

CHAPTER 8: THE UNIX SYSTEM INTERFACE

The material in this chapter is concerned with the interface between C programs and the UNIX¹ operating system. Since most C users are on UNIX systems, this should be helpful to a majority of readers. Even if you use C on a different machine, however, you should be able to glean more insight into C programming from studying these examples.

The chapter is divided into three major areas: input/output, file system, and a storage allocator. The first two parts assume a modest familiarity with the external characteristics of UNIX.

[Chapter 7](#) was concerned with a system interface that is uniform across a variety of operating systems. On any particular system the routines of the standard library have to be written in terms of the I/O facilities actually available on the host system. In the next few sections we will describe the basic system entry points for I/O on the UNIX operating system, and illustrate how parts of the standard library can be implemented with them.

The dual nature of C and UNIX has been on display throughout the book and while this chapter is called "The UNIX System Interface", in a sense is less about UNIX itself and very much about why C is such a great programming language. Let me explain.

Before UNIX and C became the norm, operating systems and operating system utilities (commands used interactively and in batch jobs) were quite often written in the assembly language of computer which it was supporting. Often there were not well documented "API" calls between utilities in assembly language and the assembly language which implemented the operating system. Smart programmers would just look at the operating system code and write their utility code to work with it.

This section shows that a language that has features like structures, arrays, pointers, a pre-processor, and unions was *sufficiently rich* so that we could document all of the intricate interfaces with an operating system using a high level language and then we could write our utility code (like `cat`) in a high level language.

In this chapter the authors are almost shouting, "Quit using assembly language to build your operating system and utility code!". Further they are showing us examples designed to answer the question that might come from programmers used to the old ways like, "Can C do XYZ?". Their emphatic answer in the (increasingly intricate) code samples is that "C is not a toy language that is only something used by a few AT&T computer scientists in a research lab".

If you are doing serious system stuff that needs maximum performance and readability - use C.