Document number N2884=09-0074
Date: 2009-06-10
Project: Programming Language C++, Library Working Group
Reply-to: Beman Dawes <bdawes at acm.org>

C++0x Stream Positioning

Solutions for library issues:
573: C++0x file positioning should handle modern file sizes
255: Why do basic_streambuf<>::pbump() and gbump() take an int?

Introduction

C and C++ I/O streams predate modern file sizes, which may be too large to be represented by an `int` or a `long`, and predate the introduction of `long long` into the languages. That has resulted in several issues addressed by this paper.

Library issue 255, *Why do basic_streambuf<>::pbump() and gbump() take an int?*, identifies a problem:

- `pbump()` and `gbump()` needs to take an argument type that supports modern file sizes.

Library issue 573, *C++0x file positioning should handle modern file sizes*, boils down to two problems:

- The specification of types relating to file sizes, positions, and offsets (`fpos_t`, `fpos`, `pos_type`, `off_type`, `streamoff`, `OFF_T`, `streamsize`, `SZ_T`, `streampos`, `wstreampos`, and perhaps more) are so intertwined and difficult to follow that understanding is very difficult.
- It isn't clear that `off_type` (also known as `streamoff` and `OFF_T`) is currently required to support modern file sizes.

In committee reflector message c++std-lib-24002, Howard Hinnant identified a further problem:

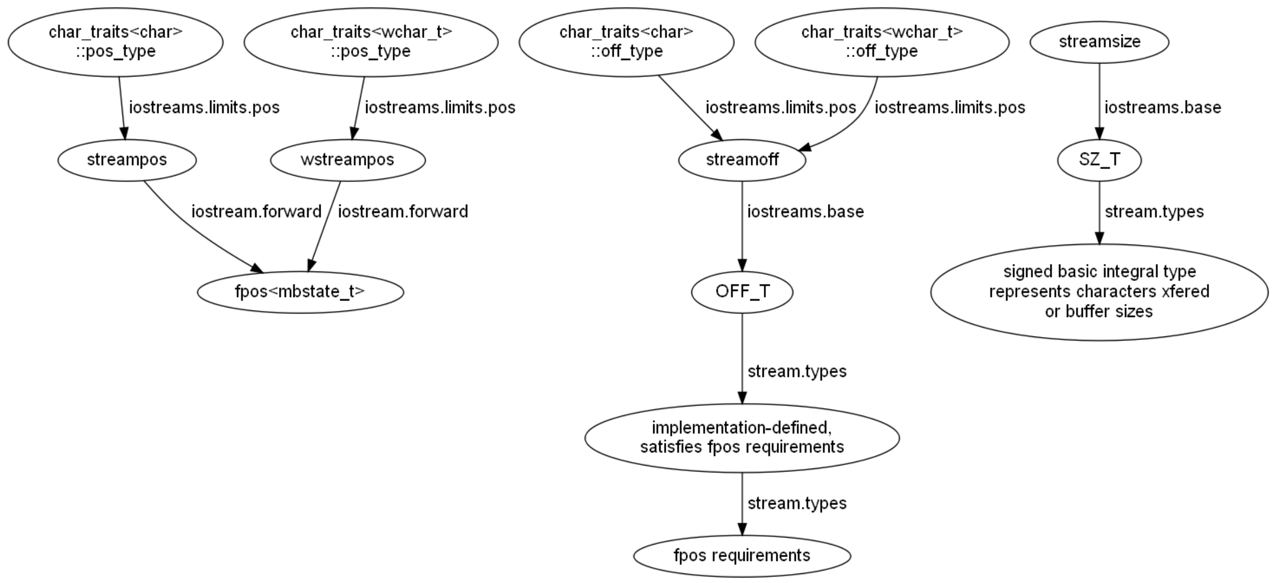
- [filebuf.virtuals]/13, in specifying seek offset for `filebuf`'s, mandate use of `std::fseek`, which specified offset via a `long`. This will result in truncation, and thus wrong effects for large files on systems where `off_type` is `long long`.

This proposal attempts to resolve all of these problems in a consistent way. Solutions are based on existing practice in several current standard library implementations, although no current implementation implements all of the changes. The changes often affect the way the standard is specified rather than actual interfaces.

As far as is know, the changes will break no existing user code.

Stream Position, Offset, and Size Types

The key to resolving the above problems is to understand the specification of stream position, offset, and size types:



Stream Position and Size Types in the Working Paper

A line from **A** to **B** indicates that **A** is a name for (i.e. typedef or specialization), or is defined in terms of, B.

Several simplifications and clarifications are possible:

- `OFF_T` can be replaced by `streamoff`. `OFF_T` is used only in three places in the `fpos` operations table, and replacement by `streamoff` will increase clarity.
- The requirements for the `streamoff`/`OFF_T` type currently must be deduced indirectly from the `fpos` operational requirements. That is very roundabout, hard to understand, and gives little hint as to the maximum file sizes that must be supported. It is much simpler and clearer to state the `streamoff` size requirements directly.
- `SZ_T` can be eliminated. It is never referenced.
- `streamsize`/`SZ_T` is specified as "signed integral basic type", which isn't a defined term. That should be "signed integer type".
- The addition of a non-normative figure and text illustrating the relationship between types is proposed to make these relationships easier to understand.
- The addition of example code is proposed to show how to position a stream at a location expressed as a `long long`.

Proposed Wording

Change 21.2.2 traits typedefs [char.traits.typedefs] as indicated:

```
typedef ~~OFF_T~~ unspecified-type off_type;
typedef ~~POS_T~~ unspecified-type pos_type;
```

Requires: Requirements for `off_type` and `pos_type` are described in 27.2.2.

Change 21.2.3.1 struct char_traits<char> [char.traits.specializations.char] as indicated:

```
typedef streamoff off_type;
typedef streampos pos_type;
...
```

The type `streampos` shall be an implementation-defined type that satisfies the requirements for ~~`POS_T`~~ `pos_type` in 21.2.2.

The type `streamoff` shall be an implementation-defined type that satisfies the requirements for ~~`OFF_T`~~ `off_type` in 21.2.2.

At the end of 27.1 General [input.output.general], add:

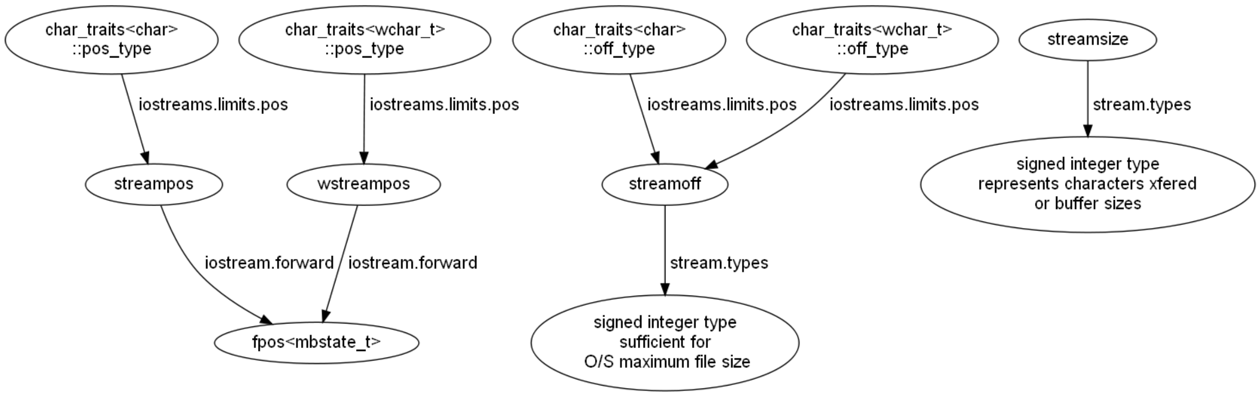

char_traits<char> ::pos_type
char_traits<wchar_t> ::pos_type
char_traits<char> ::off_type
char_traits<wchar_t> ::off_type
streamsize
iostreams.limits.pos
stream.types
streampos
wstreampos
streamoff
signed integer type represents characters xfered or buffer sizes
iostream.forward
signed integer type sufficient for O/S maximum file size
fpos<mbstate_t>

Figure 1: Stream position, offset, and size types [non-normative]

Figure 1 illustrates relationships among various types described in this clause. A line from **A** to **B** indicates that **A** is an alias (e.g., a typedef), or that **A** is defined in terms of, **B**.

Change 27.5 Iostreams base classes [iostreams.base], Header <ios> synopsis, as indicated:

typedef ~~`OFF_T`~~ ***implementation-defined*** `streamoff;`
typedef ~~`SZ_T`~~ ***implementation-defined*** `streamsize;`

Change 27.5.1 Types [stream.types] as indicated:

typedef ~~`OFF_T`~~ ***implementation-defined*** `streamoff;`

The type `streamoff` is ~~an implementation-defined type that satisfies the requirements of 27.5.3.2.~~ a synonym for a signed integer type of sufficient size to represent the maximum possible file size for the operating system.[footnote]

footnote) Typically `long long`.

typedef ~~`SZ_T`~~ ***implementation-defined*** `streamsize;`

The type `streamsize` is a synonym for ~~one of the signed basic integral types~~ a signed integer type. It is used to represent the number of characters transferred in an I/O operation, or the size of I/O buffers.[footnote]

footnote) `streamsize` is used in most places where ISO C would use `size_t`. Most of the uses of `streamsize` could use `size_t`, except for the `strstreambuf` constructors, which require negative values. It ~~should probably be~~ is typically the signed type corresponding to `size_t` (which is what Posix.2 calls `ssize_t`).

Change 27.5.3.2 fpos requirements [fpos.operations], Position type requirements, as indicated:

Expression	Return Type	Operational semantics	Assertion/note pre-/post-condition
`O(p)`	~~`OFF_T`~~ `streamoff`	converts to `offset`	`P(O(p)) == p`
`o = p - q`	~~`OFF_T`~~ `streamoff`	distance	`q + o == p`

`streamsize(o)`	`streamsize`	converts	`streamsize(O(sz)) == sz`
`O(sz)`	`~~OFF_T~~ streamoff`	converts	`streamsize(O(sz)) == sz`

At the end of 27.5.3.2 fpos requirements [fpos.operations], add:

[Example:

```
  // open a file
  std::fstream file("test.file", std::ios_base::in | std::ios_base::binary);

  // seek to position 10 000 000 000 by passing a streamoff
  file.seekg(10000000000LL, std::ios_base::beg);
  ...

  // seek to position 10 000 000 000 passing a streampos
  //  constructed from a streamoff
  file.seekg(std::streampos(10000000000LL));
  ...
```

--end example]

Change 27.6.2 Class template basic_streambuf<charT,traits> [streambuf], basic_streambuf synopsis as indicated:

```
void gbump(~~int~~ streamsize n);
...
void pbump(~~int~~ streamsize n);
```

Change 27.6.2.3.2 Get area access [streambuf.get.area] as indicated:

```
void gbump(~~int~~ streamsize n);
```

Effects: Adds n to the next pointer for the input sequence.

Change 27.6.2.3.3 Put area access [streambuf.put.area] as indicated:

```
void pbump(~~int~~ streamsize n);
```

Effects: Adds n to the next pointer for the output sequence.

Change 27.9.1.5 Overridden virtual functions [filebuf.virtuals], paragraph 13, as indicated:

```
pos_type seekoff(off_type off, ios_base::seekdir way,
    ios_base::openmode which = ios_base::in | ios_base::out);
```

Effects: Let `width` denote `a_codecvt.encoding()`. If `is_open() == false`, or `off != 0 && width <= 0`, then the positioning operation fails. Otherwise, if `way != basic_ios::cur` or `off != 0`, and if the last operation was output, then update the output sequence and write any unshift sequence. Next, seek to the new position: if `width > 0`, ~~call~~ as if by calling ~~`std::fseek`~~ *`seekfunc`*`(file, width * off, whence)`, otherwise ~~call~~ as if by calling ~~`std::fseek`~~ *`seekfunc`*`(file, 0, whence)`, where *`seekfunc`* has the same behavior as `std::fseek` except having a second argument type of `off_type`.

Change D.6 Old iostreams members [depr.ios.members] as indicated:

```
    typedef ~~OFF_T~~ implementation-defined streamoff;
    typedef ~~POS_T~~ implementation-defined streampos;
```

The type `streamoff` is an implementation-defined type that satisfies the requirements of type ~~OFF_T~~ `streamoff` ([stream.types]).

The type `streampos` is an implementation-defined type that satisfies the requirements of type ~~POS_T~~ `streampos` ([iostream.forward]).

Other possible changes

Issue 573 also raised the question of adding additional member functions to `fpos`. In light of the proposed WP changes above, that does not appear to be necessary and is not proposed here.

Implementation Experience

Microsoft VC++ 2010 beta 1 implements the proposed wording. The following program runs without error on VC++ 2010 beta 1, but reports failures with earlier releases that did not implement the proposed wording:

```
#include <fstream>
#include <iostream>
#include <iosfwd>

const long long max = 800000000LL;

int main()
{
  std::fstream file("test.file",
  std::ios_base::in | std::ios_base::out | std::ios_base::binary | std::ios_base::trunc);
  if ( !file )
    std::cout << "Could not open test.file\n";

  // create the test file
  for (long long i = 0; i < max; ++i)
    file.write(reinterpret_cast<char*>(&i), sizeof(i));

  // test seekg with offset
  long long x;
  file.seekg((max-1)*sizeof(x), std::ios_base::beg);
  file.read(reinterpret_cast<char*>(&x), sizeof(x));
  if (x!=(max-1))
    std::cout << "seekg with offset failed to position the file correctly\n";

  // test seekg with pos_type
  std::fstream::pos_type pos((max-2)*sizeof(x));
  file.seekg(pos);
  file.read(reinterpret_cast<char*>(&x), sizeof(x));
  if (x!=(max-2))
    std::cout << "seekg with pos_type failed to position the file correctly\n";

  return 0;
}
```

Acknowledgements

Alberto Ganesh Barbati, Walter Brown, Howard Hinnant, Daniel Krügler, Bill Plauger, Jerry Schwarz, and Martin Sebor contributed helpful comments and suggestions.