CHAPTER 7: INPUT AND OUTPUT

Chapter 7

Input and output facilities are not part of the C language, so we have de-emphasized them in our presentation thus far. Nonetheless, real programs do interact with their environment in much more complicated ways than those we have shown before. In this chapter we will describe "the standard I/O library," a set of functions designed to provide a standard I/O system for C programs. The functions are intended to present a convenient programming interface, yet reflect only operations that can be provided on most modern operating systems. The routines are efficient enough that users should seldom feel the need to circumvent them "for efficiency" regardless of how critical the application. Finally, the routines are meant to be "portable," in the sense that they will exist in compatible form on any system where C exists, and that programs which confine their system interactions to facilities provided by the standard library can be moved from one system to another essentially without change.

We will not try to describe the entire I/O library here; we are more interested in showing the essentials of writing C programs that interact with their operating system environment.

7.1 Access to the Standard Library

Each source file that refers to a standard library function must contain the line

```
#include <stdio.h>
```

near the beginning. The file `stdio.h` defines certain macros and variables used by the I/O library. Use of the angle brackets `<` and `>` instead of the usual double quotes directs the compiler to search for the file in a directory containing standard header information (on UNIX, typically `/usr/include`).

Furthermore, it may be necessary when loading the program to specify the library explicitly; for example, on the PDP-11 UNIX system, the command to compile a program would be

```
cc source files, etc. -lS
```

where `-lS` indicates loading from the standard library. (The character `l` is the letter ell.)

7.2 Standard Input and Output - Getchar and Putchar

The simplest input mechanism is to read a character at a time from the "standard input," generally the user's terminal, with `getchar`. `getchar()` returns the next input character each time it is called. In most environments that support C, a file may be substituted for the terminal by using the `<` convention: if a program `prog` uses `getchar`, then the command line

```
prog <infile
```

causes `prog` to read `infile` instead of the terminal. The switching of the input is done in such a way that `prog` itself is oblivious to the change; in particular, the string `<infile` is not included in