

Tip Of The Month: How To Track Down Memory Leaks

Version 1.0; May 15, 2005

This article presents a method of tracking down a memory leak in cases where the address of that heap allocation keeps changing. The solution consists of the following steps:

- Start the application and run it until it panics.
- Take note of the address that wasn't deallocated. (Assume it's 0x1046a432.)
- Restart the application and run it until it panics. Make sure you run it exactly as you did the first time.
- Again take note of the address. (Assume it's 0x1a52a432.)
- Run it one more time, again until it panics and in exactly the same way.
- Again take note of the address. (Assume it's 0x245ea432.)

Now looking at the numbers you will find a correlation.

Constant Progression

In some cases you will find that the address increases by the same amount with each restart of the application. In the case of the addresses assumed above, the address for the same panic increased from 0x1046a432 to 0x1a52a432, that is, by 0xa0c00000 bytes. Then, the next address was at place of the same increase, that is, 0x245ea432. So now you know that when you restart the application, there will be a panic at $0x245ea432 + 0xa0c00000 = 0xc51ea432$.

Now view that memory address in the memory window and put a breakpoint in the code just before it panics, that is, in a final destructor call, or some function call that you know is pretty much at the end of the application. When you hit the breakpoint, view what's in that address space. If it looks like some text, then you'll know it is some descriptor that wasn't cleaned up properly. If it contains another address or looks like it's some kind of an object, then try typecasting it in your watcher window like this: (CBase*)0xc51ea432. In many cases, the debugger then will show you what object it is.

Irregular Progression

However, if the application restarts didn't show a constant increase in the memory address, then try restarting the emulator. After restarting the emulator and the application, run the application until it panics in exactly the same way as previously. If the address is the same as in the previous tests, that is, 0x1046a432, then the next panic will probably be in the address space of the last test round, that is, 0x1a52a432, so try viewing and watching this memory address with the breakpoint enabled when you restart the application.

Related Addresses

If that doesn't find the leak, the final thing left is to put a few breakpoints around the code and run the initial tests again, going through about three iterations of panics. But this time in addition to noting the panicking address, also take down the addresses of a few objects that are created in your code, and then try to find any correlation between the known object addresses and the place where the panic occurs. Check for correlations where the first four bits in the address are the same, or perhaps the last four. If you find some correlation, use that to calculate the address of the next failed deallocation.

Process of Elimination

If all that doesn't help, you will need to go through your code and disable everything that's allocated, and then enable things step by step until you find the allocation that never gets deallocated.

Copyright © 2005 Nokia Corporation. All rights reserved.

Nokia and Nokia Connecting People are registered trademarks of Nokia Corporation. Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. Other product and company names mentioned herein may be trademarks or trade names of their respective owners.

Disclaimer

The information in this document is provided “as is,” with no warranties whatsoever, including any warranty of merchantability, fitness for any particular purpose, or any warranty otherwise arising out of any proposal, specification, or sample. Furthermore, information provided in this document is preliminary, and may be changed substantially prior to final release. This document is provided for informational purposes only.

Nokia Corporation disclaims all liability, including liability for infringement of any proprietary rights, relating to implementation of information presented in this document. Nokia Corporation does not warrant or represent that such use will not infringe such rights.

Nokia Corporation retains the right to make changes to this specification at any time, without notice.

License

A license is hereby granted to download and print a copy of this specification for personal use only. No other license to any other intellectual property rights is granted herein.