Document number: P1944R0
Project: Programming Language C++
Audience: LEWGI, LEWG, LWG

Daniil Goncharov neargye@gmail.com

Antony Polukhin antoshkka@gmail.com

Date: 2019-10-24

Add Constexpr Modifiers to Functions in <cstring> and <cwchar> Headers

I. Introduction and Motivation

Headers `<cstring>` and `<cwchar>` have popular functions for string manipulation. In C++20 those functions are not `constexpr`. The paper proposes to make some of the functions usable in constexpr context.

Consider the simple example:

```
int main() {
    constexpr char[] str = "abcd"; // OK
    constexpr auto str_len = std::strlen(str); // Fail
}
```

II. Impact on the Standard

This proposal is a pure library extension. It proposes changes to existing headers `<cstring>` and `<cwchar>` such that the changes do not break existing code and do not degrade performance. It does not require any changes in the core language in simple cases of non assembly optimized Standard Library, and it could be implemented in standard C++, except for the memchr, memcmp, memset, memcpy and memmove functions.

III. Design Decisions

A. `<cstring>` must have `constexpr` additions

All the functions from `<cstring>` header must be marked with `constexpr`, except the `strcoll`, `strxfrm`, `strtok`, `strerror` functions.

`strcoll`, `strxfrm` use locale that is non usable in `constexpr` context. `strtok` touches a static or global variable. `strerror` touches a thread local buffer and also can not be made `constexpr`.

B. `std::memchr`, `std::memcmp`, `std::memchr`, `std::memset`, `std::memcpy`, `std::memmove` must have `constexpr` additions

`std::memchr`, `std::memcmp`, `std::memchr`, `std::memset`, `std::memcpy`, `std::memmove` accept `void*` and `const void*` parameters. This makes them impossible to implement in pure C++ as `constexpr`, because constant expressions can not evaluate a conversion from type cv `void *` to a pointer-to-object type according to [expr.const].

However those functions are not only popular, but also are widely used across Standard Library to gain better performance. Not making them `constexpr` will force standard Library developer to have compiler intrinsics for them anyway. This is a hard step that must be done.

Clang already support `constexpr` __builtin_memchr, __builtin_memcmp, __builtin_memcpy, __builtin_memmove https://reviews.llvm.org/rL338941.

Note that std::bit_cast and std::is_constant_evaluated() could be used to implement those functions in pure C++ (in theory).

C. Add `strtok(char* str, const char* delim, char** ptr)`

Unlike `strtok(char* str, const char* delim)`, this function does not update static storage: it stores the parser state in the user-provided location, so it can be `constexpr`.

This function is analogous to the existing `std::wcstok` function, but works with char.

```
constexpr char* strtok(char* str, const char* delim, char** ptr);
```

D. Apply the `constexpr` to the analogs in

As well as similar functions from `<cstrings>` for char, these functions from `<cwchar>` are useful when working with wchar_t in `constexpr`. Note that we do not propose to constexprify the functons that touch global state or work with locales.

IV. Proposed wording

All the additions to the Standard are marked with underlined green.

A. Modifications to "21.8 Null-terminated sequence utilities" [c.strings]

constexpr char* strcpy(char* dest, const char* src);

constexpr char* strncpy(char* dest, const char* src, std::size_t count);

constexpr char* strcat(char* dest, const char* src);

```
constexpr char* strncat(char* dest, const char* src, std::size_t count);

constexpr std::size_t strlen(const char* str);

constexpr int strcmp(const char* lhs, const char* rhs);

constexpr int strncmp(const char* lhs, const char* rhs, std::size_t count);

constexpr const char* strchr(const char* str, int ch);

constexpr char* strchr(char* str, int ch);

constexpr const char* strrchr(const char* str, int ch);

constexpr char* strrchr(char* str, int ch);

constexpr std::size_t strspn(const char* dest, const char* src);

constexpr std::size_t strcspn(const char* dest, const char* src);

constexpr const char* strpbrk(const char* dest, const char* breakset);

constexpr char* strpbrk(char* dest, const char* breakset);

constexpr const char* strstr(const char* str, const char* target);

constexpr char* strstr(char* str, const char* target);

constexpr char* strtok(char* str, const char* delim, char** ptr);

constexpr const void* memchr(const void* ptr, int ch, std::size_t count);

constexpr void* memchr(void* ptr, int ch, std::size_t count);

constexpr int memcmp(const void* lhs, const void* rhs, std::size_t count);

constexpr void* memset(void* dest, int ch, std::size_t count);

constexpr void* memcpy(void* dest, const void* src, std::size_t count);

constexpr void* memmove(void* dest, const void* src, std::size_t count);
```

B. Modifications to "21.8 Null-terminated sequence utilities" [c.wchar]

```
constexpr wchar_t* wcscpy(wchar_t* dest, const wchar_t* src);

constexpr wchar_t* wcsncpy(wchar_t* dest, const wchar_t* src, std::size_t count);

constexpr wchar_t* wcscat(wchar_t* dest, const wchar_t* src);

constexpr wchar_t* wcsncat(wchar_t* dest, const wchar_t* src, std::size_t count);

constexpr std::size_t wcslen(const wchar_t* str);
```

constexpr int wcscmp(const wchar_t* lhs, const wchar_t* rhs);

constexpr int wcsncmp(const wchar_t* lhs, const wchar_t* rhs, std::size_t count);

constexpr const wchar_t* wcschr(const wchar_t* str, wchar_t ch);

constexpr wchar_t* wcschr(wchar_t* str, wchar_t ch);

constexpr const wchar_t* wcsrchr(const wchar_t* str, wchar_t ch);

constexpr wchar_t* wcsrchr(wchar_t* str, wchar_t ch);

constexpr std::size_t wcsspn(const wchar_t* dest, const wchar_t* src);

constexpr std::size_t wcscspn(const wchar_t* dest, const wchar_t* src);

constexpr const wchar_t* wcspbrk(const wchar_t* dest, const wchar_t* breakset);

constexpr wchar_t* wcspbrk(wchar_t* dest, const wchar_t* breakset);

constexpr const wchar_t* wcsstr(const wchar_t* str, const wchar_t* target);

constexpr wchar_t* wcsstr(wchar_t* str, const wchar_t* target);

constexpr wchar_t* wcstok(wchar_t* str, const wchar_t* delim, wchar_t** ptr);

constexpr wchar_t* wmemcpy(wchar_t* dest, const wchar_t* src, std::size_t count);

constexpr wchar_t* wmemmove(wchar_t* dest, const wchar_t* src, std::size_t count);

constexpr int wmemcmp(const wchar_t* lhs, const wchar_t* rhs, std::size_t count);

constexpr const wchar_t* wmemchr(const wchar_t* ptr, wchar_t ch, std::size_t count);

constexpr wchar_t* wmemchr(wchar_t* ptr, wchar_t ch, std::size_t count);

constexpr wchar_t* wmemset(wchar_t* dest, wchar_t ch, std::size_t count);

C. Modify [support.limits.general]/3

#define __cpp_lib_constexpr ~~201811L~~ DATE OF ADOPTION

V. Revision History

Revision 0: * Initial proposal

VI. References:

- [neargye] Proof of concept for <cstring> and <cwchar> functions https://github.com/Neargye/cstring-constexpr-proposal.
- [P0202R0] A Proposal to Add Constexpr Modifiers to Functions in <cwchar> and <cstring> Headers http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2016/p0202r0.html.