WG21/N0930
X3J16/96-0112

Thomas Plum
[Revision #1] May 26, 1996

Locales [clause 22] active issues list

After the Santa Cruz decisions, there remained three active issues
in the Locales issues list:

22-004, 22-046, 22-057.

Refer to N0889=96-0073 (Santa Cruz mid-meeting paper) for cross-check.

Since then, more issue(s) have been added:

22-070

Those 4 issues above were distributed in email lib-4569 .

One more issue is now added:

22-071

Please let Plum, or the -lib reflector, know if you see any errors. Thanks.

```
** Work Group:     Library: Localization Clause 22
** Issue Number:   22-004
** Title:          Description of _byname facets too vague
** Sections:       [lib.locale.facet]
** Status:         active
```

** Description:
Paragraph 4, where _byname<> classes are described, leaves some
(unspecified) issues unresolved.

** Discussion:

** Proposed Resolution:

** Requestor: ANSI public comment
** Owner:

```
** Work Group:     Library: Localization Clause 22
** Issue Number:   22-046
** Title:          Locale facets for money/time/messages need semantics
** Sections:       22.2.4
** Status:         active
```

** Description:
>From Plauger:
  Template classes collate, time_get, time_put, money_get, money_put,
  money_punct, messages, and their support classes still have only
  sketchy semantics -- over a year after they were originally
  accepted into the draft. They are based on little or no prior
  art, and they present specification problems that can be addressed
  properly only with detailed descriptions, which do not seem to be
  forthcoming. Even if adequate wording were to magically appear
  on short notice, the public still deserves the courtesy of a
  proper review. For all these reasons, and more, the remainder

of clause 22 from this point on should be struck.

** Discussion:
>From Plum:
I believe the status quo is that these classes essentially define
the syntax of an interface, with relatively little detail regarding
semantics. At this point, any proposal to add further
semantic restrictions would be a substantive change to the CD,
requiring an explicit proposal.

The text of this issue itself contains a proposal to strike the
classes itemized above. In order to reach closure on the issue,
that proposal should be addressed by the Library group next meeting.
My opinion is that removing library classes from the CD will be
considered to be a more drastic course than providing an interface
with no required semantics behind it, and that the status quo will
therefore remain in place.

** Proposed Resolution:

** Requestor: Plauger
** Owner: Plum

```
** Work Group:     Library: Localization Clause 22
** Issue Number:   22-057
** Title:          money_put<> and _get<> facets missing from table
** Sections:       [lib.locale.category], [lib.locale.money.get],
                   [lib.locale.money.put]
** Status:         active
```

** Description:

In the table of locale facet categories, and the facets that
are required to be provided as part of each, the facets
money_[get|put]<[char|wchar_t],true> are not mentioned.
They either need to be added, or the second template parameter
eliminated in favor of an argument on the put() and get()
members.

[This part has already been added to the WP, but the proposals
below have not been decided yet.]

** Discussion:
It is easy to imagine code in which the choice of local
or international monetary format is made at runtime, so
a template parameter to num_put and num_get would make
such code unnecessarily clumsy.

** Proposed Resolution:
In both money_put<> and money_get<> (but not in moneypunct<>):

1. Eliminate the second template parameter "bool Intl = false",
   and the static const member "bool intl".

2. Add to the members money_get<>::get() and money_put<>::put()
   another argument, immediately after the "const locale&" argument,
   of type bool:

```
      ...const locale&, const bool intl, ...
```

   Specify that if true, the function uses the moneypunct<charT,true>
   facet, else it uses the moneypunct<charT,false> facet, of the locale
   argument to control [parsing|formatting].

```
** Requestor:
** Owner:          Plum

----------------

** Work Group:     Library: Localization Clause 22
** Issue Number:   22-070
** Title:          Desires  conversion_type()  function
** Sections:       [lib.locale.facet]
** Status:         active
```

In paper WG21/N0859 = X3J16/96-0041 Locale codecvt<> shift-state Predicate,
Nathan wrote:

> The Standard C Library function mblen() (found in <stdlib.h>), when called
> with a magic combination of argument values, reveals whether the character
> set encoding has "State-dependent encodings". The C++ locale facilities
> currently provide no corresponding semantics.

> Filebuf needs this information for uses that include helping determine
> whether seeking is possible on a sequence using the encoding. The simplest
> way to provide this information is by adding a predicate member to codecvt<>.

I think this is a good idea, but I would like to be able to obtain also
another type of information from the codecvt<> facet. I would like to know
if the conversion is what I call a "constant conversion" and if yes what is
its size.

For intance if I have an extern file in unicode and I want to convert it
internaly as EUC complete two-byte format. I know that the conversion is
a "constant conversion". Each external unicode character as a size of two
bytes and each internal EUC character as a size of two bytes. Therefore
the filebuf underflow function needs only to read two tiny characters
for each internal EUC character and the filebuf overflow function will
have the same ratio.

These extra informations, allow you to seek and insert characters in streams
using "constant conversion". The filebuf implementation can also be more
efficient for these kind of conversions.

-- Proposed Resolution:

Add the following member functions to the locale codecvt<> facet:

```
protected:

virtual int do_conversion_type() const throw();
```

Returns: -1 if the encoding applied to *from* sequences has state-dependent
  encoding; otherwise the constant number of fromT (or toT) character(s) needed
  to produce a toT (or fromT) character or 0 if this number is not a constant.

```
public:

int conversion_type() const throw();
```

Returns: do_conversion_type().

-- end of proposed resolution

```
Note: if fromT is char and toT is wchar_t
      JIS to UNICODE returns -1
      UNICODE to EUC complete two-byte format returns 2
      shift JIS to UNICODE returns 0
```

** Requestor: le Mouel
** Owner: Plum

** Work Group: Library: Localization Clause 22
** Issue Number: 22-071
** Title: messages_base isn’t really needed
** Sections: [lib.locale.messages]
** Status: active

** Description:
The messages_base class contains a single typedef:

```
   class messages_base {
   public:
     typedef int catalog;
   };
```

There is no need for this typedef to be in its own class, nor is there a
need for this base class.

** Discussion:

** Proposed Resolution:
Move the catalog typedef into the messages class and remove the messages_base
class, as follows:

```
   template <class charT>
   class messages : public locale::facet {
   public:
      <existing typedefs...>
      typedef int catalog;
      ...
   };
```

** Further discussion [from Nathan Myers]:

I would suggest instead:

Eliminate the base class messages_base, and eliminate the
typedef as well; replace any argument or return types that
say "catalog" with "int".

** Requestor:
"Cathy Kimmel Joly, Language RTLs, 381-0247" <kimmel@cxxc.ENET.dec.com>
** Owner: Plum