Doc: X3J16/96-0145=WG21/N0963
Date: July 9, 1996
By: Philippe Le Mouël

Stringstreams clarification

Discussion

The *strinbuf* model is different from both the *filebuf* and the *strstreambuf* models. When created in input mode, the *strinbuf* has to associate the input sequence with the underlying sequence of characters. The *streambuf eback*, *gptr,* and *egptr* pointers represent respectively the beginning, current position, and end of the input sequence. When created in output mode, the *stringbuf* has to associate the output sequence with the underlying sequence of characters. The streambuf *pbase*, *pptr,* and *epptr* pointers represent respectively the beginning, current position, and end of the output sequence. When created in both input and output mode, the *stringbuf* has to associate both the input sequence (*eback*, *gptr,* and *egptr*) and the output sequence (*pbase*, *pptr,* and *eppt*r) with the underlying sequence of characters. In this case, the input and output sequences are independent from each other as in the *strstreambuf* (in contrast to *filebuf,* where both sequences are tied together). An implementation may also have to maintain other information, like the end of the underlying character sequence, which may not be equal to the end of the input or output sequence.

Issue 27-702

Description:

The string streams are currently templatized on the character type (charT) and the traits type (ios_traits). String template parameters need to be added.

Proposed Resolution:

The Santa Cruz meeting fixes part of the problem by accepting doc: 96-0036R1=N0854R1 (Unification of Traits Revision1). But we are still left with the problem of taking or returning string arguments using a different allocator than the default. See *basic_stringbuf* , *basic_istringstream*, *basic_ostringstream,* and *basic_stringstream* constructors and *str* functions.

The solution is to add *Allocator* as a third template parameter to class *basic_stringbuf* , *basic_istringstream*, *basic_ostringstream,* and *basic_stringstream,* with a default value of *allocator<charT>*.

Changes to the WP:

In **27.7 String-based streams [lib.string.streams]**

change : template <class charT, class traits = char_traits<charT> >
class basic_stringbuf ;

to : template <class charT, class traits = char_traits<charT>, class Allocator = allocator<charT> >
class basic_stringbuf ;

change : template <class charT, class traits = char_traits<charT> >
class basic_istringstream;

to : template <class charT, class traits = char_traits<charT>, class Allocator = allocator<charT> >
class basic_istringstream;

change : template <class charT, class traits = char_traits<charT> >
class basic_ostringstream;

to : template <class charT, class traits = char_traits<charT>, class Allocator = allocator<charT> >
class basic_ostringstream;

add : template <class charT, class traits = char_traits<charT>, class Allocator = allocator<charT> >
class basic_stringstream;
typedef basic_stringstream<char> stringstream;
typedef basic_stringstream<wchar_t> wstringstream;

In **27.7.1 Template class basic_stringbuf [lib.stringbuf]**

change : template <class charT, class traits = char_traits<charT> >
class basic_stringbuf : public basic_streambuf<charT, traits > {

to : template <class charT, class traits = char_traits<charT>, class Allocator = allocator<charT> >
class basic_stringbuf : public basic_streambuf<charT, traits > {

In **27.7.1 Template class basic_stringbuf [lib.stringbuf]** and **27.7.1.1 basic_stringbuf constructors [lib.stringbuf.cons]**

change : explicit basic_stringbuf(const basic_string<char_type>& str, ios_base::openmode
which = ios_base::in | ios_base::out);

to : explicit basic_stringbuf(const basic_string<charT, traits, Allocator>& str,
ios_base::openmode which = ios_base::in | ios_base::out);

In **27.7.1 Template class basic_stringbuf [lib.stringbuf]** and **27.7.1.2 Member functions [lib.stringbuf.members]**

change : basic_string<char_type> str( ) const;

to : basic_string<charT, traits, Allocator> str( ) const;

change : void str(const basic_string<char_type>& s);

to : void str(const basic_string<charT, traits, Allocator>& s);

In **27.7.2 Template class basic_istringstream [lib.istringstream]**

change : template <class charT, class traits = char_traits<charT> >
class basic_istringstream : public basic_istream<charT, traits > {

to : template <class charT, class traits = char_traits<charT>, class Allocator = allocator<charT> >
class basic_istringstream : public basic_istream<charT, traits > {

change: // basic_stringbuf<charT,traits> sb; **exposition only**

to: // basic_stringbuf<charT, traits, Allocator> sb; **exposition only**

change: The class basic_istringstream<charT, traits> supports reading objects of class basic_string<charT, traits>. It uses a basic_stringbuf object to control the associated storage.

to : The class basic_istringstream<charT, traits, Allocator> supports reading objects of class basic_string<charT, traits, Allocator>. It uses a basic_stringbuf<charT, traits, Allocator> object to control the associated storage.

In **27.7.2 Template class basic_istringstream [lib.istringstream]** and **27.7.2.1 basic_istringstream constructors [lib.istringstream.cons]:**

change : explicit basic_istringstream(const basic_string<char_type>& str, ios_base::openmode
which = ios_base::in);

to : explicit basic_istringstream(const basic_string<charT, traits, Allocator>& str,
ios_base::openmode which = ios_base::in);

In **27.7.2 Template class basic_istringstream [lib.istringstream]** and **27.7.2.2 Member functions [lib.istringstream.members]:**

change : basic_string<charT> str( ) const;

to : basic_string<charT, traits, Allocator> str( ) const;

change : void str(const basic_string<charT>& s);

to : void str(const basic_string<charT, traits, Allocator>& s);

change : basic_stringbuf<charT, traits>* rdbuf( ) const;

to : basic_stringbuf<charT, traits, Allocator>* rdbuf( ) const;

In **27.7.2.3 Class basic_ostringstream [lib.ostringstream]:**

change : template <class charT, class traits = char_traits<charT> >
class basic_ostringstream : public basic_ostream<charT, traits > {

to : template <class charT, class traits = char_traits<charT>, class Allocator = allocator<charT> >
class basic_ostringstream : public basic_ostream<charT, traits > {

change: // basic_stringbuf<charT,traits> sb; **exposition only**

to: // basic_stringbuf<charT, traits, Allocator> sb; **exposition only**

change: The class basic_ostringstream<charT, traits> supports writing objects of class basic_string<charT, traits>. It uses a basic_stringbuf object to control the associated storage.

to : The class basic_ostringstream<charT, traits, Allocator> supports writing objects of class basic_string<charT, traits, Allocator>. It uses a basic_stringbuf<charT, traits, Allocator> object to control the associated storage.

In **27.7.2.3 Class basic_ostringstream [lib.ostringstream]** and **27.7.2.4 basic_ostringstream constructors [lib.ostringstream.cons]:**

change : explicit basic_ostringstream(const basic_string<char_type>& str, ios_base::openmode
which = ios_base::out);

to : explicit basic_ostringstream(const basic_string<charT, traits, Allocator>& str,
ios_base::openmode which = ios_base::out);

In **27.7.2.3 Class basic_ostringstream [lib.ostringstream]** and **27.7.2.5 Member functions [lib.ostringstream.members]:**

change : basic_string<charT> str( ) const;

to : basic_string<charT, traits, Allocator> str( ) const;

change : void str(const basic_string<charT>& s);

to : void str(const basic_string<charT, traits, Allocator>& s);

change : basic_stringbuf<charT, traits>* rdbuf( ) const;

to : basic_stringbuf<charT, traits, Allocator>* rdbuf( ) const;

In **27.7.3 Template class basic_stringstream [lib.stringstream]**

change : template <class charT, class traits = char_traits<charT> >
class basic_stringstream : public basic_iostream<charT, traits > {

to : template <class charT, class traits = char_traits<charT>, class Allocator = allocator<charT> >
class basic_stringstream : public basic_iostream<charT, traits > {

change: // basic_stringbuf<charT,traits> sb; **exposition only**

to: // basic_stringbuf<charT, traits, Allocator> sb; **exposition only** change: The class basic_stringstream<charT, traits> supports reading and writing from objects of class basic_string<charT, traits>. It uses a basic_stringbuf<charT, traits> object to control the associated storage.

to : The class basic_stringstream<charT, traits, Allocator> supports reading and writing objects of class basic_string<charT, traits, Allocator>. It uses a basic_stringbuf<charT, traits, Allocator> object to control the associated storage.

In **27.7.3 Template class basic_stringstream [lib.stringstream]** and **27.7.4 basic_stringstream constructors [lib.stringstream.cons]:**

change : explicit basic_stringstream(const basic_string<charT>& str, ios_base::openmode which = ios_base::out | ios_base::in);

to : explicit basic_ostringstream(const basic_string<charT, traits, Allocator>& str, ios_base::openmode which = ios_base::out | ios_base::in);

In **27.7.3 Template class basic_stringstream [lib.stringstream]** and **27.7.5 Member [functions]:**

change : basic_string<charT> str( ) const;

to : basic_string<charT, traits, Allocator> str( ) const;

change : void str(const basic_string<charT>& s);

to : void str(const basic_string<charT, traits, Allocator>& s);

change : basic_stringbuf<charT, traits>* rdbuf( ) const;

to : basic_stringbuf<charT, traits, Allocator>* rdbuf( ) const;

Issue 27-701

Description:

"Table 15 in [lib.stringbuf.members] describes the return values of basic_stringbuf::str(). What does the "otherwise" mean?. Does it mean neither ios_base::in nor ios_base::out is set? What is the return value supposed to be if _both_ bits are set?"

Proposed Resolution:

The description of function basic_string<charT, traits, Allocator> str( ) const; should be:

Returns: A *basic_string* object which contents is equal to the *basic_stringbuf* underlying character sequence. If the buffer is only created in input mode, the underlying character sequence is equal to the input sequence;

otherwise, it is equal to the output sequence. In case of an empty underlying character sequence, the function returns basic_string<charT, traits, Allocator> ( ).

Note: the table has to be removed.

Issue 27-703

Description:

basic_stringbuf::str(basic_string s) Postconditions requires that str() == s. This is true only if which had in set at construction time. Condition should be restated.

Proposed Resolution:

The description of function void str(const basic_string<charT, traits, Allocator>& s); should be:

Effects: If the *basic_stringbuf* underlying character sequence is not empty, deallocats it. Then copies the content of *s* into the *basic_stringbuf* underlying character sequence and initializes the input and output sequences according to the mode stored when creating the *basic_stringbuf* object. If *(mode & ios_base::out)* is true, initializes the output sequence with the underlying sequence. If *(mode & ios_base::in)* is true, initializes the input sequence with the underlying sequence.

Postcondition: *str( ) == s*.

Note: the table has to be removed.

Issue 27-704

Description:

basic_stringbuf::basic_stringbuf(basic_string str, openmode which) Postconditions requires that str() == str. This is true only if which has in set. Condition should be restated.

Proposed Resolution:

The description of constructor explicit: basic_stringbuf(const basic_string<charT, traits, Allocator>& str,
ios_base::openmode which = ios_base::in | ios_base::out);
should be:

Effects: Constructs an object of class *basic_stringbuf*, initializing the base class with *basic_streambuf( )* (27.5.2.1), and initializing *mode* with *which*. Then copies the content of *str* into the *basic_stringbuf* underlying character sequence and initializes the input and output sequences according to *which*. If *(which & ios_base::out)* is true, initializes the output sequence with the underlying sequence. If *(which & ios_base::in)* is true, initializes the input sequence with the underlying sequence.

Postcondition: *str( ) == str.*

Note: the table has to be removed.

Issues 27-705 and 706

Solved by proposed resolutions for issues 27-703 and 704.

basic_stringbuf::seekpos 27.7.1.3 Overridden virtual functions

The description of function: pos_type seekpos(pos_type sp, ios_base::openmode which = ios_base::in | ios_base::out);

should be :

Effects: Alters the stream position within the controlled sequences, if possible, to correspond to the stream position stored in *sp* (as described below).

- if *(which & ios_base::in) != 0*, position the input sequence.
- if *(which & ios_base::out) != 0*, position the output sequence.

If *sp* is an invalid stream position, or if the function positions neither sequence, the positioning operation fails. If *sp* has not been obtained by a previous successful call to one of the positioning functions (*basic_stringbuf::seekoff, basic_stringbuf::seekpos, basic_istream::tellg, basic_ostream::tellp*), no validity of the operation is ensured.

Returns: *sp* to indicate success, or pos_type(off_type(-1)) to indicate failure.