

```

    while (*s)
        s++;
    return s - p;
}

```

This version of `strlen` computes the length of the string by locating the position of the null character, then subtracting from it the position of the first character in the string. The improvement in speed comes from not having to increment `n` inside the `while` loop. Note the appearance of the word `const` in the declaration of `p`, by the way; without it, the compiler would notice that assigning `s` to `p` places the string that `s` points to at risk.

The statement

**idiom**    `while (*s)`  
           `s++;`

and the related

**idiom**    `while (*s++)`  
           `;`

are idioms meaning “search for the null character at the end of a string.” The first version leaves `s` pointing to the null character. The second version is more concise, but leaves `s` pointing just past the null character.

## Copying a String

Copying a string is another common operation. To introduce C’s “string copy” idiom, we’ll develop two versions of the `strcat` function. Let’s start with a straightforward but somewhat lengthy version:

```

char *strcat(char *s1, const char *s2)
{
    char *p = s1;

    while (*p != '\0')
        p++;
    while (*s2 != '\0') {
        *p = *s2;
        p++;
        s2++;
    }
    *p = '\0';
    return s1;
}

```

This version of `strcat` uses a two-step algorithm: (1) Locate the null character at the end of the string `s1` and make `p` point to it. (2) Copy characters one by one from `s2` to where `p` is pointing.

The first `while` statement in the function implements step (1). `p` is set to point to the first character in the `s1` string. Assuming that `s1` points to the string “abc”, we have the following picture: