You can pronounce `gets()` as "get-string" in your head. "Get a string of text from the keyboard." However, it probably stands for "Get stdin," which means "Get from standard input." "Get string" works for me, though.

And now, the bad news about `gets()`

The latest news from the C language grapevine is *not* to use the `gets()` function, at least not in any serious, secure programs you plan on writing. That's because `gets()` is not considered a safe, secure function to use.

The reason for this warning — which may even appear when you compile a program using `gets()` — is that you can type more characters at the keyboard than were designed to fit inside the `char` variable associated with `gets()`. This flaw, known as a *keyboard overflow,* is used by many of the bad guys out there to write worms and viruses and otherwise exploit well-meaning programs.

For the duration of this book, don't worry about using `gets()`. It's okay here as a quick way to get input while finding out how to use C. But for "real" programs that you write, I recommend concocting your own keyboard-reading functions.

The Virtues of `puts()`

In a way, the `puts()` function is a simplified version of the `printf()` function. `puts()` displays a string of text, but without all `printf()`'s formatting magic. `puts()` is just a boneheaded "Yup, I display this on the screen" command. Here's the format:

```
puts(text);
```

`puts()` is followed by a left paren, and then comes the *`text`* you want to display. That can either be a string variable name or a string of text in double quotes. That's followed by a right paren. The `puts()` function is a complete C language statement, so it always ends with a semicolon.

The `puts()` function's output always ends with a newline character, `\n`. It's like `puts()` "presses Enter" after displaying the text. You cannot avoid this side effect, though sometimes it does come in handy.