What is the type of std::cout?

Where does a stream go?

Paul and Jez's Stream-a-poloza

Paul Grenyer
paul@paulgrenyer.co.uk

Jez Higgins
jez@jezuk.co.uk

Let's talk about you ...

Let's talk about you ...

In the last three months, have you used ...

- std::string

Let's talk about you ...

In the last three months, have you used ...

- std::string
- std::vector

Let's talk about you ...

In the last three months, have you used ...

- std::string
- std::vector
- std::map

Let's talk about you ...

In the last three months, have you used ...

- std::string
- std::vector
- std::map
- an iterator

Let's talk about you ...

In the last three months, have you used ...

- std::string
- std::vector
- std::map
- an iterator
- an algorithm

Let's talk about you ...

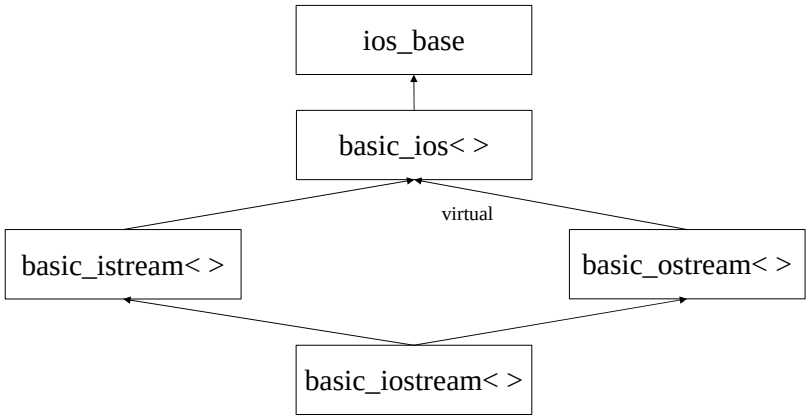
In the last three months, have you used ...

- std::string
- std::vector
- std::map
- an iterator
- an algorithm
- operator<<

Let's talk about you ...

In the last three months, have you used ...

- std::string
- std::vector
- std::map
- an iterator
- an algorithm
- operator<<
- a custom stream



- Streams the write

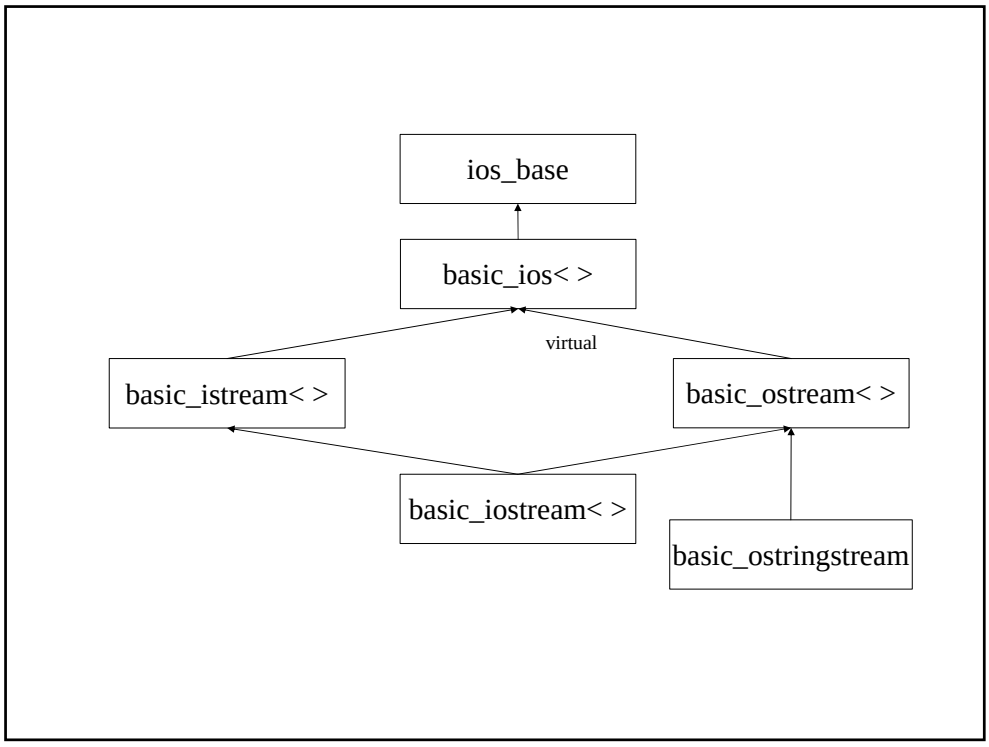
```
std::cout << “Hello, world!”;
```

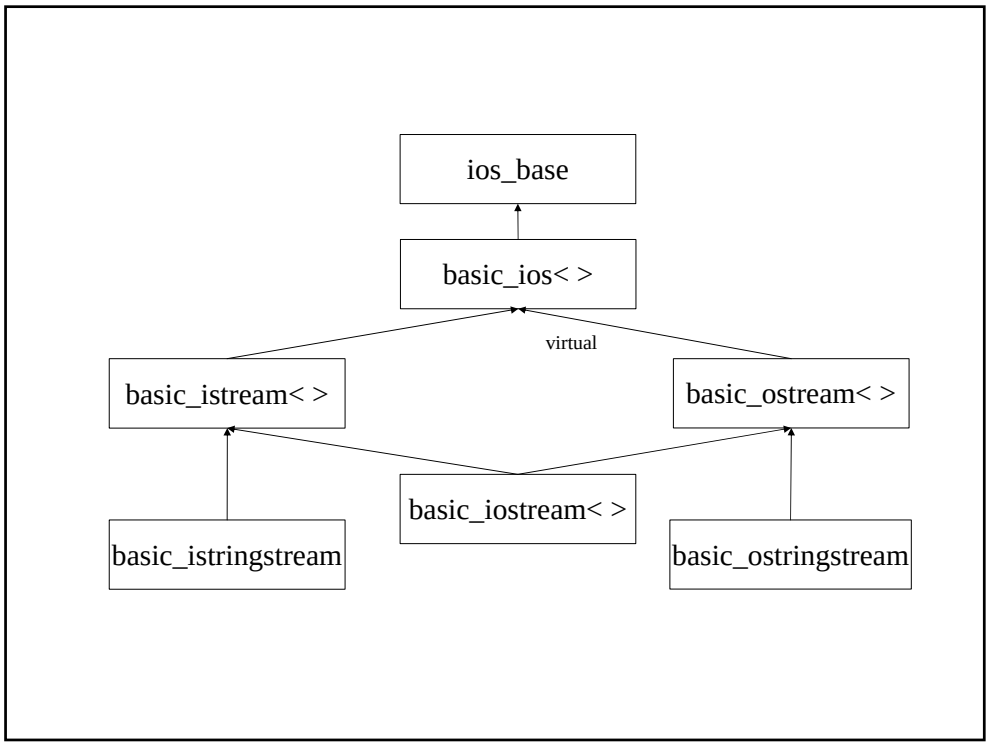
- Stream the read

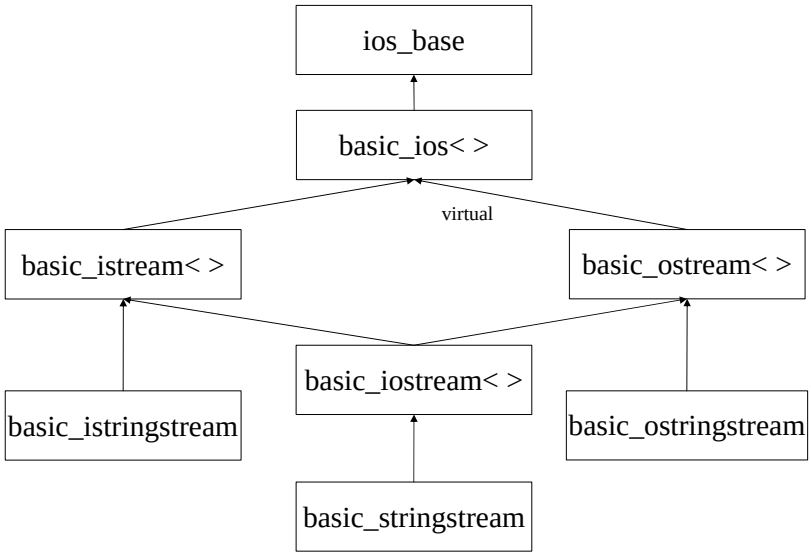
```
int n;
std::cin >> n;
```

- Streams that do both

```
std::stringsteam ss;
...
ss << “Hello, world!”;
...
ss >> some_string;
```

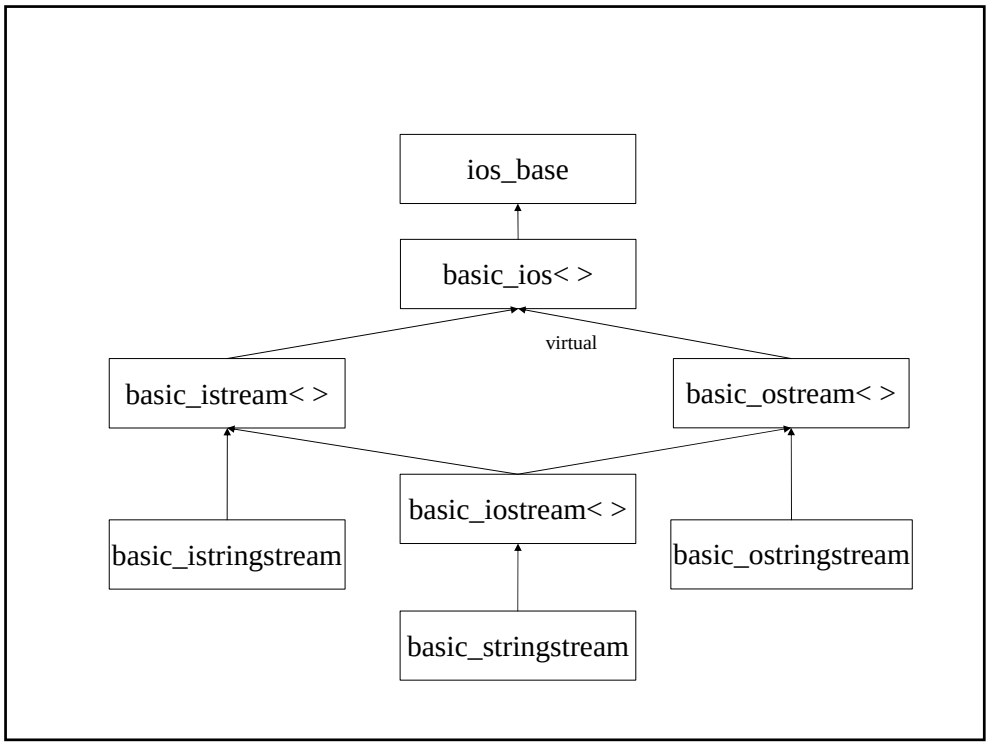

ios_base
basic_ios< >
virtual
basic_istream< >
basic_ostream< >
basic_iostream< >
basic_ostringstream

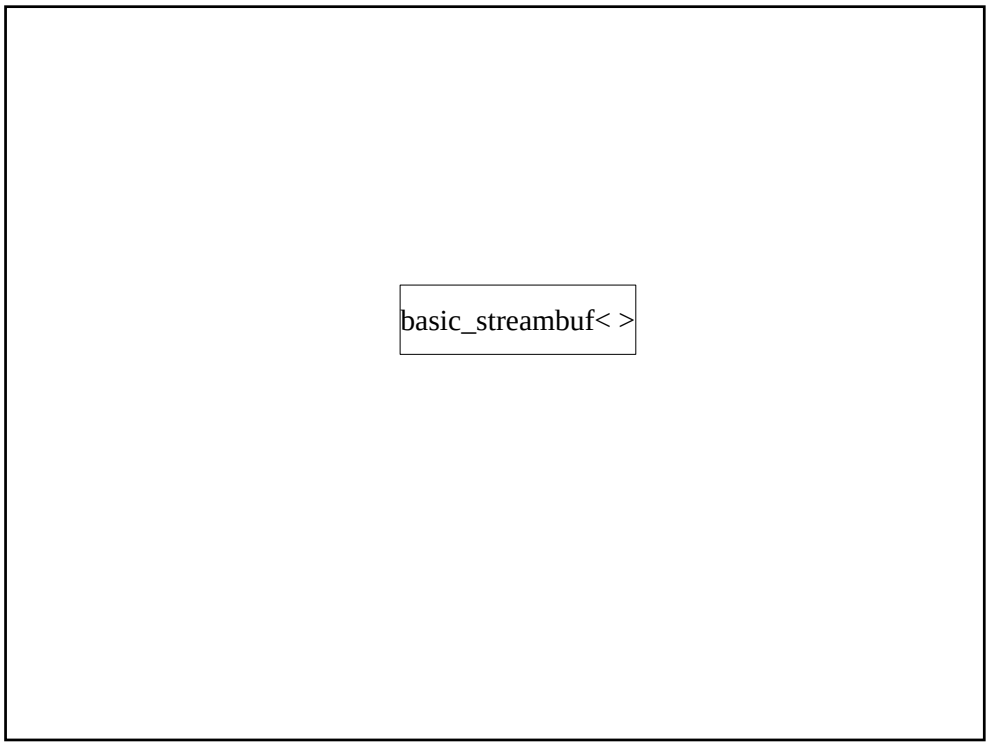

ios_base
basic_ios< >
virtual
basic_istream< >
basic_ostream< >
basic_iostream< >
basic_istringstream
basic_ostringstream



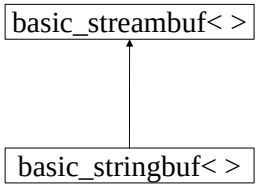
27.6.2 - “The header <ostream> defines a type and several function signatures that control output to a stream buffer.”

```
explicit basic_ostream(
  basic_streambuf<char_type, traits>* sb);
```


ios_base
basic_ios< >
virtual
basic_istream< >
basic_ostream< >
basic_iostream< >
basic_istringstream
basic_ostringstream
basic_stringstream


basic_streambuf< >

27.5 - “The header <streambuf> defines types that control input from and output to character sequences”



This is all very well, but why?

- Known interface
- All the formatting stuff works
- Do you want printf/sprintf/fprintf/snprintf?
- Write to something other than the console, memory or a file?

Digging A Ditch

Writing a Custom Stream

A Logging Stream must:

- Buffer the characters streamed to it
- Flush to another output stream
- Send time and date characters to an output stream

Stream Buffer

The heart of a stream

- Inherit from `std::basic_streambuf`
- Implement a buffer
- Override virtual functions:
  - `overflow`
  - `sync`
  - `xsputn`
- Flush to another output stream

Stream Buffer

Implement a buffer

```
#include <streambuf>
#include <vector>

template< class charT, class traits = std::char_traits< charT > >
class logoutbuf : public std::basic_streambuf< charT, traits >
{
private:
    typedef std::vector< charT > buffer_type;
    buffer_type buffer_;

    ...
};
```

Stream Buffer

Adding to the buffer

```
virtual int_type overflow( int_type c )
{
    if ( !traits::eq_int_type( c, traits::eof() ) )
    {
        buffer_.push_back( c );
    }
    return traits::not_eof( c );
}
```

Stream Buffer

Setting the output stream

```
private:
    typedef typename
         std::basic_streambuf< charT, traits >::int_type int_type;
    std::basic_streambuf < charT, traits >* out_;

public:
    logoutbuf() : out_( 0 ), buffer_()
    {
    }

    void init( std::basic_ostream< charT, traits >* out )
    {
        out_ = out;
    }
```

Stream Buffer

Flushing the buffer

```
    virtual int sync()
    {
        if ( !buffer_.empty() && out_ )
        {
            out_->sputn( &buffer_[0],
              static_cast< std::streamsize >(
                                  buffer_.size() ) );
            buffer_.clear();
        }
        return 0;
    }
```

Stream Buffer

When to flush the buffer

```
template< class charT, class traits = std::char_traits< charT > >
class logoutbuf : public std::basic_streambuf< charT, traits >
{
    ...
public:
    ...

    ~logoutbuf()
    {
        sync();
    }
    ...
};
```

Stream Buffer

Generating time data string

```
private:
    std::basic_string< charT, traits > format_time()
    {
        // Get current time and date
        time_t ltime;
        time( &ltime );

        // Convert time and date to string
        std::basic_stringstream< charT, traits > time;
        time << asctime( gmtime( &ltime ) );

        // Remove LF from time date string and
        // add separator
        std::basic_stringstream< char_type > result;
        result << time.str().erase( time.str().length() - 1 )
               << " - ";

        return result.str();
    }
```

Stream Buffer

Prepending the time date string

```
virtual int sync()
{
  if ( !buffer_.empty() && out_ )
  {
    const std::basic_string< charT, traits > time = format_time();
    out_->sputn( time.c_str(),
                 static_cast< std::streamsize >( time.length() ) );
    out_->sputn( &buffer_[0],
                 static_cast< std::streamsize >( buffer_.size() ) );
    buffer_.clear();
  }
  return 0;
}
```

Stream Buffer

In the name of efficiency

```
virtual std::streamsize xsputn( const char_type* s,
                                std::streamsize num )
  {
      std::copy( s, s + num,
                 std::back_inserter< buffer_type >( buffer_ ) );
      return num;
  }
```

The Stream

The conventional Method

```
template< class charT, class traits = std::char_traits< charT > >
class ostream : public std::basic_ostream< charT, traits >
{
private:
    logoutbuf< charT, traits > buf_;

public:
    ostream()
      : std::basic_ostream< charT, traits >( &buf_ ), buf_()
    {
    }

};
```

What’s wrong with this?

The Stream

Private Base Class (1)

```
template< class charT, class traits = std::char_traits< charT > >
struct logoutbuf_init
{

private:
    logoutbuf< charT, traits > buf_;

public:
    logoutbuf< charT, traits >* buf()
    {
        return &buf_;
    }

};
```

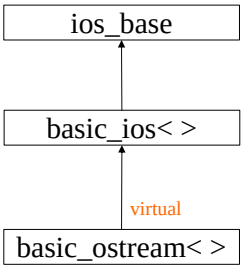
The Stream

Private Base Class (2)

```
template< class charT, class traits = std::char_traits< charT > >
class logostream : private logoutbuf_init < charT, traits >,
                   public std::basic_ostream< charT, traits >
{

private:
    typedef outbuf_init< charT, traits > logoutbuf_init;

public:
    logostream()
      : logoutbuf_init(),
      std::basic_ostream< charT, traits >( logoutbuf_init::buf()
)
    {
    }

};
```

The Stream

`basic_ios`



The Stream

Initialisation Order (1)

```
...
basic_ios
logoutbuf
logoutbuf_init
basic_ostream
logostream
```

The Stream

Inherit Virtually and Privately

```
template< class charT, class traits = std::char_traits< charT > >
class logostream : private virtual logoutbuf_init <charT, traits>,
public std::basic_ostream< charT, traits >
{

private:
    typedef outbuf_init< charT, traits > logoutbuf_init;

public:
    logostream()
      : logoutbuf_init(),
     std::basic_ostream< charT, traits >( logoutbuf_init::buf ) )
    {
    }

};
```

The Stream

Initialisation Order (2)

```
logoutbuf
logoutbuf_init
…
basic_ios
basic_ostream
Logostream
```

The Stream

Initialising the output stream

```
public:
    logostream( std::basic_ostream< charT, traits >& out )
      : logoutbuf_init(),
        std::basic_ostream< charT, traits >
                                          ( logoutbuf_init::buf() )
    {
        logoutbuf_init::buf()->init( out.rdbuf() );
    }
```

Using the Stream

```
typedef logostream< char > clogostream;
typedef logostream< wchar_t > wlogostream;

...

int main()
{
    clogostream;out( std::cout );
    out << "31 hexadecimal: " << std::hex << 31 << std::endl;
    return 0;
}
```

Using the Stream

Output

Fri Apr 20 16:00:00 2005 - 31 hexadecimal: 1f

Custom IOStream Applications

- Click here to get audience to write slide

Transcoding

- ISO8859-2 to UTF-8
- ASCII to UTF-16
- Windows-1252 to ISO8859-1
- Newline conversions
- HTML entity escaping

Compression/Encryption

- Base64
- zlib
- LZH
- Error correction

Sources and Sinks

- sockets
- a database
- a CVS repository
- memory-mapped file
- a vector
- file descriptor
- an array
- a window

Wacky Stuff

- Occam2 style channels

Code you can use

- Boost::IOStreams
  http://boost.org/
  http://home.comcast.net/~jturkanis/iostreams/
- Aeryn
  http://www.paulgrenyer.dyndns.org/aeryn/
- Arabica
  http://www.jezuk.co.uk/arabica
- Standard C++ IOStreams and Locales
  by Langer & Kreft

You. Again.

In the last three months, might you use ...

- a custom IOStream