~thread Should Join
Herb Sutter

This paper extracts a separable portion of paper N3630, “async, ~future, and “thread.”

Summary
SG1 discussion of N3630 resulted in direction in favor of the proposal that ~thread calls join, not terminate, if the thread was not already joined.

This has no effect on programs that do not currently terminate. It just replaces the requirement to call terminate with the requirement to instead call join.

Proposed Wording
Change 30.3.1.3 as follows:

```cpp
~thread();
1 If joinable(), calls join() std::terminate(). Otherwise, has no effects. [Note: Either implicitly detaching or joining a joinable() thread in its destructor could result in difficult to debug correctness (for detach) or performance (for join) bugs encountered only when an exception is raised. Thus the programmer must ensure that the destructor is never executed while the thread is still joinable. —end note] [Note: Because ~thread is required to be noexcept (17.6.5.12), if join() throws then std::terminate() will be called. —end note]
```

Change 30.3.1.4 as follows:

```cpp
thread& operator=(thread&& x) noexcept;
1 Effects: If joinable(), calls join() std::terminate(). Otherwise,Then assigns the state of x to *this and sets x to a default constructed state. [Note: If join() throws then std::terminate() will be called. —end note]

2 Postconditions: x.get_id() == id() and get_id() returns the value of x.get_id() prior to the assignment.

3 Returns: *this
```