

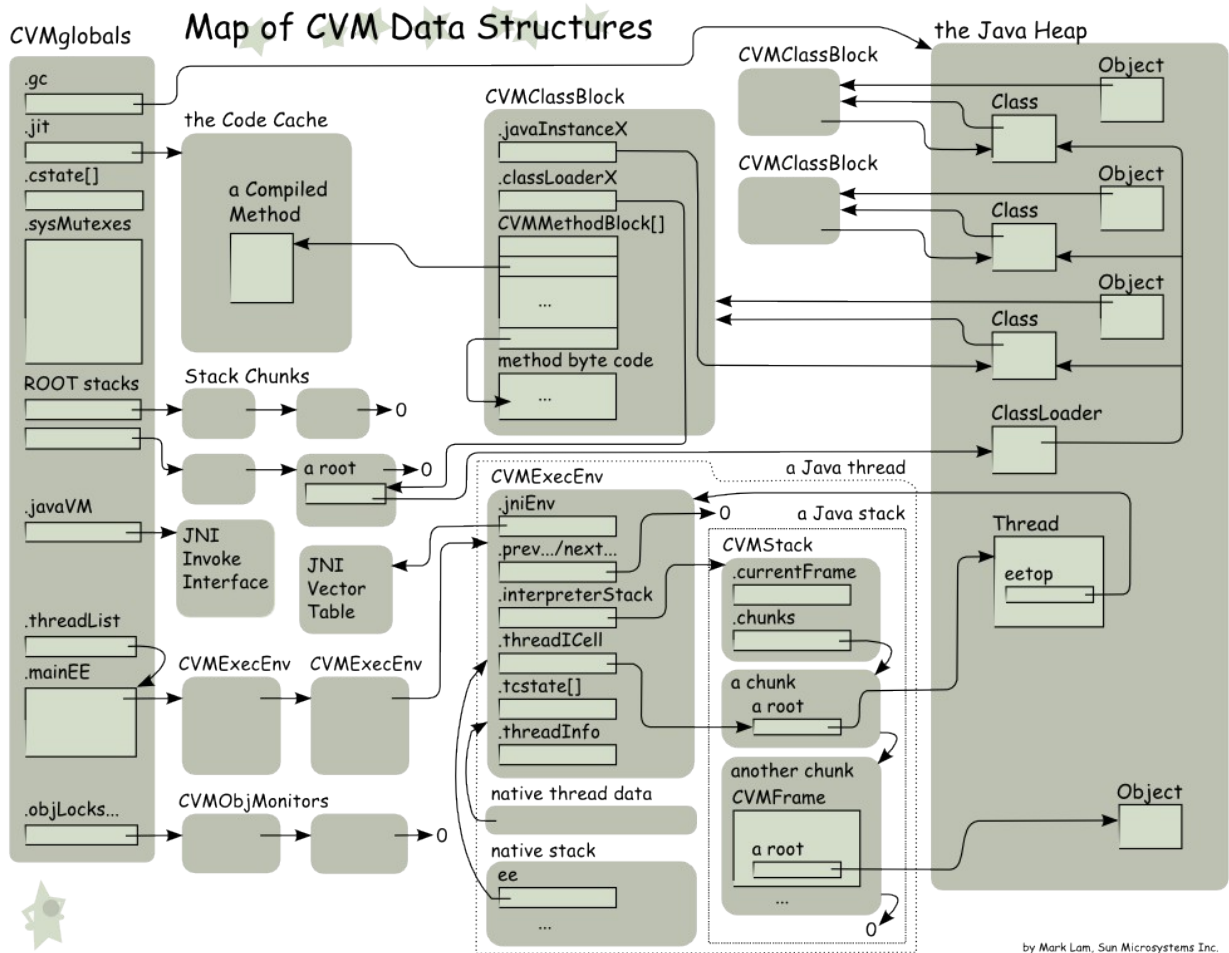
JVM 的 ARM 移植

当今 JVM 种类很多，如 IBM JVM、MS JVM 等等。但要把它用到 ARM 上，并非易事。几经过测试和对比，最终还是选择了 SUN 的 JVM，其中，J2ME (Java 2 Platform Micro Edition) 是专为移动设备设计的，在移动通信设备上已经有了很广的应用，非常适合我们在嵌入式中使用。J2ME 根据硬件资源的限制，分为 CLDC (Connected Limited Device Configuration) 和 CDC (Connected Device Configuration)，无论是 CLDC 还是 CDC，其核心都是 Java 虚拟机。CLDC 采用了 KVM，它是一个真正的最小的而又基本上完整的 Java 虚拟机，小到可以装入几十 KB 的内存中。CDC 则采用了 CVM，它是一个具有完整的 J2SE 1.3 VM 支持能力的 Java 虚拟机，因此也更适合使用、更有效，但又比标准的 J2SE VM 小。作为一个全功能的 Java 虚拟机，CVM 几乎支持 J2SE VM 的所有先进特性，包括最底层的调错处理和本地语言接口。

SUN 的 JVM 开源后，作为 JAVA 开发社区的一个项目，为避免版权问题，J2ME 改名为 phoneME。CLDC 对应 phoneME Feature，而 CDC 对应 phoneME Feature，虚拟机则称为 CVM。由于我们有较为丰富的硬件资源，并且为今后更为复杂的开发考虑，我们选择了 phoneME Feature，本文的描述，也是针对 phoneME Feature 展开的。

一、CVM SYSTEM MAP

首先，让我们来看张图。



by Mark Lam, Sun Microsystems Inc.

这张图很好表述了 CVM 的整个运行过程。遗憾的是本文不打算变成婴儿读物，对 JVM 的实现不打算加以介绍，其实 SUN 官方网站上有份《JAVA 虚拟机规范》，比我在这唠叨一半天来得给系统准确。同时这张图来自 Mark Lam 的 Blog，他是一位 SUN 的工程师，在 CVM 的小组中工作，他对这张图有了很全面的说明，但扫兴的是我不打算在这里引用，因为我不愿就此把这篇文章变成虚拟机实现的专著……

二、如何获得源码

JAVA 开源后第一个受益的就是 J2ME，我也希望你能称为受益者之一。首先，你得获取到最新的源代码：

```
# svn co https://phoneme.dev.java.net/svn/phoneme/components/cldc/trunk cldc
# svn co https://phoneme.dev.java.net/svn/phoneme/components/pcs1/trunk pcs1
# svn co https://phoneme.dev.java.net/svn/phoneme/components/midp/trunk midp
# svn co https://phoneme.dev.java.net/svn/phoneme/components/cdc/trunk cdc
# svn co https://phoneme.dev.java.net/svn/phoneme/components/tools/trunk tools
# svn co https://phoneme.dev.java.net/svn/phoneme/components/jump/trunk jump
# svn co https://phoneme.dev.java.net/svn/phoneme/components/abstractions/trunk
abstractions
```

三、编译

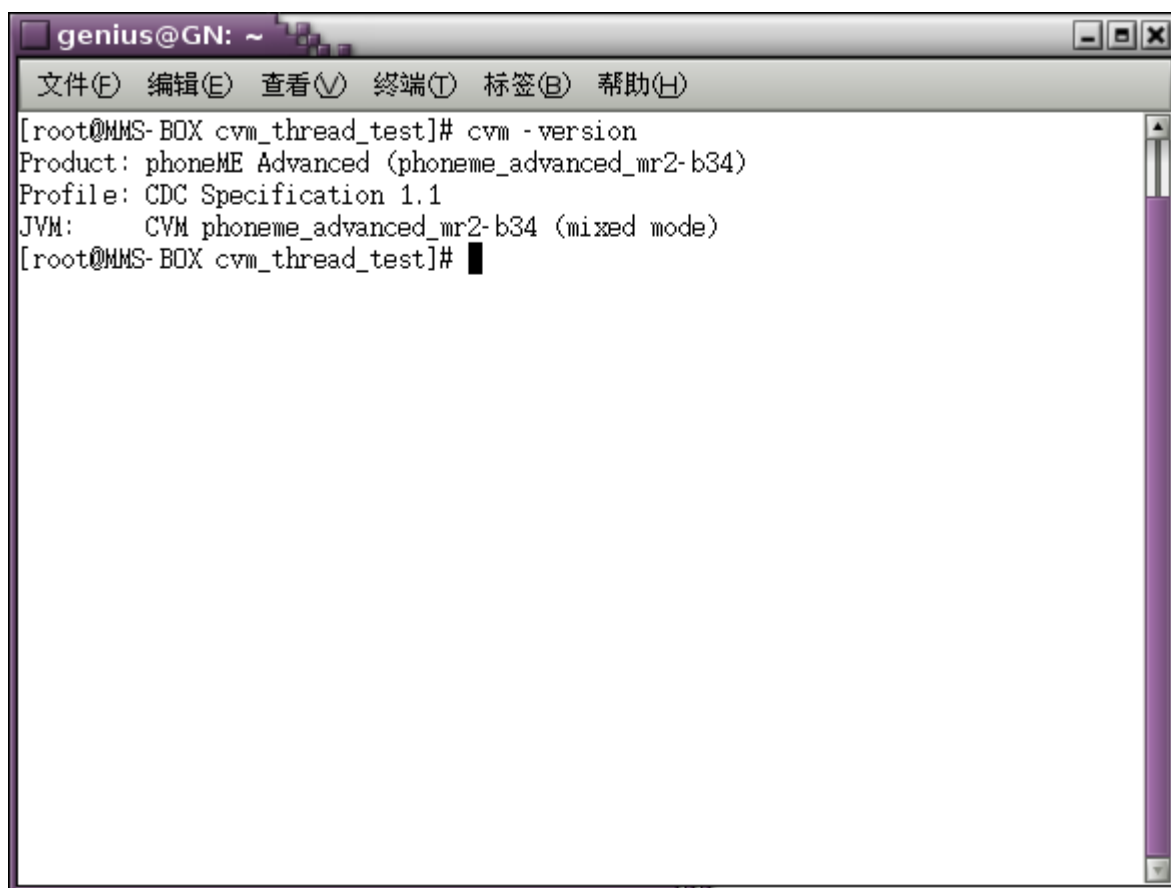
一般地，你直接 `make` 就可以了，如果报错，可能是你系统缺少一些环境变量，如

JDK_HOME、HOST_CC 等，具体请参阅官方网站上的《CDC Build System Guide》。一帆风顺的话，不出 10 分钟（对我的破机子来说），你就能编译出 CVM 了。不过我估计你会出许多问题，譬如说少了某些 Linux 库文件或某些工具，这个和你编译时使用的系统相关，因人而异，我用的是 Ubuntu 7.10，少了什么系统都会很友善的提示你，并告诉你在线安装的方法，装完重新 **make** 吧，总能成功的。虽然 CVM 之支持 JAVA 1.4 的规范，但你仍然可以用最新的 J2SE 1.6 来编译，编译时脚本会自动指定生成 JAVA 1.4 的 class 文件。

四、安装

在 **make** 完成之后，我们可以用 **make bin** 来把编译好的 CVM 和相关库文件做成安装包。比较无奈的是，**make bin** 时必须连上 Internet 网，脚本会从 SUN 的服务器上下载 3 个几 KB 的使用协议文件，否则无法打包。

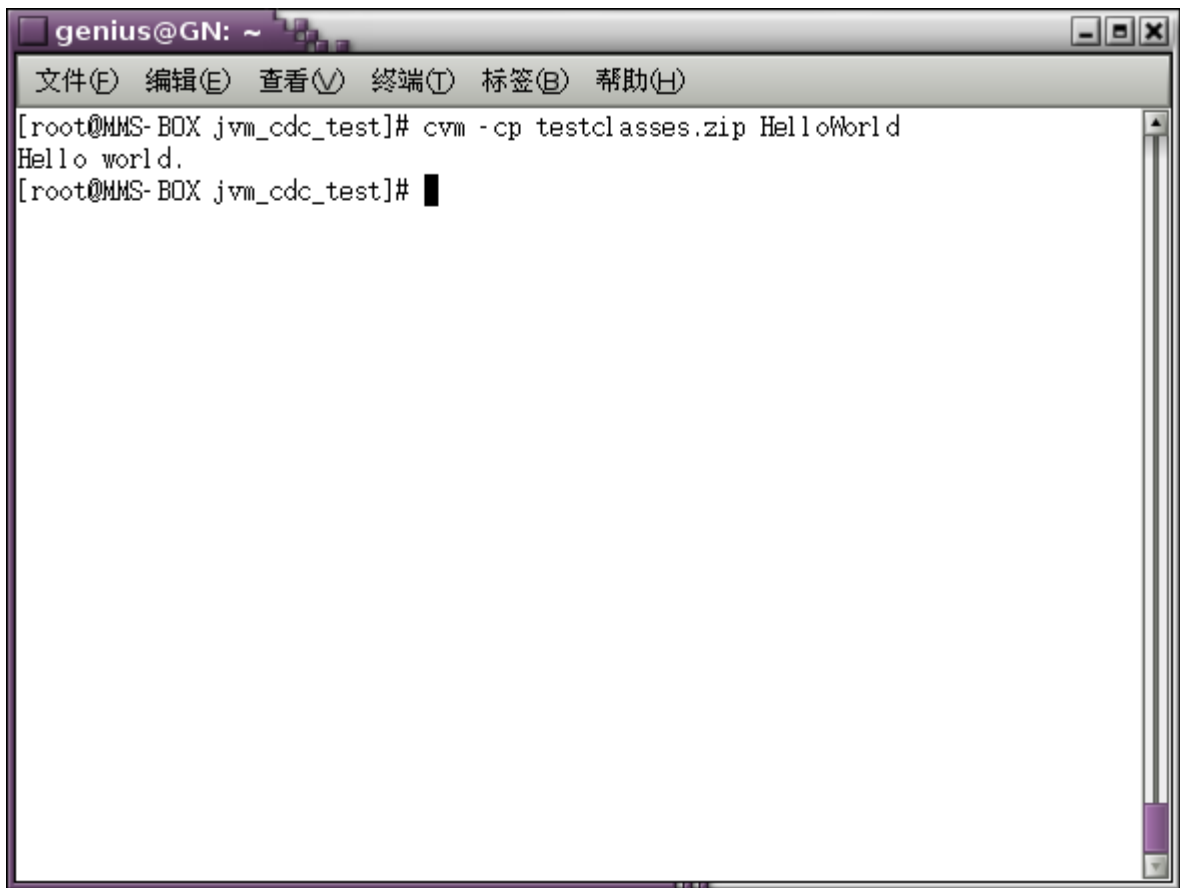
现在，你就可以把安装包下载到 ARM 平台上，并且解压安装，将其路径加入到环境变量中去。执行 **cvm -version** 如果能得到如下输出的话证明你的 CVM 能运行

A terminal window titled 'genius@GN: ~' with a menu bar containing '文件(E)', '编辑(E)', '查看(V)', '终端(T)', '标签(B)', and '帮助(H)'. The terminal content shows the command '[root@MMS-BOX cvm_thread_test]# cvm -version' and its output: 'Product: phoneme Advanced (phoneme_advanced_mr2-b34)', 'Profile: CDC Specification 1.1', and 'JVM: CVM phoneme_advanced_mr2-b34 (mixed mode)'. The prompt returns to '[root@MMS-BOX cvm_thread_test]#'.

```
genius@GN: ~
文件(E) 编辑(E) 查看(V) 终端(T) 标签(B) 帮助(H)
[root@MMS-BOX cvm_thread_test]# cvm -version
Product: phoneme Advanced (phoneme_advanced_mr2-b34)
Profile: CDC Specification 1.1
JVM: CVM phoneme_advanced_mr2-b34 (mixed mode)
[root@MMS-BOX cvm_thread_test]#
```

我还是建议你作人要保持低调，这只说明 CVM 能启动，我们还得看看它是否运行 JAVA 程序。来个经典的 Hello World，运行并观察输出

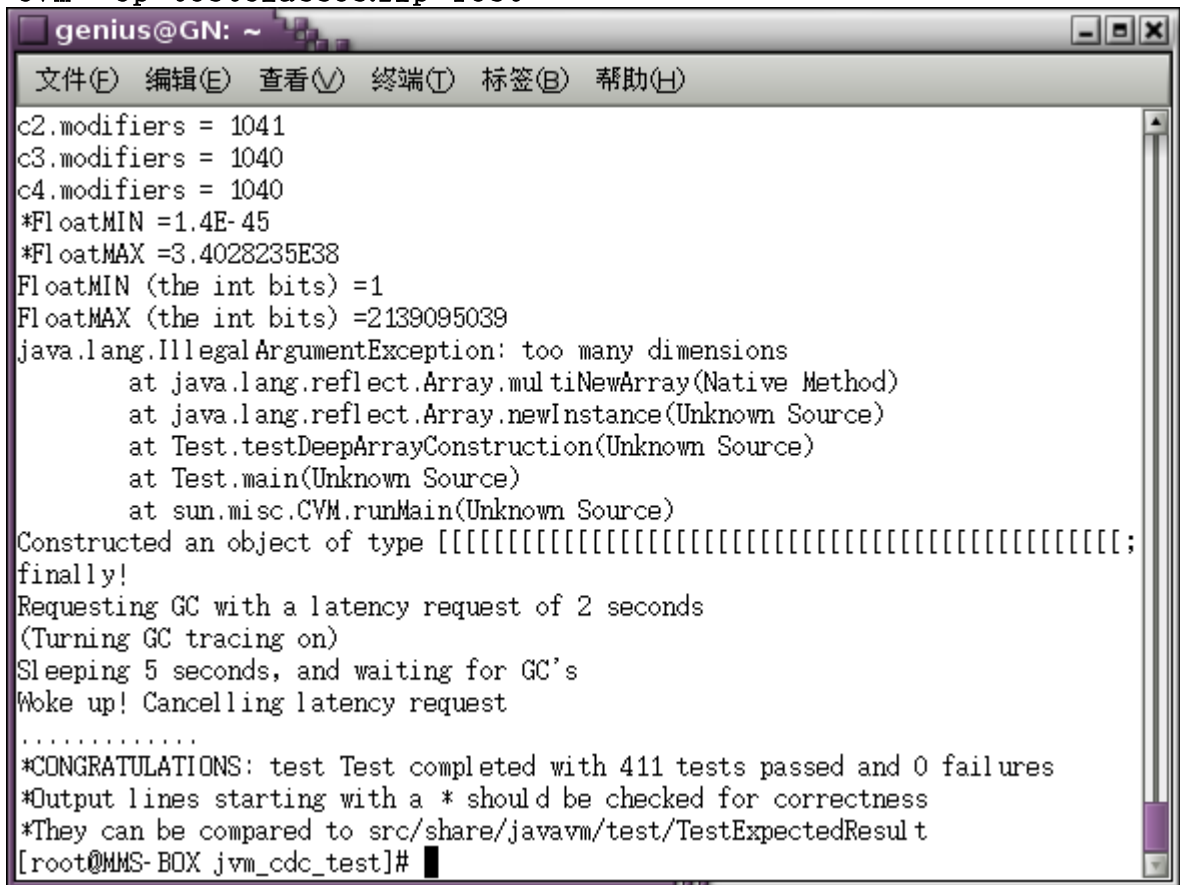
```
# cvm -cp testclasses.zip HelloWorld
```



```
genius@GN: ~
文件(E) 编辑(E) 查看(V) 终端(T) 标签(B) 帮助(H)
[root@mms-box jvm_cdc_test]# cvm -cp testclasses.zip HelloWorld
Hello world.
[root@mms-box jvm_cdc_test]#
```

笑到最后的人才能笑得最灿烂，所以我还是建议你在做一次全面的测试，运行

```
# cvm -cp testclasses.zip Test
```



```
genius@GN: ~
文件(E) 编辑(E) 查看(V) 终端(T) 标签(B) 帮助(H)
c2.modifiers = 1041
c3.modifiers = 1040
c4.modifiers = 1040
*FloatMIN =1.4E-45
*FloatMAX =3.4028235E38
FloatMIN (the int bits) =1
FloatMAX (the int bits) =2139095039
java.lang.IllegalArgumentException: too many dimensions
    at java.lang.reflect.Array.multiNewArray(Native Method)
    at java.lang.reflect.Array.newInstance(Unknown Source)
    at Test.testDeepArrayConstruction(Unknown Source)
    at Test.main(Unknown Source)
    at sun.misc.CVM.runMain(Unknown Source)
Constructed an object of type [;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
finally!
Requesting GC with a latency request of 2 seconds
(Turning GC tracing on)
Sleeping 5 seconds, and waiting for GC's
Woke up! Cancelling latency request
.....
*CONGRATULATIONS: test Test completed with 411 tests passed and 0 failures
*Output lines starting with a * should be checked for correctness
*They can be compared to src/share/javavm/test/TestExpectedResult
[root@mms-box jvm_cdc_test]#
```

不要被那个抛出的异常所吓到，那是 CVM 正在测试异常处理呢，不抛出异常才有问

题呢！好了，从最后的提示可以看到：411 tests passed and 0 failures。这意味这什么不用我讲了吧……

五、实验

乘着这股热乎劲，我决定在作一些实验。我想看看 CVM 的多线程怎么样，于是我写了下面代码

```
public class ThreadTester {
    public static void main (String args[]){
        System.out.println("/nGenius Test Java Thread:/n");
        PrintThread thread1 = new PrintThread( "thread1" );
        PrintThread thread2 = new PrintThread( "thread2" );
        PrintThread thread3 = new PrintThread( "thread3" );
        System.out.println( "Starting threads" );
        thread1.start();
        thread2.start();
        thread3.start();
        try {
            Thread.sleep(500);
        } catch (Throwable exception) {
            exception.printStackTrace();
        }
        System.out.println( "Threads started, main ends/n" );
    }
}

class PrintThread extends Thread {
    private int sleepTime;
    public PrintThread (String name) {
        super (name);
        sleepTime = (int) ( Math.random() * 3500 + 500);
    }
    public void run() {
        try {
            System.out.println(getName() + " going to sleep for " + sleepTime);
            Thread.sleep(sleepTime);
        }
        catch (InterruptedException exception) {
            exception.printStackTrace();
        }
        System.out.println(getName() + " do sleeping");
    }
}
```

主线程将创建三个线程并运行，之后将休眠 0.5 秒后结束，而每个线程启动后，随机休眠 0.5-4 秒后结束。用下面命令编译

```
# javac -source 1.4 ThreadTester.java
```

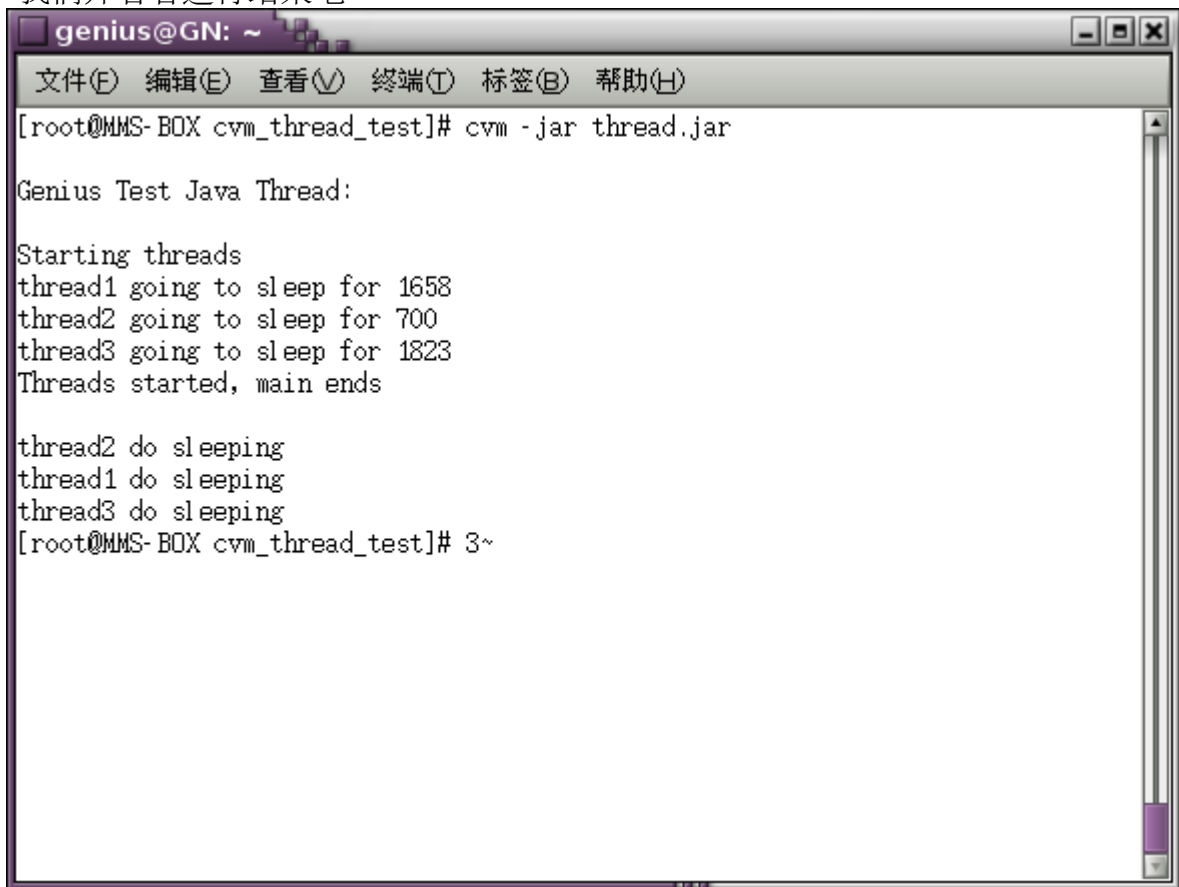
因为 CVM 之支持 JAVA 1.4 标准，所以使用高于 J2SE 1.4 版本时，请务必加上参数 -source 1.4 告诉编译器这个 1.4 的代码，请按 1.4 的标准编译，否则注定失败！编译完后再打个包方便下载

```
# jar -cvf thread.jar ThreadTester.class PrintThread.class
```

再把生成的 thread.jar 下载到 ARM 平台上，运行命令

```
# cvm -jar thread.jar
```

我们开看看运行结果吧



```
genius@GN: ~
文件(E) 编辑(E) 查看(V) 终端(T) 标签(B) 帮助(H)
[root@MMS-BOX cvm_thread_test]# cvm -jar thread.jar

Genius Test Java Thread:

Starting threads
thread1 going to sleep for 1658
thread2 going to sleep for 700
thread3 going to sleep for 1823
Threads started, main ends

thread2 do sleeping
thread1 do sleeping
thread3 do sleeping
[root@MMS-BOX cvm_thread_test]# 3~
```

因为线程二的休眠时间小于线程一的小于线程三的，所以线程的结束顺序是线程二先于线程一先于线程三，完全正确！

补充

CVM 是个可裁减的 JAVA 虚拟机，能够按照需求和资源进行定制，包括网络、GUI 等等。CVM 对 QT 的图形界面有很好的支持。想想你在 ARM 开发板硕大的一个屏幕上打手机 JAVA 游戏是多么有趣的一件事啊！

其他

本文由“湘大抽要饭的”原创，欢迎技术交流：hackerboygn(at)gmail.com