

The return value is the difference between the first “mismatched” characters in the `s` and `t` strings, which will be negative if `s` points to a “smaller” string than `t` and positive if `s` points to a “larger” string. There’s no guarantee that `strcmp` is actually written this way, though, so it’s best not to assume that the magnitude of its return value has any particular meaning.

Q: My compiler issues a warning when I try to compile the `while` statement in the `strcat` function:

```
while (*p++ = *s2++)
    ;
```

What am I doing wrong?

A: Nothing. Many compilers—but not all, by any means—issue a warning if you use `=` where `==` is normally expected. This warning is valid at least 95% of the time, and it will save you a lot of debugging if you heed it. Unfortunately, the warning isn’t relevant in this particular example; we actually *do* mean to use `=`, not `==`. To get rid of the warning, rewrite the `while` loop as follows:

```
while ((*p++ = *s2++) != 0)
    ;
```

Since the `while` statement normally tests whether `*p++ = *s2++` is not 0, we haven’t changed the meaning of the statement. The warning goes away, however, because the statement now tests a condition, not an assignment. With the GCC compiler, putting a pair of parentheses around the assignment is another way to avoid a warning:

```
while ((*p++ = *s2++))
    ;
```

Q: Are the `strlen` and `strcat` functions actually written as shown in Section 13.6?

A: Possibly, although it’s common practice for compiler vendors to write these functions—and many other string functions—in assembly language instead of C. The string functions need to be as fast as possible, since they’re used often and have to deal with strings of arbitrary length. Writing these functions in assembly language makes it possible to achieve great efficiency by taking advantage of any special string-handling instructions that the CPU may provide.

Q: Why does the C standard use the term “program parameters” instead of “command-line arguments”? [p. 302]

A: Programs aren’t always run from a command line. In a typical graphical user interface, for example, programs are launched with a mouse click. In such an environment, there’s no traditional command line, although there may be other ways of passing information to a program; the term “program parameters” leaves the door open for these alternatives.