一种简化鲁棒的 ARM-Linux 交叉编译器构建方法*

李云飞，刘军清，雷帮军，龚国强
（三峡大学 智能视觉与图像信息研究所，湖北 宜昌 443002）

摘 要：针对分步方式构建交叉编译器过程复杂且成功率低的问题，提出一种简化鲁棒的交叉编译器分步式构建方法，该方法利用功能完整的交叉编译器代替静态 GCC 编译 Glibc，减少构建过程中的步骤和参数配置数目，降低整个过程的复杂度，提高成功率。结果证明，该方法可以有效地降低分步式方法的复杂度，提高成功率。为了提高该方法的普适性，文中还对该方法中的参数设置、版本选择、库依赖等问题进行了分析和总结。

关键词：ARM；LINUX；交叉编译器；嵌入式；分步式
中图分类号：TP31 **文献标志码**：A **文章编号**：0529－6579（2014）02－0033－05

A Simplified and Robust Method of Building ARM-Linux Cross-Compiler

LI Yunfei, *LIU Junqing*, *LEI Bangjun*, *GONG Guoqiang*
(Institute Intelligent Vision and Image Information, China Three Gorges University, Yichang 443002, China)

Abstract: The method of cross-compiler building step by step has high complexity and low rate of success. To address the problem, a simplified and robust method is proposed. The method utilizes a full-featured cross-compiler to compiler Glibc instead of the static GCC, which reduces the build process steps, the number of parameters and complexity of the entire process, and improves the rate of success. The results show that the method can effectively reduce the complexity of the method to build a cross-compiler step by step and improve the rate of success of the method. In order to enhance the universality of the method, the parameter setting, version selection, dependencies of library files and other issues are analyzed and summarized also on the method proposed.

Key words: ARM; LINUX; cross-compiler; embedded; step by step

在嵌入式应用系统中，ARM-Linux 嵌入式平台技术正在迅速发展，其应用范围越来越广泛，有着很好的发展前景，在手机处理器中 ARM 处理器已经占了 90% 的份额。从事以 ARM-Linux 为平台的嵌入式开发工作，针对 ARM-Linux 平台的交叉编译器是必备的工具。最常用的针对 ARM-Linux 平台的交叉编译器是在 Linux 环境下的 ARM-Linux 交叉编译器。构建 ARM-Linux 交叉编译器可以利用 Crosstool 脚本工具构建，也可以采用分步方式构建。与利用 Crosstool 脚本工具构建交叉编译器的方式相比，分步方式较复杂，但分步方式的构建过程完全可控，具有高度的可定制性，可以构建出符合特定要求的交叉编译器。因此，利用分步式构建的交叉编译器可以和目标机有非常好的吻合，具有高可靠性和便利性[1－3]。

利用分步方式构建交叉编译器，构建过程较复

* **收稿日期**：2013－08－09
基金项目：湖北省自然科学基金资助项目（2011CDC100）；湖北省教育厅重点资助项目（D20131306）；三峡大学楚天学者基金资助项目（KJ2012B001）；三峡地区地质灾害与生态环境湖北省协同创新中心 PI 团队资助项目（CICGE2013PI-3-103）
作者简介：李云飞（1989 年生），男；**研究方向**：嵌入式系统设计；**通讯作者**：刘军清；E-mail：junqingliu@ctgu.edu.cn

杂，因此编译成功率低，本文对分步式构建交叉编译器的方法进行了改进，并对构建过程中遇到的参数设置、版本选择、库依赖等问题进行了分析和总结，以提高利用分步式构建交叉编译器的成功率。

1 常规的构建方法

分步式构建交叉编译器的基本工作是利用交叉编译器所需的程序源码并按照一定步骤编译出所要求的交叉编译器。利用源码编译程序有参数配置、编译、安装三个步骤[4]，如图 1 所示。三个步骤中参数配置是最重要的一步，参数配置是否正确，决定着后续编译工作是否能顺利进行。

图 1 利用源码编译安装程序步骤

Fig. 1 The steps of compiling and installing program form source

图 2 为 ARM-Linux 交叉编译器的常规分步方式构建流程。分为 7 个步骤。前三个步骤为准备阶段，为后续工作提供必要的程序源码、工具软件和头文件，后面四个步骤是核心工作。这四个步骤之间有严格依赖的关系，所以它们的执行顺序必须严格遵从图所示的顺序。Binutils 提供一系列交叉编译器的二进制工具，后续阶段都需要利用它。静态 GCC 的作用是用来编译 Glibc，是一个过度性工具，这个阶段的 GCC 只支持 C 语言，不支持动态库、线程库等一些特性。Glibc 提供 C 语言标准的静态库和动态库。完整版 GCC 支持动态库、线程库等一些特性，这些特性是需要有 Glibc 库的支持才可以编译成功[5-8]。

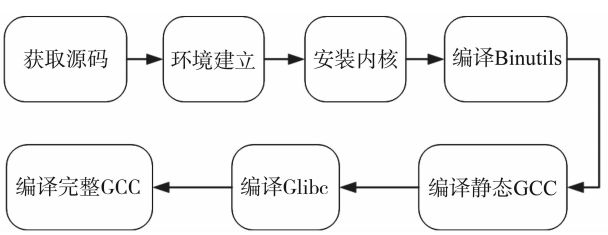

图 2 常规分步式构建交叉编译器流程图

Fig. 2 The flowchart of the conventional method to build a cross-compiler step by step

2 改进分步式构建方法

图 3 为改进的分步式构建交叉编译器的构建流程。相对于常规的方法，改进方法中没有了编译静态 GCC 的步骤，但在准备工作阶段，改进方法需要安装功能完整的交叉编译器，在编译 Glibc 时使用的编译器是安装的功能完整 ARM-Linux 交叉编译器。在常规分步式构建交叉编译器的过程中，静态 GCC 是一个过度性工具，它的作用只是编译 Glibc，因此可以利用功能完整的交叉编译器代替静态 GCC 来编译 Glibc。与常规方法中的静态 GCC 不同，替代的功能完整 ARM-Linux 交叉编译器的选择不是唯一的，所有能编译对应 Glibc 的功能完整 ARM-Linux 交叉编译器都可以使用。一般情况下，最后编译的完整版 GCC 的特性是要符合特定要求的，参数配置中关于处理器架构的配置参数和替代的功能完整交叉编译器的可能不同，所以在利用替代的交叉编译器编译 Glibc 时，要检查替代的交叉编译器的配置参数中与处理器架构相关的配置参数和将要编译的完整版的 GCC 中关于处理器架构的参数配置是否一样，如果不一样，在编译 Glibc 时需要使用如 -march、-mcpu、-mfloat 等与处理器架构相关的编译参数，使替代交叉编译器用与将要编译完整 GCC 的参数配置中相同的处理器架构相关的编译参数编译 Glibc[9-15]。

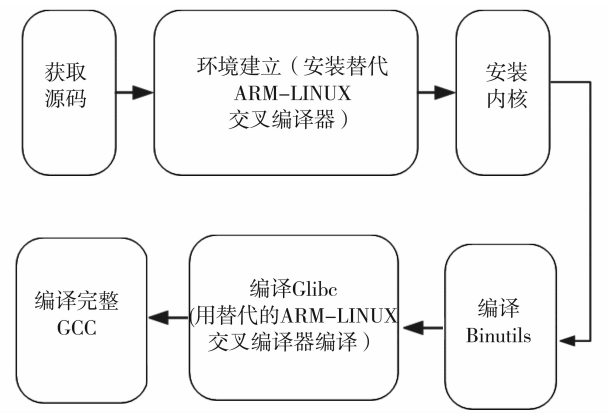

图 3 改进的分步式构建交叉编译器流程图

Fig. 3 The flowchart of the improved method to build a cross-compiler step by step

在常规分步式构建交叉编译器的过程中，在编译静态 GCC 时，还没有编译 Glibc，参数配置时需要关闭需要 Glibc 库支持的特性，开启必需的功

能，使参数配置变得复杂，容易出错，从而导致编译失败率增加。在编译 Glibc 时，由于此时编译它的静态 GCC 功能不完整，需要在参数配置时进行相应的设置和修改相关文件，才能使编译通过。这些原因增加构建交叉编译器的复杂，并降低了成功率。在改进的方法中，利用已功能完整的交叉编译器代替静态 GCC，没有编译静态 GCC 的阶段，因此可以避免编译静态 GCC 时复杂的参数配置和易出现的失败。由于编译 Glibc 的编译器功能完整，在编译 Glibc 时，无需进行特定参数设置和修改相关文件，因此降低了编译 Glibc 是参数配置的复杂度和失败率。与常规方法中的静态 GCC 不同，替代的功能完整 ARM-Linux 交叉编译器选择不是唯一的，所有能编译对应 Glibc 的功能完整 ARM-Linux 交叉编译器都可以使用，因此，在编译 Glibc 时，可以避免由于 GCC 和 Glibc 版本不匹配导致的错误。在准备阶段，改进方法需要另外安装功能完整的交叉编译器，但功能完整的交叉编译器容易获得，安装简单，对整体工作量和难度的影响很小。

为了验证分析结果，在最终编译出的完整 GCC 特性要求相同，主机环境相同条件下，分别用常规的方法和改进的方法做编译实验，获得两种方法核心步骤中每个步骤的所必须的参数配置数目。实验环境如表 1 所示。

表 1　实验环境

Table 1　Experimental environment

主机系统	Fedora 16
源码包	binutils-2. 21. 1. tar. gz、gcc-4. 6. 0. tar. gz、glibc-2. 11. tar. gz、Glibc-ports-2. 11. tar. gz、linux-2. 6. 25. 8. tar. gz、mpfr-3. 1. 1. tar. gz、gmp-5. 0. 5. tar. bz2、mpc-0. 9. tar. gz
替代的 ARM-Linux 交叉编译器	版本为 4. 3 的 ARM-Linux-gcc 交叉编译器
完整 GCC 特性	支持 C、C + +；支持动态库；支持线程库

实验结果如表 2 所示。表 2 显示了两种方法的在最终完整 GCC 特性要求相同的情况下核心步骤的参数配置的数目。从表 2 可以看出，改进方法需要配置的参数数目总共 39 个，比常规方法少了 20 个，很大程度上减少了参数配置的复杂度。分步式构建交叉编译器的过程中产生的大部分错误都是由参数配置不当而引起的，参数配置的复杂度越低，产生错误的概率就越小。因此改进的方法可以很大程度地提高整个编译过程的成功率和稳定性。

表 2　核心步骤参数配置数目

Table 2　Number of parameters in the core steps

核心步骤	参数配置项目数/个	
	常规方法	改进方法
编译 Binutils	9	8
编译静态 GCC	18	无
编译 Glibc	16	12
完整 GCC	18	17
总共	61	37

图 4 显示用柱状图来比较两种方法的核心步骤的参数配置数目。从图中可以直观的看出改进的方法的参数数目比常规的方法要少很多。本文方法不存在编译静态 GCC 的步骤，编译过程中总的参数配置数目要比传统方法少 20 个以上。

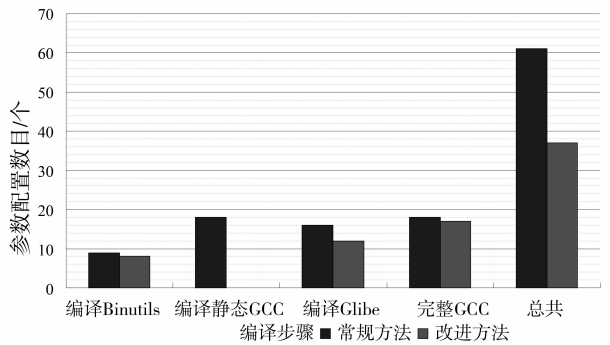

图 4　核心步骤参数配置数目的柱状图

Fig. 4　The histogram of number of parameters in the core steps

3　问题分析及解决方法

下面就本文方法在实际应用中存在的三个主要问题进行分析，并给出相应的解决方法。

3. 1　参数配置

每个阶段的编译工作都需要通过 configure 脚本文件进行参数配置，可以使用 configure - help 命令查看参数的简要说明。每个步骤有很多配置参数，主要包括安装目录参数（如 - prefix）、系统类型参数（ - build、 - host、 - target）、相关特性和功能参数（以 enable、disable、with、without 为前缀）等。

安装目录参数中最常用的是 - prefix，作用是将所有的文件安装到指定目录下。其他相关的安装目录参数作用是将不同类型文件分别安装到不同目录下。系统类型参数包括 - build、 - host 和 - target，但 Glibc 的系统类型参数只有 - build 和

–host。参数–build 指定编译此工具的平台，参数–host 指定运行此工具的平台，参数–target 指定此工具产生的文件运行的平台。通常情况，参数–build、–host 的值是编译此工具的平台，参数–build、–host 的值可以通过命令 echo ${MACHTYPE} 获得。–target 参数的值可以根据表3设置。每个工具的相关特性和功能参数有所不同，而且此类型的参数较多，对于它们的设置要根据具体要求设置。

表3 Target 值

Table 3 Value of target

处理器	字节序	应用程序二进制接口（ABI）	参数 target 的值
Generic arm	little	OABI	arm-unknown-linux-gnu
Generic arm, version 4	little	OABI	armv4-unknown-linux-gnu
Generic arm, version 5	little	EABI	armv5l-unknown-linux-gnueabi
Generic arm, version 5	big	EABI	armv5b-unknown-linux-gnueabi
Generic arm	little	EABI	arm-unknown-linux-gnueabi
Generic arm, version 7-a	little	EABI	armv7a-unknown-linux-gnueabi

3.2 版本选择

在制作交叉编译器之前，需要选择满足要求的 Binutils、GCC、Glibc 和主机 GCC 的版本，否则，会导致编译失败。一般情况，由于版本选择问题导致的编译失败，主要表现为符号未定义、函数名未定义等语法错误。这是由于在新版本中存在一定程度的改动或添加了一些新特性，导致程序之间会存在兼容性问题。

在利用常规的分步式构建交叉编译器过程中，如果在编译 Glibc 时出现符号未定义、函数名未定义等语法错误，主要原因是由于前一阶段编译的静态 GCC 的版本不满足要求，因为编译 Glibc 是利用前一阶段的静态 GCC 来进行的。如果在编译静态 GCC 和完整版 GCC 出现符号未定义、函数名未定义等语法错误，主要是由于主机 GCC 版本不满足要求，因为静态 GCC 和完整版 GCC 由主机 GCC 编译的。在选择每个工具的版本时，首先选择 Glibc 的版本，尽量不要选择最新的版本。然后，根据 Glibc 源码根目录下的 INSTALL 文件里说明，选择 GCC 和 Binutils 版本。再根据 GCC 源码目录下 INSTALL 文件夹下 prerequisites. html 文件里的说明选择主机 GCC 的版本。

在改进的方法中，在编译 Glibc 时出现符号未定义、函数名未定义等语法错误，主要原因是替代静态 GCC 的完整交叉编译器的版本选择不满足要求。要根据 Glibc 源码根目录下的 INSTALL 文件里说明选择其版本，建议选择和将要编译的 GCC 的版本相近的版本。对于 GCC 和 binutils 版本的选择不需要根据 Glibc 源码根目录下的 INSTALL 文件中的要求选择，选择自由度大。宿主机的 GCC 编译器的版本要根据 GCC 源码目录下 INSTALL 文件夹下 prerequisites. html 文件里的要求选择。

3.3 库依赖问题

库依赖问题，是指编译器搜索不到所需的库文件。在编译交叉编译器的过程中，库依赖问题经常出现。出现库依赖问题，一般会有“ld：cannot find –lx”（x 为库名）的错误提示，然而一些情况，一些其他错误提示也可能由库依赖问题引起，这种情况，需要到出错的目录下，查看 config. cache 文件。在此文件中，在出现错误信息的前面几行会显示出错的具体原因，如果是库依赖问题引起的，会有相关错误提示。

在制作交叉编译器的过程中出现库依赖问题的原因主要有三种。相关库的路径不在编译器的搜索路径里会产生库依赖问题。这种情况可以通过变量 LD_ LIBRARY_ PATH 或 LDFLAGS 将库文件的路径添加到编译器的搜索路径里；库文件的链接文件失效或不存在会产生库依赖问题。这种情况，要重新建立库文件的链接文件。在利用常规的分步式构建交叉编译器过程中，静态 GCC 编译完成后，需要对 libgcc. a 静态库文件在同目录下建立两个名为 libgcc_ sh. a、libgcc_ eh. a 的链接文件，否则，会引起由于库文件的链接文件不存在导致的库依赖问题；在利用常规的分步式和改进的方法构建交叉编译器过程中，在编译 Glibc 完成后，都需要对安装目录下 lib 文件夹下的 libc. so 和 libpthread. so 进行修改，这两个文件需要在终端里用 VI 编辑器打开，并将两个文件中 GROUP 后面每个库文件的绝对路径删除。如果不删除绝对路径，会导致编译器找不到这两个文件中描述的库文件[16]。

4 总 结

本文对分步式构建交叉编译器的方法做了改进，在改进的方法中，去除了编译静态 GCC 的阶段，所以可以省去在这阶段中复杂的参数配置，而且，由于编译 Glibc 的交叉编译器功能完整，在

Glibc 的参数配置中无需配置一些相应的参数，所以编译 Glibc 阶段的参数配置也变得相对简单。不仅参数配置变得简单，而且版本选择相对容易。同时，也可以避免编译静态 GCC 时出现的失败，和由于静态 GCC 的不完整导致编译 Glibc 失败。因此，改进的方法可以很好的弥补分步式方法构建交叉编译器时过程复杂、成功率低的不足。本文构建的交叉编译器的方法是针对 ARM 处理器的，但同样也适合 MIPS、POWERPC 等其他处理器。

参考文献：

[1] 刘二刚. 利用 crosstool-ng 构建交叉编译工具链[J]. 电脑知识与技术,2011,7(19):4553 - 4554 + 4567.

[2] 张欢庆,高丽,宋承祥. 基于 ARM 的嵌入式 Linux 交叉编译环境的研究与实现[J]. 计算机与数字工程,2012,40(2):151 - 153.

[3] 李文,张建泽. 基于 S3C2440 的嵌入式 Linux 系统移植[J]. 化工自动化及仪表,2010,37(9):88 - 92.

[4] 孟庆昌,牛欣源. Linux 教程[M]. 北京,电子工业出版社,2011.

[5] WANG W H, GAO M Y. Design of embedded media player based on S3C2440 and SDL-FFMPEG[C]// International Conference on Electrical and Control Engineering, 2011:2979 - 2982.

[6] 尤盈盈,孟利民. 构建嵌入式 Linux 交叉编译环境[J]. 计算机与数字工程,2006,34(6):30 - 34.

[7] 林炎,张友益. Windows 平台下构建嵌入式 Linux 交叉编译环境[J]. 单片机与嵌入式系统应用,2013(2):74 - 78.

[8] 许青林,解争龙. 基于 ARM 的 Linux 系统移植研究与实现[J]. 物联网技术,2013(1):37 - 42.

[9] 陈永强,陶品,王笃强. 嵌入式 Linux 移植[J]. 实验室研究与探索,2012,31(9):67 - 72.

[10] 张瑞,于德海,马明龙. 基于 ARM 的嵌入式 Linux 的交叉编译环境的建立[J]. 科技信息,2009(25):508 - 509.

[11] 何春山,谭剑. 基于 Redhat AS4 并行计算机群的安装研究[J]. 中山大学学报:自然科学版,2006,45(3):114 - 116.

[12] 谭会生. ARM 嵌入式系统原理及应用开发[M]. 西安:西安电子科技大学出版社,2012.

[13] 李佳佳,马永杰. Linux 系统在 S3C2440 上移植的分析与实现[J]. 计算机系统应用,2012,21(10):214 - 219.

[14] 刘发贵,林恺,柴阳阳. GDIXEAP:面向服务的嵌入式软件开发平台[J]. 中山大学学报:自然科学版,2008,47(2):37 - 41.

[15] 冯钢,郑扣银. 基于 GCC 的交叉编译研究与开发[J]. 计算机工程与设计,2004,25(11):1880 - 1883.

[16] 阳富明,李文海,涂刚. 嵌入式动态库小型化技术研究[J]. 华中科技大学学报,2004,32(9):6 - 8.

(上接第 32 页)

[8] LEE S Y, KIM H K, KIM J K, et al. Phase transition characteristics and electrical properties of nitrogen-doped GeSb thin films for PRAM applications[J]. J Mater Sci, 2009,44(16): 4354 - 4359.

[9] FORST M, DEKORSY T, TRAPPE C, et al. Phase change in $Ge_2Sb_2Te_5$ films investigated by coherent phonon spectroscopy[J]. Appl Phys Lett, 2000,77(13): 1964 - 1966.

[10] ZHU W L, WANG C Z, SUN M C, et al. Characterization of Femtosecond laser-irradiation crystallization and structure of multiple periodic Si/$Sb_{80}Te_{20}$ nanocomposite films by coherent phonon spectroscopy[J]. Opt Express, 2011, 19 (23): 22684 - 22691.

[11] GARRENT G A, ALBRECHT T F, WHITAKER J F, et al. Coherent THz phonons driven by light pulse and the Sb problem: What is the mechanism? [J]. Phys Rev Lett, 1996, 77(17):3661 - 3664.

[12] ZEIGER H J, VIDAL J, CHENG T K, et al. Theory for displacive excitation of coherent phonons[J]. Phys Rev Lett, 1992, 45(2): 768 - 778.

[13] KIM H K, KIM N H, ROH J S, et al. Considerable changes in crystallization process delivered by N doping in Te-free Sb-rich GeSb binary allay[J]. Curr Appl Phys,2011, 11(3):S404 - S409.

[14] Li S M, HUANG H, ZHU W L, et al. Femtosecond laser-induced crystallization of amorphous Sb_2Te_3 film and coherent phonon spectroscopy characterization and optical injection of electron spins[J]. J Appl Phys, 2011, 110(5):053523.

[15] LI S M, ZHOU D, WEN T, et al. Femtosecond laser-irradiated crystallization of amorphous $Si_2Sb_2Te_3$ films and its in-situ characterization by coherent phonon spectroscopy[J]. Opt Express, 2013,21(8): 10222 - 10227.

[16] WANG Y G, XU X F, VENKATASUBRAMANIAN R, et al. Reduction in coherent phonon lifetime in Bi_2Te_3/Sb_2Te_3 superlattices[J]. Appl Phys Lett, 2008, 93(11):113114.

[17] ZHU W L, LU Y G, LI S M et al. Femtosecond laser-induced crystallization of amorphous Ga-Sb-Se films and coherent phonon dynamics[J]. Opt Express, 2012, 20(17):18585 - 18590.

[18] HASE M, MIYAMOTO Y, TOMINAGA J, et al. Ultrafast dephasing of coherent optical phonons in atomically controlled GeTe/Sb_2Te_3 superlattices[J]. Phys Rev B, 2009,79(17):174112.