std::mblen

Defined in header <cstdlib>

```
int mblen( const char* s, std::size_t n );
```

Determines the size, in bytes, of the multibyte character whose first byte is pointed to by s.

If s is a null pointer, resets the global conversion state and determines whether shift sequences are used.

This function is equivalent to the call `std::mbtowc(nullptr, s, n)`, except that conversion state of std::mbtowc is unaffected.

Notes

Each call to mblen updates the internal global conversion state (a static object of type std::mbstate_t, only known to this function). If the multibyte encoding uses shift states, care must be taken to avoid backtracking or multiple scans. In any case, multiple threads should not call mblen without synchronization: std::mbrlen may be used instead.

Parameters

s - pointer to the multibyte character
n - limit on the number of bytes in s that can be examined

Return value

If s is not a null pointer, returns the number of bytes that are contained in the multibyte character or `-1` if the first bytes pointed to by s do not form a valid multibyte character or `0` if s is pointing at the null charcter `'\0'`.

If s is a null pointer, resets its internal conversion state to represent the initial shift state and returns `0` if the current multibyte encoding is not state-dependent (does not use shift sequences) or a non-zero value if the current multibyte encoding is state-dependent (uses shift sequences).

Example

Run this code

```
#include <clocale>
#include <cstdlib>
#include <iomanip>
#include <iostream>
#include <stdexcept>
#include <string_view>

// the number of characters in a multibyte string is the sum of mblen()'s
// note: the simpler approach is std::mbstowcs(nullptr, s.c_str(), s.size())
std::size_t strlen_mb(const std::string_view s)
{
    std::size_t result = 0;
    const char* ptr = s.data();
    const char* end = ptr + s.size();
    std::mblen(nullptr, 0); // reset the conversion state
    while (ptr < end) {
        int next = std::mblen(ptr, end-ptr);
        if (next == -1) {
            throw std::runtime_error("strlen_mb(): conversion error");
        }
        ptr += next;
        ++result;
    }
    return result;
}

void dump_bytes(const std::string_view str)
{
    std::cout << std::hex << std::uppercase << std::setfill('0');
```

```
    for (unsigned char c : str)
        std::cout << std::setw(2) << static_cast<int>(c) << ' ';
    std::cout << std::dec << '\n';
}

int main()
{
    // allow mblen() to work with UTF-8 multibyte encoding
    std::setlocale(LC_ALL, "en_US.utf8");
    // UTF-8 narrow multibyte encoding
    const std::string_view str = "z\u00df\u6c34\U0001f34c"; // or u8"zß水🍌"
    std::cout << std::quoted(str) << " is " << strlen_mb(str)
              << " characters, but as much as " << str.size() << " bytes: ";
    dump_bytes(str);
}
```

Possible output:

```
"zß水🍌" is 4 characters, but as much as 10 bytes: 7A C3 9F E6 B0 B4 F0 9F 8D 8C
```

See also

mbtowc	converts the next multibyte character to wide character (function)
mbrlen	returns the number of bytes in the next multibyte character, given state (function)
C documentation for **mblen**	