The `printf()` function requires a text string enclosed in double quotes. Your compiler knows that. After the second double quote in the string was encountered (before the word *Ta*), the compiler expected something else — something other than "Ta." Therefore, an error was generated. Figure 4-1 illustrates this in a cute way.

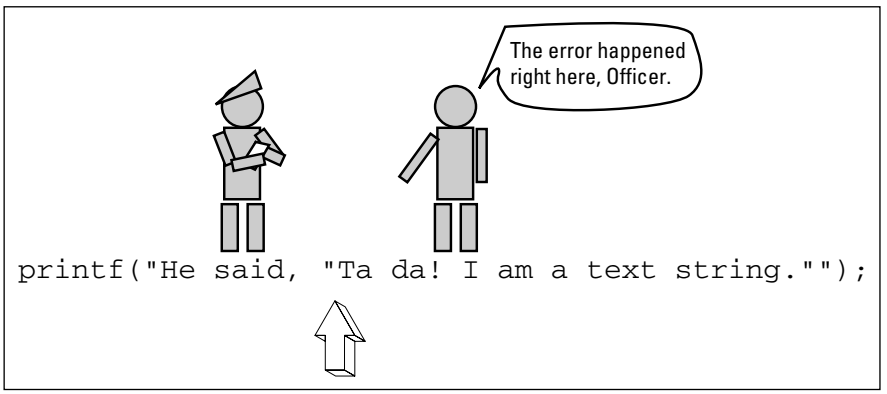

Figure 4-1: The C compiler detects something amiss in the `printf()` statement.

Obviously, there is a need to use the double-quote character in a string of text. The question is how to pass that character along to `printf()` without it ruining the rest of your day. The answer is to use an escape sequence.

In the olden days, programmers would have simply gone without certain characters. Rather than trip up a string with a double quote, they would have used two single quotes. Some ancient programmers who don't know about escape sequences still use these tricks.

Escape from `printf()`!

Escape sequences are designed to liven up an otherwise dull action picture with a few hard-cutting, loud-music moments of derring-do. In a programming language, *escape sequences* are used to sneak otherwise forbidden characters, or characters you cannot directly type at the keyboard, into text strings.

In the C language, escape sequences always begin with the backslash character (`\`). Locate this character on your keyboard now. It should be above the Enter key, though they often hide it elsewhere.

The backslash character signals the `printf()` function that an escape sequence is looming. When `printf()` sees the backslash, it thinks, "Omigosh, an escape sequence must be coming up," and it braces itself to accept an otherwise forbidden character.