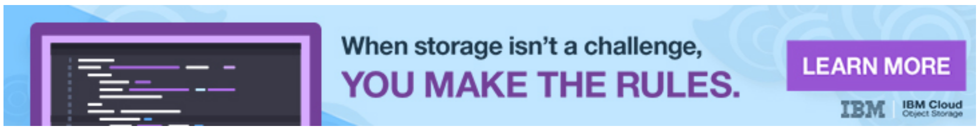
What causes a SIGABRT fault?



Could you please tell me what could possibly cause a SIGABRT fault in C++?

c++

asked Jun 22 '12 at 17:20
user1444426
80 1 2 7

7 We are not your personal search assistants. If you can't even be bothered to try looking this up yourself, why should we try to answer for you? – Marc B Jun 22 '12 at 17:21

6 Read en.wikipedia.org/wiki/SIGABRT and come back if you have any more questions. – ecatmur Jun 22 '12 at 17:22

2 Good point!!:)I tried looking up it says its a signal that is sent to abort the process from compiler to the system, but my compiler does not say which portion is causing this unusual behavior, To narrow down on this, I asked for possible reasons.My code is 500 lines long. – user1444426 Jun 22 '12 at 17:23

@user1444426 - Compile it in debug mode (-g with g++), use dbx and it will find the problem. – Ed Heal Jun 22 '12 at 17:26

1 @Ecatmur, the Wikipedia page isn't really all that helpful. All it says is that `abort` raises that signal, but it doesn't go beyond that. I get `SIGABRT` signals in my programs sometimes, but I've *never* directly called `abort` . – Rob Kennedy Jun 22 '12 at 17:29

3 Answers

Per Wikipedia,

> `SIGABRT` is sent by the process to itself when it calls the `abort` libc function, defined in `stdlib.h` . The `SIGABRT` signal can be caught, but it cannot be blocked; if the signal handler returns then all open streams are closed and flushed and the program terminates (dumping core if appropriate). This means that the `abort` call never returns. Because of this characteristic, it is often used to signal fatal conditions in support libraries, situations where the current operation cannot be completed but the main program can perform cleanup before exiting. It is used when an assertion fails.

That means that if your code is not calling `abort` directly nor sending itself the `SIGABRT` signal via `raise` , and you don't have any failing assertions, the cause must be that a support library (which could be libc) has encountered an internal error. If you provide the details of your program we might be able to suggest possible causes. Even better, if you examine a core or run your program in a debugger you should be able to collect a stack trace, which will show which library caused your program to abort.

(It is also possible that another program on your system is sending your program `SIGABRT` , but this is in most cases vanishingly unlikely.)

answered Jun 24 '12 at 22:32
ecatmur
97.5k 12 163 255

This usually happens when libraries encounter internal error, so they call abort(), because they cant continue. This might happen when you overwrite one of it's data structures (the one which belongs to the function from libc for example). So this might be e.g. libc calling because you did overwrite something. And the application must then terminate because it's impossible to continue or handle it, which is called failed assertion.

edited Jun 22 '12 at 17:37

answered Jun 22 '12 at 17:31
Andrew
703 5 13

In practice this is usually triggered by the assert macro:

```
char* foo = NULL;
assert( foo != NULL );
```

would result in SIGABRT

answered Jun 24 '12 at 22:39
Rafael Baptista
6,851 1 19 37