Using a Link Map to Track Down a Symbol’s Origin

This thread has been locked by a moderator.

69

Link maps might be old and obscure, but they’re still super useful when debugging weird problems. I’ve talked about them before many times here no DevForums, but they came up again today, so I decided to write up a more detailed example.

If you have any questions or comments about this, start a new thread and tag it with *Linker* so that I see it.

Share and Enjoy

—
Quinn “The Eskimo!” @ Developer Technical Support @ Apple
`let myEmail = "eskimo" + "1" + "@" + "apple.com"`

Using a Link Map to Track Down a Symbol’s Origin

Imagine you encounter a crash report with a backtrace like this:

```
Thread 0 Crashed::  Dispatch queue: com.apple.main-thread
0 libsystem_kernel.dylib  … __pthread_kill + 10
1 libsystem_pthread.dylib … pthread_kill + 263
2 libsystem_c.dylib       … abort + 123
3 MyApp                   … Mystery::Mystery() + 51
4 MyApp                   … Mystery::Mystery() + 21
5 MyApp                   …
6 MyApp                   …
7 dyld                    … invocation function for block in dyld4::Loader::findAndRunAllInitializers(…) …
```

Frame 7 indicates that the program has crashed running a dynamic linker initialisation function. Frame 2 shows that the crash was caused by a call to `abort`. But what about frames 3 and 4? You have a symbol name but no source file or line number.

In an ideal world all crash reports would be fully symbolicated. For hints and tips on how to work out that info, see Adding Identifiable Symbol Names to a Crash Report. However, this process doesn’t always work.

In fact, sometimes it *can’t* work. If the `Mystery::Mystery()` routine comes from a static library that doesn’t have symbol information, there’s no way to symbolicate it. Furthermore, it’s possible that the symbol is internal to that library, so you can’t even tell what library it comes from. You might search your entire source code and find no references to `Mystery`.

However, there is a way forward here, by looking at a *link map*. This is an old school linker feature that records the location, size, and origin of every symbol added to the linker’s output.

To generate a link map, enable the Write Link Map File build setting. By default this puts the link map into a text (`.txt`) file within the derived data directory. To find the exact path, look at the Link step in the build log. You can also customise this using Path to Link Map File build setting.

Note If you’re running `ld` directly, use the `-map` option. See the `ld` man page for details.

In the link map you’ll see an entry like this:

```
0x100003270 0x00000040  [  2] __ZN7MysteryC2Ev
```

The first item, 0x100003270, is the start of the symbol and the second, 0x00000040, is its size. The last item, `__ZN7MysteryC1Ev`, is the symbol name. This is C++, so it’s mangled. Use `c++filt` to unmangle it:

```
% c++filt __ZN7MysteryC1Ev
Mystery::Mystery()
```

The remaining item, `[ 2]`, gives the origin of the symbol. At the top of the link map is an index:

```
# Object files:
[  0] linker synthesized
[  1] …/main.o
[  2] …/libEnigma.a(Enigma.o)
…
```

So, the `Mystery::Mystery()` routine came from the `Enigma.o` object file within the `libEnigma.a` static library. This is the info you need to continue your investigation into the crash.

Linker

Reply

Posted 2 weeks ago by eskimo

Add a Comment

Developer › Forums

Platforms
iOS
iPadOS
macOS
tvOS
watchOS

Tools
Swift
SwiftUI
SF Symbols
Swift Playgrounds
TestFlight
Xcode
Xcode Cloud

Topics & Technologies
Accessibility
Accessories
App Extensions
App Store
Audio & Video
Augmented Reality
Business
Design
Distribution
Education
Fonts
Games
Health & Fitness
In-App Purchase
Localization
Maps & Location
Machine Learning
Security
Safari & Web

Resources
Documentation
Curriculum
Downloads
Forums
Videos

Support
Support Articles
Contact Us
Bug Reporting
System Status

Account
Apple Developer
App Store Connect
Certificates, IDs, & Profiles
Feedback Assistant

Programs
Apple Developer Program
Apple Developer Enterprise Program
App Store Small Business Program
MFi Program
News Partner Program
Video Partner Program
Security Bounty Program
Security Research Device Program

Events
App Accelerators
App Store Awards
Apple Design Awards
Apple Developer Academies
Entrepreneur Camp
Tech Talks
WWDC

To view the latest developer news, visit News and Updates.