminated comments, since program text will have a different color if it's accidentally included in a comment. A program such as `lint` can also help.

`lint` ➤1.2

Q: Is it legal to nest one comment inside another?

A: Old-style comments (`/* ... */`) can't be nested. For instance, the following code is illegal:

```
/*
    /*** WRONG ***/
*/
```

The `*/` symbol on the second line matches the `/*` symbol on the first line, so the compiler will flag the `*/` symbol on the third line as an error.

C's prohibition against nested comments can sometimes be a problem. Suppose we've written a long program containing many short comments. To disable a portion of the program temporarily (during testing, say), our first impulse is to "comment out" the offending lines with `/*` and `*/`. Unfortunately, this method won't work if the lines contain old-style comments. C99 comments (those beginning with `//`) can be nested inside old-style comments, however—another advantage to using this kind of comment.

C99