

## <<Android深度探索（卷1）>>

### 图书基本信息

书名：<<Android深度探索（卷1）>>

13位ISBN编号：9787115298027

10位ISBN编号：7115298025

出版时间：2013-1

出版单位：人民邮电出版社

作者：李宁

页数：637

字数：969000

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

## <<Android深度探索 (卷1)>>

### 内容概要

全书分为4篇，分别从搭建开发环境，Linux驱动和Android HAL的基础知识，开发Linux驱动的高级技术和分析典型的Linux驱动源代码4个方面介绍Android和Linux的底层开发。

本书使用的试验环境是Ubuntu

Linux12.04

LTS、Android模拟器和S3C6410开发板。

在第1篇详细介绍了如何搭建和使用这3个试验环境。

第2篇通过3个Linux驱动的完整案例(统计单词个数驱动、LED驱动和蜂鸣器驱动)从不同角度来讨论如何开发一个完整的Linux驱动。

并且通过完整的案例介绍了如何编写Android

HAL，以及如何与Linux驱动交互。

第3篇则介绍了开发Linux驱动所需要的高级技术，这些技术包括并发控制、阻塞和非阻塞I/O、异步编程、Linux中断和底半部、时间管理、内存管理和I/O访问。

最后一部分分析了一些典型Linux驱动的源代码(RTC驱动、LCD驱动、音频驱动、块设备驱动、网络设备驱动和USB驱动)。

《Android深度探索(卷1)：HAL与驱动开发》注重理论和实践相结合。

在介绍了大量的基础知识的同时，为每一个知识点提供了完整的案例，使读者可以通过实际的代码更好地理解Linux驱动和Android底层技术。

为了使读者更好地实践本书提供的实例代码，在随书光盘中除了提供源代码文件外，还提供了一个VMWare

Ubuntu Linux12.04 LTS的虚拟环境。

读者可以在Windows、Linux和Mac OS

X上，通过VMWare打开该虚拟机文件来学习和测试本书的例子(虚拟环境中也带了一套本书提供的例子代码)。

《Android深度探索(卷1)：HAL与驱动开发》适合底层开发的程序员和编程爱好者使用，也适合作为相关培训学校的Android底层开发培训教材。

## <<Android深度探索（卷1）>>

### 作者简介

拥有超过10年的软件开发经验，曾任某知名企业项目经理，对Android有深入的研究，是国内第一批Android实践者。

## <<Android深度探索（卷1）>>

### 书籍目录

#### 第一篇 Android驱动开发前的准备

##### 第1章 Android系统移植与驱动开发概述

- 1.1 Android系统架构
- 1.2 Android系统移植的主要工作
- 1.3 查看Linux内核版本
- 1.4 Linux内核版本号的定义规则
- 1.5 如何学习Linux驱动开发
- 1.6 Linux设备驱动
  - 1.6.1 设备驱动的发展和作用
  - 1.6.2 设备的分类及特点
- 1.7 见识一下什么叫Linux驱动：LED
- 1.8 小结

##### 第2章 搭建Android开发环境

- 2.1 Android底层开发需要哪些工具
- 2.2 安装JDK
- 2.3 搭建Android应用程序开发环境
  - 2.3.1 安装Android SDK
  - 2.3.2 安装Eclipse
  - 2.3.3 安装ADT
  - 2.3.4 配置ADT
  - 2.3.5 建立AVD
- 2.4 安装Android NDK开发环境
  - 2.4.1 下载Android NDK
  - 2.4.2 安装CDT
  - 2.4.3 命令行方式编译Android NDK程序
  - 2.4.4 导入Android NDK的例子
  - 2.4.5 配置Android NDK的集成开发环境
- 2.5 安装交叉编译环境
- 2.6 小结

##### 第3章 Git使用入门

- 3.1 安装Git
- 3.2 查看Git文档
- 3.3 源代码的提交与获取
  - 3.3.1 创建版本库：git init
  - 3.3.2 将文件提交到本地版本库：git commit
  - 3.3.3 创建本地分支：git branch
  - 3.3.4 切换本地分支：git checkout
  - 3.3.5 在GitHub上创建开源项目
  - 3.3.6 上传源代码到GitHub：git push
  - 3.3.7 从GitHub下载源代码：git clone
- 3.4 小结

##### 第4章 源代码的下载和编译

- 4.1 下载、编译和测试Android源代码
  - 4.1.1 配置Android源代码下载环境
  - 4.1.2 Android源代码目录结构解析

## <<Android深度探索 (卷1)>>

- 4.1.3 下载Android源代码中的一部分
- 4.1.4 编译Android 源代码
- 4.1.5 out目录结构分析
- 4.1.6 将自己的APK作为Android内置程序发布
- 4.1.7 用模拟器测试system.img文件
- 4.2 下载和编译Linux内核源代码
  - 4.2.1 下载Linux内核源代码
  - 4.2.2 Linux内核源代码的目录结构
  - 4.2.3 安装Android内核的编译环境
  - 4.2.4 配置和编译Linux内核
- 4.3 小结
- 第5章 搭建S3C6410开发板的测试环境
  - 5.1 S3C6410开发板简介
  - 5.2 安装串口调试工具：minicom
  - 5.3 烧写Android系统
  - 5.4 配置有线网络
  - 5.5 小结
- 第二篇 Android底层开发入门
- 第6章 第一个Linux驱动程序：统计单词个数
  - 6.1 Linux驱动到底是个什么东西
  - 6.2 编写Linux驱动程序的步骤
  - 6.3 第一个Linux驱动：统计单词个数
    - 6.3.1 编写Linux驱动程序前的准备工作
    - 6.3.2 编写Linux驱动程序的骨架(初始化和退出驱动)
    - 6.3.3 指定与驱动相关的信息
    - 6.3.4 注册和注销设备文件
    - 6.3.5 指定回调函数
    - 6.3.6 实现统计单词数的算法
    - 6.3.7 编译、安装、卸载Linux驱动程序
  - 6.4 使用多种方式测试Linux驱动
    - 6.4.1 使用Ubuntu Linux测试Linux驱动
    - 6.4.2 在Android模拟器上通过原生(Native)C程序测试Linux驱动
    - 6.4.3 使用Android NDK测试Linux驱动
    - 6.4.4 使用Java代码直接操作设备文件来测试Linux驱动
    - 6.4.5 使用S3C6410开发板测试Linux驱动
    - 6.4.6 将驱动编译进Linux内核进行测试
  - 6.5 使用Eclipse开发和测试Linux驱动程序
    - 6.5.1 在Eclipse中开发Linux驱动程序
    - 6.5.2 在Eclipse中测试Linux驱动
  - 6.6 小结
- 第7章 LED将为我闪烁：控制发光二级管
  - 7.1 LED驱动的实现原理
  - 7.2 编写LED驱动
    - 7.2.1 体验LED驱动的奇妙
    - 7.2.2 创建LED驱动的设备文件
    - 7.2.3 卸载LED驱动的设备文件
    - 7.2.4 设置寄存器与初始化LED驱动

## <<Android深度探索 (卷1)>>

- 7.2.5 控制LED
- 7.2.6 LED驱动模块参数
- 7.2.7 LED驱动的完整代码
- 7.3 测试LED驱动
  - 7.3.1 编写测试I/O控制命令的通用程序
  - 7.3.2 使用NDK测试LED驱动
  - 7.3.3 使用Java测试LED驱动
- 7.4 LED驱动的移植
- 7.5 小结
- 第8章 让开发板发出声音：蜂鸣器驱动
  - 8.1 Linux驱动的代码重用
    - 8.1.1 编译是由多个文件组成的Linux驱动
    - 8.1.2 Linux驱动模块的依赖(导出符号)
  - 8.2 强行卸载Linux驱动
  - 8.3 蜂鸣器(PWM)驱动
    - 8.3.1 蜂鸣器驱动的原理
    - 8.3.2 实现蜂鸣器驱动
    - 8.3.3 测试蜂鸣器驱动
  - 8.4 小结
- 第9章 硬件抽象层：HAL
  - 9.1 为什么要在Android中加入HAL
  - 9.2 Android HAL架构
  - 9.3 为LED驱动增加HAL
    - 9.3.1 编写一款支持HAL的Linux驱动程序步骤
    - 9.3.2 颠覆Linux驱动的设计理念：精简LED驱动
    - 9.3.3 测试读写寄存器操作
    - 9.3.4 编写调用LED驱动的HAL模块
    - 9.3.5 编写调用HAL模块的Service
    - 9.3.6 HAL模块的存放路径和命名规则
    - 9.3.7 编写调用Service的Java库
    - 9.3.8 测试LED驱动
  - 9.4 小结
- 第10章 嵌入式Linux的调试技术
  - 10.1 打印内核调试信息：printk
  - 10.2 防止printk函数降低Linux 驱动性能
  - 10.3 通过虚拟文件系统(/proc)进行数据交互
  - 10.4 调试工具
    - 10.4.1 用gdb调试用户空间程序
    - 10.4.2 用gdbserver远程调试用户空间程序
    - 10.4.3 用kgdb远程调试内核程序
  - 10.5 小结
- 第三篇 Linux驱动开发高级技术
- 第11章 Linux驱动程序中的并发控制
  - 11.1 并发和竞态
  - 11.2 原子操作
    - 11.2.1 整型原子操作
    - 11.2.2 64位整型原子操作

## <<Android深度探索 (卷1)>>

- 11.2.3 位原子操作
- 11.2.4 用原子操作阻止设备文件被多个进程打开
- 11.3 自旋锁(Spin Lock)
  - 11.3.1 自旋锁的使用方法
  - 11.3.2 使用自旋锁保护临界区
  - 11.3.3 读写自旋锁
  - 11.3.4 使用读写自旋锁保护临界区
  - 11.3.5 顺序锁(seqlock)
  - 11.3.6 使用顺序锁写入正在读取的共享资源
- 11.4 读—复制—更新(RCU)机制
  - 11.4.1 RCU的原理
  - 11.4.2 RCU API
  - 11.4.3 RCU的应用
- 11.5 信号量(Semaphore)
  - 11.5.1 信号量的使用
  - 11.5.2 信号量用于同步
  - 11.5.3 读写信号量
  - 11.5.4 使用读写信号量保护临界区
- 11.6 互斥体(Mutex)
- 11.7 完成量(Completion)
- 11.8 小结
- 第12章 Linux驱动程序中的阻塞和非阻塞I/O
  - 12.1 等待队列
    - 12.1.1 等待队列原理
    - 12.1.2 等待队列的API
    - 12.1.3 等待队列的使用方法
    - 12.1.4 支持休眠和唤醒的Linux驱动
  - 12.2 轮询操作
    - 12.2.1 用户空间的select函数
    - 12.2.2 内核空间的poll函数
    - 12.2.3 以非阻塞的方式访问Linux驱动
  - 12.3 小结
- 第13章 Linux驱动程序中的异步编程
  - 13.1 信号与异步通知
    - 13.1.1 Linux信号
    - 13.1.2 接收Linux信号
    - 13.1.3 发送信号
  - 13.2 异步I/O(AIO)
    - 13.2.1 异步操作的API
    - 13.2.2 异步读写本地文件
    - 13.2.3 Linux驱动中的异步函数(aio\_read和aio\_write)
    - 13.2.4 接收信号时异步读取数据
    - 13.2.5 AIO中的回调函数
  - 13.3 小结
- 第14章 Linux中断和底半部
  - 14.1 什么是中断
  - 14.2 中断处理程序

## <<Android深度探索 (卷1)>>

14.3 Linux 中断处理的核心：顶半部和底半部

14.4 获取Linux 系统的中断统计信息

14.5 Linux 中断编程

14.5.1 注册中断处理程序

14.5.2 注销中断处理程序

14.5.3 编写中断处理函数

14.5.4 共享中断处理程序

14.5.5 禁止和激活中断

14.5.6 禁止和激活中断线

14.5.7 获取中断系统的状态

14.5.8 与中断编程相关的函数和宏

14.6 实例：S3C6410实时钟中断

14.7 中断中下文

14.8 中断的实现原理

14.9 底半部

14.9.1 为什么要使用底半部

14.9.2 实现底半部的机制

14.9.3 软中断

14.9.4 Tasklet

14.9.5 实例：Tasklet演示

14.9.6 软中断处理线程(ksoftirqd)

14.9.7 工作队列(work queue)

14.9.8 与工作队列相关的API

14.9.9 实例：工作队列演示

14.10 小结

第15章 时间管理

15.1 Linux内核中的时间概念

15.1.1 时钟频率

15.1.2 提高时钟频率的优点和缺点

15.2 节拍总数(jiffies)

15.2.1 访问jiffies

15.2.2 jiffies、时间和时钟频率之间的转换

15.2.3 jiffies的回绕

15.2.4 用户空间和时钟频率

15.3 实时时钟和定时器

15.4 时钟中断处理程序的实现

15.5 读写本地时间

15.6 内核定时器

15.6.1 如何使用内核定时器

15.6.2 实例：秒表定时器

15.7 内核延迟

15.7.1 忙等待

15.7.2 短延迟

15.7.3 休眠延迟(schedule\_timeout)

15.8 小结

第16章 内存管理与I/O访问

16.1 内存管理模式



## <<Android深度探索 (卷1)>>

- 16.1.1 内存的基本单位：页(Page)
- 16.1.2 页的逻辑划分：区(zone)
- 16.1.3 获取页
- 16.1.4 释放页
- 16.2 分配连续的内存空间(Kmalloc)
- 16.2.1 gfp\_mask标志
- 16.2.2 释放内存(kfree)
- 16.3 分配不连续的内存空间(vmalloc)
- 16.4 全局缓存(slab)
- 16.4.1 Slab层的实现原理
- 16.4.2 Slab分配器
- 16.4.3 示例：从Slab高速缓存中分配和释放对象
- 16.5 Linux内存池
- 16.5.1 内存池的实现原理
- 16.5.2 示例：从内存池获取对象
- 16.6 虚拟地址与物理地址之间的转换
- 16.7 设备I/O端口与I/O内存
- 16.7.1 读写I/O端口
- 16.7.2 读写I/O内存
- 16.7.3 将I/O端口映射为I/O内存
- 16.7.4 申请和释放设备I/O端口和I/O内存
- 16.7.5 使用设备I/O端口和I/O内存的一般步骤
- 16.8 内核空间与用户空间共享数据
- 16.8.1 内存映射与VMA
- 16.8.2 示例：用户程序读取内核空间数据
- 16.9 I/O内存静态映射
- 16.10 小结

### 第四篇 Linux设备驱动与Android底层开发

#### 第17章 RTC驱动

- 17.1 实时时钟(RTC)结构与移植内容
- 17.1.1 RTC系统的结构
- 17.1.2 RTC驱动主要的移植工作
- 17.2 RTC系统中的Android部分
- 17.2.1 警报管理：AlarmManager
- 17.2.2 警报服务：AlarmManagerService
- 17.2.3 直接与Alarm驱动交互的JNI代码
- 17.3 Alarm驱动的分析与移植
- 17.3.1 Alarm驱动简介
- 17.3.2 Alarm驱动中的关键数据结构
- 17.3.3 Alarm驱动的应用层接口(alarm\_dev.c)代码分析
- 17.3.4 Alarm驱动的通用文件(alarm.c)代码分析
- 17.4 RTC驱动的分析与移植
- 17.4.1 实时时钟(RTC)的特性
- 17.4.2 RTC的结构
- 17.4.3 RTC芯片的寄存器
- 17.4.4 RTC驱动的用户空间接口
- 17.4.5 RTC系统组件之间的调用关系

## <<Android深度探索 (卷1)>>

- 17.4.6 设备文件(/dev/rtc0)的I/O命令
- 17.4.7 sysfs虚拟文件处理函数
- 17.4.8 proc虚拟文件处理函数
- 17.5 小结
- 第18章 LCD驱动
- 18.1 LCD简介
  - 18.1.1 液晶的工作原理
  - 18.1.2 LCD的种类
  - 18.1.3 LCD的技术参数
  - 18.1.4 LCD时序图
- 18.2 LCD驱动结构分析和移植要点
- 18.3 帧缓冲(FrameBuffer)驱动设计与实现
  - 18.3.1 FrameBuffer设备
  - 18.3.2 示例：通过dd命令与FrameBuffer设备文件交互
  - 18.3.3 示例：编写访问FrameBuffer设备文件的程序
  - 18.3.4 FrameBuffer驱动的架构
  - 18.3.5 FrameBuffer驱动主要的数据结构
  - 18.3.6 如何在Linux内核中查找指定的内容
  - 18.3.7 FrameBuffer驱动设备事件的处理(fbmemb.c)
  - 18.3.8 FrameBuffer驱动源代码分析与移植
- 18.4 FrameBuffer驱动的HAL层分析
  - 18.4.1 Gralloc库
  - 18.4.2 初始化HAL Gralloc的核心结构体
  - 18.4.3 获取Gralloc HAL模块
  - 18.4.4 与FrameBuffer设备文件交互
- 18.5 调用Gralloc HAL库
- 18.6 小结
- 第19章 音频驱动
- 19.1 音频驱动基础
  - 19.1.1 数字音频简介
  - 19.1.2 ALSA架构简介
  - 19.1.3 ALSA设备文件
  - 19.1.4 数字采样与数字录音
  - 19.1.5 混音器
  - 19.1.6 音频驱动的目录结构
  - 19.1.7 音频设备硬件接口
  - 19.1.8 ALSA架构支持的声卡芯片
- 19.2 AC97芯片的寄存器
  - 19.2.1 控制寄存器
  - 19.2.2 状态寄存器
  - 19.2.3 编解码器命令寄存器
  - 19.2.4 编解码器状态寄存器
  - 19.2.5 PCM输出/输入通道FIFO数据寄存器
  - 19.2.6 MIC输入通道FIFO地址寄存器
  - 19.2.7 PCM输出/输入通道FIFO数据寄存器
  - 19.2.8 MIC输入通道FIFO数据寄存器
- 19.3 创建声卡

## <<Android深度探索 (卷1)>>

- 19.3.1 声卡的顶层数据结构
- 19.3.2 创建声卡的步骤
- 19.3.3 示例：基于ARM的AC97音频驱动
- 19.4 音频逻辑设备
  - 19.4.1 创建PCM设备
  - 19.4.2 创建录音和播放设备文件节点
  - 19.4.3 创建Control设备数据结构
  - 19.4.4 创建Control设备
  - 19.4.5 注册与打开音频字符设备
- 19.5 嵌入式设备中的ALSA(ASoC)
  - 19.5.1 什么是ASoC
  - 19.5.2 ASoC的硬件架构
  - 19.5.3 ASoC的软件架构
  - 19.5.4 如何确定S3C开发板使用了哪个音频驱动
  - 19.5.5 ASoC架构中的Machine
  - 19.5.6 ASoC架构中的Codec
  - 19.5.7 ASoC架构中的Platform
- 19.6 音频驱动的HAL分析
  - 19.6.1 实现HAL Library
  - 19.6.2 调用HAL Library
- 19.7 小结
- 第20章 Linux块设备驱动
  - 20.1 块设备简介
  - 20.2 块设备的体系架构
  - 20.3 块设备的数据结构与相关操作
    - 20.3.1 磁盘设备(gendisk结构体)
    - 20.3.2 block\_device\_operations结构体
    - 20.3.3 I/O请求(request结构体)
    - 20.3.4 请求队列(request\_queue结构体)
    - 20.3.5 块I/O(bio结构体)
  - 20.4 块设备的加载和卸载
  - 20.5 块设备的打开和释放
  - 20.6 块设备的ioctl函数
  - 20.7 块设备驱动的I/O请求处理
    - 20.7.1 依赖请求队列
    - 20.7.2 不依赖请求队列
  - 20.8 实例1：依赖请求队列的RamDisk
  - 20.9 在嵌入式设备上测试块设备驱动
    - 20.9.1 编译、配置和安装Busybox
    - 20.9.2 测试块设备驱动
  - 20.10 实例2：不依赖请求队列的RamDisk
  - 20.11 扇区与磁盘碎片整理
  - 20.12 小结
- 第21章 网络设备驱动
  - 21.1 Linux网络设备驱动的结构
    - 21.1.1 网络协议接口层
    - 21.1.2 网络设备接口层

## <<Android深度探索（卷1）>>

- 21.1.3 设备驱动功能层
- 21.1.4 网络设备与媒介层
- 21.2 网络设备驱动设计与实现
  - 21.2.1 网络设备的注册与注销
  - 21.2.2 网络设备的初始化
  - 21.2.3 网络设备的打开与释放
  - 21.2.4 发送数据
  - 21.2.5 接收数据
  - 21.2.6 网络连接状态
- 21.3 示例：DM9000网卡设备驱动
  - 21.3.1 如何确定S3C6410开发板使用的网络设备
  - 21.3.2 DM9000网卡硬件描述
  - 21.3.3 网络设备驱动的定义与安装
  - 21.3.4 初始化DM9000网卡设备驱动
  - 21.3.5 移出网络设备
  - 21.3.6 打开和停止DM9000网卡
  - 21.3.7 发送数据
  - 21.3.8 接收数据
  - 21.3.9 设置广播地址
- 21.4 小结
- 第22章 USB驱动
  - 22.1 USB设备简介
  - 22.2 USB驱动与USB核心之间的交互
    - 22.2.1 端点(Endpoint)
    - 22.2.2 接口(Interfaces)
    - 22.2.3 配置(Config)
  - 22.3 USB设备的核心数据结构
    - 22.3.1 USB设备：usb\_device结构体
    - 22.3.2 USB驱动：usb\_driver结构体
    - 22.3.3 识别USB设备：usb\_device\_id结构体
    - 22.3.4 USB端点：usb\_host\_endpoint结构体
    - 22.3.5 USB接口：usb\_interface结构体
    - 22.3.6 USB配置：usb\_host\_config结构体
  - 22.4 描述符数据结构
    - 22.4.1 设备描述符
    - 22.4.2 配置描述符
    - 22.4.3 接口描述符
    - 22.4.4 端点描述符
    - 22.4.5 字符串描述符
    - 22.4.6 查看描述符信息
  - 22.5 USB和sysfs
  - 22.6 URB(USB请求块)
    - 22.6.1 URB结构体
    - 22.6.2 URB的处理流程
    - 22.6.3 简单的批量与控制URB
  - 22.7 USB驱动程序的结构
  - 22.8 鼠标驱动分析

## 22.9 小结

## &lt;&lt;Android深度探索（卷1）&gt;&gt;

## 章节摘录

版权页： 插图： 那么液晶为什么会显示字符、图像呢？

原来这种液态光电显示材料，利用液晶的电光效应把电信号转换成字符、图像等可见信号。

液晶在正常隋况下，其分子排列很有秩序，显得清澈透明，一旦加上直流电场后，分子的排列被打乱，一部分液晶变得不透明，颜色加深，因而能显示字符和图像。

液晶的电光效应是指它的干涉、散射、衍射、旋光、吸收等受电场调制的光学现象。

一些有机化合物和高分子聚合物，在一定温度或浓度的溶液中，既具有液体的流动性，又具有晶体的各向异性，这就是液晶。

液晶光电效应受温度条件控制的液晶称为热致液晶；溶致液晶则受控于浓度条件。

显示用液晶一般是低分子热致液晶。

根据液晶会变色的特点，人们利用它来指示温度、报警毒气等。

例如，液晶能随着温度的变化，使颜色从红变绿、蓝。

这样可以指示出某个实验中的温度。

液晶遇上氯化氢、氢氰酸之类的有毒气体，也会变色。

在化工厂，人们把液晶片挂在墙上，一旦有微量毒气逸出，液晶变色了，就提醒人们赶紧去检查、补漏。

液晶的工作原理如图18—1所示。

从背景灯射出的光线穿过液晶体后（可能只有部分光线穿过），再由滤光器处理色彩像素。

18.1.2 LCD的种类 LCD是LiquidCrystalDisplay的简称，目前被广泛应用在手机等低耗电的计算设备中。

手机的彩色屏幕因为LCD品质和研发技术不同而有所差异，其种类大致有STN、UFB、TFD、TFT和OLED几种。

一般来说能显示的颜色越多越能显示复杂的图像，画面的层次也更丰富。

STN是SuperTwistedNematic的缩写，是我们接触得最多的LCD了，因为我们过去使用的灰阶手机的屏幕都是STN的（即FSTN）。

和TFT相比STN型液晶属于被动矩阵式LCD器件，它的好处是功耗小，具有省电的最大优势。

彩色STN的显示原理是在传统单色STN液晶显示器上加一层彩色滤光片，并将单色显示矩阵中的每一像素分成三个子像素，分别通过彩色滤光片显示红、绿、蓝三原色，就可显示出彩色画面。

与TFT不同，STN属于被动矩阵式LCD，一般最高能显示65536种色彩，色泽不是特别好，亮度不高，所以一般在阳光强的地方，图像看起来比较费劲。

由于价格低廉，是众多中低端彩屏手机的选择，少量高端机型也采用STN。

撇开灰阶STN不提，现在STN主要有CSTN和DSTN之分。

CSTN即Color STN，一般采用传送式（transmissive）照明方式，传送式屏幕要使用外加光源照明，称为背光（backlight），照明光源要安装在LCD的背后。

传送式LCD在正常光线及暗光线下，显示效果都很好；但在户外，尤其在日光下，很难辨清显示内容；而背光需要电源产生照明光线，要消耗电功率。

DSTN（double—layer super—twisted nematic）即双层STN，过去主要应用在一些笔记本电脑上。

也是一种无源显示技术，使用两个显示层，这种显示技术解决了传统STN显示器中的漂移问题，而且由于DSTN还采用了双扫描技术，因而显示效果较STN有大幅度的提高。

由于DSTN分上下两屏同时扫描，所以在使用中有可能在显示屏中央出现一条亮线。

## <<Android深度探索（卷1）>>

### 媒体关注与评论

一直想找一个在三种平台上开发的书（Ubuntu Linux、Android模拟器和开发板），本书正好符合我的需求，翻看了一下，作者对Android底层和驱动层的介绍十分详细，配图和代码片段都是选取的很典型很有特点的，说明作者具有很强的理论基础和实战能力，是本不错的驱动开发工具书，总体来说，物有所值~~赞一个！

## <<Android深度探索（卷1）>>

### 编辑推荐

3大真实的实验环境：Ubuntu Linux12.04 LTS、Android模拟器和S3C6410开发板。

必知必会的驱动开发技术，包括并发控制、阻塞和非阻塞I/O、异步编程、Linux中断和底半部、时间管理、内存管理和I/O访问等。

6大核心Linux驱动代码分析与实战：RTC驱动、LCD驱动、音频驱动、块设备驱动、网络设备驱动和USB驱动



版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:<http://www.tushu007.com>