Doc: X3J16/96-0128=WG21/N0946
Date: June 10, 1996
By: Philippe Le Mouël

Pre-Stockholm
iostreams WG proposals

1) 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> ( ).

If we feel that a table is still necessary, we should modify the existing one to:

Table X—str return values

Condition	**Setting**
(mode & ios_base::out & ios_base::in)!=0 and (pptr( ) != 0)	basic_string<charT, traits, Allocator >(pbase(), epptr() - pbase())
(mode & ios_base::out)!=0 and (pptr( ) != 0)	basic_string<charT, traits, Allocator >(pbase(), epptr() - pbase())
(mode & ios_base::in)!=0 and (gptr( ) != 0)	basic_string<charT, traits, Allocator >(eback(), egptr() - eback())
Otherwise	basic_string<charT, traits, Allocator > ( )

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 if *s.length()* is zero, executes:

setg(0,0,0);
setp(0,0);

Otherwise, if *s.length() > 0*, 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.*
If *str.length( )* is zero, executes:

setg(0,0,0);
setp(0,0);

Otherwise, if *str.length()* > 0, 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 outp 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 should 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.

2) Others issues

Issue 27-206

Description:

The function *clear()* should set badbit (independent of its argument) if *rdbuf()* returns null.
Note: This means *clear()* must not be moved to ios_base; it must remain in basic_ios.

Proposed Resolution:

The description of function void clear(iostate state = goodbit); should be:

Postcondition: If rdbuf() != 0, rdstate( ) == state; otherwise, rdstate( ) == state | ios_base::badbit.

Effects: If (rdstate( ) & exceptions( )) == 0, returns. Otherwise, the function throws an object of class ios_base::failure (27.4.3.1.1), constructed with implementation-defined argument values.

This means that the following functions have to stay in basic_ios:

- void clear(iostate state = goodbit);
- void setstate(iostate state);
- iostate rdstate( ) const;
- bool good( ) const;
- bool eof( ) const;
- bool fail( ) const;
- bool bad( ) const;
- operator bool( ) const;
- bool operator! ( ) const;

The following functions have to be moved from ios_base to basic_ios:

- iostate exceptions( ) const;
- void exceptions(iostate except);

Issue 27-601

Description:

The ios_base manipulators 27.4.5.1**[lib.std.ios.manip]** will not work as written. They won't work because there is no conversion from ios_base to basic_ios.

Proposed Resolution:

Add to basic_istream:

basic_istream<charT, traits>& operator>>(ios_base& (*pf)(ios_base&));

Effects: Calls (*pf)(*this)
Returns: *this.

Add to basic_ostream:

basic_ostream<charT, traits>& operator<<(ios_base& (*pf)(ios_base&));

Effects: Calls (*pf)(*this)
Returns: *this.

Change footnote 9 in 27.4.5.3 **basefield manipulators [lib.basefield.manip]** to:

The function signature dec(ios_base& str) can be called by the function signature basic_ostream<charT,traits>& basic_ostream<charT,traits>::operator << (ios_base& (*)(ios_base&)) to permit expressions of the form cout << dec to change the format flags stored in cout.

Issue 27-815

Description:

basic_filebuf::seekpos has no semantics. Needs to be supplied.

Proposed Resolution:

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 file position, if possible, to correspond to the position stored in *sp* (as described below).

- if *(which & ios_base::in) != 0*, set the file position to *sp*, then update the input sequence.
- if *(which & ios_base::out) != 0*, set the file position to *sp*, then update 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_filebuf::seekoff, basic_filebuf::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.

Issue 27-816

Description:

(i)(o)fstream *open* functions should not use *is_open* to determine if the operation fails (and as a result setting *failbit*). The problem arises if you do not close the (i)(o)fstream and then try to open another file with it. In this case the *filebuf open* function will fail, but *is_open* will still return true.

Proposed Resolution:

In **27.8.1.6 basic_ifstream constructors [lib.ifstream.cons]:**

change : explicit basic_ifstream(const char* s, openmode mode = in);

> **Effects:** Constructs an object of class basic_ifstream, initializing the base class with basic_istream(&sb) and initializing sb with basic_filebuf<charT,traits>( ) (27.6.1.1.1, 27.8.1.2), then calls rdbuf( )->open(s, mode).

to : explicit basic_ifstream(const char* s, openmode mode = in);

> **Effects:** Constructs an object of class basic_ifstream, initializing the base class with basic_istream(&sb) and initializing sb with basic_filebuf<charT,traits>( ) (27.6.1.1.1, 27.8.1.2), then calls rdbuf( )->open(s, mode). If that function returns a null pointer, calls setstate(failbit), (which may throw ios_base::failure).

In **27.8.1.7 Member functions [lib.ifstream.members]:**

change: void open(const char* s, openmode mode = in);

> **Effects:** Calls rdbuf( )->open(s, mode). If is_open( ) returns false, calls setstate(failbit), (which may throw ios_base::failure (27.4.4.3)).

to: void open(const char* s, openmode mode = in);

> **Effects:** Calls *rdbuf()->open(s,mode).* If that function returns a null pointer, calls *setstate(failbit)* (which may throw *ios_base::failure).*

In **27.8.1.9 basic_ofstream constructors [lib.ofstream.cons]:**

change : explicit basic_ofstream(const char* s, openmode mode = out);

> **Effects:** Constructs an object of class basic_ofstream, initializing the base class with basic_ostream(&sb) and initializing sb with basic_filebuf<charT,traits>( ) (27.6.1.1.1, 27.8.1.2), then calls rdbuf( )->open(s, mode).

to : explicit basic_ofstream(const char* s, openmode mode = out);

> **Effects:** Constructs an object of class basic_ofstream, initializing the base class with basic_ostream(&sb) and initializing sb with basic_filebuf<charT,traits>( ) (27.6.1.1.1, 27.8.1.2), then calls rdbuf( )->open(s, mode). If that function returns a null pointer, calls setstate(failbit), (which may throw ios_base::failure).

In **27.8.1.10 Member functions [lib.ofstream.members]:**

change: void open(const char* s, openmode mode = out);

> **Effects:** Calls rdbuf( )->open(s, mode). If is_open( ) returns false, calls setstate(failbit), (which may throw ios_base::failure (27.4.4.3)).

to: void open(const char* s, openmode mode = out);

> **Effects:** Calls *rdbuf()->open(s,mode).* If that function returns a null pointer, calls *setstate(failbit)* (which may throw *ios_base::failure).*

In **27.8.1.11 basic_fstream constructors [lib.fstream.cons]**

change : explicit basic_fstream(const char* s, ios_base::openmode mode);

> **Effects:** Constructs an object of class basic_fstream, initializing the base class with basic_iostream(&sb) and initializing sb with basic_filebuf<charT,traits>( ) (27.6.1.1.1, 27.8.1.2), then calls rdbuf( )->open(s, mode).

to : explicit basic_fstream(const char* s, ios_base::openmode mode);

> **Effects:** Constructs an object of class basic_fstream, initializing the base class with basic_iostream(&sb) and initializing sb with basic_filebuf<charT,traits>( ) (27.6.1.1.1, 27.8.1.2), then calls rdbuf( )->open(s, mode). If that function returns a null pointer, calls setstate(failbit), (which may throw ios_base::failure).

In **27.8.1.13 Member functions [lib.fstream.members]**

change: void open(const char* s, ios_base::openmode mode);

> **Effects:** Calls rdbuf( )->open(s, mode). If is_open( ) returns false, calls setstate(failbit) (which may throw ios_base::failure (27.4.4.3))

to: void open(const char* s, ios_base::openmode mode);

> **Effects:** Calls *rdbuf()->open(s,mode).* If that function returns a null pointer, calls *setstate(failbit),* (which may throw *ios_base::failure).*

Issue 27-919

Description:

27.1.2.4 **[lib.iostreams.pos.t]**: table 2: first row has assertion "p == P(i)" but p does not appear in the expression for that row; also, that row has the note "a destructor is assumed" -- what does this mean?

The first row of table 2 should be deleted. The second row already specifies the construction and assignment from an integer value.

The pre-Stockholm iostreams WG recommends accepting the above resolution. A larger issue is that the table was voted out of the iostreams chapter as part of traits consolidation, but needs to be included somewhere. The discussions of OFF_T and POS_T should be consolidated in the iostreams chapter, with a note added to the string chapter referring to the iostreams chapter.

Proposed Resolution:

In **21.1.4 traits typedefs [lib.char.traits.typedefs]:**

Change the description of *typedef OFF_T off_type*; to:

typedef OFF_T off_type;

Requires: Used by the iostreams classes to represent a signed displacement relative to a specified position within a sequence, or an absolute position within a sequence. The type or class OFF_T is used to define off_type. The description and properties of OFF_T are described in section 27.1.2.3 and in the Table 2-position type requirements in chapter 27.

Change the description of *typedef POS_T pos_type*; to:

typedef POS_T pos_type;

Requires: Used by the iostreams classes to represent a positional information. The type or class POS_T is used to define pos_type. The description and properties of POS_T are described in section 27.1.2.4 and in the Table 2-position type requirements in chapter 27.

Change the description of *typedef STATE_T state_type*; to:

typedef STATE_T state_type;

Requires: Used by the iostreams classes to represent the conversion state type or class which is applied to the *codecvt*< > facet defined in chapter 22. The type or class STATE_T is used to define state_type, and is described in section 27.1.2.6 in chapter 27.

Keep sections **27.1.2.3 Type OFF_T, 27.1.2.4 Type POS_T** and **27.1.2.6 Type STATE_T** unchanged.

Remove the first row of **Table 2—Position type requirements.**