命名规则

交叉编译工具链的命名规则为：**arch [-vendor] [-os] [-(gnu)eabi]**

> **arch** - 体系架构，如 ARM，MIPS
>
> **vendor** - 工具链提供商
>
> **os** - 目标操作系统
>
> **eabi** - 嵌入式应用二进制接口（Embedded Application Binary Interface）

根据对操作系统的支持与否，ARM GCC 可分为支持和不支持操作系统，如

> **arm-none-eabi**：这个是没有操作系统的，自然不可能支持那些跟操作系统关系密切的函数，比如 fork(2)。他使用的是 newlib 这个专用于嵌入式系统的 C 库。
>
> **arm-none-linux-eabi**：用于 Linux 的，使用 Glibc

实例

1、arm-none-eabi-gcc

（ARM architecture，no vendor，not target an operating system，complies with the ARM EABI）
用于编译 ARM 架构的裸机系统（包括 ARM Linux 的 boot、kernel，不适用编译 Linux 应用 Application），一般适合 ARM7、Cortex-M 和 Cortex-R 内核的芯片使用，所以不支持那些跟操作系统关系密切的函数，比如 fork(2)，他使用的是 newlib 这个专用于嵌入式系统的 C 库。

2、arm-none-linux-gnueabi-gcc

(ARM architecture, no vendor, creates binaries that run on the **Linux** operating system, and uses the GNU EABI)

主要用于基于 ARM 架构的 Linux 系统，可用于编译 ARM 架构的 u-boot、Linux 内核、linux 应用等。arm-none-linux-gnueabi 基于 GCC，使用 Glibc 库，经过 Codesourcery 公司优化过推出的编译器。arm-none-linux-gnueabi-xxx 交叉编译工具的浮点运算非常优秀。一般 ARM9、ARM11、Cortex-A 内核，带有 Linux 操作系统的会用到。

3、arm-eabi-gcc

Android ARM 编译器。

4、armcc

ARM 公司推出的编译工具，功能和 arm-none-eabi 类似，可以编译裸机程序（u-boot、kernel），但是不能编译 Linux 应用程序。armcc 一般和 ARM 开发工具一起，Keil MDK、ADS、RVDS 和 DS-5 中的编译器都是 armcc，所以 armcc 编译器都是收费的（爱国版除外，呵呵~~）。

5、arm-none-uclinuxeabi-gcc 和 arm-none-symbianelf-gcc

arm-none-uclinuxeabi 用于 **uCLinux**，使用 Glibc。

arm-none-symbianelf 用于 **symbian**，没用过，不知道 C 库是什么 。

Codesourcery

Codesourcery 推出的产品叫 Sourcery G++ Lite Edition，其中基于 command-line 的编译器是免费的，在官网上可以下载，而其中包含的 IDE 和 debug 工具是收费的，当然也有 30 天试用版本的。

目前 CodeSourcery 已经由明导国际(Mentor Graphics)收购，所以原本的网站风格已经全部变为 Mentor 样式，但是 Sourcery G++ Lite Edition 同样可以注册后免费下载。

Codesourcery 一直是在做 ARM 目标 GCC 的开发和优化，它的 ARM GCC 在目前在市场上非常优秀，很多 patch 可能还没被 gcc 接受，所以还是应该直接用它的（而且他提供 Windows 下[mingw 交叉编译的]和 Linux 下的二进制版本，比较方便；如果不是很有时间和兴趣，不建议下载 src 源码包自己编译，很麻烦，Codesourcery 给的 shell 脚本很多时候根本没办法直接用，得自行提取关键的部分手工执行，又费精力又费时间，如果想知道细节，其实不用自己编译一遍，看看他是用什么步骤构建的即可，如果你对交叉编译器感兴趣的话。

ABI 和 EABI

ABI：二进制应用程序接口(Application Binary Interface (ABI) for the ARM Architecture)。在计算机中，应用二进制接口描述了应用程序（或者其他类型）和操作系统之间或其他应用程序的低级接口。

EABI：嵌入式 ABI。嵌入式应用二进制接口指定了文件格式、数据类型、寄存器使用、堆积组织优化和在一个嵌入式软件中的参数的标准约定。开发者使用自己的汇编语言也可以使用 EABI 作为与兼容的编译器生成的汇编语言的接口。

两者主要区别是，ABI 是计算机上的，EABI 是嵌入式平台上（如 ARM，MIPS 等）。

arm-linux-gnueabi-gcc 和 arm-linux-gnueabihf-gcc

两个交叉编译器分别适用于 armel 和 armhf 两个不同的架构，armel 和 armhf 这两种架构在对待浮点运算采取了不同的策略（有 fpu 的 arm 才能支持这两种浮点运算策略）。

其实这两个交叉编译器只不过是 gcc 的选项 -mfloat-abi 的默认值不同。gcc 的选项 -mfloat-abi 有三种值 **soft**、**softfp**、**hard**（其中后两者都要求 arm 里有 fpu 浮点运算单元，soft 与后两者是兼容的，但 softfp 和 hard 两种模式互不兼容）：

soft： 不用 fpu 进行浮点计算，即使有 fpu 浮点运算单元也不用，而是使用软件模式。

softfp： armel 架构（对应的编译器为 arm-linux-gnueabi-gcc ）采用的默认值，

用 fpu 计算，但是传参数用普通寄存器传，这样中断的时候，只需要保存普通寄存器，中断负荷小，但是参数需要转换成浮点的再计算。

hard： armhf 架构（对应的编译器 arm-linux-gnueabihf-gcc ）采用的默认值，用 fpu 计算，传参数也用 fpu 中的浮点寄存器传，省去了转换，性能最好，但是中断负荷高。

把以下测试使用的 C 文件内容保存成 mfloat.c：

```
#include <stdio.h>
int main(void)
{
   double a,b,c;
   a = 23.543;
   b = 323.234;
   c = b/a;
   printf("the 13/2 = %f\n", c);
   printf("hello world !\n");
   return 0;
}
```

1、使用 arm-linux-gnueabihf-gcc 编译，使用"-v"选项以获取更详细的信息：

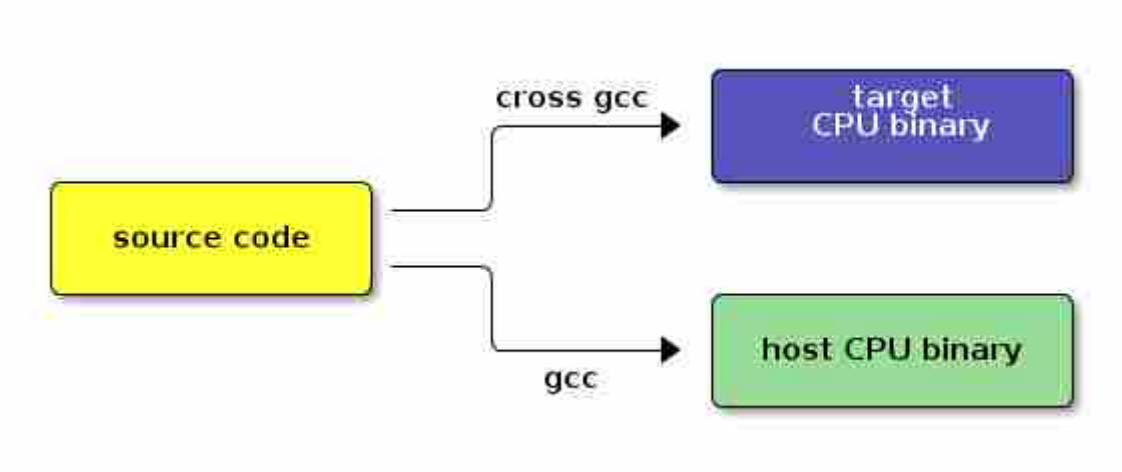
```
# arm-linux-gnueabihf-gcc -v mfloat.c
COLLECT_GCC_OPTIONS='-v' '-march=armv7-a' '-mfloat-abi=hard' '-
mfpu=vfpv3-d16' '-mthumb'
-mfloat-abi=hard
```

可看出使用 hard 硬件浮点模式。

2、使用 arm-linux-gnueabi-gcc 编译：

```
# arm-linux-gnueabi-gcc -v mfloat.c
COLLECT_GCC_OPTIONS='-v' '-march=armv7-a' '-mfloat-abi=softfp' '-
mfpu=vfpv3-d16' '-mthumb'
-mfloat-abi=softfp
```

可看出使用 softfp 模式。



交叉编译工具

参考资料

1. 交叉编译器 arm-linux-gnueabi 和 arm-linux-gnueabihf 的区别：http://www.cnblogs.com/xiaotlili/p/3306100.html
2. arm-none-linux-gnueabi，arm-none-eabi 与 arm-eabi 区别：http://blog.csdn.net/mantis_1984/article/details/21049273
3. What's the difference between arm-linux- / arm-none-linux-gnueabi- / arm-fsl-linux-gnueabi- in LTIB?https://community.freescale.com/thread/313490