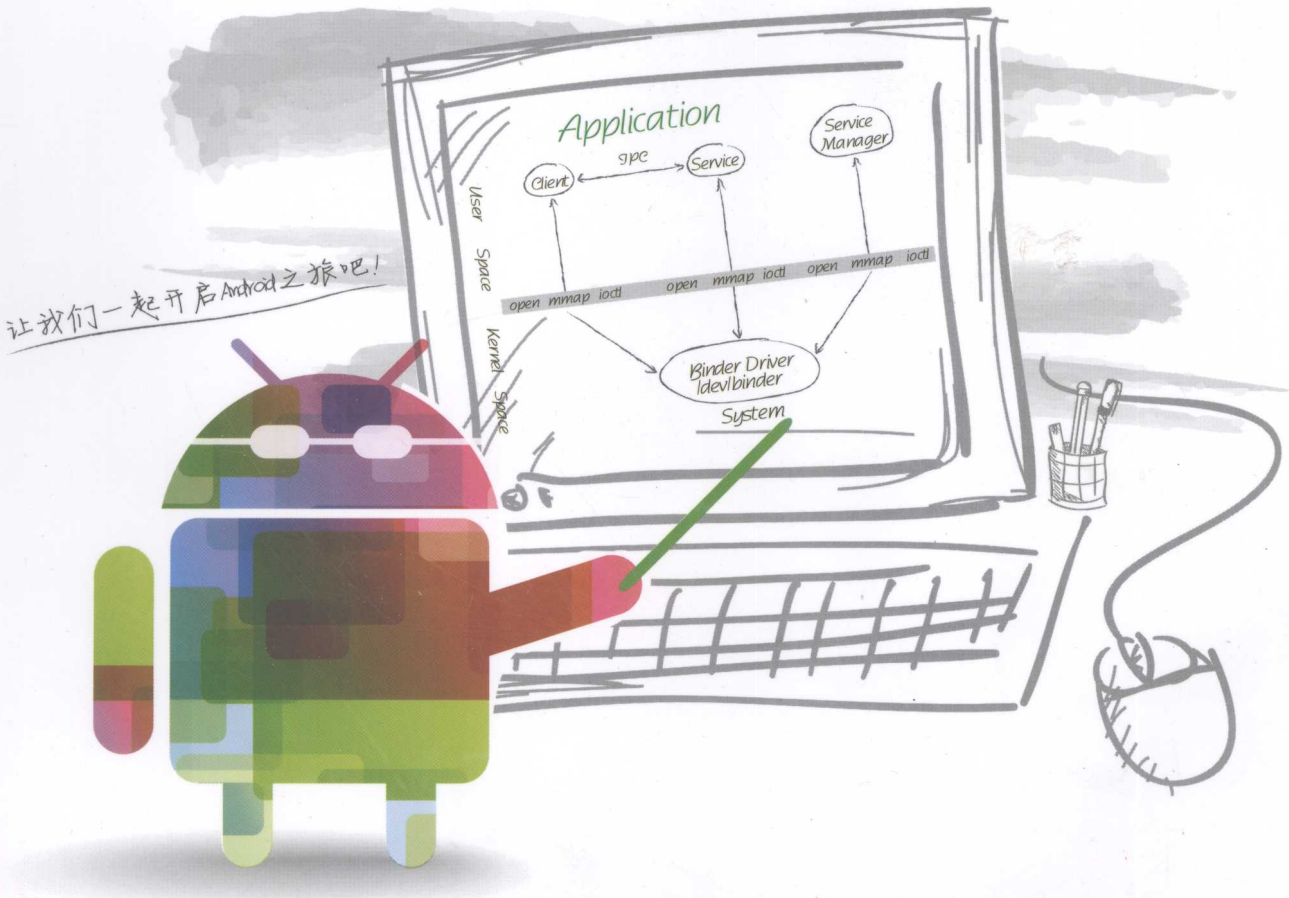
Broadview®
www.broadview.com.cn

全面、深入、细致地掌握Android系统，引领移动互联网新时代！

Android
系统源代码情景分析

◎罗升阳 著◎



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Android
系统源代码情景分析

◎罗升阳 著◎

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内容简介

在内容上，本书结合使用情景，全面、深入、细致地分析了Android系统的源代码，涉及到Linux内核层、硬件抽象层（HAL）、运行时库层（Runtime）、应用程序框架层（Application Framework）以及应用程序层（Application）。

在组织上，本书将上述内容划分为初识Android系统、Android专用驱动系统和Android应用程序框架三大篇。初识Android系统篇介绍了参考书籍、基础知识以及实验环境搭建；Android专用驱动系统篇介绍了Logger日志驱动程序、Binder进程间通信驱动程序以及Ashmem匿名共享内存驱动程序；Android应用程序框架篇从组件、进程、消息以及安装四个维度对Android应用程序的框架进行了深入的剖析。

通过上述内容及其组织，本书能使读者既能从整体上把握Android系统的层次结构，又能从细节上掌握每一个层次的要点。

图书在版编目（CIP）数据

Android系统源代码情景分析 / 罗升阳著. —北京：电子工业出版社，2012.10
ISBN 978-7-121-18108-5

Ⅰ. ①A… Ⅱ. ①罗… Ⅲ. ①移动终端—应用程序—程序设计 Ⅳ. ①TN929.53

中国版本图书馆CIP数据核字（2012）第204380号

策划编辑：符隆美
责任编辑：葛　娜
印　　刷：北京京科印刷有限公司
装　　订：三河市皇庄路通装订厂
出版发行：电子工业出版社
　　北京市海淀区万寿路173信箱　　邮编：100036
开　　本：850×1168　1/16　　印张：52.5　　字数：1570千字
印　　次：2012年12月第2次印刷
印　　数：4000册　　定价：109.00元（含光盘1张）

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

虽然Android系统自2008年9月发布第一个版本1.0以来，截至2011年10月发布最新版本4.0，一共存在十多个版本，但是据官方统计，截至2012年3月5日，占据首位的是Android 2.3，市场占有率达到66.5%；其次是Android 2.2，市场占有率为25.3%；第三位是Android 2.1，市场占有率为6.6%；而最新发布的Android 3.2和Android 4.0的市场占有率只有3.3%和2%。因此，在本书中，我们选择了Android 2.3的源代码来分析Android系统的实现，一是因为从目前来说，它的基础架构是最稳定的；二是因为它是使用最广泛的。

本书内容

全书分为初识Android系统篇、Android专用驱动系统篇和Android应用程序框架篇三个部分。

初识Android系统篇包含三个章节的内容，主要介绍Android系统的基础知识。第1章介绍与Android系统有关的参考书籍，以及Android源代码工程环境的搭建方法；第2章介绍Android系统的硬件抽象层；第3章介绍Android系统的智能指针。读者可能会觉得奇怪，为什么一开始就介绍Android系统的硬件抽象层呢？因为涉及硬件，它似乎是一个深奥的知识点。其实不然，Android系统的硬件抽象层无论是从实现上，还是从使用上，它的层次都是非常清晰的，而且从下到上涵盖了整个Android系统，包括Android系统在用户空间和内核空间的实现。内核空间主要涉及硬件驱动程序的编写方法，而用户空间涉及运行时库层、应用程序框架层及应用程序层。因此，尽早学习Android系统的硬件抽象层，有助于我们从整体上去认识Android系统，以便后面可以更好地分析它的源代码。在分析Android系统源代码的过程中，经常会碰到智能指针，第3章我们就重点分析Android系统智能指针的实现原理，也是为了后面可以更好地分析Android系统源代码。

Android专用驱动系统篇包含三个章节的内容。我们知道，Android系统是基于Linux内核来开发的，但是由于移动设备的CPU和内存配置都要比PC低，因此，Android系统并不是完全在Linux内核上开发的，而是在Linux内核里面添加了一些专用的驱动模块来使它更适合于移动设备。这些专用的驱动模块同时也形成了Android系统的坚实基础，尤其是Logger日志驱动程序、Binder进程间通信驱动程序，以及Ashmem匿名共享内存驱动程序，它们在Android系统中被广泛地使用。在此篇中，我们分别在第4章、第5章和第6章分析Logger日志系统、Binder进程间通信系统和Ashmem共享内存系统的实现原理，为后面深入分析Android应用程序的框架打下良好的基础。

Android应用程序框架篇包含十个章节的内容。我们知道，在移动平台中，Android系统、iOS系统和Windows Phone系统正在形成三足鼎立之势，谁的应用程序更丰富、质量更高、用户体验更好，谁就能取得最终的胜利。因此，每个平台都在尽最大努力吸引第三方开发者来为其开发应用程序。这就要求平台必须提供良好的应用程序架构，以便第三方开发者可以将更多的精力集中在应用程序的业务逻辑上，从而开发出数量更多、质量更高和用户体验更好的应用程序。在此篇中，我们将从组件、进程、消息和安装四个维度来分析Android应用程序的实现框架。第7章到第10章分析Android应用程序

四大组件Activity、Service、Broadcast Receiver和Content Provider的实现原理；第11章和第12章分析Android应用程序进程的启动过程；第13章到第15章分析Android应用程序的消息处理机制；第16章分析Android应用程序的安装和显示过程。学习了这些知识之后，我们就可以掌握Android系统的精髓了。

本书特点

本书从初学者的角度出发，结合具体的使用情景，在纵向和横向上对Android系统的源代码进行了全面、深入、细致的分析。在纵向上，采用从下到上的方式，分析的源代码涉及了Android系统的内核层（Linux Kernel）、硬件抽象层（HAL）、运行时库层（Runtime）、应用程序框架层（Application Framework）以及应用程序层（Application），这有利于读者从整体上掌握Android系统的架构。在横向上，从Android应用程序的组件、进程、消息以及安装四个角度出发，全面地剖析了Android系统的应用程序框架层，这有利于读者深入地理解Android应用程序的架构以及运行原理。

作　者

目　录

第1篇　初识Android系统

第2篇 Android专用驱动系统

第1篇 初识Android系统

Linux内核的鼻祖Linus在回答一个Minix系统[1]的问题时，第一句话便是“Read The Fucking Source Code”。这句话虽然颇有调侃的味道，但是它道出了阅读源代码的重要性。由于Android系统的源代码是开放的，因此，认识Android系统的最好方法莫过于是阅读它的源代码了。

我们知道，Android系统是基于Linux内核来开发的，因此，在阅读它的源代码之前，需要掌握Linux内核的基础知识。有了Linux内核的基础知识之后，就可以下载Android源代码来阅读了。阅读Android源代码时，宜采用动静结合的方法。所谓静，就是研究和思考源代码的实现；而所谓动，就是通过运行系统来证实自己对源代码的研究和思考。运行系统并不是单纯地将系统运行起来就可以了，还需要亲自动手编写一些应用程序来验证系统的行为。因此，在第1章中，我们首先介绍几本Linux内核的经典参考书籍，然后介绍如何搭建Android源代码工程环境，最后介绍如何在Android源代码工程环境中开发应用程序。

在阅读Android源代码之前，首先需要从整体上了解Android系统的架构。Android系统大致可以划分为五个层次，它们分别是Linux内核层、硬件抽象层、运行时库层、应用程序框架层和应用程序层。其中，硬件抽象层从实现到使用上涉及了这五个层次，因此，在第2章中，我们通过具体的实例来介绍Android系统的硬件抽象层，以便对Android系统的实现层次有一个感性的认识。

虽然Android应用程序是使用Java语言来开发的，但是在Android应用程序框架层中，有相当多的一部分代码是使用C++语言来编写的，在阅读这些C++代码时，经常会碰到智能指针。Android系统的智能指针对于C++开发者来说，是比较容易理解的；但是对于Java开发者来说，就比较苦涩了。因此，在第3章中，我们同样是通过具体的实例来分析Android系统的智能指针实现原理。

1 Minix是一个类UNIX的操作系统，见http://en.wikipedia.org/wiki/MINIX。

第 1 章

准备知识

Android系统的源代码非常庞大和复杂，我们不能贸然进入，否则很容易在里面迷失方向，进而失去研究它的信心。为了有条不紊地对Android系统的源代码进行全面、深入、细致的分析，我们需要准备一些参考书籍，搭建好Android系统源代码工程环境，以及对Android系统有一个感性认识。

1.1 Linux内核参考书籍

在阅读分析Android系统的源代码时，经常会碰到诸如管道（pipe）、套接字（socket）和虚拟文件系统（VFS）等知识。此外，Android系统通过模块的形式在Linux内核中增加了一些专用的驱动程序，如Logger日志驱动程序、Binder进程间通信驱动程序以及Ashmem匿名共享内存驱动程序等，这些都是Linux内核的基础知识，涉及进程、内存管理等内容。由于本书的重点是分析Android系统的源代码，因此，下面推荐四本介绍Linux内核基础知识的经典书籍。

（1）*Linux Kernel Development*

这本书的作者是Robert Love，目前最新的版本是第3版。它对Linux内核的设计原理和实现思路提供了一个总览视图，并且对Linux内核的各个子系统的设计目标进行了清晰的描述，非常适合初学者阅读。从软件工程的角度来看，这本书相当于Linux内核的概要设计文档。

（2）*Understanding the Linux Kernel*

这本书的作者是Daniel P. Bovet和Marco Cesati，目前最新的版本是第3版。它对Linux内核的实现提供了更多的细节，详细地描述了内核开发中用到的各种重要数据结构、算法以及编程技巧等，非常适合中、高级读者阅读。从软件工程的角度来看，这本书相当于Linux内核的详细设计文档。

（3）*Linux Device Drivers*

这本书的作者是Jonathan Corbet, Alessandro Rubini和Greg Kroah-Hartman，目前最新的版本是第3版。它更加注重于实际操作，详细地讲解了Linux内核驱动程序的实现原理，对分析Android系统的专用驱动模块有非常大的帮助。

（4）*Linux内核源代码情景分析*

这本书的作者是毛德操和胡希明，是中国人自己编写的一本经典的Linux内核书籍。它最大的特点是从使用情景出发，对Linux内核作了详细的分析，为读者在Linux内核源代码的汪洋大海中指明方向。

1.2 Android应用程序参考书籍

分析Android系统的源代码时，应该带着问题或者目标。要把问题或者目标挖掘出来，最好的方法就是在Android系统中编写应用程序。通过编写Android应用程序，我们可以知道系统提供了哪些功能，并且如何去使用这些功能，进而激发我们去了解这些功能是如何实现的。这样我们就可以获得分析Android系统源代码所需要的问题或者目标。下面推荐两本介绍Android应用程序开发的书籍。

（1）*Professional Android 2 Application Development*

（2）*Google Android SDK开发范例大全*

这两本书的特点是都使用了大量的例子来描述如何使用Android SDK来开发Android应用程序。我们可以根据实际情况来熟悉Android应用程序的开发方法，主要掌握Android应用程序四大组件（Activity、Service、Broadcast Receiver和Content Provider）的用法。在学习的过程中，如果遇到其他问题，还可以参考官方API文档，其网址为：

http://developer.android.com/index.html

1.3 下载、编译和运行Android源代码

目前，Android源代码工程环境只能在Linux系统上使用，本书推荐使用Ubuntu系统。Ubuntu系统是一个广受称道的Linux发行版本，它具有强大的软件包管理系统，并且简单易用，官方下载地址为：

http://www.ubuntu.com/

虽然Android源代码工程环境要求Ubuntu系统的最低版本是8.04，但还是建议读者从官方网站上下载最新的版本来安装。

如果读者习惯使用Windows系统，那么就可以考虑先在Windows上安装虚拟机，然后在虚拟机上安装Ubuntu系统。虚拟机推荐使用VMWare，官方网站为：

http://www.vmware.com/

安装VMWare时，最好选择6.0以上的版本，因为较旧版本的VMWare在网络连接支持上比较差，而我们在下载Android源代码时，是必须要联网的。

安装好Ubuntu系统之后，我们还需要安装一些工具和依赖包，然后才可以正常下载、编译和运行Android源代码。这些工具和依赖包包括Git和Java SDK等，接下来我们就介绍它们的安装方法。

1. Git工具

Android源代码采用Git工具来管理。Git是一种分布式的源代码管理工具，它可以有效、高速地对项目源代码进行版本管理[1]，它的安装方法如下：

```
USER@MACHINE:~$ sudo apt-get install git-core gnupg
```

2. Java SDK

编译Android源代码时，需要使用到Java SDK，它的安装方法如下：

```
USER@MACHINE:~$ sudo add-apt-repository ppa:ferramroberto/java[2]
USER@MACHINE:~$ sudo apt-get update
```

1 Git工具的使用方法可以参考官方网站http://git-scm.com/。

2 由于License问题，官方的JDK 6已经不能在Ubuntu上发布了。因此，现在可能已经无法从下载源ferramrobert上安装JDK 6了。可以选择其他下载源来解决这个问题，或者直接从官方网站下载JDK 6到本地来手动安装。官方JDK 6的下载地址为http://www.oracle.com/technetwork/java/javase/downloads/index.html。

```
USER@MACHINE:~$ sudo apt-get install sun-java6-jre sun-java6-plugin
USER@MACHINE:~$ sudo apt-get install sun-java6-jdk
```

3. 其他依赖包

编译Android源代码时，还需要用到其他的工具包，它们的安装方法如下：

```
USER@MACHINE:~$ sudo apt-get install flex bison gperf libsdl-dev libesd0-dev libwxgtk2.6-dev
build-essential zip curl valgrind
```

安装好这些工具和依赖包之后，接下来就可以下载、编译和运行Android源代码了。

1.3.1 下载Android源代码

为了方便开发者下载Android源代码，Google提供了一个repo工具。这个工具实际上是一个脚本文件，里面封装了用来下载Android源代码所需要的git命令。它的下载和安装方法如下：

```
USER@MACHINE:~$ wget https://dl-ssl.google.com/dl/googlesource/git-repo/repo
USER@MACHINE:~$ chmod a+x repo
USER@MACHINE:~$ sudo mv repo /bin/
```

安装好repo工具之后，我们就可以创建一个空目录，然后进入到这个目录中执行repo命令来下载Android源代码了。

```
USER@MACHINE:~$ mkdir Android
USER@MACHINE:~$ cd Android
USER@MACHINE:~/Android$ repo init -u https://android.googlesource.com/platform/manifest
USER@MACHINE:~/Android$ repo sync
```

下载的过程可能会比较漫长，这取决于网络连接速度，期间还可能会碰到网络中断的现象，这时候只需要重复执行repo sync命令就可以继续下载了。

上述命令下载的是主线上的Android源代码，即最新版本的Android源代码。一般来说，主线上的源代码是正在开发的版本，它是不稳定的，编译和运行时都可能会遇到问题。如果想下载稳定的版本，就需要选择某一个支线上的代码。例如，如果我们想下载Android 2.3.1版本的代码，就可以在执行repo init命令时指定-b选项。

```
USER@MACHINE:~/Android$ repo init -u https://android.googlesource.com/platform/manifest -b
android-2.3.1_r1
```

在本书接下来的内容中，如果没有特别声明，我们所分析的Android源代码都是基于Android 2.3版本的，并且位于~/Android目录中。

1.3.2 编译Android源代码

要编译Android源代码，只需在Android源代码目录下执行make命令就可以了。

```
USER@MACHINE:~/Android$ make
```

第一次编译Android源代码时，花费的时间会比较长，同时也可能会遇到各种各样的问题，这时候一般都可以通过搜索引擎来找到解决方案。例如，如果我们是在32位机器上编译主线上的Android源代码，则会碰到下面这个错误提示。

```
build/core/main.mk:76: ************************************************************
build/core/main.mk:77: You are attempting to build on a 32-bit system.
build/core/main.mk:78: Only 64-bit build environments are supported beyond froyo/2.2.
build/core/main.mk:79: ************************************************************
```

这时候可以使用关键词“You are attempting to build on a 32-bit system”在搜索引擎上找到解决方案。原来，主线上的Android源代码默认只能在64位的机器上编译，如果在32位的机器上编译，就会出现上述错误提示。如果我们仍然想在32位的机器上编译Android源代码，就可以按照下面方法来修改编译脚本。

（1）打开build/core/main.mk文件，并且找到下面内容：

```
ifeq ($(BUILD_OS),linux)
     build_arch := $(shell uname -m)
     ifneq (64,$(findstring 64,$(build_arch)))
     $(warning ************************************************************)
     $(warning You are attempting to build on a 32-bit system.)
     $(warning Only 64-bit build environments are supported beyond froyo/2.2.)
```

将第3行修改为：

```
ifneq (i686,$(findstring i686,$(build_arch)))
```

（2）打开external/clearsilver/cgi/Android.mk、external/clearsilver/cs/Android.mk、external/clearsilver/java-jni/Android.mk和external/clearsilver/util/Android.mk这四个文件，并且找到下面内容：

```
# This forces a 64-bit build for Java6
LOCAL_CFLAGS += -m64
LOCAL_LDFLAGS += -m64
```

将后面两行修改为：

```
LOCAL_CFLAGS += -m32
LOCAL_LDFLAGS += -m32
```

经过这样的修改之后，在32位的机器上编译Android源代码产生的问题就可以解决了。

编译成功后，可以看到下面的输出：

```
Target system fs image: out/target/product/generic/obj/PACKAGING/systemimage_intermediates/
system.img
Install system fs image: out/target/product/generic/system.img
Target ram disk: out/target/product/generic/ramdisk.img
Target userdata fs image: out/target/product/generic/userdata.img
Installed file list: out/target/product/generic/installed-files.txt
```

编译结果输出目录为out/target/product/$(TARGET_PRODUCT)，其中，TARGET_PRODUCT是一个环境变量，它的默认值为generic。

Android源代码编译成功之后，可以执行以下命令将它打包成SDK：

```
USER@MACHINE:~/Android$ make sdk
```

打包成功后，可以看到下面的输出：

```
Package SDK: out/host/linux-x86/sdk/android-sdk_eng.$USER$_linux-x86.zip
```

其中，$USER$表示当前登录到系统中的用户名。有了这个SDK包之后，我们就可以在IDE环境中开发Android应用程序了。

1.3.3 运行Android模拟器

Android源代码编译成功之后，我们就可以运行它了。为了方便起见，我们使用Android模拟器emulator来运行编译出来的Android源代码。执行以下命令来启动Android模拟器：

```
USER@MACHINE:~/Android$ export PATH=$PATH:~/Android/out/host/linux-x86/bin
USER@MACHINE:~/Android$ export ANDROID_PRODUCT_OUT=~/Android/out/target/product/generic
USER@MACHINE:~/Android$ emulator
```

Android模拟器位于源代码根目录下的子目录out/host/linux-x86/bin中，因此，为了方便使用，我们把这个目录添加到环境变量PATH中。

启动Android模拟器需要四个文件，它们分别是zImage、system.img、userdata.img和ramdisk.img，其中，前面一个是Linux内核镜像文件，而后面三个是Android系统镜像文件。如果不带任何参数来运行emulator命令，那么Android模拟器默认使用的zImage文件是位于源代码根目录下的子目录out/host/linux-x86/bin中的kernel-qemu文件，而默认使用的system.img、userdata.img和ramdisk.img文件则位于ANDROID_PRODUCT_OUT目录中。ANDROID_PRODUCT_OUT是一个环境变量，我们将它的值设置为Android源代码编译结果输出目录；如果不设置ANDROID_PRODUCT_OUT环境变量，就需要指定上述四个文件来启动Android模拟器，如下所示：

```
USER@MACHINE:~/Android$ emulator -kernel ./prebuilt/android-arm/kernel/kernel-qemu -sysdir ./
out/target/product/generic -system system.img -data userdata.img -ramdisk ramdisk.img
```

在本书接下来的内容中，当我们运行不带参数的emulator命令时，如果没有特别声明，都是假设已经设置好PATH和ANDROID_PRODUCT_OUT两个环境变量，其中PATH环境变量包含了~/Android/out/host/linux-x86/bin目录，而ANDROID_PRODUCT_OUT环境变量指向~/Android/out/target/product/generic目录。

Android模拟器emulator启动起来后，它的界面如图1-1所示。

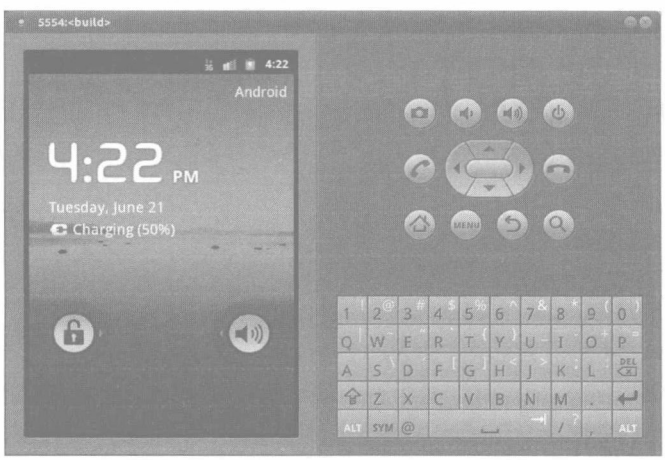


图1-1 Android模拟器界面

1.4 下载、编译和运行Android内核源代码

Android源代码工程默认是不包含它所使用的Linux内核的源代码的，因此，如果我们需要运行定制的Linux内核，就要下载它的源代码，并且对它进行编译。接下来，我们就详细介绍如何为Android模拟器下载、编译和运行Linux内核源代码。

1.4.1 下载Android内核源代码

Android模拟器所使用的Linux内核源代码也是通过Git工具来管理的，不过它不像Android源代码有一个repo脚本工具来支持下载，我们需要手动执行git命令来下载。

```
USER@MACHINE:~/Android$ mkdir kernel
USER@MACHINE:~/Android$ cd kernel
USER@MACHINE:~/Android/kernel$ git clone http://android.googlesource.com/kernel/goldfish.git
```

首先在Android源代码根目录下创建一个用来保存Linux内核源代码的子目录kernel，然后调用git clone命令来下载Linux内核源代码。取决于网络连接速度，整个下载过程可能需要较长时间的等待。

下载完成之后，在kernel目录下就会看到一个空的goldfish子目录，这时候需要执行git checkout命令来指定所需要的支线代码。

```
USER@MACHINE:~/Android/kernel/goldfish$ git branch -a
* master
  remotes/origin/HEAD -> origin/master
  remotes/origin/android-goldfish-2.6.29
  remotes/origin/master
USER@MACHINE:~/Android/kernel/goldfish$ git checkout remotes/origin/android-goldfish-2.6.29
```

在执行git checkout命令之前，可以先执行git branch命令来查看有哪些支线代码。由于我们使用Android模拟器来运行Android系统，因此，我们选择android-goldfish-2.6.29支线代码来作为Android系统的内核源代码。

执行完成git checkout命令之后，我们就可以在goldfish目录中看到Linux内核源代码了。在本书接下来的内容中，如果没有特别声明，我们分析的Android内核源代码都是基于android-goldfish-2.6.29版本的，并且位于~/Android/kernel/goldfish目录中。

1.4.2 编译Android内核源代码

在编译Android内核源代码之前，我们首先要修改它的Makefile文件。Android模拟器所使用的CPU体系结构是arm的，因此，我们需要将Makefile文件中ARCH变量的值设置为arm。又由于我们是在PC上为Android模拟器编译内核的，因此，还需要在Makefile文件中指定交叉编译工具，即修改里面的CROSS_COMPILE变量的值。

打开Android内核源代码目录下的Makefile文件，并且找到下面的内容：

```
export KBUILD_BUILDHOST := $(SUBARCH)
ARCH                    ?= $(SUBARCH)
CROSS_COMPILE           ?=
```

将后面两行修改为：

```
ARCH                    ?= arm
CROSS_COMPILE           ?= arm-eabi-
```

Android源代码目录为我们准备了一个适用于编译Android模拟器内核的交叉编译工具，它位于Android源代码目录下的prebuilt/linux-x86/toolchain子目录中。在Makefile文件中，我们将ARCH变量的值设置为arm，表示编译的Linux内核是适用于arm体系结构的；而将CROSS_COMPILE变量的值设置为arm-eabi-，表示所使用的交叉编译工具名称是以“arm-eabi-”来作为前缀的。

为Android模拟器编译内核分为三个步骤。其中，第一步是将交叉编译工具所在的目录添加到环境变量PATH中；第二步是修改硬件配置文件goldfish_defconfig[2]；第三步是执行make命令。

2 取决于Android模拟器的CPU体系架构版本，如果是arm v5版本的，则使用goldfish_defconfig文件；如果是arm v7版本的，则使用goldfish_armv7_defconfig文件。可以使用Android源代码目录下的prebuilt/android-arm/kernel子目录中的kernel-qemu文件来启动Android模拟器，然后使用adb工具来连接Android模拟器，并且通过cat命令来查看/proc/cpuinfo文件的内容，就可以知道当前所使用的Android模拟器的CPU体系架构版本。