13 Strings

It's difficult to extract sense from strings, but they're the only communication coin we can count on.

Although we've used `char` variables and arrays of `char` values in previous chapters, we still lack any convenient way to process a series of characters (a *string*, in C terminology). We'll remedy that defect in this chapter, which covers both string *constants* (or *literals*, as they're called in the C standard) and string *variables*, which can change during the execution of a program.

Section 13.1 explains the rules that govern string literals, including the rules for embedding escape sequences in string literals and for breaking long string literals. Section 13.2 then shows how to declare string variables, which are simply arrays of characters in which a special character—the null character—marks the end of a string. Section 13.3 describes ways to read and write strings. Section 13.4 shows how to write functions that process strings, and Section 13.5 covers some of the string-handling functions in the C library. Section 13.6 presents idioms that are often used when working with strings. Finally, Section 13.7 describes how to set up arrays whose elements are pointers to strings of different lengths. This section also explains how C uses such an array to supply command-line information to programs.

13.1 String Literals

A *string literal* is a sequence of characters enclosed within double quotes:

```
"When you come to a fork in the road, take it."
```

We first encountered string literals in Chapter 2; they often appear as format strings in calls of `printf` and `scanf`.