Table 25.11
Formatted Wide-Character Input/Output Functions and Their `<stdio.h>` Equivalents

<wchar.h> Function	*<stdio.h> Equivalent*
`fwprintf`	`fprintf`
`fwscanf`	`fscanf`
`swprintf`	`snprintf, sprintf`
`swscanf`	`sscanf`
`vfwprintf`	`vfprintf`
`vfwscanf`	`vfscanf`
`vswprintf`	`vsnprintf, vsprintf`
`vswscanf`	`vsscanf`
`vwprintf`	`vprintf`
`vwscanf`	`vscanf`
`wprintf`	`printf`
`wscanf`	`scanf`

fwprintf Additional differences between `fwprintf` and `fprintf` include the following:

- The `%c` conversion specifier is used when the corresponding argument has type `int`. If the `l` length modifier is present (making the conversion `%lc`), the argument is assumed to have type `wint_t`. In either case, the corresponding argument is written as a wide character.
- The `%s` conversion specifier is used with a pointer to a character array, which may contain multibyte characters. (`fprintf` has no special provision for multibyte characters.) If the `l` length modifier is present, as in `%ls`, the corresponding argument should be an array containing wide characters. In either case, the characters in the array are written as wide characters. (With `fprintf`, the `%ls` specification also indicates an array of wide characters, but they're converted to multibyte characters before being written.)

fwscanf Unlike `fscanf`, the `fwscanf` function reads wide characters. The `%c`, `%s`, and `%[` conversions require special mention. Each of these causes wide characters to be read and then converted to multibyte characters before being stored in a character array. `fwscanf` uses an `mbstate_t` object to keep track of the state of the conversion during this process; the object is set to zero at the beginning of each conversion. If the `l` length modifier is present (making the conversion `%lc`, `%ls`, or `%l[`), then the input characters are not converted but instead are stored directly in an array of `wchar_t` elements. Thus, it's necessary to use `%ls` when reading a string of wide characters if the intent is to store them as wide characters. If `%s` is used instead, wide characters will be read from the input stream but converted to multibyte characters before being stored.

swprintf `swprintf` writes wide characters into an array of `wchar_t` elements. It's similar to `sprintf` and `snprintf` but not identical to either one. Like `snprintf`, it uses the parameter `n` to limit the number of (wide) characters that it will write. However, `swprintf` returns the number of wide characters actually written, not including the null character. In this respect, it resembles `sprintf` rather than `snprintf`, which returns the number of characters that would have been written (not including the null character) had there been no length restriction.