Document number: P1944R1
Project: Programming Language C++
Audience: LEWGI, LEWG, LWG, SG16

Daniil Goncharov neargye@gmail.com
Antony Polukhin antoshkka@gmail.com

Date: 2020-03-28

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`. Other functions by C standard do not use locale see 7.11.1.1 [N1570].

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.
GCC already support `constexpr __builtin_mem*` https://gcc.gnu.org/bugzilla/show_bug.cgi?id=80265.

C. Apply the `constexpr` to the analogs in `<cwchar>`

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.

D. ABI compatibility

Implementations that use their own C++ implementation of `<cstring>` and `<cwchar>` as per using.linkage.2, could mark functions constexpr without ABI break.

Implementations that use the C standard library and use "C" linkage for the `<cstring>` and `<cwchar>` headers could also avoid ABI break. One of the solutions is to special case `<cstring>` and `<cwchar>` functions at the compiler level. In that case in constexpr-context compiler will return compile-time result, in runtime compiler will emit regular C function call.

IV. Proposed wording relative to [N4835]

Modifications to "21.5 Null-terminated sequence utilities" [c.strings]

All the additions to the Standard are marked with green.

A. Modifications to "21.5.3 Header <cstring> synopsis" [cstring.syn]

```
constexpr void* memcpy(void* s1, const void* s2, size_t n);

constexpr void* memmove(void* s1, const void* s2, size_t n);

constexpr char* strcpy(char* s1, const char* s2);

constexpr char* strncpy(char* s1, const char* s2, size_t n);

constexpr char* strcat(char* s1, const char* s2);

constexpr char* strncat(char* s1, const char* s2, size_t n);

constexpr int memcmp(const void* s1, const void* s2, size_t n);

constexpr int strcmp(const char* s1, const char* s2);
```

```
int strcoll(const char* s1, const char* s2);

constexpr int strncmp(const char* s1, const char* s2, size_t n);

size_t strxfrm(char* s1, const char* s2, size_t n);

constexpr void* memchr(void* s, int c, size_t n);

constexpr const void* memchr(const void* s, int c, size_t n);

constexpr const char* strchr(const char* s, int c);

constexpr char* strchr(char* s, int c);

constexpr size_t strcspn(const char* s1, const char* s2);

constexpr const char* strpbrk(const char* s1, const char* s2);

constexpr char* strpbrk(char* s1, const char* s2);

constexpr const char* strrchr(const char* s, int c);

constexpr char* strrchr(char* s, int c);

constexpr size_t strspn(const char* s1, const char* s);

constexpr const char* strstr(const char* s1, const char* s2);

constexpr char* strstr(char* s1, const char* s2);

char* strtok(char* s1, const char* s2);

constexpr void* memset(void* s, int c, size_t n);

char* strerror(int errnum);

constexpr size_t strlen(const char* s);
```

B. Modifications to "21.5.4 Header <cwchar> synopsis" [cwchar.syn]

```
constexpr wchar_t* wcscpy(wchar_t* s1, const wchar_t* s2);

constexpr wchar_t* wcsncpy(wchar_t* s1, const wchar_t* s2, size_t n);

constexpr wchar_t* wmemcpy(wchar_t* s1, const wchar_t* s2, size_t n);

constexpr wchar_t* wmemmove(wchar_t* s1, const wchar_t* s2, size_t n);

constexpr wchar_t* wcscat(wchar_t* s1, const wchar_t* s2);

constexpr wchar_t* wcsncat(wchar_t* s1, const wchar_t* s2, size_t n);

constexpr int wcscmp(const wchar_t* s1, const wchar_t* s2);

int wcscoll(const wchar_t* s1, const wchar_t* s2);
```

```
constexpr int wcsncmp(const wchar_t* s1, const wchar_t* s2, size_t n);

size_t wcsxfrm(wchar_t* s1, const wchar_t* s2, size_t n);

constexpr int wmemcmp(const wchar_t* s1, const wchar_t* s2, size_t n);

constexpr const wchar_t* wcschr(const wchar_t* s, wchar_t c);

constexpr wchar_t* wcschr(wchar_t* s, wchar_t c);

constexpr size_t wcscspn(const wchar_t* s1, const wchar_t* s2);

constexpr const wchar_t* wcspbrk(const wchar_t* s1, const wchar_t* s2);

constexpr wchar_t* wcspbrk(wchar_t* s1, const wchar_t* s2);

constexpr const wchar_t* wcsrchr(const wchar_t* s, wchar_t c);

constexpr wchar_t* wcsrchr(wchar_t* s, wchar_t c);

constexpr size_t wcsspn(const wchar_t* s1, const wchar_t* s2);

constexpr const wchar_t* wcsstr(const wchar_t* s1, const wchar_t* s2);

constexpr wchar_t* wcsstr(wchar_t* s1, const wchar_t* s2);

constexpr wchar_t* wcstok(wchar_t* s1, const wchar_t* s2, wchar_t** ptr);

constexpr const wchar_t* wmemchr(const wchar_t* s, wchar_t c, size_t n);

constexpr wchar_t* wmemchr(wchar_t* s, wchar_t c, size_t n);

constexpr size_t wcslen(const wchar_t* s);

constexpr wchar_t* wmemset(wchar_t* s, wchar_t c, size_t n);
```

C. Modify to "17.3.2 Header <version> synopsis" [version.syn]

```
#define __cpp_lib_constexpr_cstring DATE OF ADOPTION

#define __cpp_lib_constexpr_cwchar DATE OF ADOPTION
```

V. Revision History

Revision 0:

- Initial proposal
- Prague voting
    - We think this is important enough that we want to spend more time on this problem.

SF	WF	B	WA	SA
8	6	1	1	0

    - We want to add `std::strtok(char*, char*, char**)`.

SF	WF	B	WA	SA
2	4	5	4	0

    - Consensus: Bring a revision of P1944R0 (constexpr `<cstring>` and `<cwchar>`), with the guidance below, to LEWGI for further design review.
        - Remove `std::strtok(char*, char*, char**)`.
        - Find the prior discussions on adding constexpr to `<cstuff>` headers, consult implementors, and add what you learn to the next revision.
        - Consult SG16 on this paper regarding locales.

Revision 1:

- Add notice about locale.
- Remove `strtok(char* str, const char* delim, char** ptr)`.
- Add paragraph about ABI compatibility.
- Updated order functions in the proposal for matches the order in the [N4835].

VI. References

- [N4835] Working Draft, Standard for Programming Language C++. Available online at https://github.com/cplusplus/draft/raw/master/papers/n4835.pdf.
- [N1570] Committee Draft, Standard for Programming Language C. Available online at http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1570.pdf
- [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.