p0573r0 - Manipulators for C++ Synchronized Buffered Ostream (see p0053)

Peter Sommerlad, Pablo Halpern

2017-07-15

Document Number:	p0573r0
Date:	2017-07-15
Project:	Programming Language C++
Audience:	LEWG (LWG to re-check the wording)

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.

1.1 Items to be discussed by LEWG

— Naming of the manipulators

— Should the manipulators be in header `<osyncstream>` instead of globally available in `<ostream>` as are `flush` and `endl`? 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 implementaions).

— 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? I believe both should be addressed when moved, like with p0053.

2 Wording

This wording is relative to the current C++ working draft and refers to the specification in p0053r6. It could be integrated into a Concurrency TS accordingly when p0053 gets adopted.

2.1 30.7.5.4 Standard `basic_ostream` manipulators [ostream.manip]

Add the following three manipulators.

```
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>` pointer `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 takes care its `emit_on_sync` flag. — *end note*]

2 *Returns:* `os`.

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

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

4 *Returns:* `os`.

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

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

6 *Returns:* `os`.

2.2 Implementation

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