Interacting with the Command Line

In Chapter 22, you may have read briefly about how the `main()` function returns a value to the operating system when the program quits. That's one way that a program can communicate with the operating system. The other way is to read in options directly from the command line. For example:

```
grep pirntf *.c
```

This shell command searches for misspellings in your C language source code. The command has two command-line arguments: `pirntf` and `*.c`. These two strings of text are passed to the `main()` function as arguments as well, which the program can then evaluate and act on, just as arguments passed to any function.

The problem with introducing such a thing in this book is that you need to understand more about arrays and pointers to be able to deal with the information passed to the `main()` function. That too will have to wait for another day.

Disk Access

One of the reasons you have a computer is to store information and work on it later. The C language is equipped with a wide variety of functions to read and write information from and to the disk drives. You can save data to disk using C just as you can with any program that has a File➪Save command — though in C, it is *you* who writes the File➪Save command.

Interacting with the Operating System

The C language also lets you perform operating system functions. You can change directories, make new directories, delete files, rename files, and do a host of other handy tasks.

You can also have your programs run other programs — sometimes two at once! Or, your program can run operating system commands and examine the results.

Finally, you can have your program interact with the environment and examine the state of your computer. You can even run services or prowl out on the network. Just about anything the computer can do, you can add into your program and do it yourself.