Like the `strchr` function, `memchr` returns a pointer to the first occurrence of the character. If it can't find the desired character, `memchr` returns a null pointer.

strrchr

`strrchr` is similar to `strchr`, but it searches the string in *reverse* order:

```
char *p, str[] = "Form follows function.";

p = strrchr(str, 'f');    /* finds last 'f' */
```

In this example, `strrchr` will first search for the null character at the end of the string, then go backwards to locate the letter `f` (the one in `function`). Like `strchr` and `memchr`, `strrchr` returns a null pointer if it fails to find the desired character.

strpbrk

`strpbrk` is more general than `strchr`; it returns a pointer to the leftmost character in the first argument that matches *any* character in the second argument:

```
char *p, str[] = "Form follows function.";

p = strpbrk(str, "mn");    /* finds first 'm' or 'n' */
```

In this example, `p` will point to the letter `m` in `Form`. `strpbrk` returns a null pointer if no match is found.

strspn
strcspn

Q&A

The `strspn` and `strcspn` functions, unlike the other search functions, return an integer (of type `size_t`), representing a position within a string. When given a string to search and a set of characters to look for, `strspn` returns the index of the first character that's *not* in the set. When passed similar arguments, `strcspn` returns the index of the first character that's *in* the set. Here are examples of both functions:

```
size_t n;
char str[] = "Form follows function.";

n = strspn(str, "morF");      /* n = 4 */
n = strspn(str, " \t\n");     /* n = 0 */
n = strcspn(str, "morF");     /* n = 0 */
n = strcspn(str, " \t\n");    /* n = 4 */
```

strstr

`strstr` searches its first argument (a string) for a match with its second argument (also a string). In the following example, `strstr` searches for the word `fun`:

```
char *p, str[] = "Form follows function.";

p = strstr(str, "fun");    /* locates "fun" in str */
```

`strstr` returns a pointer to the first occurrence of the search string; it returns a null pointer if it can't locate the string. After the call above, `p` will point to the letter `f` in `function`.

strtok

`strtok` is the most complicated of the search functions. It's designed to search a string for a "token"—a sequence of characters that doesn't include certain delimiting characters. The call `strtok(s1, s2)` scans the `s1` string for a nonempty sequence of characters that are *not* in the `s2` string. `strtok` marks the end