1 Introduction

After Kona, Pablo asked me to add ostream manipulators for basic_osyncstream to allow users of such streams to modify their flushing behavior, when those stream objects are only know via ostream\& down the call chain.

The wording for these manipulators was reviewed by LWG in Toronto (p0053r5), but their names were never discussed in LEWG, therefore I followed Jeffrey’s suggestion to split them from p0053r6. For more information see that paper.

After the blessing of the names by LEWG in Albuquerque, LWG looked at it again in Jacksonville where this version was prepared to be put in the working draft.

1.1 Items discussed by LEWG/LWG

— Naming of the manipulators (in ABQ). OK.
— Should the manipulators be in header <osyncstream> instead of globally available in <iostream> as are flush and end1? Putting them in <osyncstream> (only), will increase dependence on basic_osyncstream, where basic_syncbuf would suffice for inline implementation of the manipulators. That dependency could even be mitigated by non-inline implementations of the manipulators (providing their instantiations for the supported character types as is done with many other things in the iostream implementations). LEWG could not answer it, but LWG confirmed putting them in <iostream> is the way.
— re-check wording (done be LWG in Toronto, but minor adaptations were made, because of LWG’s feedback. Pablo is OK with the edits)
— What should be the delivery vehicle for this feature: C++20 or the concurrency TS? working draft of the standard, because p0053 was put there already in ABQ.
1.2 Changes from r0/r1 - as given by LWG in Jacksonville

— replaced pointer with * in manipulator specification
— direct call flush() member instead of manipulator function std::flush(os)
— add to synopsis of <ostream> part for putting the manipulators in
— check with Jonathan Coe wrt BSI (no further request for change)
— add #include <ostream> header to <osynstream> synopsis as a fly-by fix to get the manipulators whenever <osynostream> is used.

2 Wording

This wording is relative to the current C++ working draft.

2.1 30.7.2 Header <ostream> synopsis [ostream.syn]
Add the following manipulator declarations to the synopsis of the header <ostream>

```cpp
namespace std {
    template<class charT, class traits = char_traits<charT>>
    class basic_ostream;

    using ostream = basic_ostream<char>;
    using wostream = basic_ostream<wchar_t>;

    template<class charT, class traits>
    basic_ostream<charT, traits>& endl(basic_ostream<charT, traits>&& os);
    template<class charT, class traits>
    basic_ostream<charT, traits>& ends(basic_ostream<charT, traits>&& os);
    template<class charT, class traits>
    basic_ostream<charT, traits>& flush(basic_ostream<charT, traits>&& os);
    template <class charT, class traits>
    basic_ostream<charT, traits>& emit_on_flush(basic_ostream<charT, traits>&& os);
    template <class charT, class traits, class T>
    basic_ostream<charT, traits>& operator<<(basic_ostream<charT, traits>&& os, const T& x);
}
```

2.2 30.7.5.4 Standard basic_ostream manipulators [ostream.manip]
Add the following three manipulators.

```cpp
template <class charT, class traits>
basic_ostream<charT, traits>& emit_on_flush(basic_ostream<charT, traits>&& os);
```

1 Effects: If os.rdbuf() is a basic_osyncbuf<charT, traits, Allocator>* buf, calls buf->set_emit_on_sync(true). Otherwise this manipulator has no effect. [Note: To work around
the issue that the Allocator template argument can not be deduced, implementations can introduce an intermediate base class to basic_osyncbuf that manages its emit_on_sync flag.
—end note]

Returns: os.

template <class charT, class traits>
basic_ostream<charT, traits>& noemit_on_flush(basic_ostream<charT, traits>&& os);

Effects: If os.rdbuf() is a basic_osyncbuf<charT, traits, Allocator>* buf, calls buf->set_-_emit_on_sync(false). Otherwise this manipulator has no effect.

Returns: os.

template <class charT, class traits>
basic_ostream<charT, traits>& flush_emit(basic_ostream<charT, traits>&& os);

Effects: os.flush(). Then, if os.rdbuf() is a basic_osyncbuf<charT, traits, Allocator>* buf, calls buf->emit().

Returns: os.

2.3 30.10.1 Header <syncstream> synopsis [syncstream.syn]

Add the following header include directive to the synopsis:

```
#include <ostream> // see 30.7.2 [ostream.syn]
```

namespace std {
    template<class charT, class traits, class Allocator>
    class basic_syncbuf;
    ...

2.4 Implementation

An example implementation is available on https://github.com/PeterSommerlad/SC22WG21_Papers/tree/master/workpace/p0053_basic_osyncstreambuf