MCLinker: A Linker Solution for Mobile

MediaTek White Paper

February 2015

Introduction

MCLinker, a MediaTek-initiated open source project[1], is a full-fledged system linker designed particularly for mobile devices. In mobile software framework Ahead-of-time (AOT) compilation technology is introduced to improve the application performance and brings the advantages to reduce the runtime compilation efforts. The needs of on-device toolchains are raised.

MCLinker is designed to be an on-device linker under the "UIUC" BSD-Style license. Its small code size and fast linking speed can survive the tough environment of mobile devices with limited memory, hard-disk space, and computing power. To fulfill the requirements of current mobile environment, MCLinker supports ARM, X86 and MIPS targets with both 32-bit and 64-bits, which are the Android mainstream targets.

Linking Flow

Normally a linker requires several steps to complete linking, as shown in Figure 1. First the Reader reads in the input files according to their formats and transforms them to linker internal data structures. Secondly, the Normalization step examines the validity of input command line options and enables the corresponding setting for linking. The command line options of linkers are numerous and complicated, hence Normalization is usually a complex and difficult part. The third step is Symbol Resolution, which is the most important part in a linker. It scans all the symbols and binds the same symbols together to decide their final attributes. Fourthly, Layout decides the output file layout after all inputs are read and parsed. After the layout is decided and the final address of the output program is determined, Relocation is performed to assign the final address to the program for function or data referenced. Finally, Writers write out the output file and the whole linking has been done.

[1] MCLinker is an open source project. All source and related materials can be found in the public website, https://code.google.com/p/mclinker/

In MCLinker, every step and the intermediate data structures are carefully designed to limit the memory usage and speed up linking time. For example, during Normalization, MCLinker introduces the linker command line language to simplify the complicated analysis and efficiently transform the input options to the corresponding handler during linking. MCLinker traverses all the symbols very few times when compared with other linkers. In as much as there are a large number of symbols, limiting the symbol traversing time can efficiently reduce the linking time.

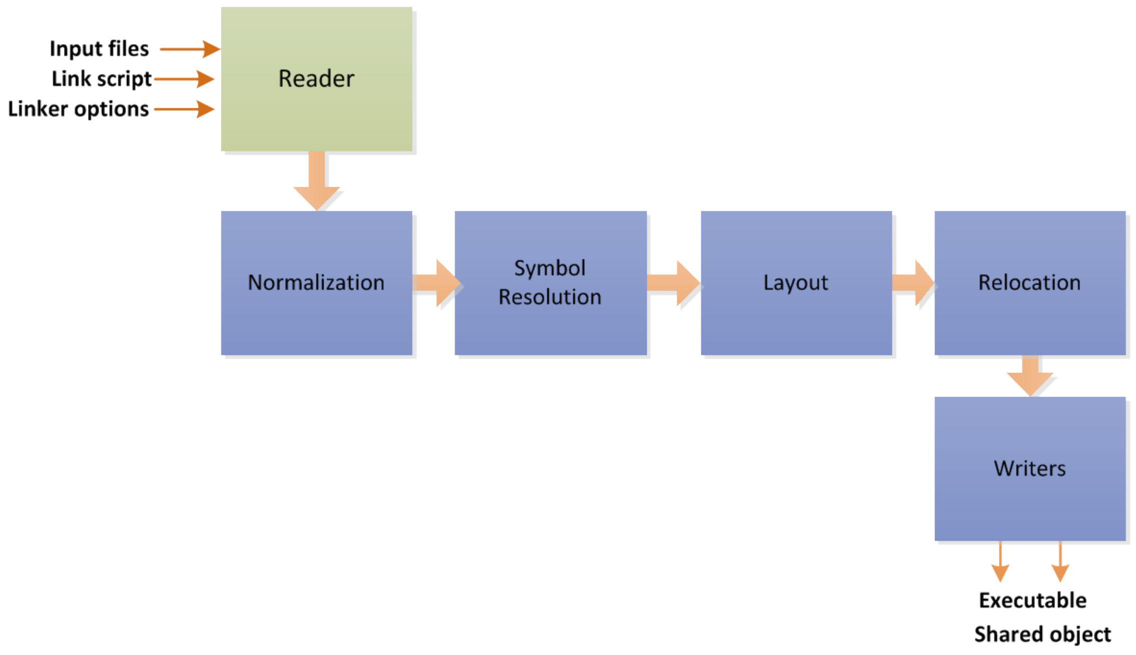

Figure 1. The Linking Processes

Linker Library

MCLinker is an integrated linker. That is, it can be used as a reusable linker library. The whole linking processes can be achieved by calling a set of APIs provided by MCLinker library. With this functionality, it is possible to integrate compilation and linking to reduce translation time and memory usage, and to provide the more efficient and flexible usage of linkers.

Multiple Targets Support

MCLinker supports multiple targets including ARM, X86, MIPs and Hexagon, with both 32-bit and 64-bit on ARM, X86 and MIPs. And the MCLinker architecture is designed for cross platform linking. The core linking framework and the target related functionalities are clearly defined and separated. Hence it is easy to add other targets' backend support into MCLinker.

Performance

The figures below provide code size and linking speed comparison with two other recognized open-source linkers—the GNU-ld and Google gold linker. MCLinker has the smallest size and the linking time competes with Google gold linker. As MCLinker is the only one under non-GPL license, it is the best choice of on-device linkers among three.

Linker	**Size** **ARM target only**	**Size** **All targets**
GNU-ld	2M	7M
Google gold linker	4M	4M
MCLinker	1.2M	1.4M

Figure 2. Code Size Comparison

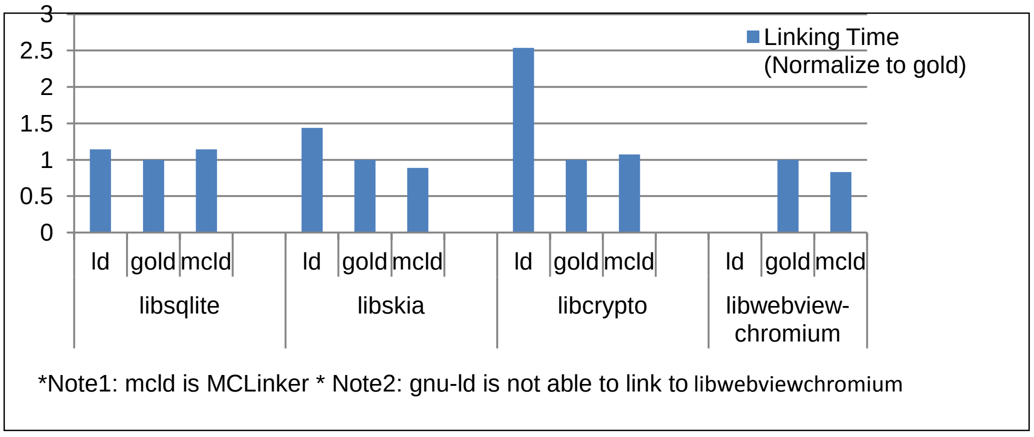

Figure 3. Linking Speed Comparison

MCLinker in Android

MCLinker has been adopted by Android NDK Revision 9[2] and AOSP (Android Open Source Project) as an alternative linker.

Android has faced the problem of target devices fragmentation for quite some time. Beginning with Android Lollipop, ART (Android Runtime) has replaced the original Dalvik runtime as the default runtime. ART introduces AOT compilation technology to improve the application performance and the significant benefits and improvements are seen. Moreover, the AOT compilation could be the way to solve Android fragmentation problem.

Nevertheless, linking on-device is challenging. Linkers usually require a lot of time and memory space to complete the linking processes, which may not be acceptable on a device. Another concern is that the on-device toolchain requires a friendlier license for commercial use. MCLinker, as a BSD-Style license, can satisfy the needs perfectly.

Conclusion

MCLinker is a solid multi-arch supported linker toolchain for on-device linking. It is now adopted by Google Android. Moreover, with the design of modular linker, many opportunities to integrate compiler and linker are introduced.

[2] https://developer.android.com/tools/sdk/ndk/index.html