第 3 章　英特尔 Android 应用开发流程和工具链

本教材的重点是介绍基于 Intel 硬件平台的 Android 应用开发，而采用 Android 软件平台（即操作系统）的 Atom 设备，如智能手机、平板电脑等，都属于嵌入式系统。嵌入式系统的应用开发有着通用计算机应用开发所不具备的特殊性和困难性。嵌入式系统应用开发会用到一些专用的开发、调试、性能分析工具，而其开发环境、目标代码格式等与通用桌面计算机都有所不同。因此在进行 Android 应用开发之前，我们先来熟悉一下嵌入式应用开发流程。

Android 为其应用开发提供了一套完整的工具链（或称工具集）。早期的 Android 都是针对 ARM 架构硬件平台的，最近才开始支持 Atom 硬件平台。为支持 Atom 架构应用的开发，Intel 在 Android 工具链基础上增加了重要的插件、库等辅助模块。除此之外，为了更能发挥英特尔硬件的性能优势，英特尔补充了一些诸如编译器、调优等有特色的开发工具。这些都是原始的 Android 开发工具所不具备的，因此我们把加入 Intel 模块、支持 Atom 架构高软件性能的 Andoid 开发工具称为 Intel Android 应用工具链。

关于 Intel Android 应用工具链，我们准备分成两部分来介绍，本章主要介绍 Atom 平台上的 Android 应用开发的一般流程与方法，而后续的专门章节，我们会介绍利用 Intel 专用工具实现性能优化、低功耗设计的方法。

3.1　嵌入式应用开发流程

3.1.1　嵌入式应用开发环境

3.1.1.1　交叉开发

前面说过，通用计算机的软件开发一般都是以本地（Native）编译或开发的方式进行，嵌入式系统一般不支持本地环境开发，其软件开发通常采用交叉（Cross）开发的方式。典型的交叉开发配置如图 3-1 所示。交叉开发环境建立在宿主机（Host）上，宿主机又称开发机，一般是一台通用计算机，如 PC。对应嵌入式系统称为目标（Target）机。目标机指各式各样的嵌入式设备，例如手机、掌上电脑等；或者是嵌入式厂商提供的一套专用于开发的评估板（Evaluation Board），所以又称开发板；甚至是基于软件的模拟器。开发时使用宿主机上的交叉编译、汇编和链接工具形成可在目标机上执行的二进制代码，然后把可执行文件下载到目标机上运行。嵌入式系统开发过程中不仅编译（包括链接等步骤）采用交叉的方式，调试也常采用交叉的方式。

图 3-1　嵌入式系统的交叉开发配置

嵌入式系统开发之所以采用交叉开发的主要原因在于目标机上往往无法进行有效的本地编译。首先，目标机硬件本身在开发过程中还不能使用或还不够稳定；其次，目标机平台上缺乏完整的本地编译工具、环境；再者，目标机系统本身性能不够，导致编译太慢。嵌入式系统的软件编译，与桌面机（如 Windows）开发不一样，不仅要编译应用程序，还要编译相应的依赖库、操作系统内核等。所以一次完整的编译非常费时。比如编译一个 Linux 内核在奔 4 级别的 PC 上都需要十几分钟。而从硬件角度，决定编译速度的主要因素是 CPU 速度、内存容量和文件系统 I/O 速度。而这些方面在嵌入式系统上往往都要比 PC 差很多。这样会造成目标机系统上进行本地编译效率低下。综合以上多方面因素的考虑，嵌入式系统开发一般都采用交叉开发方式，如交叉编译（包括交叉链接）、交叉调试等。

宿主机和目标机的配置、功能、体系结构和使用环境有很大的不同，它们之间一般通过串口、并口、USB 或以太网等连接线进行连接。在宿主机上会安装用于嵌入式软件开发的一系列工具集，例如代码编辑工具、编译器、连接器、调试工具、软件配置管理工具等。开发者在宿主机一端完成代码编写和调试。

宿主机和目标机通常有如下不同：

- 体系结构的不同。宿主机和目标机通常是异构的。宿主机一般采用 x86 体系结构，但是目标机的体系结构则可能非 x86 的，如是 ARM、MIPS、PowerPC 等各式各样。
- 处理能力不同。通常宿主机的处理速度、存储容量等会远远大于目标机。
- 运行的操作系统不同。宿主机一般运行通用操作系统，而目标机通常运行各种嵌入式操作系统。
- 输入输出方式不同。相对宿主机，目标机的输入输出功能可能比较单一。

要说明的是，以上特点仅仅是一般嵌入式开发环境常见的特征，对某些嵌入式系统而言，以上某些特征可能不具备或表现不明显。例如凌动系统开发，宿主机和目标机多半会使用相同的体系结构——x86 系统，当然执行的指令集可能不一样，如宿主机（如 Core 2 Duo 处理器）可能支持 SSE4，而凌动仅仅支持到 SSE3 和 SSSE3，因此编译时还是要考虑到目标机型号和指令集的选择。此外，如果目标机是诸如平板电脑之类功能较为强大的系统，这些系统的开发也可以不采用交叉开发的方式，可以直接在本机上编辑—编译—链接生成应用程序执行。考虑到凌动系统一般都是资源受限的，因此我们推荐使用交叉开发的方式。

3.1.1.2　编程语言

通用计算机系统应用的编程语言很多，在过去四十多年中，人们开发研制出了上百种

计算机编程语言，从底层的汇编语言，到 FORTRAN、C/C++、ADA、Module 高级语言，再到与平台无关的 Java、C#.NET 等，数不甚数。从面向过程的 PASCAL、C，发展到面向对象(Object-Oriented Programming，OOP)的 C++、Java、C#，到泛型编程(Generic Programming，GP)的 C++等，其反映的编程思想也各异。对于编程语言来说，并没有优劣之分，只有对具体需求适用不适用之别。考察一门语言是否适用，需要从多个方面进行考虑。不同的语言，都有自己的特色，很难将其进行全面比较。另外任何一门语言的运行特性都与运行环境密切相关，因此选择语言时应综合考虑。考虑到多个方面的因素和现实的发展现状，嵌入式系统比较常用的编程语言有 C/C++、Java 和 Python，在底层方面又不得不用到汇编语言，虽然其使用量不大。复杂的嵌入式系统一般由多种语言混合编写而成，常见的编程语言选择可能如表 3-1 所示。

表 3-1　　常见的编程语言选择

级别	常用的编程语言
应用程序	C/C++、Java、.NET、脚本、Python
操作系统级别	C/C++、汇编
驱动程序级别	C/C++、汇编
启动代码，硬件抽象层(HAL)	汇编、C/C++

Java 是一种可以撰写跨平台应用软件的面向对象的程序设计语言，是由 Sun Microsystems 公司于 1995 年 5 月推出的 Java 程序设计语言和 Java 平台(即 JavaSE、JavaEE、JavaME)的总称。Java 编程语言的风格十分接近 C、C++语言，它是一个纯的面向对象的程序设计语言，它继承了 C++语言面向对象技术的核心，舍弃了 C++语言中容易引起错误的指针(以引用取代)、运算符重载(Operator Overloading)、多重继承(以接口取代)等特性，增加了垃圾回收器功能用于回收不再被引用的对象所占据的内存空间，使得程序员不用再为内存管理而担忧。在 Java 1.5 版本中，Java 又引入了泛型编程(Generic Programming)、类型安全的枚举、不定长参数和自动装/拆箱等语言特性。

与一般的编译执行计算机语言不同，Java 是一种解释执行计算机语言。Java 编译器产生的二进制字节代码(Byte Code)，而不是本地可以直接执行的机器代码。编译后的 Java 程序由 Java 虚拟机(Java Virtual System，缩写为 JVM)解释成本地机器码执行。虚拟机在不同平台上来解释执行字节码，从而实现了“一次编译、到处执行”的跨平台特性。不过，每次的执行编译后的字节码需要消耗一定的时间，这同时也在一定程度上降低了 Java 程序的运行效率。总的来说，Java 是一个简单、面向对象、分布式、解释性、健壮、安全与系统无关、可移植、高性能、多线程和动态的编程语言。考虑到 Java 诸多优点，Android 应用开发的首选语言就是 Java。

选定一个高级语言之后，也未必在项目中使用其所有功能。Android 虽然使用 Java 语言作为开发工具，但是在实际开发中还是与传统(即桌面型)Java SDK 有一些不同的地方。Android SDK 引用了大部分的 Java SDK，少数部分被 Android SDK 抛弃，比如说界面部分，java.awt package 除了 java.awt.font 被引用外，其他都被抛弃。例如，将 Java 游戏移植到 Android 平台的过程中，很可能变得不可用。

前面说过 Java 是一种跨平台的解释执行计算机语言，这种特征虽然造就了它的平台

无关的高度可移植性，但是也带来了一定弊病，其中之一是无法利用平台、架构相关的特征与潜能。而对此，编译 C/C++、汇编语言得到的与机器相关的目标代码却可以实现，这种情况在性能优化时尤显突出。为了在某些地方利用机器硬件特点，挖掘其性能潜能时，我们往往需要用 C/C++、汇编等 Java 外的语言来编写应用代码。当然这些部分所占的代码比例并不高，其编程的繁杂度也远远高于 Java，因此只在少数地方使用，正所谓“好钢用在刀刃上”。后面可以看到，Intel 平台的 Android 应用开发采用的是一种以改进的 Java 为主，以 C、汇编等语言为辅的混合编程模式。对此编程方法的介绍也是分两个部分来进行，本章和后续介绍 Android 应用通用功能开发的章节，我们采用 Java 语言；而在再后面的性能优化等专门章节里面，我们再介绍混合编程的知识。

3.1.2　嵌入式应用开发流程

一般来说，嵌入式软件开发依次要经历编辑、编译、链接、打包、部署、调试、优化等步骤，在某些嵌入式系统中可能还需要测试和验证等步骤。从流程上说，大致可分为编码阶段、构建阶段、部署阶段、调优和其他阶段。典型的开发流程如图 3-2 所示。

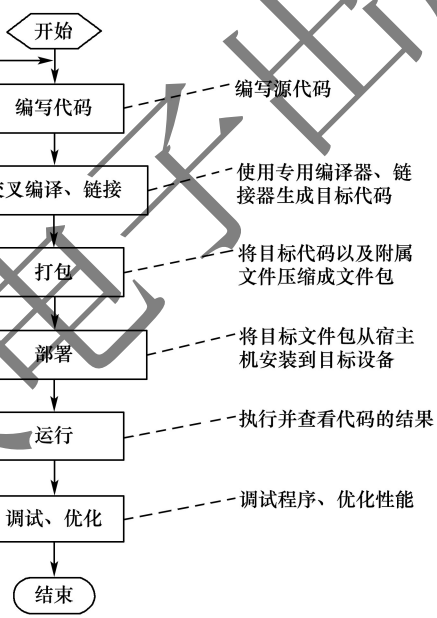


图 3-2　嵌入式软件开发流程

1. 编码阶段

编码阶段是软件开发的开始，其任务是编写软件的源代码，使用的工具是各种编辑器。在 Android 开发中，这部分工作主要是编辑.java 代码、.xml 等各种源文件。

2. 构建阶段

构建阶段的任务是把代码转化成可以在嵌入式硬件上可执行程序的过程。此阶段包括编译、链接、打包等子步骤。详细过程如图 3-3 所示。

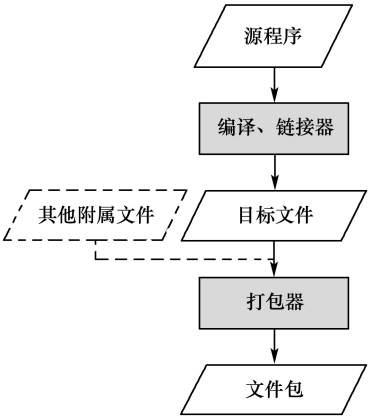


图 3-3 软件构建阶段

构建的第一步是生成(Build),即将各种源代码文件翻译成目标文件。有些目标文件是与机器相关的,如 C/C++的目标文件,它们对应机器的执行指令。而有些是与机器无关的,如 Java 的目标代码,它们不是机器的执行指令。在 Android 应用开发中,这些目标文件一般是以.class 为后缀名的文件。

对于目标文件为机器执行指令的语言的生成,一般要包括编译和链接两个阶段,前者将程序设计语言编写的源代码翻译成特定处理器上等效的一系列操作码,后者所有目标文件链接成一个目标文件,该目标文件可以在目标机上直接执行。

构建的第二步是打包。打包的目标是将目标文件以及附属文件整合为一个安装到目标机上文件包。在 Android 中就是将.class 文件、资源文件等多个文件打包成一个.apk 文件,该文件可以安装目标机外存中。打包操作通常由专门的打包工具完成。

3. 部署阶段

软件开发的最后一个阶段是部署,即把安装包从宿主机拷贝、解开并安装到嵌入式设备的只读型存储器上,这样是目标文件最终运行于嵌入式设备上。

不少嵌入式系统采用嵌入的 ROM 或闪存(Flash)等只读型存储器来保存数据。将数据写入到这些存储器中需要有一个专门的烧写过程(又称烧录),因此在这些系统中部署又称为烧录,实现此过程也需借助专门的烧写工具。而 Android 的主要应用硬件平台,如上网本、智能手机、平板电脑等,都安装有操作系统,操作系统对用户屏蔽了只读型存储器的底层细节,用户对只读型存储器的操作如同对文件的操作一样,因此我们还是在此阶段采用"部署"(Deploy)这一通用术语。

从实现的形式来说,部署可以分为脱线(Off-line)和在线编程模式(In-System Program,ISP)两种方式。在脱线部署中,需要将 Flash ROM、SD 卡等只读型存储器芯片从目标机上取下,使用专门的编程器和配套软件来完成烧写,然后将烧写好的只读型存储器插回目标机,这种方式中宿主机与目标机是分离的。而在在线编程模式中,宿主机与目标机通过 JTAG-ICE(在后续章节介绍)、网络、USB 等方式相连,宿主机上的目标文件通过网络、JTAG-ICE、USB 线等直接写到目标机的只读存储器中。

Android 应用开发既可以采用在线编程模式方式,也可以采用脱线方式,如图 3-4 所

示。在线编程模式中，宿主机与目标机采用 USB 线连接，目标机运行的是 Android 操作系统，而宿主机采用 Windows 或 Linux 操作系统，交叉编译生成的目标文件文件包——.apk文件——复制到目标机文件系统的某个目录下，解开并安装之，这样完成部署。在脱线方式上，交叉编译生成的目标文件包先从宿主机辅助到 SD 卡等移动存储器上，然后从宿主机上拔出来再插到目标机上，最后在目标机上将移动存储器上的目标文件包解开并安装到目标机上。

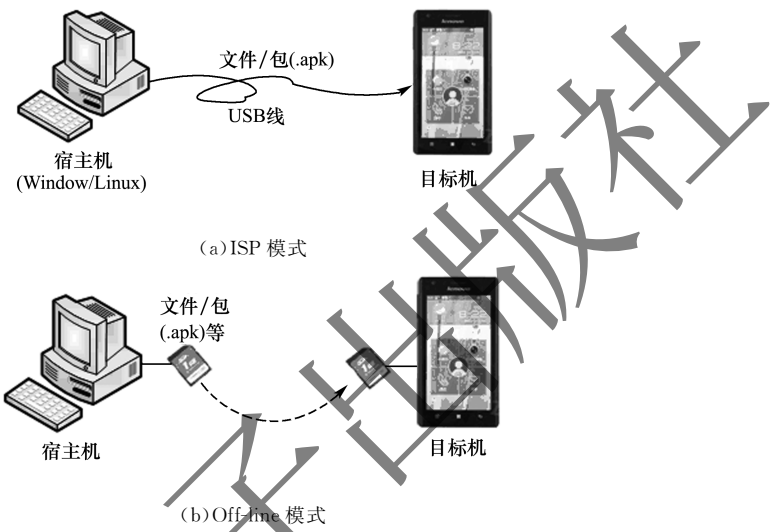


图 3-4　Android 应用的部署

在线编程模式中，根据宿主机和目标机之间文件拷贝的方向不同，对文件拷贝有不同别称。例如下载/上载，在 Android 中称为推(Push)/拉(Pull)，其中从宿主机拷贝文件到目标机称为推，而从目标机拷贝文件到宿主机称为拉。

脱线方式可以完成交叉编译文件的部署，但是难以完成交叉在线调试(在下面部分介绍)，其提供的功能不如 ISP 方式那么强，部署的速度和方便性都不如 ISP 方式，因此其使用面比在线方式广。本教材的例子如果不加特别说明，一般都是用 ISP 方式。当然一些特殊场合，如操作系统的定制安装等，一些特殊设备，如不带 USB 连线的机器，只能使用脱线方式，对此我们会有专门的说明。

4. 调优阶段(调试、优化阶段)

此阶段主要是对软件进行调试和性能优化。

即使再有经验的软件工程师，要编写完全没有错误的程序也是不可能的。当代码不能按照希望的方式运行时，就需要开发人员对代码进行调试。熟练掌握调试技术对嵌入式软件开发至关重要。嵌入式软件调试的效率都比较低，因为即使改动一行代码，都可能需要把生成、打包、部署等步骤完整走一遍，这也是调优阶段最显著的特点，快速地定位到问题可以节省大量的时间。其次，如果将错误带到最终的嵌入式系统产品中，其后果可能是致命的。对于 PC 用户，用户或许可以忍受自己的机器每天死机一两次，但是想象在 ATM 上取款，在医疗系统上手术或在卫星发射时，软件发生异常，其带来的后果有多严重。

嵌入式软件开发用到的调试技术和技巧繁多,有许多方法是在通用计算机系统中难以见到的,而且所用到的许多设备也是嵌入式系统所独有的。对此我们会在专门章节来介绍。

对于软件产品最低目标是能正确地运行,但是这一目标对于嵌入式软件来说是不够的,因为嵌入式系统是一个资源受限的系统,它对程序的运行空间和时间要求比桌面系统要苛刻得多,为适应此需求,嵌入式软件在正确运行这一基本要求上,还需要使其能以性能最优的方式运行,这就是软件优化所进行的工作。优化目标主要包括速度性能、功耗性能、空间性能等方面,达到运行更快、更节能、占用空间更小等目的。以上目标可能存在互相抵触的地方,难以做到熊掌和鱼兼得,对此人们需要根据实际情况选择某种折中。在大多数情况下,人们会突出对运行速度的要求。

改进应用程序的运行性能是一项非常耗时耗力的工作,但是究竟程序中是哪些函数消耗掉了大部分执行时间,这通常不是非常明显的,对此需要借助专门的工具来分析其代码。借助工具分析程序代码,精确分析性能瓶颈,据此引导和建议开发者进行改进,此过程常称为代码简档(Code Profiling),对应的工具常常称为简档器(Profiler)或性能分析器(Performance Analyzer)。

通常,简档器优化运行性能的原理是尽量优化软件中被频繁调用的部分。例如,假设应用程序花了 50%的时间在字符串处理函数上,如果可以对这些函数进行优化,提高 10%的效率,那么应用程序的总体执行时间就会改进 5%。使用简档器工具可以精确测量应用程序执行过程中时间都花费到什么地方去了,这样做的目的是了解一下在什么地方进行优化效果最佳。有些简档器还能根据处理器的特点提出相应的改进建议,例如,英特尔的 VTune 能在适合的热点区域(Hotspot)给出用向量运算指令代替原指令来加快处理速度的建议。

5. 其他阶段

在某些特殊的嵌入式系统上,软件开发还需进行一些特殊的步骤。

软件验证是其中之一,软件验证是验证程序逻辑上的正确性和常见的错误。这对于某些难以测试和调试的环境,例如航空,逻辑验证显得至关重要。

此外还有软件测试步骤。软件测试中借助专门的工具来帮助测试人员找出程序中存在的错误。在一定程度上人力难以实现此工作,例如压力测试和自动测试。

3.1.3 嵌入式系统调试与仿真

由于嵌入式系统调试的独特性以及困难性,人们专门研制了一些方法和设备来进行调试。嵌入式常见的几种调试方法如下。

3.1.3.1 全系统仿真器

全系统仿真器(System Simulator)早期是通过指令集模拟器(Instruction Set Emulator)来实现的,即在一种体系结构的机器上通过使用软件模拟另外一种体系结构的指令集的技术。换言之,也就是使用软件来解释执行以往需要通过 CPU 来执行的机器码,从而达到模拟某种处理器的执行目的。而现代的全系统仿真器除了模拟 CPU 外,还模拟外设,因而达到模拟全系统的效果。有些书中又将仿真器称为虚拟机、模拟器等。

指令集模拟可分为同构模拟和异构模拟两种。同构模拟指的是在某种体系架构的处理器上，使用软件模拟出另外一个与自己相同的虚拟机。目前常见的 Microsoft Virtual PC 或 VmWare，都是在 x86 上模拟 x86 处理器的执行，属于同构模拟。异构模拟则是在一个体系结构的处理器上模拟另外一种处理器的执行。绝大多数指令集模拟机都属于异构模拟。如 Device Emulator 就是在 x86 处理器上模拟 ARM 处理器的执行。一些常见的全系统仿真器可以参考表 3-2 所示。

表 3-2　　常见的全系统仿真器

模拟器名称	模拟的目标平台	备注
Microsoft Virtual PC/Virtual Server	x86	
VMWare	x86	支持 Windows、Mac 和 Linux 等多平台
Bochs	x86	开源项目
Device Emulator	ARM	模拟 SMDK2410 开发板
SkyEye	ARM	国产
VirtualBox Advance	ARM	模拟任天堂 GBA 游戏机
ePSXe	MIPS	模拟 sony 公司 PS 游戏机
VirtualNES	MOS6502	模拟任天堂 FC 游戏机
GDB Sim	ARM、1750、SPARC、MIPS、PowerPC、Z80、…	GDB 自带的多种体系结构全系统仿真器，支持直接运行 ELF 文件

对于嵌入式系统调试而言，在宿主机（通常是 PC）运行全系统仿真器，然后将目标机的软件运行在全系统仿真器中，这样不需要额外的硬件，就可方便地在 PC 上运行嵌入式软件来完成调试工作。在这里仿真器代替了交叉开发中实际的目标机。从物理上看，宿主机和目标机同在一台机器上实现，俗称“一机两用”。所以现在再回过来头看前面介绍的交叉开发环境时，我们提到了目标机不一定是实际机，有可能是基于软件的模拟器，正是说的这一点。采用仿真器不仅节省了硬件开支，而且方便了调试。

Android 开发工具为我们提供了优秀的仿真器，称为 AVD（Android Vitural Device，Android 虚拟设备），俗称模拟器。AVD 不仅能模拟 IA32 指令（即 x86 指令），还能模拟 ARM 指令。图 3-5 是运行 Windows 环境下 AVD 的一个界面截图。

Android 模拟器称为 Goldfish，在 Android SDK 1.5 版以后的 Android 开发中，必须创建至少一个 AVD。每个 AVD 模拟了一套移动设备（目前主要是手机）来运行 Android 平台，这个平台至少要有自己的内核、系统图像和数据分区，还可以自己的 SD 卡和用户数据以及外观显示等。Android 的模拟器是基于 Qemu 开发的，Qemu 是一个有名的开源虚拟机项目。Android 模拟器所对应的源代码主要位于 external/qemu 目录下。

AVD 除了模拟 CPU、屏幕、键盘、音频输出等目标机的常用设备外，还要模拟出 GPS、触摸、重力加速度等等传感器设备，以期最接近实际机的效果。例如一个 x86 结构的 AVD 模拟的设备包括：Atom CPU、MMC、RTC、Keyboard、USB Gadget、Framebuffer、TTY driver、NAND Flash。当然 AVD 与实际机还是有不同之处，与实际机相比，AVD 的差距包括：

- 不支持呼叫和接听实际来电；但可以通过控制台模拟电话呼叫（呼入和呼出）。

图 3-5 AVD(Android 模拟器)运行界面

- 不支持 USB 连接。
- 不支持数码相机/视频捕捉。
- 不支持音频输入(捕捉),但是支持输出(重放)。
- 不支持扩展耳机。
- 不能确定电池电量水平和交流电充电状态。
- 不能确定 SD 卡的插入/弹出。
- 不支持蓝牙。

此外,AVD 不仅模拟宿主机与目标机之间的 USB 连接,还模拟目标机与宿主机的网络,在 AVD 将宿主机作为一个缺省网关和 NAT(地址转换器)来连接网络。也就是说只要宿主机可以上网,AVD 模拟的目标机就可以上网。

3.1.3.2 其他调试手段

除了全系统仿真器,嵌入式系统还有其他一些调试手段,这些手段虽然在 Android 中不用,为了对嵌入式开发有全面的了解,我们也介绍一下。

(1)驻留监控程序。

驻留监控软件(Resident Monitors)是一段运行在目标机上的程序,运行在宿主机的调试程序通过以太网口、并行端口、串行端口等通信端口与目标机驻留监控软件进行交互,由调试软件发布命令通知驻留监控软件控制程序的执行、读写存储器、读写寄存器、设置断点等。驻留监控程序就是一个简单的系统管理程序,一般驻留在内存闪存中,当机器启动后,它立即接管控制权,它控制着目标机外部消息的接收,内部消息的发送与程序的

运行，同时对目标机上的各种异常进行监控和管理。

(2)在线仿真器。

在线仿真器(In-Circuit Emulator，ICE)使用仿真头完全取代目标机上的 CPU，实现对嵌入式芯片的完全仿真，达到一种深入调试的效果。在线仿真器能够完全监控总线的行为，功能比较强。

在线仿真器在早期的嵌入式系统中使用比较多，但随着集成电路技术的发展，新一代嵌入式处理器的工作频率越来越高，封装越来越贴片化，在线仿真器要通过探头来取代处理器和目标系统的连接变得越来越困难。此外，这类仿真器为了能够全速仿真时钟速度高于 100MHz 的处理器，必须采用极为复杂的设计和工艺，因而价格比较昂贵。在线仿真器通常用在嵌入式的硬件开发中，在软件开发中较少使用，其价格高昂是其不可忽视的原因之一。

(3)JTAG 仿真器。

JTAG(Joint Test Action Group，联合测试行动小组)是一种国际标准测试协议，与 IEEE 1149.1 标准兼容，主要用于芯片内部测试。现在多数大规模芯片器件都支持 JTAG 协议，如 SoC、DSP、FPGA 器件等。它在芯片内部封装了专门的测试电路测试访问端口(Test Access Port，TAP)，通过专用的 JTAG 测试工具对内部节点进行测试。标准的 JTAG 接口是 4 线：TMS、TCK、TDI、TDO，分别为模式选择、时钟、数据输入和数据输出线。基于 JTAG 接口的边界扫描测试技术，是在芯片引脚和芯片内部逻辑之间增加串行链接的边界扫描测试单元，实现对芯片引脚状态的设定和读取，使芯片引脚状态具有可控性和可观测性。

JTAG 的原始功能有两种：一种是用于测试芯片的电气特性，检测芯片是否有问题；另一种用于调试，对各类芯片及其外围设备进行调试。一个含有 JTAG 调试接口模块的 CPU，只要时钟正常，就可以通过 JTAG 接口访问 CPU 的内部寄存器、挂在 CPU 总线上的设备以及内置模块的寄存器。因此，JTAG 仿真器也称为 JTAG 调试器。现在 JTAG 接口还实现了 ISP 在线编程功能，可完成对目标机在线交叉开发。还有不少 JTAG 还具有在线仿真器的功能，被称为 JTAG-ICE。

3.1.3.3　交叉调试

在对有操作系统支持的嵌入式应用进行调试时，更多采用的是交叉调试的方法。交叉调试与交叉编译的思路一样：程序的运行在目标机上，而调试的显示、监控、控制等交互工作都在宿主机上进行。

交叉调试一般只能采用在线模式，不能采用脱线模式。这时，从硬件上来说，宿主机通过 USB 线、网络、JTAG-ICE 等与目标机相连。从软件上说，目标机一般要运行一个调试服务器，在 GNU 工具链中称为调试代理(Stub)，而在宿主机上运行调试的前端，实际是调试的客户端。前端负责与用户交互，将用户的交互转换为向调试服务器的请求；而调试服务器接收前端来的命令，控制应用的运行，并将结果返回前端，前端最终将结果显示出来，如图 3-6 所示。

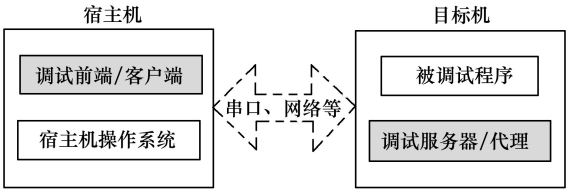


图 3-6 交叉调试的软件环境

例如,前端需要在某处设置一个断点,观察某变量的值。调试服务器接收前端的“设置断点”的请求,在被调程序的相应代码处插入一个中断,当应用运行到该处时产生中断,调试服务器接管运行,暂停应用的运行,然后将对应变量的值返回给前端,前端将该值显示出来。

很多开发工具支持交叉调试,如 GNU 调试器。Android 的 adb 调试器也支持交叉调试。adb(Android Debug Bridge)是 Android 提供一个通用的调试工具。它只能工作在线编程模式下,不能采用脱线方式。adb 也是基于客户端/服务器的设计模式,其工作原理是,宿主机平台上运行 adb 客户端,而目标机上 adb 服务器,如图 3-7 所示。adb 既可以以命令行的形式来运行,也可以集成在 Eclipse IDE(Integrated Development Environment,集成开发环境)来使用。前者命令行解释器充当 adb 的客户端,后者 Eclipse 是 adb 的客户端。

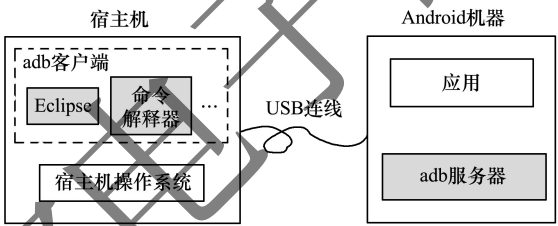


图 3-7 adb 的软件结构

借助 adb,远程(即目标机上的)应用的调试变得与本地一样简单。adb 不仅可以管理设备和模拟器的状态,还可以进行如下操作。

- 快速更新设备或手机模拟器中的代码,如应用或 Android 系统升级。
- 在设备上运行 shell 命令。
- 管理设备或手机模拟器上的预定端口。
- 在设备或手机模拟器上复制和粘贴文件。

关于 adb 的命令行方式的使用方式,以及调试以外的功能,我们将在后面的章节中予以介绍。本章先介绍 adb 作为插件集成到 Eclipse 等 IDE 中的使用。使用 Eclipse 对 Android 应用调试的截图如图 3-8 所示。通过 Android 调试,我们运用常见的调试手段,如设置断点、观察变量、单步执行、查看调试输出等,来调试目标机上的应用,而且整个调试过程如同调试本地应用一样,用户几乎感觉不到应用是运行在目标机而不是本机上。

关于 adb 命令使用,以及 Eclipse 如何调试,我们将在后续的章节结合例子进行详细介绍。

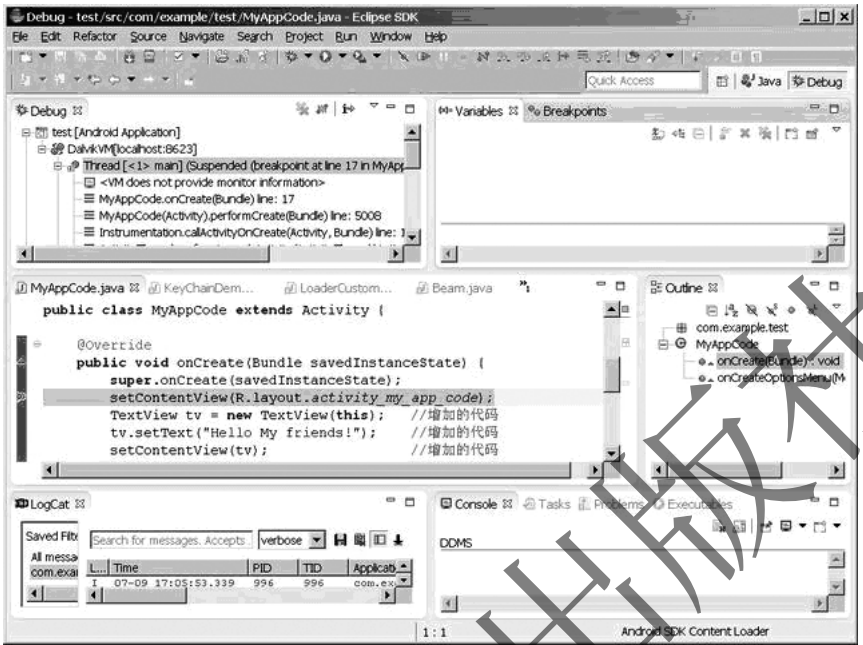


图 3-8　Eclipse 的 Android 应用调试截图

3.2　典型的开发工具链

在嵌入式软件开发的各个阶段都有相应的工具帮助用户来完成其工作，这些工具形成与开发流程对应的工具链(Toolchain)，又称工具集。嵌入式软件开发各阶段典型工具例子如表 3-3 所示。

表 3-3　　嵌入式软件开发典型工具

开发阶段	功能描述	典型工具例子
编辑	编辑程序的源代码	Vi、Emacs、Windows 记事本
编译、链接	将源程序编译、链接成可执行的二进制文件	gcc、Intel C Compiler
烧写	把可执行的二进制程序烧写到嵌入式系统内部的 ROM 或 Flash 中，以便系统开机即运行	JFlash、SJflash
调试	在程序运行的时候动态地跟踪程序的运行状态。查看程序的执行情况，以找出程序错误的原因	gdb、idb、Kernel Debugger
优化	分析程序性能，帮助开发者建立运行更快、更节能，占用空间更小程序	gprof、VTune
测试	帮助测试人员自动寻找程序中存在的错误，减少人力花费	CETK
验证	验证程序逻辑上的正确性和常见错误，特别是对某些难于测试和调试的环境	Application Verifier
模拟/仿真	模拟和仿真嵌入式软硬件运行环境，便于开发者开发和调试	VmWare、Device Emulator

目前工具集的类型各种各样，它们由不同的公司或组织提供，各有特色。例如表 3-3

所示，Intel C/C++ Compiler、VTune、idb 由英特尔公司提供，而 gcc、gdb、gprof 由自由软件组织 GNU 提供，CETK、Application Verifier、Device Emulator 则由微软公司提供。这些工具有些是免费的，如 GNU 工具集；有些是商业的，如微软工具集。基于的平台也不一样，例如 JFlash 是在 Linux 平台上运行，而微软工具集大部分基于 Windows 平台（包括桌面 Windows 操作系统和嵌入式操作系统 Windows CE/Mobile），还有的是跨平台的，如 GNU 工具集就可以在 Linux、Windows、Mac OS 等多种平台下运行。工具集的运行方式也不一样，目前主要有两种类型：一种是命令行方式；另一种是集成开发环境（Integrated Development Environment，IDE）。命令行方式工具集的各个应用工具是在命令行中以单独的命令来运行的。而集成开发环境将编辑、编译、链接、部署、调试等功能做在一个工具中，这样在一个应用中就可完成开发的全过程。大多数 GNU 工具是以命令行方式运行，而最典型的集成开发环境是微软的 Visual Studio，它是用于 Windows 系统的，而 Linux 下的 IDE 有 Anjuta 等。Android 的开发工具 Eclipse 是一种 IDE，它可在 Windows、Linux 等多种操作系统下运行。本教材的例子都采用 Windows 下运行的版本。

GNU 工具集具有跨平台、开放性、使用广、与其他工具兼容等特点，使得其成为嵌入式应用开发的首选工具。下面我们来介绍一下 GNU 工具集。

小提示：GNU GPL 和 LGPL

GNU 是当前最大的，也是最有名、最有影响力的自由软件组织。它最早由 Richard Stallman 于 1975 年创建，倡导 FSF（Free Software Foundation）的宗旨，目的是要脱离商业软件的种种枷锁。使用 GNU 软件必须遵守 GNU 软件许可。

GPL 是 GNU 通用公共许可证（GNU General Public License）的意思，GNU 软件许可之一。GPL 许可允许公众享有运行、复制软件的自由；发行传播软件的自由；获得软件源码的自由，以及改进软件并将自己作出的改进版本向社会发行传播的自由。GPL 还规定，只要这种修改文本的整体或者其某个部分来源于遵循 GPL 的程序，则该修改文本的整体就必须按照 GPL 流通，不仅该修改文本的源码必须向社会公开，而且对于这种修改文本的流通不准许附加修改者自己做出的限制。换句话说就是，使用 GPL 声明下的自由软件开发出来的软件也一定是自由软件。Linux 操作系统及其相关的大量软件是在 GPL 的推动下开发和发布的。

LGPL 是较宽松 GPL（Lesser GPL）的意思，它也是 GNU 软件许可之一，是 GPL 的变种。与 GPL 的最大不同是，用户可以私有使用 LGPL 授权的自由软件，开发出来的新软件可以是私有的而不要求是自由软件。用户在使用自由软件之前应该保证是在 LGPL 或其他 GPL 变种的授权下。LGPL 最早是用于一些 GNU 程序库（软件库），所以最早又称库 GPL（Library GPL）。使用 LGPL 典型的有 Mozilla 和 OpenOffice.org。

GNU 的开发工具都是免费的，遵循 GPL 协议，任何人都可以从网上获取。同样 GNU 也为嵌入式系统、x86 系统下的软件开发提供了完整的工具链。这些工具包含编译器、汇编器、链接器、调试工具等。这些工具它们可以以命令行形式单独运行，也可以集成到 Eclipse 等集成开发环境中。GNU 的工具链如表 3-4 所示。

表 3-4　　GNU 工具链

功能	组件	说明
编辑	Vi、Emacs、Ed 等	文本编辑器，用于编辑源代码
编译、链接	gcc	一组多种编程语言的编译器
调试	gdb	调试器
优化	gprof	分析程序性能，帮助开发者建立运行更快的程序
项目管理	make	自动管理软件编译
系统构建	autotools	构建项目所需的材料和文件

各组件的功能如下。

1. 编辑器

对源代码的编辑可以采用任何一个文本编辑工具。在 Linux 平台下，系统提供了一个完整的编辑器家族系列，按功能它们分为两大类：一类是行编辑器，如 Ed、Ex 等；另一类是全屏幕编辑器，如 Vi、Emacs、Gedit 等。行编辑器每次只能对一行进行操作；全屏幕编辑器可以对整个屏幕进行编辑，用户编辑的文件直接显示在屏幕上，从而克服了行编辑的那种不直观的操作方式，便于用户学习和使用，功能较前者强大。

要说明的是在集成开发环境中，编辑器是集成在工具中的，无需单独使用编辑器来编写源代码。

2. 编译、链接器

编译过程包括词法、语法和语义分析，中间代码的生成与优化，符号表的管理和出错处理。GNU 的编译器是 gcc，可以说，gcc 始终是 Linux 标配的编译器。

gcc 最初是 GNU C 语言编译器，目前该编译器的基本框架已支持 C、C++、Object-C、FORTRAN、Java 和 ADA。在某种程度上说，gcc 是 GNU 编译器的集合。gcc 不仅可以完成源代码的编译，还能实现链接。用户可选择相应的命令参数来编译，链接或直接生成可执行文件。

3. 调试器

调试器可以方便程序员调试程序，但不是代码执行的必要工具。在编程过程中，调试所消耗的时间远远大于编写代码的时间。因此一个功能强大、使用方便的调试器是很有必要的。

GNU 提供的调试器是 gdb(GNU Debugger 的简称)，它也是开放源代码的。gdb 是基于命令行的调试器，所有调试命令都是通过控制台的命令来完成的。

4. 构建管理器

类似于 Windows 系统中 Visual C++里的工程，GNU 提供了一种构建管理器 make，它是一种控制编译或者重复编译软件的工具。另外，它还能自动管理软件编译的内容、方式和时机，使程序员能把精力集中在代码的编写而不是源代码的编译次序的组织上。

make 会按用户定义的配置文件 makefile 调用 gcc 等命令对源代码进行编译、链接生成目标可执行文件。

5. Makefile(配置文件)自动生成工具

配置文件 makefile 可以帮助 make 完成目标文件生成任务，但是编写 makefile 并不

是一件轻松的事，尤其是对一个较大的项目来说。对 GNU 提供一系列自动工具(autotools)来制作 makefile，此外这些工具还可以完成系统配置的收集，从而使用户方便地处理各种移植性问题。autotools 包括 aclocal、autoscan、autoconf、autoheader、automake、libtool 等系列工具。

综合上面的内容，我们可以总结出从源代码文件生成目标文件的几种方法，这一过程如图 3-9 所示。

- 方法一：使用 gcc 命令分别编译、链接各源代码文件，生成可执行的目标文件。
- 方法二：用户编写 makefile 等配置文件，然后用 make 一步生成可执行的目标文件。
- 方法三：使用系统构建自动工具 autotools 制作 makefile 等配置文件，然后用 make 一步生成可执行的目标文件。

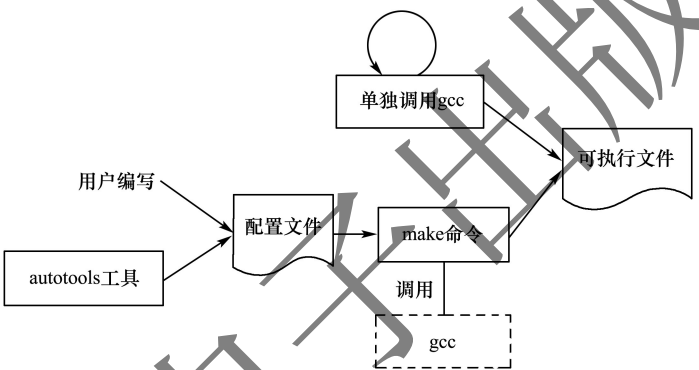


图 3-9 GNU 目标文件生成的几种方法

6. 优化工具

为了帮助用户优化程序，GNU 提供了性能分析器 gprof，它是 GNU binutils 的工具之一。

gprof 测量程序的性能，记录每个函数被调用的次数以及相应的执行时间，这样就能锁定程序执行时花费时间最多的部分，对程序的优化就可集中于对它们的优化。此外它还能产生程序运行时候的函数调用关系，包括调用次数，可以帮助程序员分析程序的运行流程。借助函数的调用关系，开发者不用费心地去一点点找出程序的运行流程，这样可以大大提高工作效率，而且此功能对于维护旧代码或者是分析开源(Open Source)项目来说也是很有益的，有了调用图，对程序的运行框架也就有了一个大体了解，知道了程序的“骨架”，分析它也就不再那么茫然，尤其是对自己不熟悉的代码和开源项目。

3.3 英特尔 Android 应用开发工具链的概述、安装与配置

3.3.1 英特尔 Android 应用开发工具链的概述

Android 为其应用开发提供了一套完整的工具链(或称工具集)。早期的 Android 都

是针对 ARM 架构硬件平台的，最近才开始支持 Atom 硬件平台。为支持 Atom 架构应用的开发，Intel 在 Android 工具链基础上增加了重要的插件、库等辅助模块。除此之外，为了更能发挥英特尔硬件的性能优势，英特尔补充了一些诸如编译器、调优等有特色的开发工具。这些都是原始的 Android 开发工具所不具备的，因此我们把加入 Intel 模块、支持 Atom 架构高软件性能的 Andoid 开发工具称为 Intel Android 应用工具链。

关于 Intel Android 应用工具链，我们准备分成两部分来介绍，本章主要介绍 Atom 平台上的 Android 应用开发的一般流程与方法，而后续的专门章节，我们介绍一下利用 Intel 专用工具实现性能优化、低功耗设计的方法。

Intel Android 开发工具链与 GNU 开发工具链，对应嵌入式交叉开发各阶段功能，如表 3-5 所示。

表 3-5　　GNU、英特尔工具链对比

交叉开发的阶段	GNU 工具链	Intel Android 开发工具链	备注
编辑	Vi、Emacs、Ed 等	Eclipse、Android SDK 等	Android 开发工具 ＋Intel 相关插件
编译、链接	gcc		
工程管理	make		
makefile 自动生成工具	autotools		
部署			
调试	gdb		
模拟/仿真		AVD	
优化	gprof	GPA、VTune	Intel 系列工具

英特尔工具除了表 3-5 中所列的与 GNU 工具的不同外，它还提供了一些专用的性能库来方便用户使用，如 IPP(Intel Integrated Performance Primitives，英特尔集成性能原语)、MKL(Intel Math Kernel Library，英特尔数学核心库)、TBB(Intel Threading Building Blocks，英特尔线程生成模块)等，这些库有的提供了专用服务，如 TBB 中基于 C＋＋模版的线程服务 API，有的则提供了能充分利用 x86 指令潜力实现性能优化的功能，如 IPP 中的快速傅里叶变换 FFT(Fast Fourier Transform)等。这些库目前还没有提供直接的 Java 接口，为了使用这些库我们必须使用一种专门的方式，对此我们会在后续的章节专门予以介绍。

从表 3-5 可以看出，Intel Android 开发工具链基本分为两部分，一部分是 Android 开发工具，这里 Intel 的工具，如 x86 模拟器、开发库等仅仅作为其插件的形式来出现；另一部分是独立的 Intel 工具。Android 开发工具涉及到一个应用开发的大部分步骤，如编辑—生成—打包—部署—调试等。而 Intel 系列工具主要涉及到优化工作，独立性强，基本上可以不依赖于 Eclipse 而单独运行。对于第一部分，由于 Intel 工具仅仅是作为 Android 开发工具的一种辅助组件的方式出现的，我们就简称其为 Android 开发工具了。

Android 开发工具是由 Java 开发包 JDK(Java SE Development Kit)、Android 开发包 Android SDK(software development kit)以及集成开发环境 Eclipse 等多个软件包构成的软件环境。Android 开发工具可以运行于 Linux、Windows 等多个操作系统平台下。本教材的例子都是在 Windows 环境开发的。

Android 开发工具既可以以命令行的形式出现，也可以以 IDE(Integrated Development Environment，集成开发环境)的形式出现。Android 命令行工具在 Android SDK 中，其一般的开发流程如图 3-10 所示。IDE 模式主要是由 Eclipse 来承担。Eclipse 是一个集成了编辑、编译、链接、部署、调试等功能的图形用户界面的工具。下面介绍的方法主要是基于集成开发环境的。

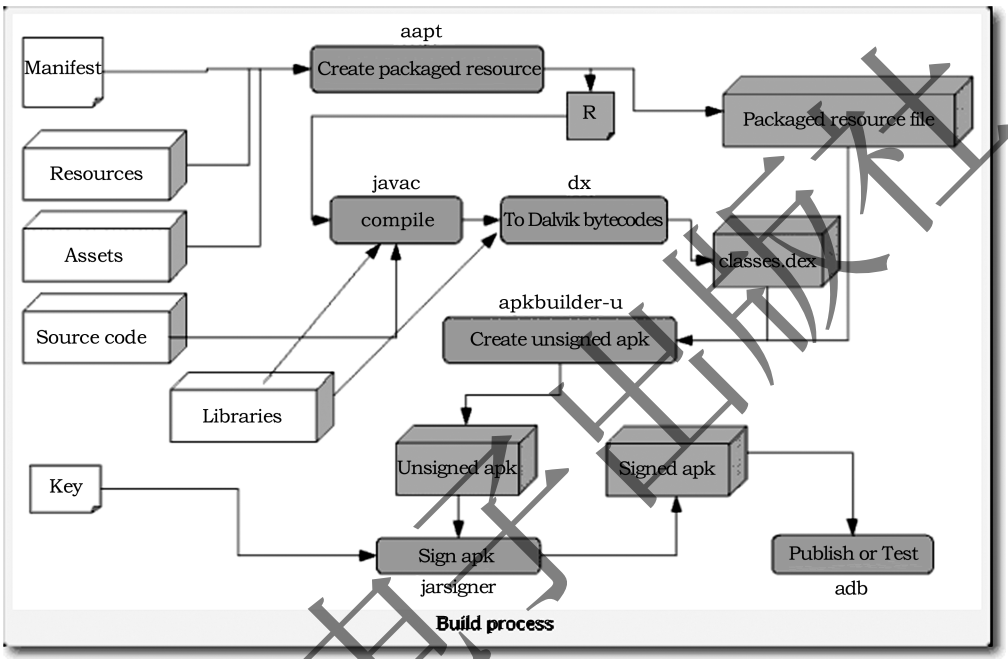


图 3-10 Android SDK 命令行开发步骤

Android SDK 的目录结构如下(可以在命令行运行 tree 命令得到)：

```
├─add-ons
│  └─addon-google_apis-google-16
├─docs
│  ├─about
│  ├─assets
│  ├─design
│  ├─develop
│  ├─distribute
│  ├─guide
│  ├─images
│  ├─intl
│  ├─live
│  ├─out
│  ├─reference
│  ├─resources
```

```
|   ├──samples
|   ├──sdk
|   ├──shareables
|   ├──tools
|   └──training
├──extras
|   ├──android
|   └──google
├──platform-tools
|   ├──api
|   ├──lib
|   └──renderscript
├──platforms
|   └──android-16
├──samples
|   └──android-16
├──sources
|   └──android-16
├──system-images
|   └──android-16
├──temp
└──tools
    ├──ant
    ├──apps
    ├──Jet
    ├──lib
    ├──proguard
    ├──support
    ├──systrace
    └──templates
```

主要文件夹的内容如下：

• add-ons：里面包含了 google 提供的 API 包，最主要的是 google MAP 的 API。

• docs：里面包含了文档，即帮助文档和说明文档。

• platforms：针对每个版本的 SDK 版本提供了和其对应的 API 包以及一些示例文件，其中包含了各个版本的 android。

• tools：包含了一些通用的工具文件。

• usb_driver：包含了 AMD64 和 X86 下的驱动文件。

其主要的文件及其作用介绍如下：

• android.jar

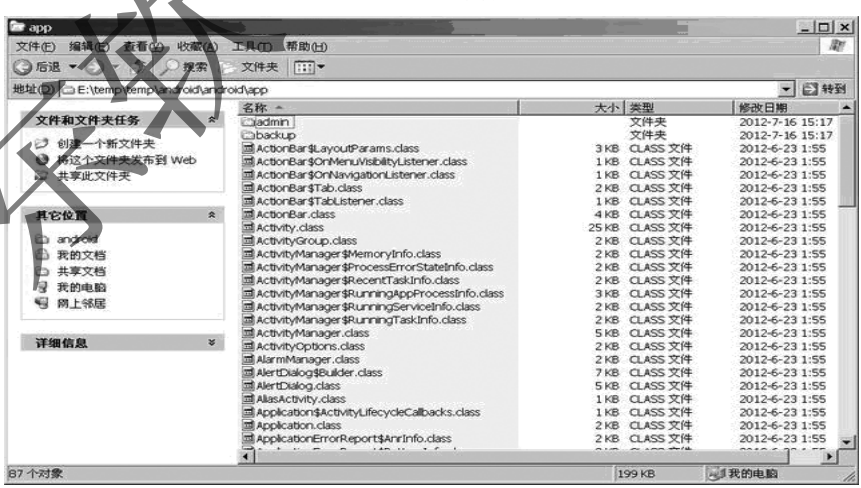
该文件位于%android-sdk%\platforms 目录下，每个 Android 版本中都有一个 an-

droid.jar。通过查看该文件可以了解内部的 API 包结构和组织方式。这里的%android-sdk%是 Android SDK 的安装目录，而版本 16 对应的目录为 android-16，例如笔者的某个 android.jar 所在的位置为：

```
C:\Documents and Settings>dir D:\Android\android-sdk\platforms\android-16
……
2012-07-08  20:02            18,325,478 android.jar
```

android.jar 是一个标准的压缩包，里面包含了编译后的压缩文件，包括全部的 API。使用 Winrar 可以查看其内部结构，其结构如图 3-11 所示。它对 API 包进行了详细的划分，分为了 app、content、database 等。

(a)根目录结构



(b)app 的内部结构

图 3-11　android.jar 文件的内容结构

• index. html

index. html 文件位于 docs 目录下。

```
C:\Documents and Settings> dir D:\Android\android-sdk\docs
……
2012-07-08  19:58              11,742 index.html
```

如果要深入了解各个文件包内的 API 及其具体用法，可以使用浏览器 index. html 文件，如图 3-12 所示。

图 3-12　Android 帮助文档首页

• ddms. bat

调试监视服务 ddms. bat 集成在 Dalvik(Android 平台的虚拟机)中，用于管理运行在模拟器或设备的进程，并协助调试工作。它可以去除一些进程，选择一个特定的程序来调试，生成跟踪数据，查看堆和线程数据，对模拟器或设备执行屏幕快照等操作。

```
C:\Documents and Settings>dir D:\Android\android-sdk\tools
……
2012-06-25  04:35               3,419 android.bat
```

```
2012-06-25  04:35             2,304 ddms.bat
2012-06-25  04:35           508,416 sqlite3.exe
```

• adb.exe

Android 调试桥(adb)是多种用途的工具,该工具可以帮助你管理设备或模拟器的状态。对此前面也有所介绍,该文件位于%android-sdk%\platform-tools 目录下,例如笔者的 adb.exe 位于 D:\Android\android-sdk\platform-tools 目录下:

```
C:\Documents and Settings>dir D:\Android\android-sdk\platform-tools
……
2012-07-08  19:44           846,848 aapt.exe
2012-07-08  19:44           191,488 adb.exe
2012-07-08  19:44           275,456 aidl.exe
2012-07-08  19:44             2,618 dx.bat
```

• aapt.exe

Android 资源打包工具(aapt.exe),使用它可以创建 apk 文件。apk 文件中包含了 Android 应用程序的二进制文件和资源文件。文件所在位置与 adb.exe 相同。

• aidl.exe

Android 接口描述语言(aidl.exe),用于生成进程间接口代码。文件所在位置与 adb.exe 相同。

• sqlite3.exe

SQLite3 数据库(sqlite3.exe)。Android 可以创建和使用 SQLite 数据文件,开发人员和用户可以方便地访问这些 SQLite 数据文件。其位置与 ddms.bat 文件相同。

• dx.bat

将 class 字节码重写为 Android 字节码(被存储在 dex 文件中)。文件所在位置与 adb.exe 相同。

• android.bat

该文件与 ddms.bat 同一目录下。该命令用于显示、创建 AVD。

```
D:\Android\android-sdk\tools>android list targets
Available Android targets:
----------
id: 1 or "android-15"
     Name: Android 4.0.3
     Type: Platform
     API level: 15
     Revision: 3
     Skins: HVGA, QVGA, WQVGA400, WQVGA432, WSVGA, WVGA800 (default), WVGA854, W
XGA720, WXGA800
     ABIs : x86
----------
id: 2 or "Google Inc.:Google APIs:15"
```

```
Name: Google APIs
Type: Add-On
Vendor: Google Inc.
Revision: 2
Description: Android + Google APIs
Based on Android 4.03 (API level 15)
Libraries:
  * com.google.android.media.effects (effects.jar)
      Collection of video effects
  * com.android.future.usb.accessory (usb.jar)
      API for USB Accessories
  * com.google.android.maps (maps.jar)
      API for Google Maps
Skins: WVGA854, WQVGA400, WSVGA, WXGA720, HVGA, WQVGA432,
WVGA800 (default), QVGA, WXGA800
ABIs : armeabi-v7a
```

以上命令显示了本机安装的目标机开发库有两个。

3.3.2　宿主机 Android 开发工具的安装

Android 开发工具运行于宿主机上，其安装的一般步骤如下。

3.3.2.1　下载相关软件

1. 下载 JDK

用户可以到 http://www.oracle.com/technetwork/java/javase/downloads/index.html 上去下载 JDK(Java Development Toolkit)，如图 3-13 所示。

图 3-13　下载 JDK

2. 下载 Android SDK

用户可以到 http://developer.android.com/sdk/index.html 下载 installer_r20-windows.exe，如图 3-14 所示。

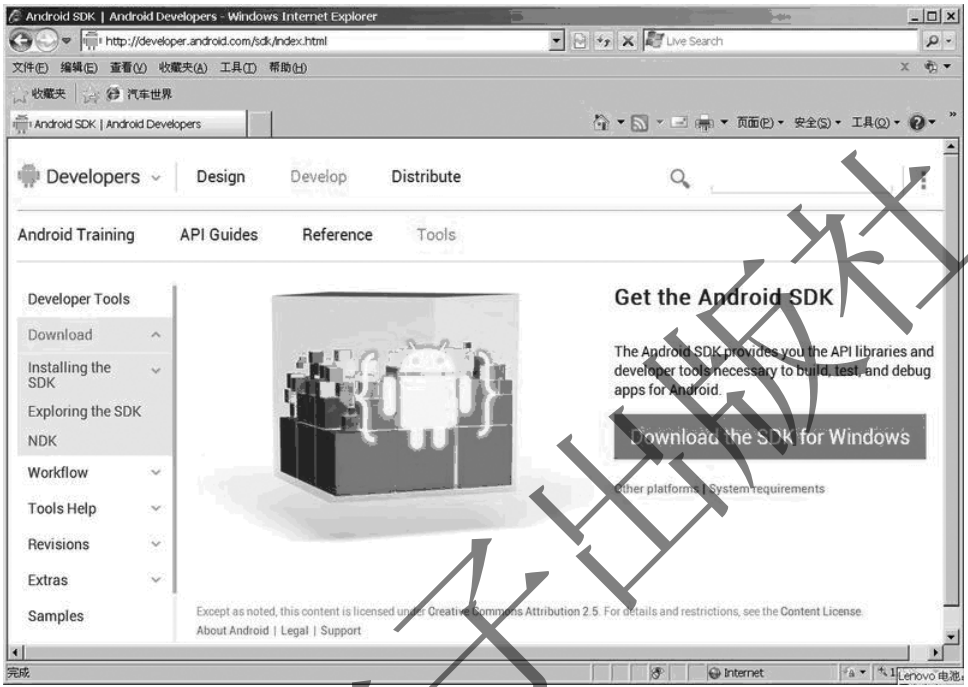


图 3-14 下载 Android SDK

3. 下载 Eclipse

到 http://www.eclipse.org/downloads/下载 eclipse-SDK-4.2-win32.zip，如图 3-15 所示。

图 3-15 下载 Eclipse