Apply the [[noreturn]] attribute to main as a hint to eliminate destructor calls for objects with static storage duration

Document #:	N4226
Date:	2014-10-10
Project:	Programming Language C++
	Evolution Working Group

Reply-to: Diego Diaz
<diego.diaz@tallertechnologies.com>
Emanuel Bringas
<emanuel.bringas@tallertechnologies.com>
Jens Maurer
<jens.maurer@gmx.net>
Daniel Gutson
<daniel.gutson@tallertechnologies.com>

1 Motivation

Overview

When a `main` function returns, the `std::exit()` function is invoked and all static storage duration objects are destroyed (§**3.6.1p5**).

Nonetheless, many deeply embedded program environments are not intended to exit from `main`. For these environments, the cleanup after the return from `main` is inefficient.

2 Proposed Solution

This document proposes small changes in the standard text to reflect that `noreturn` on `main` could be used as a hint to denote an application that never exits. When using this attribute on `main`, the destructors for objects with static storage duration and functions registered with `std::atexit` will never be invoked. Implementations may utilize this hint and take advantage of its optimization potential, like minimizing the code size and registrations, if any.

The wording proposed in this section presents new language deleted text shown in this way.

Modification to the standard text

1. **[basic.start.main] Add a note in paragraph §3.6.1p3**

   The function `main` shall not be used within a program. The linkage (3.5) of `main` is implementation-defined. A program that defines `main` as deleted or that declares `main` to be `inline`, `static`, or `constexpr` is ill-formed. The name `main` is not otherwise reserved. [*Example*: member functions, classes, and enumerations can be called `main`, as can entities in other namespaces. *end example*] [*Note:* `main` function can also be declared with noreturn attribute (7.6.3). – *end note*]

2. **[basic.start.main] Add a new paragraph §3.6.1p6**

   If the `main` function is declared with the noreturn attribute and the program odr-uses (3.2) the function `std::exit`, the program is ill-formed; no diagnostic required. [*Note:* This implies that destructors for objects with static storage duration are never invoked (3.6.3). – *end note*]

3. **[support.start.term] Modify paragraph §18.5p6**

   Implementation limits: The implementation shall support the registration of at least 32 functions. [*Note:* If the `main` function is declared with the noreturn attribute (3.6.1), calling the `atexit()` function has no effect, since the functions so registered are never invoked. – *end note*]

3 Future Work

This proposal makes no mention of the behavior with regard to `std::at_quick_exit` function since it does not involve the destruction of static storage duration objects. Nevertheless, another very likely proposal could introduce the idea of relaxing the implementation limits of at least 32 functions for `std::at_quick_exit` (**§18.5**). This will give a hint to implementers to save memory, the same as this paper suggests for `std::atexit` function.

4 Discussion

Despite the fact that `[[noreturn]]` attribute can be applied to non-void return type functions, applying this attribute to `main` function could be seen as a contradiction since it actually returns a signed int. Because of this, it might be worthwhile to discuss new signature versions of `main`, like

```
[[noreturn]] void main() { /* ... */ }
```

and

```
[[noreturn]] void main(int argc, char* argv[]) { /* ... */ }
```

5 Acknowledgements

To Christopher Kormanyos, Markus Mayer, Martin Sebor to the initial feedback, guiding and suggestions.
To Diego Novillo and Ian Lance Taylor, for their experience and suggestions as linker implementers.
To Angel Bustamente for his support and experience in writing proposals.

6 References

[1] ISOCPP Standard C++ - ISO/IEC JTC1 SC22 WG21 N 3690
https://isocpp.org/files/papers/N3690.pdf

[2] “C++ ABI or the ARM Architecture” - 3.2.4.2 Static object destruction
http://infocenter.arm.com/help/topic/com.arm.doc.ihi0041d/IHI0041D_cppabi.pdf

[3] “Itanium C++ ABI” - 3.3.5 DSO Object Destruction API
http://refspecs.linux-foundation.org/cxxabi-1.86.html#dso-dtor