Q: What does `lint` do? [p. 6]

A: `lint` checks a C program for a host of potential errors, including—but not limited to—suspicious combinations of types, unused variables, unreachable code, and nonportable code. It produces a list of diagnostic messages, which the programmer must then sift through. The advantage of using `lint` is that it can detect errors that are missed by the compiler. On the other hand, you've got to remember to use `lint`; it's all too easy to forget about it. Worse still, `lint` can produce messages by the hundreds, of which only a fraction refer to actual errors.

Q: Where did `lint` get its name?

A: Unlike the names of many other UNIX tools, `lint` isn't an acronym; it got its name from the way it picks up pieces of "fluff" from a program.

Q: How do I get a copy of `lint`?

A: `lint` is a standard UNIX utility; if you rely on another operating system, then you probably don't have `lint`. Fortunately, versions of `lint` are available from third parties. An enhanced version of `lint` known as `splint` (Secure Programming Lint) is included in many Linux distributions and can be downloaded for free from *www.splint.org*.

Q: Is there some way to force a compiler to do a more thorough job of error-checking, without having to use `lint`?

A: Yes. Most compilers will do a more thorough check of a program if asked to. In addition to checking for errors (undisputed violations of the rules of C), most compilers also produce warning messages, indicating potential trouble spots. Some compilers have more than one "warning level"; selecting a higher level causes the compiler to check for more problems than choosing a lower level. If your compiler supports warning levels, it's a good idea to select the highest level, causing the compiler to perform the most thorough job of checking that it's capable of. Error-checking options for the GCC compiler, which is distributed with Linux, are discussed in the Q&A section at the end of Chapter 2. GCC ➤*2.1*

***Q: I'm interested in making my program as reliable as possible. Are there any other tools available besides `lint` and debuggers?**

A: Yes. Other common tools include "bounds-checkers" and "leak-finders." C doesn't require that array subscripts be checked; a bounds-checker adds this capability. A leak-finder helps locate "memory leaks": blocks of memory that are dynamically allocated but never deallocated.

*Starred questions cover material too advanced or too esoteric to interest the average reader, and often refer to topics covered in later chapters. Curious readers with a fair bit of programming experience may wish to delve into these questions immediately; others should definitely skip them on a first reading.