

Chapter 7 - Input and Output Input and output are not part of the C language itself, so we have not emphasized them in our presentation thus far. Nonetheless, programs interact with their environment in much more complicated ways than those we have shown before. In this chapter we will describe the standard library, a set of functions that provide input and output, string handling, storage management, mathematical routines, and a variety of other services for C programs. We will concentrate on input and output. The ANSI standard defines these library functions precisely, so that they can exist in compatible form on any system where C exists. Programs that confine their system interactions to facilities provided by the standard library can be moved from one system to another without change. The properties of library functions are specified in more than a dozen headers; we have already seen several of these, including `<stdio.h>`, `<string.h>`, and `<ctype.h>`. We will not present the entire library here, since we are more interested in writing C programs that use it. The library is described in detail in Appendix B.

7.1 Standard Input and Output

As we said in Chapter 1, the library implements a simple model of text input and output. A text stream consists of a sequence of lines; each line ends with a newline character. If the system doesn't operate that way, the library does whatever necessary to make it appear as if it does. For instance, the library might convert carriage return and linefeed to newline on input and back again on output. The simplest input mechanism is to read one character at a time from the standard input, normally the keyboard, with `getchar`: `int getchar(void)` `getchar` returns the next input character each time it is called, or EOF when it encounters end of file. The symbolic constant EOF is defined in `<stdio.h>`. The value is typically -1, but tests should be written in terms of EOF so as to be independent of the specific value. In many environments, a file may be substituted for the keyboard by using the `<` convention for input redirection: if a program `prog` uses `getchar`, then the command `line prog <infile` causes `prog` to read characters from `infile` instead. 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 the command-line arguments in `argv`. Input switching is also invisible if the input comes from another program via a pipe mechanism: on some systems, the command `line otherprog | prog` runs the two programs `otherprog` and `prog`, and pipes the standard output of `otherprog` into the standard input for `prog`.