What is the difference between __cxa_atexit() and atexit()

Asked 6 years, 5 months ago Modified 4 years, 10 months ago Viewed 11k times



In the GCC docs I found the -fuse-cxa-atexit option and it says the following:

10

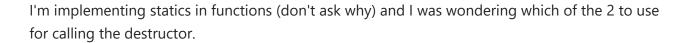
This option is required for fully standards-compliant handling of static destructors



So what is the difference between the two? In the docs for cxa atexit I found the following:



The __cxa_atexit() function is used to implement atexit()



And I guess I only have atexit() for MSVC? Is that a problem?

Can I just use atexit() everywhere and be sure that it would behave just like real static objects in functions would?

C++atexit

Share Improve this question Follow

asked Mar 20, 2017 at 19:09



ongtam

4,356 2 28 50

\$

1 Answer

Sorted by:

Highest score (default)



__cxa_atexit() is defined in the Itanium C++ ABI. The document explained the motivation behind this function:

The C++ Standard requires that destructors be called for global objects when a program exits in the opposite order of construction. Most implementations have handled this by









registered functions, although most implementations support many more. More important, it does not deal at all with the ability in most implementations to

calling the C library atexit routine to register the destructors. This is problematic because the 1999 C Standard only requires that the implementation support 32

remove [Dynamic Shared Objects] from a running program image by calling dlclose prior to program termination.

The API specified below is intended to provide standard-conforming treatment during normal program exit, which includes executing <code>atexit</code>-registered functions in the correct sequence relative to constructor-registered destructors, and reasonable treatment during early DSO unload (e.g. <code>dlclose</code>).

So:

- __cxa_atexit() is not limited to 32 functions.
- __cxa_atexit() will call the destructor of the static of a dynamic library when this dynamic library is unloaded before the program exits.

You *should* enable -fuse-cxa-atexit if you are writing a library, and your libc has this function (e.g. glibc, musl). In fact, the gcc that comes with your distro might have this flag enabled automatically already (there will be a linker error if you enable the flag and the libc doesn't support it).

Note that <u>users should not call __cxa_atexit_directly</u>: it takes arguments which only the compiler/linker is supposed to know (<u>the __dso_handle</u>).

... No user interface to __cxa_atexit is supported, so the user is not able to register an atexit function with a parameter or a home DSO.

MSVC apparently <u>does not use atexit()</u> -like functions to run the global destructors. And according to <u>Destructor of a global static variable in a shared library is not called on dlclose</u> MSVC already run destructors on <u>dlclose()</u>.

Share Improve this answer Follow

edited Jun 20, 2020 at 9:12

Community Bot

answered Mar 20, 2017 at 19:26

kennytm 511k 105 1085 1005

soooooo for emulating the lifetime of local statics - simply using atexit() will not be 100% the same as what the compiler generates? - ongtam Mar 20, 2017 at 19:49

0 @onqtam Right. Instead of atexit, you could create a local struct which the destructor calls that handler. static struct Foo { ~Foo() { handle(foo); } } foo_dtor; - kennytm Mar 20, 2017 at 19:53

There is a typo in the first line: Italium C++ ABI. should be Itanium C++ ABI. Unfortunately the stackoverflow won't allow me to edit the post changing less than 6 chars. – Morty Oct 26, 2018 at 14:04

@morty Thanks, fixed – kennytm Oct 26, 2018 at 14:47