

1.2 Strengths and Weaknesses of C

Like any other programming language, C has strengths and weaknesses. Both stem from the language's original use (writing operating systems and other systems software) and its underlying philosophy:

- *C is a low-level language.* To serve as a suitable language for systems programming, C provides access to machine-level concepts (bytes and addresses, for example) that other programming languages try to hide. C also provides operations that correspond closely to a computer's built-in instructions, so that programs can be fast. Since application programs rely on it for input/output, storage management, and numerous other services, an operating system can't afford to be slow.
- *C is a small language.* C provides a more limited set of features than many languages. (The reference manual in the second edition of K&R covers the entire language in 49 pages.) To keep the number of features small, C relies heavily on a "library" of standard functions. (A "function" is similar to what other programming languages might call a "procedure," "subroutine," or "method.")
- *C is a permissive language.* C assumes that you know what you're doing, so it allows you a wider degree of latitude than many languages. Moreover, C doesn't mandate the detailed error-checking found in other languages.

Strengths

C's strengths help explain why the language has become so popular:

- *Efficiency.* Efficiency has been one of C's advantages from the beginning. Because C was intended for applications where assembly language had traditionally been used, it was crucial that C programs could run quickly and in limited amounts of memory.
- *Portability.* Although program portability wasn't a primary goal of C, it has turned out to be one of the language's strengths. When a program must run on computers ranging from PCs to supercomputers, it is often written in C. One reason for the portability of C programs is that—thanks to C's early association with UNIX and the later ANSI/ISO standards—the language hasn't splintered into incompatible dialects. Another is that C compilers are small and easily written, which has helped make them widely available. Finally, C itself has features that support portability (although there's nothing to prevent programmers from writing nonportable programs).
- *Power.* C's large collection of data types and operators help make it a powerful language. In C, it's often possible to accomplish quite a bit with just a few lines of code.