characters. We'll now describe these functions, which belong to the `<stdlib.h>` header. C99's `<wchar.h>` and `<wctype.h>` headers, which are discussed in Sections 25.5 and 25.6, supply a number of additional multibyte and wide-character functions.

C89's multibyte/wide-character functions are divided into two groups. The first group converts single characters from multibyte form to wide form and vice versa. The behavior of these functions depends on the `LC_CTYPE` category of the current locale. If the multibyte encoding is state-dependent, the behavior also depends on the current ***conversion state.*** The conversion state consists of the current shift state as well as the current position within a multibyte character. Calling any of these functions with a null pointer as the value of its `char *` parameter sets the function's internal conversion state to the ***initial conversion state,*** signifying that no multibyte character is yet in progress and that the initial shift state is in effect. Later calls of the function cause its internal conversion state to be updated.

mblen

The `mblen` function checks whether its first argument points to a series of bytes that form a valid multibyte character. If so, the function returns the number of bytes in the character; if not, it returns –1. As a special case, `mblen` returns 0 if the first argument points to a null character. The second argument limits the number of bytes that `mblen` will examine; typically, we'll pass `MB_CUR_MAX`.

The following function, which comes from P. J. Plauger's *The Standard C Library*, uses `mblen` to determine whether a string consists of valid multibyte characters. The function returns zero if `s` points to a valid string.

```
int mbcheck(const char *s)
{
  int n;

  for (mblen(NULL, 0); ; s += n)
    if ((n = mblen(s, MB_CUR_MAX)) <= 0)
      return n;
}
```

Two aspects of the `mbcheck` function deserve special mention. First, there's the mysterious call `mblen(NULL, 0)`, which sets `mblen`'s internal conversion state to the initial conversion state (in case the multibyte encoding is state-dependent). Second, there's the matter of termination. Keep in mind that `s` points to an ordinary character string, which is assumed to end with a null character. `mblen` will return zero when it reaches this null character, causing `mbcheck` to return. `mbcheck` will return sooner if `mblen` returns –1 because of an invalid multibyte character.

mbtowc

The `mbtowc` function converts a multibyte character (pointed to by the second argument) into a wide character. The first argument points to a `wchar_t` variable into which the function will store the result. The third argument limits the number of bytes that `mbtowc` will examine. `mbtowc` returns the same value as `mblen`: the number of bytes in the multibyte character if it's valid, –1 if it's not, and zero if the second argument points to a null character.