The f means "formatted"

The function is called `printf()` for a reason. The *f* stands for *formatted.* The advantage of the `printf` function over other, similar display-this-or-that functions in C is that the output can be formatted.

Earlier in this chapter, I introduce the format for the basic `printf` function as

```
printf("text");
```

But the real format — *shhh!* — is

```
printf("format_string"[,var[,...]]);
```

What appears in the double quotes is really a *formatting string.* It's still text that appears in `printf()`'s output, but secretly inserted into the text are various *conversion characters,* or special "placeholders," that tell the `printf()` function how to format its output and do other powerful stuff.

After the format string comes a comma (still inside the parentheses) and then one or more items called *arguments.* The argument shown in the preceding example is *var,* which is short for *var*iable. You can use `printf()` to display the content or value of one or more variables. You do this by using special conversion characters in `format_string`. Figure 4-2 illustrates this concept rather beautifully.

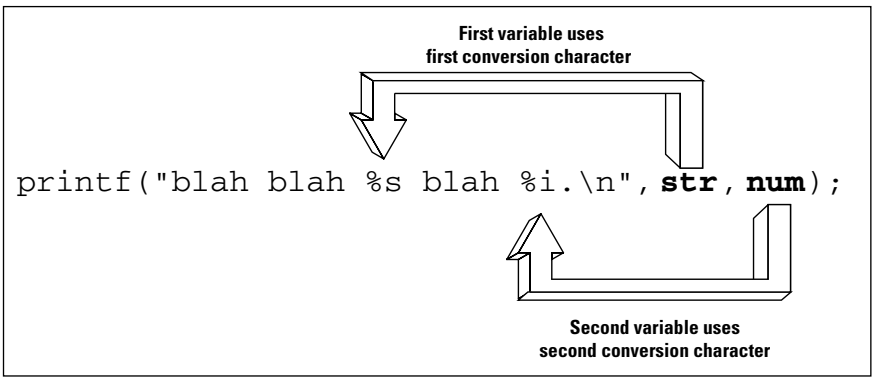

Figure 4-2: How `printf()` solves arguments.

The `[,...]` doohickey means that you can have any number of *var* items specified in a single `printf` function (before the final paren). Each *var* item, however, must have a corresponding placeholder (or conversion character) in `format_string`. They must match up, or else you get an error when the program is compiled.