# Interfacing bitset with string_view

## Abstract

This paper proposes amending the interface of `bitset` to support construction from `basic_string_view`.

## Tony Table

| Before | | Proposed | |
|---|---|---|---|
| `bitset b0{""};` | ✔ | `bitset b0{""};` | ✔ |
| `bitset b1{""sv};` | ✘ | `bitset b1{""sv};` | ✔ |
| `bitset b2{""s};` | ✔ | `bitset b2{""s};` | ✔ |
| | | | |
| `//concerning LWG2946` | | `//concerning LWG2946` | |
| `bitset b3({"", 1});` | ✘ | `bitset b3({"", 1});` | ✘ |

## Revisions

**R0:** Initial version

## Motivation

[string.view] specifies `basic_string_view`, a vocabulary type template that represents an immutable reference to some string-like object. Unless a string can be moved from source to target, it is generally advisable to pass "immutable stringy inputs" by `basic_string_view`. Doing so obviates the need for multiple overloads and enables support for user-defined types.

[template.bitset] specifies the class templates `bitset` to represent a fixed size sequence of bits. It can be initialized from the biggest fundamental unsigned type (unsigned long `long int`) and a string. As `bitset` predates the introduction of `basic_string_view`, it only supports construction from strings of two forms: `const CharT *` and `basic_string<CharT, Traits, Allocator>`, with `CharT`, `Traits` and `Allocator` being deduced in the respective constructor and then promptly discarded, as `bitset` is independent of these types.

This leads to an embarrassing problem when following the aforementioned recommendation: Every `basic_string_view` must either be:

- converted to a temporary `basic_string`, introducing an unnecessary(!) copy as `bitset` only reads from the string for initialization, or
- extracted (via `.data()`); This approach places additional burden on the user as the respective `const CharT *` may not be \0-terminated and additional constructor parameters need to be provided to prevent an out of bounds access.

---

[1] RISC Software GmbH, Softwarepark 32a, 4232 Hagenberg, Austria, michael.hava@risc-software.at

This paper aims to solve these issues by introducing direct support for `basic_string_view`.

## Design space

This paper proposes to add a new constructor taking a `basic_string_view` to `bitset`. Contrary to other extensions to similar overload sets (e.g. P2495), LWG2946 does not apply here as all existing constructors of `bitset` are explicit.

## Impact on the Standard

This proposal is a pure library addition. One existing standard library class is modified in a non-ABI-breaking way. Overload resolution for existing code is not affected by the introduced overload.

## Implementation Experience

The proposed overload set has been implemented on [https://godbolt.org/z/56aaE3qP7] for evaluation.

## Proposed Wording

Wording is relative to [N4917]. Additions are presented like <mark>this</mark>, removals like <mark>this</mark>.

## [version.syn]

In [version.syn], add:

```
#define __cpp_lib_bitset YYYYMML //also in <bitset>
```

Adjust the placeholder value as needed to denote this proposal's date of adoption.

## [template.bitset.general]

In [template.bitset.general], in the synopsis, add the proposed overload:

```
...
// 22.9.2.2, constructors
constexpr bitset() noexcept;
constexpr bitset(unsigned long long val) noexcept;
template<class charT, class traits, class Allocator>
  constexpr explicit bitset(
    const basic_string<charT, traits, Allocator>& str,
    typename basic_string<charT, traits, Allocator>::size_type pos = 0,
    typename basic_string<charT, traits, Allocator>::size_type n = basic_string<charT, traits, Allocator>::npos,
    charT zero = charT('0'),
    charT one = charT('1'));
template<class charT>
  constexpr explicit bitset(
    const charT* str,
    typename basic_string<charT>::size_type n = basic_string<charT>::npos,
    charT zero = charT('0'),
    charT one = charT('1'));
template<class charT, class traits>
  constexpr explicit bitset(
    basic_string_view<charT, traits> sv,
    typename basic_string_view<charT, traits>::size_type pos = 0,
    typename basic_string_view<charT, traits>::size_type n = basic_string_view<charT, traits>::npos,
    charT zero = charT('0'),
    charT one = charT('1'));

// 22.9.2.3, bitset operations
...
```

## [bitset.cons]

In [bitset.cons]:

```
        template<class charT>
          constexpr explicit bitset(
            const charT* str,
```

```
              typename basic_string<charT>::size_type n = basic_string<charT>::npos,
              charT zero = charT('0'),
              charT one = charT('1'));
8     Effects: As if by:
          bitset(n == basic_string<charT>::npos
                     ? basic_string<charT>(str)
                     : basic_string<charT>(str, n),
                  0, n, zero, one)

    template<class charT, class traits>
      constexpr explicit bitset(
        basic_string_view<charT, traits> sv,
        typename basic_string_view<charT, traits>::size_type pos = 0,
        typename basic_string_view<charT, traits>::size_type n = basic_string_view<charT, traits>::npos,
        charT zero = charT('0'),
        charT one = charT('1'));
9     Effects: As if by:
          bitset(basic_string<charT, traits>(sv), pos, n, zero, one)
```

# Acknowledgements