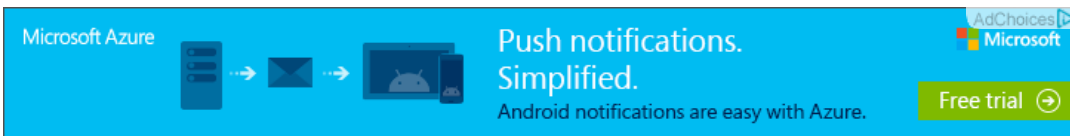


Stack Overflow is a question and answer site for professional and enthusiast programmers. It's 100% free, no registration required.

Take the 2-minute tour ×

When abort() is preferred over exit()?




Microsoft Azure Push notifications. Simplified. Android notifications are easy with Azure. Free trial

I know the differences between the two. One notable thing is that `abort()` sends SIGABRT signal, so it may be relevant when your software relies on them. But for a typical application `exit()` seems to be more safe version of `abort()`...? Are there any other concerns to use `abort()` instead of `exit()`?

c++ c exit abort

asked Sep 9 '10 at 11:48

 [doc](#)
2,549 16 37

[add a comment](#)

4 Answers

Using `abort` will dump core, if the user has core dumps enabled. So as a rule of thumb, I'd use `abort` if you're so unsure about what's gone wrong that the only way to get useful information about it is by analysing a core dump.

If you can safely `exit` from any given point, and don't need the core dump, then `exit` is a nicer approach.

answered Sep 9 '10 at 11:56

 [Andy Mortimer](#)
1,969 6 10

1 that's also good point +1 – [doc](#) Sep 9 '10 at 12:05

1 I would rephrase that as: are you expecting your users to analyze core dumps? If not, don't use `abort`. (Keeping in mind that although you the developer might want a core dump, your users might not. So perhaps `abort` should only be used in a "debug" version of your executable.) – [Ken Simon](#) Sep 9 '10 at 12:44

@Ken Simon: If users don't want core dumps, they can turn them off (`ulimit -c 0`). I think Ubuntu does that by default. – [camh](#) Sep 9 '10 at 13:12

@camh: I would agree with Ken on this. when you send software in the wild you can't always control what system parameters users will choose. What if as a software developer you never want users to get core dumps? – [kriss](#) Sep 9 '10 at 14:11

@Ken, @kriss: I'd want my users to get core dumps, so they can include them in bug reports. If they don't want to do that, they can easily suppress them. – [Mike Seymour](#) Sep 9 '10 at 15:42

[show 3 more comments](#)



Professional tools that complement your Java code Get started now

Use `abort()` if your program is in a possibly corrupt state and you consider it too dangerous to try to do anything further. `exit()` will cause any `atexit` functions, and in C++ destructors of static objects, to be called. This is usually what you want for a clean exit, but it could be catastrophic if, for example, they overwrite a file with corrupt data.

answered Sep 9 '10 at 12:01



[Mike Seymour](#)

147k 8 167 319

- 2 +1: but you also have others ways to do it. for not calling function registered as atexit you could also `_exit()` instead of `exit()`, or even send a SIGKILL to yourself for immediate exit. – [kriss](#) Sep 9 '10 at 12:05

@Kriss: `abort()` is the standard way to do so, and easy. Why would you choose a non-standard method? – [MSalters](#) Sep 9 '10 at 12:10

@kriss: `_exit()` isn't standard C or C++, and aborting by raising a signal other than SIGABRT seems a bit of an odd thing to do. – [Mike Seymour](#) Sep 9 '10 at 12:18

isn't it a paraphrase of sharptooth's answer? – [doc](#) Sep 9 '10 at 12:57

@doc: No, I wrote this before I saw the other answer. It says more or less the same thing, though. – [Mike Seymour](#) Sep 9 '10 at 13:16

show 8 more comments

Sometimes your program breaks to such extent that its state becomes inconsistent and so `exit()` will not work because it would cause global objects destruction and the latter would not function properly when the state is inconsistent. In such situations `abort()` is to be preferred.

answered Sep 9 '10 at 11:53



[sharptooth](#)

96.2k 25 224 569

I suppose it's for example when IO becomes unoperational due to hdd failure? You catch this as an exception but you can't destroy file objects because their destructors need to perform file close. Thanks for the idea. – [doc](#) Sep 9 '10 at 12:00

- 1 @doc: Yes, but it's a rather extreme example. A more C++ example: you're already handling an error and another error happens (not related to error handling process) and the handling code is reentered. That's not very good - errors happen faster that you can handle them. So you maintain a flag "I'm inside handling this kind of error already". Once code is reentered you throw in the towel - call `abort()` to terminate the program immediately. – [sharptooth](#) Sep 9 '10 at 12:04

add a comment

Abort is preferred when application doesnot able to handle the exception and not able to understand what to do scenario. Exit() mean application should must finish all task gracefully. if exception occurs and application is able to handle the same then Exit() call happens.

answered Sep 9 '10 at 13:08



[Santosh kumar](#)

64 1 5

add a comment

Not the answer you're looking for? Browse other questions tagged [c++](#) [c](#) [exit](#) [abort](#) or [ask your own question](#).