Clause 21 (Strings Library) Issues List
Revision 1

Revision History

Introduction
This document is a summary of the issues identified in Clause 21. For each issue the status, a short description, and pointers to relevant reflector messages and papers are given. This evolving document will serve as a basis of discussion and historical for Strings issues and as a foundation of proposals for resolving specific issues.

Issues
Issue Number: 1
Title: Should basic_string have a getline() function?
Section: 21.1.1.4.5 (new) [lib.string::getline]
Status: active
Description:

As identified by Beman Dawes in lib-3367, the 20 September 1994 draft of the WP does not include getline(). It was part of the 27 May 1994 draft of the WP. Beman suggested that getline() be reinstated with the semantics as specified in the earlier WP draft.

In lib-3408, Nathan Myers responded as follows:
“I’m quite concerned about the semantics implied in the string traits. There, it seems to be assumed that the end-of-line character is the same for all encodings of a character type. But, of course, even in ASCII we see an amazing variety of line-end conventions. Unicode is worse, with all the ASCII control characters and (as I recall) two more line-end characters.

“I fear that we cannot provide internationalized getline semantics with the same interface that we have had. I can imagine a getline() which takes the user’s choice of line ending, but I can imagine you may want any of the available choices to end a line. The locale object’s ctype facet does not provide an ‘is_eol()’ member, and POSIX does not provide the underlying support necessary to implement it in any case.

“It seems clear to me that the getline operation depends on the character-encoding in use, and that makes it a locale-dependent operation. It is not clear to me how to propagate the information to the place where it is needed. It
would like to avoid a ‘virtual-function-call-per-character’ when reading lines of
text, because of performance problems.”

Resolution:
Requester: Beman Dawes: beman@dawes.win.net
Owner: (none)
Emails: lib-3367, lib-3408, lib-3411, lib-3417, lib-3421
Papers: (none)

Issue Number: 2
Title: Are string_traits members char_in() and char_out() necessary?
Section: 21.1.1.1 [lib.string.char.traits]
Status: active
Description:
In lib-3398, Nathan Myers writes:

Looking at Clause 21, Strings, I find some string_traits static members:

```cpp
static basic_istream<charT>
    string_char_traits::char_in(basic_istream<charT>& is,
        charT& a)
{ return is >> a; }

static basic_istream<charT>
    string_char_traits::char_out(basic_ostream<charT>& os,
        charT& a)
{ return os << a; }
```

Are they necessary? If so, shouldn’t they be parameterized on ios_traits? And
shouldn’t they default to use streambuf put() and get()?

[Note: lib-3398 contained a typo in which char_in() and char_out() were
incorrectly specified as being members of basic_string. The slight error is
corrected above.]

Resolution:
Requester: Nathan Myers: myersn@roguewave.com
Owner: Rick Wilhelm
Emails: lib-3398
Papers: (none)

Issue Number: 3
Title: Character-oriented assign function has incorrect signature
Section: 21.1.3.6 [lib.string::assign]
Status: active
Description:
As specified in N0557=94-0170, which was accepted in Valley Forge, the
character-oriented assign member has the interface:

```cpp
basic_string<T>& assign(size_type pos, size_type n, const T c = T());
```

This interface should not take have its first parameter. This change was
inadvertently introduced and should be removed.

Requester: Rick Wilhelm: rkw@chi.andersen.com
Owner: Rick Wilhelm
Issue Number: 3
Title: Character-oriented replace function has incorrect signature
Section: 21.1.3.9 [lib.string::replace]
Status: active
Description:
As specified in N0557=94-0170, which was accepted in Valley Forge, the character-oriented replace member has the interface:

```cpp
basic_string<T>&
replace(size_type pos, size_type n, const T c = T());
```

This interface should be as follows:

```cpp
basic_string<T>&
replace(size_type pos, size_type n1,
        size_type n2, const T c = T());
```

This change was inadvertently introduced and should be removed.
(This issue will be irrelevant and closed if 2.5.5 of N0628=95-0028 is accepted.)
Requester: Rick Wilhelm: rkw@chi.andersen.com
Owner: Rick Wilhelm
Emails: (none)
Papers: 95-0028=N0628

Issue Number: 5
Title: How come the string class does not have a prepend() function?
Section: 21.1.3.5 [lib.string::append]
Status: active
Description:
(No additional information at this time.)
Requester: Judy Ward: ward@roguewave.com
Owner: 
Emails: (none)
Papers: (none)