



## 精英版 用户使用手册

版本号 : V2.1

日期 : 2015-12-14

[www.topeetboard.com](http://www.topeetboard.com)

迅为官网 : [www.topeetboard.com](http://www.topeetboard.com)

4412 开发社区 : [bbs.topeetboard.com](http://bbs.topeetboard.com)

销售热线 : 010-58957586

传 真 : 010-58957586

售后支持 : 010-58957738

开发板商城 : <http://arm-board.taobao.com>

售后 QQ 群 : 310166965

版本	19
前言	23
必须注意的问题	25
名词解释	26
— iTOP-4412 开发板介绍	27
1.1 开发板平台简介	27
1.1.1 核心板	27
1.1.1.1 POP 封装	27
1.1.1.2 SCP 1G 封装	28
1.1.1.3 SCP 2G 封装	28
1.1.2 底板	29
1.1.2.1 电源以及接口	29
1.1.2.2 拨码开关以及对应功能	31
1.1.2.3 显卡资源以及接口	31
1.1.2.4 蜂鸣器 BEEP	33
1.1.2.5 灯 LEDS	34
1.1.2.6 按键 Keys	35
1.1.2.7 摄像头 CAMERA+AVIN 扩展口	35
1.1.2.8 JTAG 扩展口	36
1.1.2.9 GPIO+CAN+485 扩展口	37
1.1.2.10 模数 A/D 转换	37
1.1.2.11 声卡资源以及接口	38
1.1.2.12 串口接口	39
1.1.2.13 MIPI 接口	40

1.1.2.14 实时时钟 RTC	41
1.1.2.15 以太网	41
1.1.2.16 WIFI 扩展口	42
1.1.2.17 串口+矩阵键盘+GPS 扩展口	43
1.1.2.18 OTG 接口	44
1.1.2.19 USB 接口	44
1.1.2.20 TF 卡接口	45
1.1.2.21 复位按键	46
1.1.3 精英版使用串口修改方法	47
1.2 光盘资料	48
1.3 网盘资料	49
Exynos4412 三星原厂资料	49
iTOP-4412 开发板搭建编译环境所需要的工具包以及补丁包	50
iTOP-4412 开发板源码（其它）	50
iTOP-4412 开发板相关文档（补充）	50
iTOP-4412 开发板视频教程及其相关	50
嵌入式学习推荐书籍及软件（第三方）	51
1.4 网盘压缩包 MD5 值的使用	51
1.5 使用 github 获取开发源码	54
二 iTOP-4412 开发平台组装以及初体验	56
2.1 开发板的组装	56
2.1.1 控制台（Console）串口	56
2.1.2 屏幕的连接	57
2.1.2.1 电阻屏的连接（4.3 寸屏幕）	58
2.1.2.2 电容屏的连接（7 寸屏幕和 9.7 寸屏幕）	58

2.1.2.3 显示器 ( HDMI ) 的连接	60
2.2.3 电源的连接	60
2.2 启动方式设置 ( 拨码开关 )	61
2.3 uboot 模式和文件系统模式	62
2.3.1 uboot 模式	62
2.3.2 文件系统模式	63
2.4 iTOP-4412 开发平台初体验	65
2.4.1 系统基本功能	65
2.4.1.1 开机	65
2.4.1.2 音量调节	66
2.4.1.3 亮度调节	68
2.4.2 USB 和 OTG 功能	68
2.4.2.1 连接 U 盘	68
2.4.2.2 鼠标及键盘	69
2.4.2.3 将开发板当做平板与 PC 相连	70
2.4.3 网络设置和连接	71
2.4.3.1 WIFI 连接 ( 选配 )	71
2.4.3.2 有线网 ( RJ45 )	72
2.4.3.3 浏览网页	73
2.4.4 多媒体	74
2.4.4.1 电影音乐	74
2.4.4.2 摄像头 ( 选配 )	75
2.4.4.3 声卡的内外放设置	76
2.4.5 GPS 功能 ( 选配 )	77
2.4.6 游戏 3D 性能	78

2.4.7 蓝牙功能 (选配) .....	79
三 iTOP-4412 平台基础软件的安装和学习.....	80
3.1 超级终端的安装和使用.....	80
3.1.1 安装 USB 转串口驱动.....	80
3.1.2 超级终端的安装.....	86
3.1.3 超级终端的设置.....	89
3.1.4 超级终端的系统配置.....	92
3.1.5 超级终端保存日志.....	95
3.2 安装虚拟机以及 Ubuntu12.04.2 等软件.....	98
3.2.1 虚拟机 VMware-workstation 的安装.....	98
3.2.1.1 安装虚拟机.....	99
3.2.1.2 安装虚拟机常见错误.....	106
3.2.1.3 卸载后重装虚拟机需要注意的问题.....	107
3.2.1.4 虚拟机安装 Ubuntu 常见问题之 64 位虚拟化.....	109
3.2.2 虚拟机加载 Ubuntu12.04.2 镜像.....	110
3.2.3 虚拟机安装 Ubuntu12.04.2 初始系统.....	114
3.2.4 虚拟机 VMware-workstation8.0.3 联网以及基本设置.....	122
3.2.5 安装和使用 SSH 软件.....	127
3.3 Ubuntu 的基本操作.....	132
3.3.1 初识 Ubuntu12.04.2 以及 Ubuntu 命令行.....	133
3.3.2 Ubuntu 中启用 root 用户.....	136
3.3.3 Linux 常用 shell 命令.....	137
用户文件夹.....	137
显示命令.....	137
查看当前工作路径.....	138

切换目录.....	139
清屏.....	139
显示和配置网络属性.....	139
新建文件夹.....	140
删除命令.....	140
压缩和解压命令.....	141
拷贝命令.....	142
帮助命令.....	143
3.3.4 Linux 的重要命令 apt-get.....	143
3.3.4.1 查看数据源文件.....	144
3.3.4.2 修改数据源地址.....	144
3.3.4.3 “apt-get update” 命令.....	148
3.3.4.4 “apt-get install” 命令.....	148
3.3.5 安装和使用 SSH 软件.....	151
3.3.5.1 安装 SSH 软件.....	151
3.3.5.2 使用 SSH 软件传文件.....	154
3.3.5.3 SSH 控制台.....	157
3.3.6 虚拟机 Ubuntu 扩展硬盘空间.....	158
3.3.7 虚拟机无法识别 USB3.0 的解决方法.....	169
3.3.8 U 盘、TF 卡与虚拟机连接.....	170
3.4 Vim 编辑器.....	174
3.4.1 安装 Vim 编辑器.....	175
3.4.2 Vim 打开文件以及新建文件.....	175
3.4.3 三种模式的切换.....	177
3.4.4 Vim 编辑器常用基本命令.....	179

3.4.4.1 命令行模式.....	179
3.4.4.2 输入模式.....	180
3.4.4.3 底行模式.....	180
3.5 Source Insight 的安装和使用.....	181
3.5.1 Source Insight 的安装.....	181
3.5.2 使用 Source Insight 查看内核代码.....	183
3.6 安卓 ADB 功能介绍.....	192
3.6.1 安卓 ADB 驱动的安装.....	192
3.6.2 ADB 的基础知识.....	197
3.6.2.1 cmd.exe 程序.....	197
3.6.2.2 fastboot.exe 程序.....	200
3.6.2.3 adb.exe 程序.....	200
3.6.3 常用的 ADB 命令.....	200
3.6.4 ADB 驱动安装常见问题解决办法汇总.....	204
3.7 win8 下基础软件的安装和学习.....	206
3.7.1 超级终端的安装和使用.....	206
3.7.1.1 关闭 win8 的自动更新.....	206
3.7.1.2 安装 USB 转串口驱动 PI2303.....	208
3.7.1.3 超级终端的安装.....	213
3.7.1.4 其它库的安装.....	216
3.7.2 win8 下安装虚拟机以及 Ubuntu12.04.2 等软件.....	217
3.7.3 win8 下的 cmd.exe 程序.....	218
四 iTOP-4412 开发板镜像的烧写.....	221
4.1 镜像文件说明.....	221
4.1.1 Android4.0.3 和三种核心板配套的镜像.....	222

4.1.2Qt 和三种核心板配套的镜像.....	223
4.2 OTG 接口烧写方式.....	225
4.2.1 硬件平台.....	225
4.2.2 软件平台.....	225
4.2.3 烧写步骤.....	226
4.3 TF 卡烧写方式.....	234
4.3.1 制作可以烧写的 TF 卡.....	234
4.3.2 使用 TF 卡烧写.....	239
4.4 开发板出厂前首次烧写.....	241
<b>五 Android 开发环境搭建以及编译.....</b>	<b>245</b>
5.1 Android4.0.3 编译环境的两种搭建方式.....	246
5.1.1 使用已经搭建好的镜像.....	246
5.1.2 自己搭建环境.....	246
5.2 搭建环境.....	247
5.2.1 安装基本软件.....	247
5.2.2 安装编译组件.....	250
交叉编译工具.....	250
修改交叉编译工具的路径 ( 修改环境变量 ) .....	252
5.2.3 安装库文件、JDK 以及降低 GCC 版本.....	254
安装库文件和 JDK.....	255
降低 GCC 版本.....	257
5.3Android4.0.3 镜像的编译.....	260
5.3.1uboot 的编译.....	260
5.3.1.1 源码目录.....	260
5.3.1.2 编译器.....	261

5.3.1.3 参数配置.....	261
5.3.1.4 编译生成 uboot 镜像举例.....	261
5.3.2 Linux 内核的编译.....	264
5.3.2.1 源码目录.....	264
5.3.2.2 编译器.....	264
5.3.2.3 参数配置.....	264
5.3.2.4 编译生成内核镜像举例.....	265
5.3.3 Android4.0.3 的编译.....	268
5.3.3.1 源码目录.....	268
5.3.3.2 编译器.....	268
5.3.3.3 参数配置.....	269
5.3.3.4 编译生成 Android4.0.3 镜像.....	269
5.4Android4.4.4 镜像的编译.....	272
5.4.1uboot 的编译.....	272
5.4.2 Linux 内核的编译.....	272
5.4.2.1 源码目录.....	272
5.4.2.2 编译器.....	273
5.4.2.3 参数配置.....	273
5.4.2.4 编译生成内核镜像举例.....	273
5.4.3 Android4.4.4 的编译.....	273
六 Qtopia2.2.0 开发环境搭建以及编译镜像.....	275
6.1 uboot 的编译.....	275
6.2 Linux 内核的编译.....	275
6.2.1 参数配置.....	276
6.2.2 编译生成内核镜像举例.....	276

6.3 Qttopia2.2.0 编译的环境以及编译.....	279
6.3.1 编译器和基本库文件的安装.....	279
6.3.2 Qttopia2.2.0 源文件和补丁文件.....	282
6.3.3 库文件和编译 Qttopia2.2.0.....	285
6.3.4 第三方库文件.....	288
6.3.5 生成 system.img.....	288
七 Qt/E4.7 的编译和使用说明.....	292
7.1 Qt/E4.7.1 编译器的安装.....	293
7.2 Qt/E4.7.1 的编译.....	294
7.3 Qt/E4.7 和 Qttopia 的切换.....	296
7.3.1 设置开发板优先运行的文件系统.....	296
7.3.2 Qt/E4.7 和 Qttopia2.2.0 的触摸校准.....	297
7.3.3 系统运行后 Qt/E4.7 和 Qttopia2.2.0 的切换.....	298
7.4 QtE 库的编译配置选项简介.....	299
八 基于 Linux-C 的测试程序.....	303
8.1 测试程序的编译和运行.....	303
8.1.1 编译环境的设置.....	303
8.1.2 编译 helloworld.....	304
8.1.3 上传 helloworld 到开发板.....	304
8.1.4 修改程序权限和运行 helloworld.....	305
8.2 Led 灯的测试.....	306
8.3 Buzzer 蜂鸣器的测试.....	307
8.4 ADC 数模转换的测试.....	308
8.5 串口的测试.....	309
8.6 全能版 485 的测试.....	312

九 定制 Linux 内核.....	315
9.1 使用缺省文件配置和编译内核.....	315
9.2 驱动程序源代码的位置.....	316
9.3 Menuconfig 的用法.....	318
9.4 手动定制 Linux 内核.....	328
9.4.1 配置 CPU 平台文件.....	329
9.4.2 Vibrator 振动器.....	330
9.4.3 蜂鸣器 Buzzer.....	331
9.4.4 Leds.....	333
9.4.5 ADC 数模转换.....	334
9.4.6 RS-485.....	336
9.4.7 GPS 导航.....	337
9.4.8 RTC 实时时钟.....	339
9.4.9 I2C 总线.....	340
9.4.10 SPI 总线.....	343
9.4.11 CAN 总线.....	344
9.4.12 串口 UART.....	346
9.4.13 串口虚拟控制台 console.....	348
9.4.14 USB 转串口 PL2303.....	352
9.4.15 触摸屏 TP.....	354
9.4.16 显卡 Graphics Card.....	356
9.4.17 背光 Backlight.....	359
9.4.18 高清晰度多媒体接口 HDMI_HPD.....	361
9.4.19 高清晰度多媒体接口 HDMI_Audio.....	363
9.4.20 USB 摄像头 Camera.....	363

9.4.21 USB 键盘和键盘.....	365
9.4.22 矩阵键盘 GIPO_KEYS.....	367
9.4.23 U 盘.....	369
9.4.24 SD 卡/eMMC.....	371
9.4.25 AVIN 驱动.....	373
9.4.26 修改开机 logo.....	377
9.4.27 200W 和 500WCMOS ( ITU ) 摄像头的配置.....	377
9.5 制作最小文件系统镜像.....	380
9.6 最小 linux 系统的存储空间修改.....	418
9.7 以模块的方式编译内核驱动.....	422
单独编译驱动模块.....	429
9.8 一键实现开关机唤醒和休眠.....	431
9.8.1 设置启动方式：.....	431
9.8.2 设置一键休眠唤醒关机.....	432
+ Android 应用开发入门指南.....	434
10.1 搭建 Android 应用的开发环境.....	434
10.1.1 下载和安装 JDK.....	434
10.1.1.1 下载 JDK.....	434
10.1.1.2 AndroidJDK 和修改 JDK 环境变量.....	437
10.1.2 下载和安装 ADT 集成开发环境以及 SDK 管理器.....	441
10.1.3 下载 SDK.....	442
10.1.4 ADT 集成开发环境.....	446
10.1.5 创建 Android 模拟器.....	448
10.1.6 创建第一个 Android 应用程序 helloworld.....	452
10.1.7 在模拟器上运行 hellworld.....	454

10.2 在 iTOP-4412 开发板上调试 helloworld 应用.....	456
10.2.1 安装 ADB 驱动.....	456
10.2.2 测试 ADB 驱动.....	458
10.2.3 通过 OTG 接口调试 helloworld 应用.....	458
10.3 Led 应用程序.....	461
10.3.1 导入 Led 应用程序工程.....	461
10.3.2 导入工程常见问题.....	465
10.3.2.1 导入工程失败的原因.....	465
10.3.2.2 解决办法.....	466
10.3.4 在模拟器上调试.....	467
10.3.5 在开发板上调试.....	470
10.4 JNI 基础概念.....	473
10.4.1 ledtest 工程中的 JNI.....	473
10.4.2 什么情况下需要使用 JNI.....	473
10.4.3 与 JNI 相关的文件.....	474
10.5 Java 程序调用 JNI 的方法和步骤.....	475
10.6 Android.MK 文件.....	479
10.7 安装 NDK 编译器以及编译 JNI 库文件.....	482
10.7.1 安装 NDK 编译器.....	483
10.7.2 编译 Android 动态链接库.....	484
10.8 其他常用 Android 应用程序.....	485
10.8.1 Buzzer 应用程序.....	486
10.8.2 ADC 应用程序.....	487
10.8.3 串口应用程序.....	489
10.8.3.1 串口应用程序.....	489

10.8.3.2 串口应用移植需要注意的问题.....	491
10.8.4 蓝牙应用.....	492
10.8.5 485 应用.....	495
10.8.6 RFID 的应用.....	496
10.9 Android 文件系统源码修改.....	498
10.9.1 更改默认休眠时间.....	498
10.9.2 去掉默认安装的 APK.....	498
10.9.3 将 APK 编译到 Android 镜像中.....	499
10.9.4 手机和平板模式（横屏竖屏）.....	499
10.9.5 设置有线网.....	500
10.9.6 设置 Android 的 GPS.....	500
10.9.7 设置 Android 的 HDMI 转 VGA.....	502
10.9.8 设备权限的修改.....	503
十一 QtE 应用开发入门指南.....	505
11.1 Qt 的下载和安装.....	505
11.1.1 下载软件.....	505
11.1.1.1 网盘下载.....	506
11.1.1.2 官网下载.....	506
11.1.2 安装 Qt Creator.....	508
11.2 使用集成开发环境开发 Qt 应用程序.....	516
11.2.1 Ubuntu 上运行 helloworld.....	516
11.2.2 在 iTOP-4412 开发板上运行 helloworld.....	522
11.2.2.1 移植.....	522
11.2.2.2 安装运行.....	525
11.2.3 开发板上修改文件.....	526

11.2.4 开机运行 helloworld.....	527
11.3 QtE 必备知识介绍.....	528
11.3.1 开机启动脚本.....	529
11.3.2 qt4 文件的移植和修改（鼠标触摸以及字体）.....	531
11.3.3 qt 挂载盘符.....	534
linux QT 系统下挂载 u 盘.....	534
linux QT 系统下挂载 tf 卡.....	535
linux QT 系统下挂载带 USB 外壳的 tf 卡.....	535
11.3.4 QT 支持 HDMI 显示.....	536
配置内核.....	536
修改 linux-QT 文件系统.....	536
11.3.5 QtE 连接 WIFI.....	538
十二 Ubuntu 的应用.....	539
12.1 烧写 Ubuntu.....	539
12.1.1 TF 卡读写速度测试.....	539
12.1.2 烧写 Ubuntu.....	541
12.2 Ubuntu 的 uboot 以及内核编译.....	550
12.2.1 uboot 的编译.....	550
12.2.1.1 参数配置.....	550
5.3.1.4 编译生成 uboot 镜像举例.....	551
12.2.2 Linux 内核的编译.....	553
12.2.2.1 参数配置.....	553
5.3.2.4 编译生成内核镜像举例.....	553
12.3 Ubuntu 下使用 wifi.....	556
12.3.1 精英版 wifi 配置.....	557

12.3.2 全功能板 wifi 配置.....	560
附录一 QT 第三方库文件的编译.....	561
交叉编译 jpeg 库.....	562
交叉编译 e2fsprogs-1.40.2.....	564
交叉编译 libpng 库.....	565
交叉编译 zlib.....	566
附录二 编译 ARM-qtopia-free-src-2.2.0 常见错误的处理.....	568
1 缺少工具错误.....	568
2 缺少类声明.....	570
3 缺少文件错误.....	571
4 无法匹配到 QSizePolicy 类的构造函数.....	572
5 类的成员函数前有额外的类名字.....	575
6. open 函数调用缺少必要的参数.....	578
7 缺少 this 指针.....	580
8 缺少系统头文件.....	581
附录三 内核配置详细说明.....	586
第一部分.....	586
第二部分.....	586
第三部分.....	588
第四部分.....	589
第五部分.....	590
第六部分.....	590
第七部分.....	590
第八部分.....	590
第九部分.....	591

第十部分	591
第十一部分	591
第十二部分	594
第十三部分	597
第十四部分	597
第十五部分	598
第十六部分	598
第十七部分	601
第十八部分	601
附录四 Linux 下多核处理器相关知识	602
附录五 Android 系统架构	605
附录六 iTOP-4412 源码的开发版本下载和使用	610
6.1 Uboot 的下载和编译	610
6.2 Kernel 源码下载及编译	611
6.3 文件系统的下载	614
6.3.1 android4.0 代码下载和编译	614
6.3.1.1 repo 下载	614
6.3.1.2 Android4.0 代码下载	615
6.3.1.3 Android4.0 源码编译	618
6.3.2 android4.4 代码下载和编译	620
6.3.2.1 repo 下载	620
6.3.2.2 Android4.4 代码下载	621
6.3.2.3 Android4.4 源码编译	625
6.3.3 Linux Qt 文件系统下载及制作	626
6.3.4 Ubuntu 文件系统	628

联系方式 ..... 629

# 版本

注意：使用手册更新后，会直接上传到迅为 QQ 技术支持群的共享文件中。

当前版本为 iTOP-4412 开发板之精英版使用手册\_V2.1

更新说明 V2.1

日期	改动
2015.12.14	添加内容
	1.1.1 小节 更加详细的核心板说明 1.2 小节 新的光盘目录
	第四章 镜像目录修改，其中几个小节的名称修改，将 Ubuntu 烧写移到第十二章
	第五章 去掉 Android4.2.2 的编译，使用新的编译方式，编译文件系统对应的 uboot 和 kernel
	第六章 使用新的编译方式，编译文件系统对应的 uboot 和 kernel
	第七章 使用新的编译方式，编译文件系统对应的 uboot 和 kernel
	第九章 9.1 小节 对新的缺省文件进行解释
	第十二章 名称改为 Ubuntu 的编译烧写应用，添加编译 Ubuntu 对应的 uboot 和 kernel，添加 Ubuntu 的烧写

当前版本为 iTOP-4412 开发板之精英版使用手册\_V2.0

更新说明 V2.0

日期	改动
2015.8.26	添加内容
	添加 1.5 使用 github 获取开发源码
	添加 3.2.1.4 虚拟机安装 Ubuntu 常见问题之 64 位虚拟化
	添加 3.3.7 虚拟机无法识别 USB3.0 的解决方法
	添加 3.3.8 U 盘、TF 卡与虚拟机连接
	第四章第五章第六章添加更详细的图文说明
	添加 11.3.5 QtE 连 WIFI
	添加附录六 iTOP-4412 源码的开发版本下载和使用
	修改其它几个小地方

当前版本为 iTOP-4412 开发板之精英版使用手册\_V1.9

更新说明 V1.9

日期	改动
2015.7.17	添加内容 1.1.3 串口使用的修改方法 10.9.8 串口设备权限的修改 1.1.4 精英版以及核心板结构 3.2 小节 添加更详细的介绍
2015.07.15	添加以下内容 3.3.7 虚拟机无法识别 USB3.0 的解决方法 3.6.4 ADB 驱动安装常见问题解决办法汇总 3.7 win8 下基础软件的安装和学习 3.7.1 超级终端的安装和使用 3.7.2 win8 下安装虚拟机以及 Ubuntu12.04.2 等软件 3.7.3 win8 下的 cmd.exe 程序

2015.07.03	添加 1.1.2 底板硬件简介
------------	--------------------

更新说明 V1.8

日期	改动
2014.5.22	Andoird4.4.4 编译小错误
	添加以下章节 2.4.4.3 声卡的内外放设置 9.4.25 AVIN 驱动 9.4.26 修改开机 logo 9.6 最小 linux 系统的存储空间修改 9.7 以模块的方式编译内核驱动 9.8 一键实现开关机唤醒和休眠 10.8.6 RFID 的应用 10.9.4 章节名称改为手机和平板模式（横屏竖屏） 10.9.6 设置 Android 的 GPS 10.9.7 设置 Android 的 HDMI 转 VGA 11.3.3 qt 挂载盘符 11.3.4 QT 支持 HDMI 显示 十三 Ubuntu 的应用 13.1.1 精英版 wifi 配置 13.1.2 全功能板 wifi 配置
	1.3 网盘资料 重新划分了网盘目录

更新说明 V1.7

2015.04.15	修改一些小错误和一些表述不清楚的地方
------------	--------------------

更新说明 V1.6

2014.12.30	修改一些小错误和不严谨的地方
	添加 “1.4 网盘压缩包 MD5 值的使用”
	添加 “3.3.6 虚拟机 Ubuntu 扩展硬盘空间”
	添加 “5.5 Android4.4.4 环境的搭建和编译”

	添加 “7.4 QtE 库的编译配置选项简介”
	新增加十一章 “QtE 应用开发入门指南” , 其中增加 3 个小节

### 更新说明 V1.5

日期	改动
2014.12.20	新增 “4.4.1 TF 卡读写速度测试”
	新增 “第七章 Qt/E4.7 的编译和使用说明”
	新增 “第八章 基于 Linux-C 的测试程序”
	新增 “第九章 定制 Linux 内核”
	新增 “第十章 Android 应用开发入门指南”
2014.12.05	将附录一到附录六合并为 “第三章 iTOP-4412 平台基础软件的安装和学习”

### 更新说明 V1.4

日期	改动
2014.11.11	添加基础部分附录一至附录六

### 更新说明 V1.3

日期	改动
2014.10.21	在整个文档中 , 每一章节的内容都做了更加详细的说明

## 前言

去年这个时候（2012年），四核ARM处理器还是个新鲜事物，但短短一年时间，嵌入式多核系统已经深入人心，并开始大面积普及应用，以极快的速度向工控、仪表等行业延伸，这些都是我们始料不及的。

面对这样的形势，我们是固守还是迎接新的挑战？答案是肯定的，世界的潮流永远是浩浩荡荡，不会停止前进的脚步。

如果我们固守，老旧的芯片价格将越来越贵，例如多年前大量使用的SDRAM价格已经很高昂，几年前的DDR2也开始不便宜，现在的DDR3反倒性价比更高，其他芯片包括存储介质也是这样。

如果我们停滞不前，一些新系统、新架构所带来的生产力将不能在新产品中体现，竞争能力以及产品优势随之会受到影响。

同时，以前掌握的知识和能力将逐渐陈旧，越来越不能适应时代不断发展、需求不断进步给我们提出的新要求。

发展是硬道理，尤其在信息技术这个领域，只有不断学习，积极改变才能把握先机，赢得更多的机遇和挑战。

从架构上来讲，多核系统比单核要复杂很多，性能也成倍增长，但掌握它真的很难吗？

实际上，从嵌入式系统开发的角度来看，所谓的‘四核’其实并没有多少改变，留给我们开发人员的工作跟以前也没有多少差别，因为Linux本身一直是支持多核的。

实际开发的时候，我们会发现，多核系统的驱动和应用程序的编程方式、编程理念、编译工具、以及开发方法和原来单核系统并无差别。

作为一款开发板的产品手册，我们将尽可能详尽的为大家讲解Exynos 4412这款处理器的系统应用特点，对于开发过程中可能碰到的难点和问题给予较全面的说明。

欢迎广大用户提出宝贵意见，便于我们更好的改进和提高！

最后祝大工作家学习愉快、开发顺利！

迅为电子 ·2015 年 11 月

## 必须注意的问题

开发板属于裸板，为了开发和学习方便，一般不会增加外壳，作为电子产品如果使用不当就容易出问题，严重时甚至会永久性损伤硬件系统，以下几点是您需要注意的：

- 1、在确认系统连接完好的情况下再上电，如果发现异常应及时关断电源。
- 2、尽量不要用手接触芯片。如果一定要摸，请注意人体放电（可以通过摸一下地面等方式来消除静电）；尤其是北方干燥的冬天，人体静电会高达几千伏，足以击穿芯片，导致系统损毁。
- 3、不要带电插拔 HDMI 接口。当用户使用 HDMI 接口连接外部显示器或者 TV 时，一定要注意在开发板和显示设备断电的情况下进行操作，切忌在开发板上电的情况下来回插拔，因为 4412 处理器直接输出 HDMI 信号，中间并没有接口芯片，如果操作不当，可能导致处理器损毁。
- 4、不要带电插拔串口 UART。同以上第 3 点描述的那样，连接和断开串口线时请在开发板设备断电的情况下进行。
- 5、拆装核心板时要格外小心。分拆核心板正确的做法应该是：用扁平的工具在核心板边缘轻轻翘起，然后在另一个方向进行相同的操作。
- 6、HDMI 接口不能连接笔记本电脑，因为笔记本电脑的 HDMI 也是输出设备！只能连接带有 HDMI 接口的电脑显示器或电视。
- 7、开发板有两个 HDMI 接口，其中一个是专门连接迅为自家屏幕的（实际是 LVDS 信号）；而另一个才是真正的 HDMI 接口，用来连接电脑显示器或带有 HDMI 接口的电视机。

## 名词解释

在 iTOP-4412 开发平台的使用过程中，会用到各种终端，各种终端的命令又有交集，为了便于用户理解，作如下说明：

用户在超级终端中，输入的命令，也就是电脑通过串口和开发板交互，它有下面两种模式：

超级终端：uboot 模式，参考 2.3.1 小节以及 3.1 小节

超级终端：文件系统模式，参考 2.3.2 小节以及 3.1 小节

在 Windows 命令行中，PC 机通过 USB 接口和开发板交互，可以输入以下两种命令：

Windows 命令行：fastboot 命令，参考 3.6 小节

Windows 命令行：adb 命令，参考 3.6 小节

在 Ubuntu 命令行中，用户可以输入 Linux 命令，参考 3.3 小节。

# — iTOP-4412 开发板介绍

## 1.1 开发板平台简要介绍

### 1.1.1 核心板

Exynos4412 有两种封装形式，其中 POP 封装的芯片内含 1GB 内存，所以不需要外扩 DDR，可大大节省 PCB 面积，功耗控制方面也更好，多用于手持设备当中；SCP 封装优点是内存扩展更灵活，生产工艺相对更容易控制。

电源芯片 S5M8767 的输入电压范围是 3.5v~5.5v，但是最佳的输入电压是 4v，这样可以使 S5M8767 芯片处于最佳的工作状态。

#### 1.1.1.1 POP 封装

长宽：5CM \* 6CM，高度 1.5MM，320 个引脚（80 \* 4）；

板载 1GB 内存，电源管理；

和底板装配的时候注意“防呆箭头”。



### 1.1.1.2 SCP 1G 封装

长宽：6CM \* 7CM，高度 1.5MM，320 个引脚（80 \* 4）；

SCP 板载 1G 内存，电源管理；

和底板装配的时候注意“防呆箭头”。

如下图所示，核心板 SCP 1G



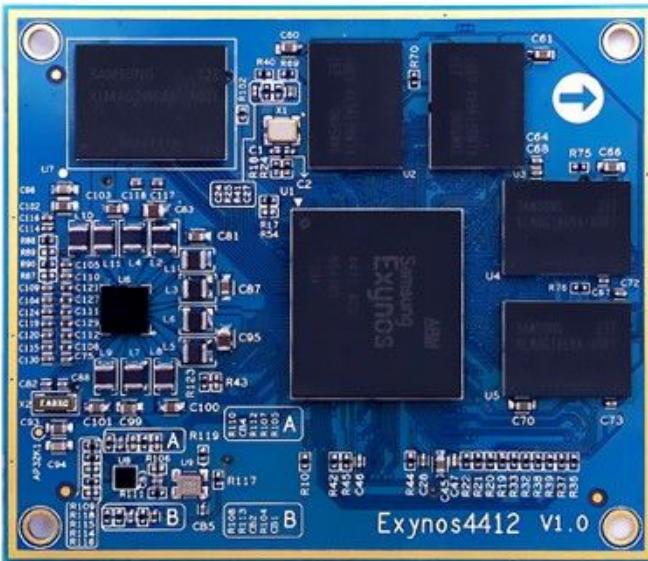
### 1.1.1.3 SCP 2G 封装

核心板 SCP1G 和 SCP2G，需要对比内存芯片，四颗芯片和 PCB 封装的丝印外框靠的远的是 1G 内存，和丝印外框靠的近的是 2G。

长宽：6CM \* 7CM，高度 1.5MM，320 个引脚（80 \* 4）；

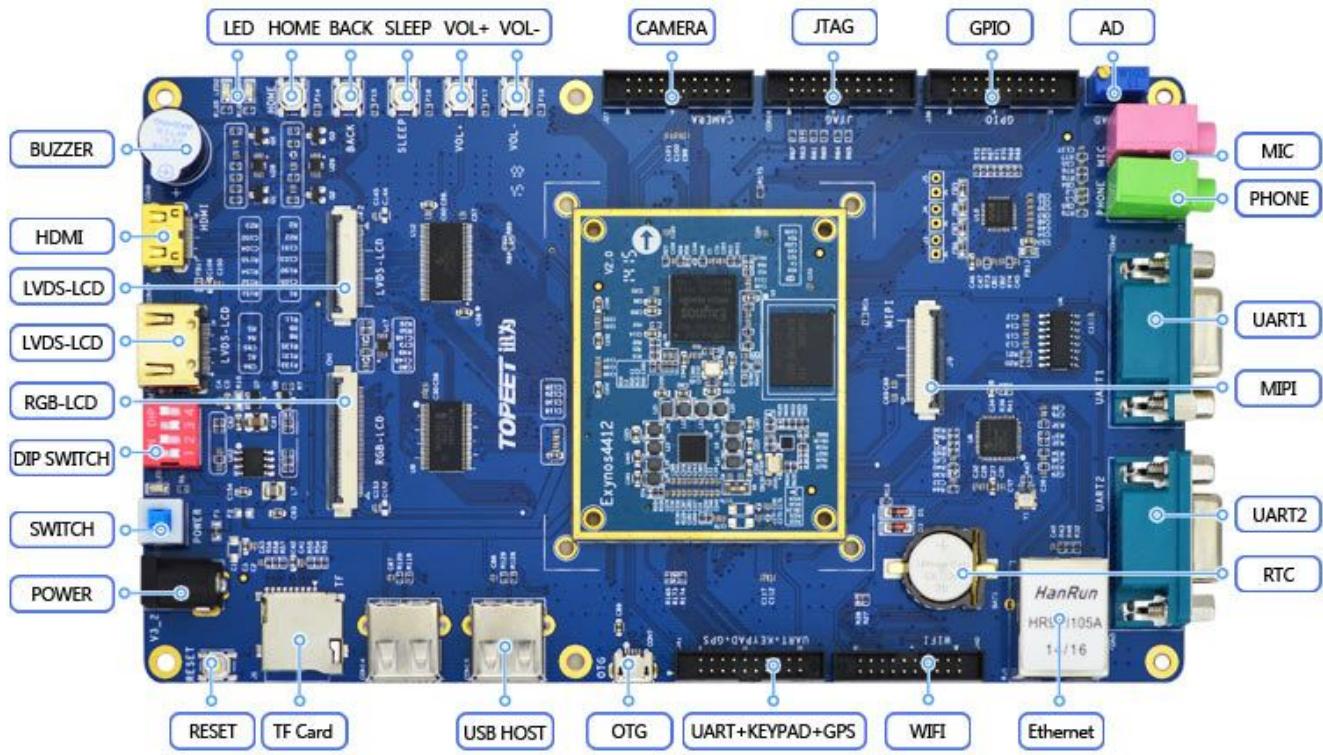
SCP 板载 2G 内存，电源管理；

和底板装配的时候注意“防呆箭头”。



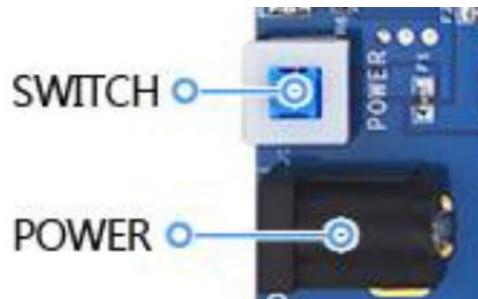
## 1.1.2 底板

iTOP-4412 精英版底板如下图所示（下图中的核心板是 POP ）：

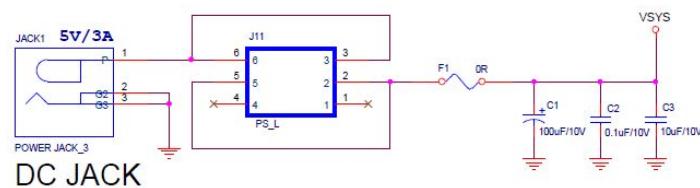


### 1.1.2.1 电源以及接口

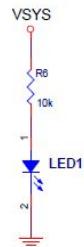
如下图所示，为电源以及电源开关，输入 5V 电源即可。



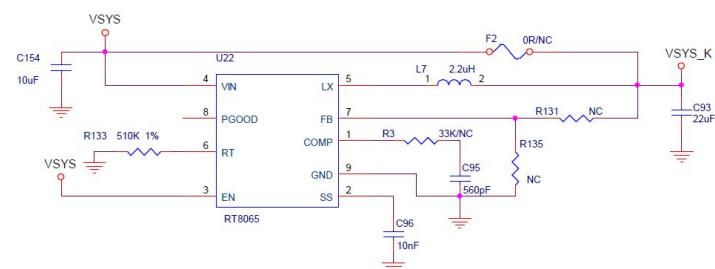
开发板电源原理图部分如下。



上电开机之后 LED1 会亮，表明有电源输入，原理图如下所示。

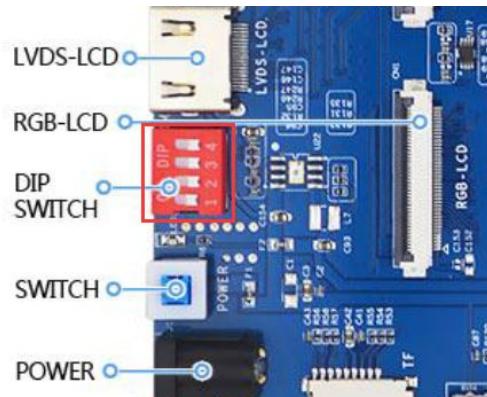


核心板供电部分如下图，建议给核心板提供 4V 电源，可以使核心板电源管理芯片 8767 处于最佳工作状态，原理图如下所示。

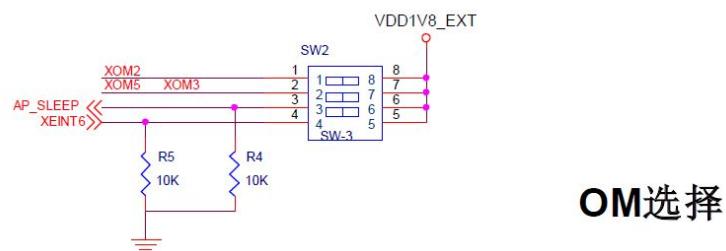


### 1.1.2.2 拨码开关以及对应功能

iTOP-4412 开发板可以通过拨码开关控制启动方式以及显卡输出。

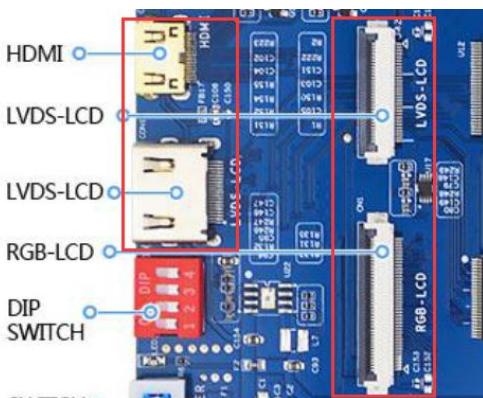


如下图，XOM2，XOM3，XOM5 用于控制 4412 启动方式，AP\_SLEEP，XEINT6 用于控制显卡输出。拨码开关的具体用法可以参考使用手册 2.2 小节。



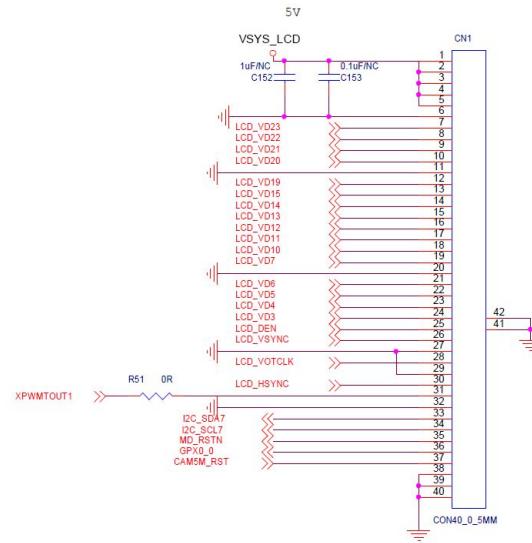
### 1.1.2.3 显卡资源以及接口

iTOP-4412 有丰富的视频输出接口，如下图所示包括 HDMI，RGB 以及 LVDS。在手册 2.1.2 小节有具体怎么进行硬件连接的详细介绍。

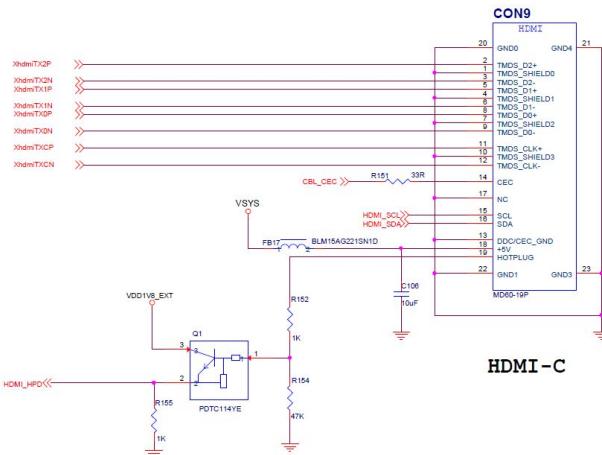


RGB-LCD 接口用于支持 4.3 寸屏幕。

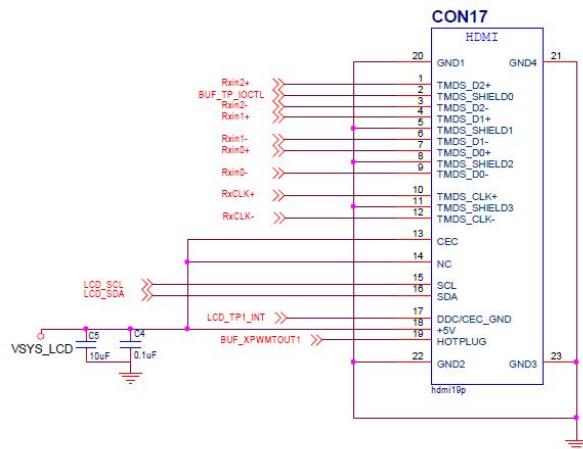
#### 4.3 LCD RGB Interface



HDMI 接口用于支持各种 HDMI 显示器电视等等。这里需要注意的是，小的 HDMI 口输出的才是 HDMI 信号，大的 HDMI 口输出的是 LVDS 信号。



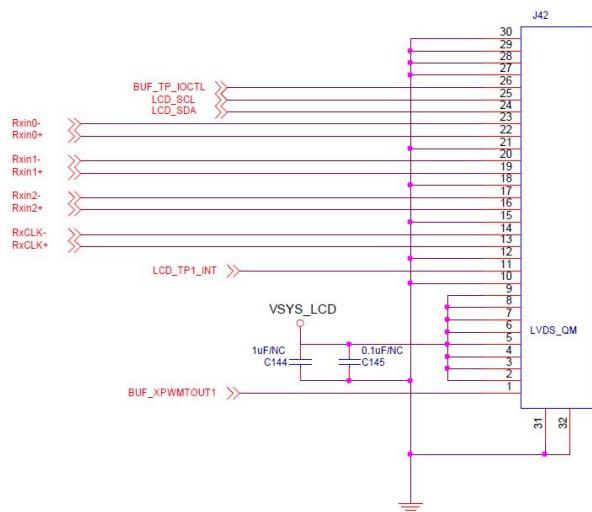
LVDS-HDMI 接口可以迅为的 7 寸屏幕或者 9.7 寸屏幕。



**LVDS LCD Interface using HDMI's Wire**

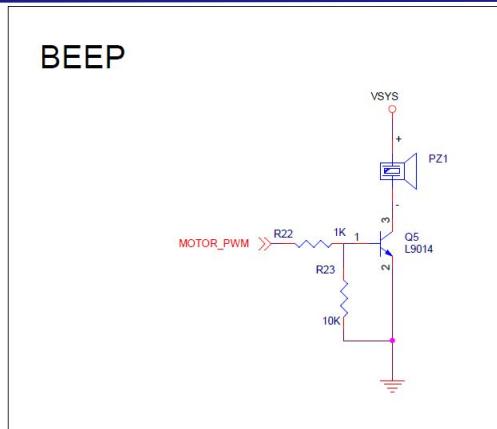
LVDS-LCD 接口使用排线连接迅为的 7 寸或者 9.7 寸屏幕。

### LVDS Interface

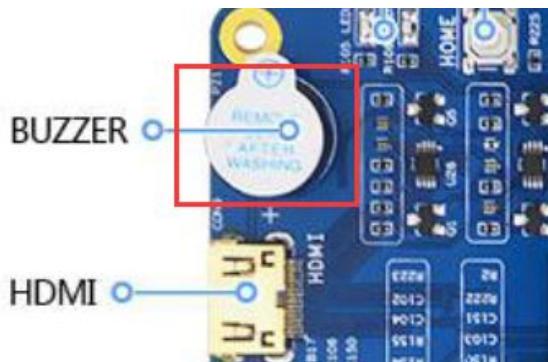


#### 1.1.2.4 蜂鸣器 BEEP

BEEP 的原理图如下图，MOTOR\_PWM 网络给高电平即可使蜂鸣器发出“滴滴”的声响。

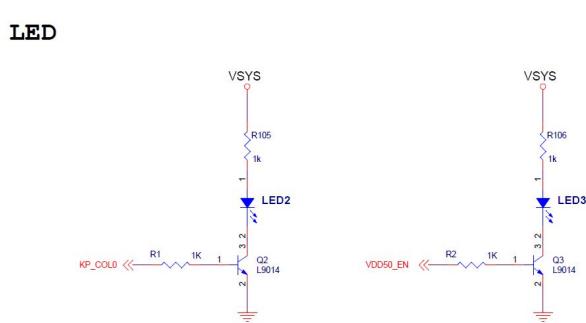


如下图示蜂鸣器在 PCB 上的位置。



### 1.1.2.5 灯 LEDs

LED 灯原理图如下图所示。

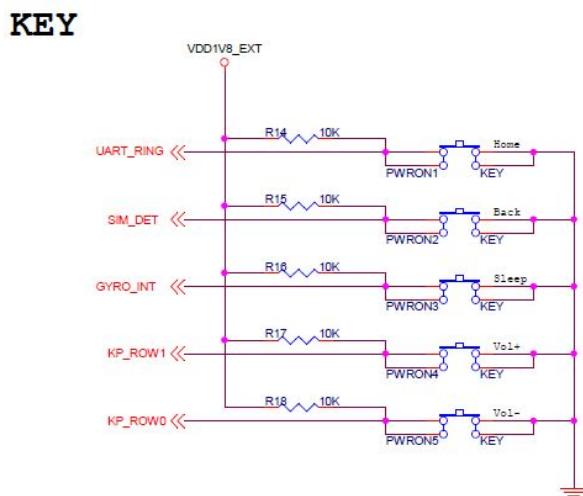


LED 在蜂鸣器旁边，高电平 LEDs 即可点亮。

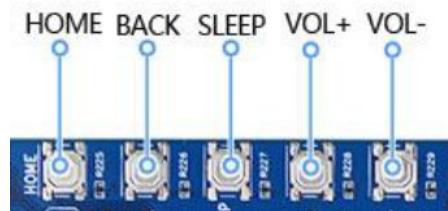


### 1.1.2.6 按键 Keys

底板有 5 个按键，原理图如下，原理比较简单。



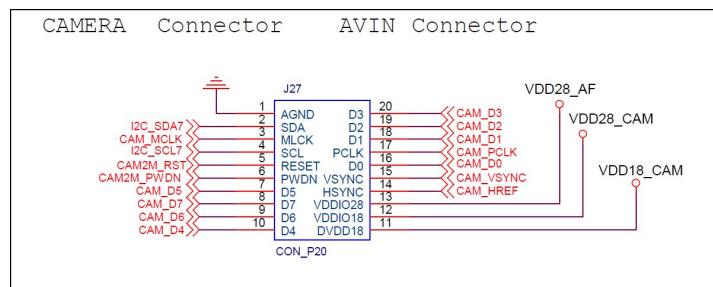
在 Android 系统中，这几个按键和平板的类似，分别是 HOME 按键，返回按键，休眠按键（休眠后唤醒也是该按键），音量加和减。



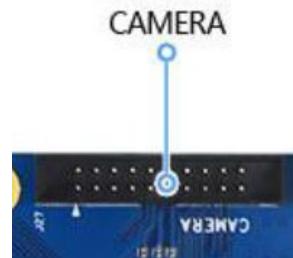
### 1.1.2.7 摄像头 CAMERA+AVIN 扩展口

摄像头连接的时候注意“三角形箭头”要和模块小箭头对应。

原理图如下图所示

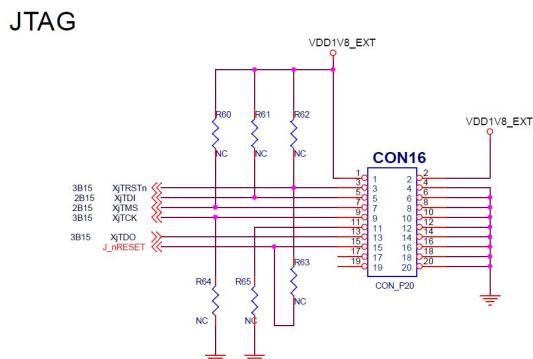


PCB 上 camera 接口，如下图所示。

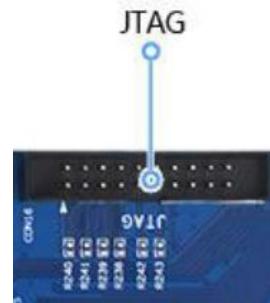


### 1.1.2.8 JTAG 扩展口

原理图如下图所示。



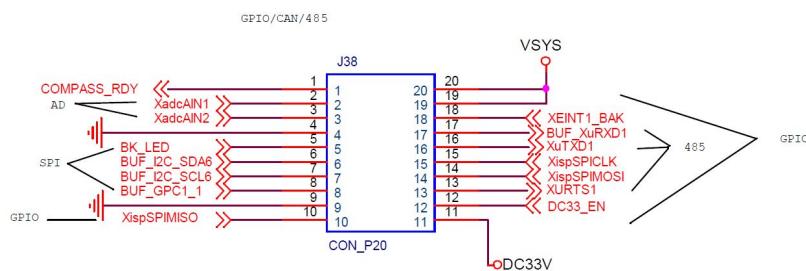
PCB 上 JTAG 接口，如下图所示。



### 1.1.2.9 GPIO+CAN+485 扩展口

CPIO 连接的时候注意“三角形箭头”要和模块小箭头对应。

原理图如下图所示



PCB 上 GPIO 接口，如下图所示。

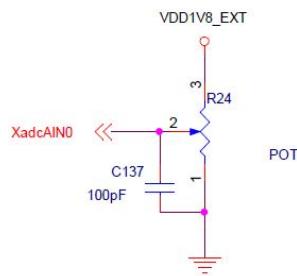


### 1.1.2.10 模数 A/D 转换

4412 有三路 A/D,另外两路在 GPIO 中引出。可以参考 1.1.2.9 小节。

原理图如下图所示

## AD



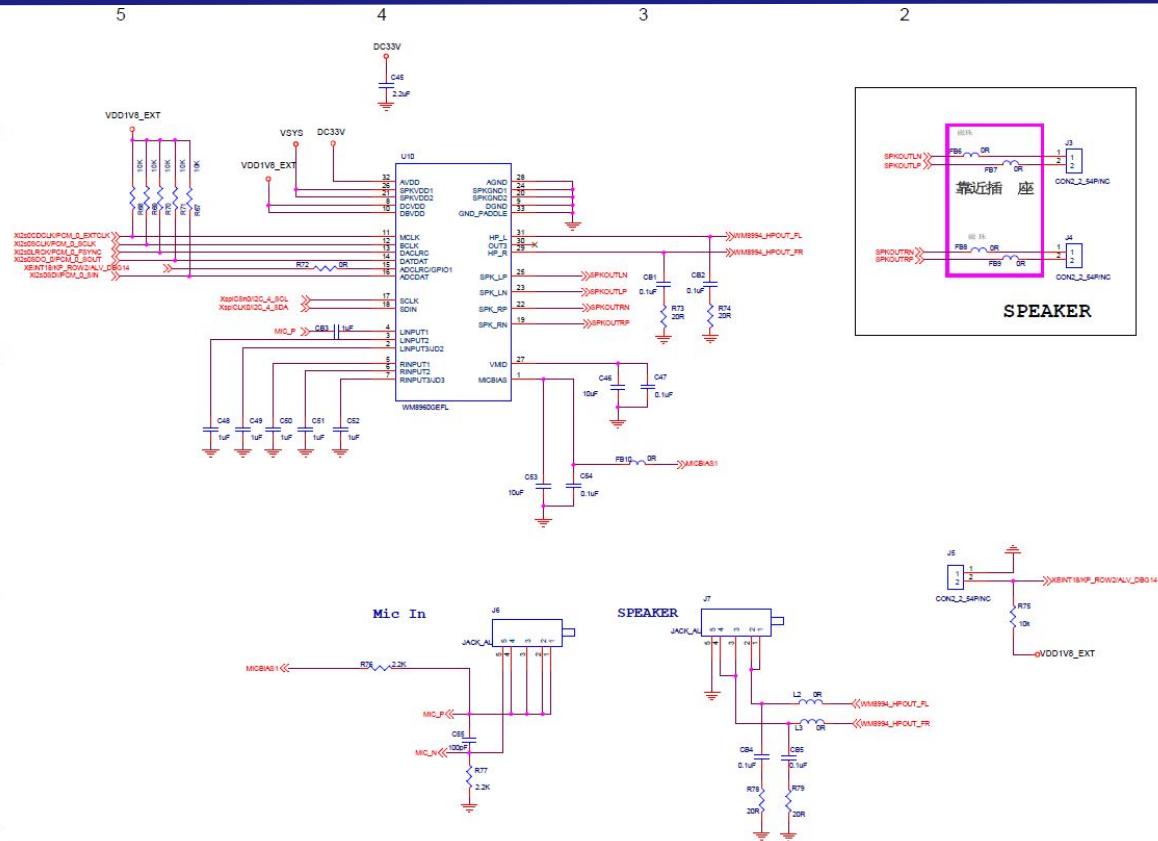
PCB 上 AD , 如下图所示。



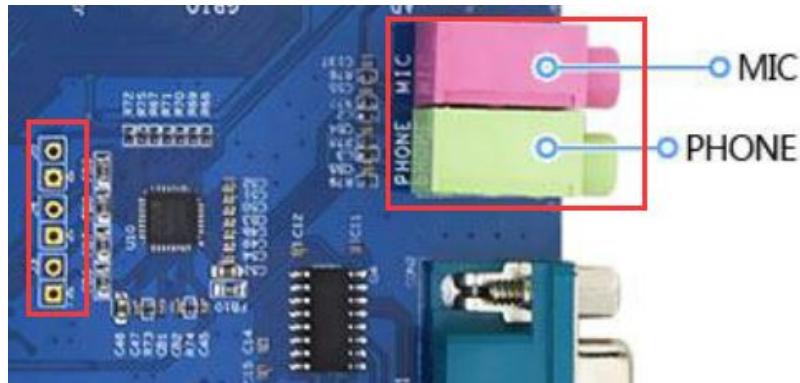
### 1.1.2.11 声卡资源以及接口

声卡有耳机和耳麦接口，还有外放如下图所示左边红色方框的 PIN2 的喇叭外放扩展口。

原理图如下图所示



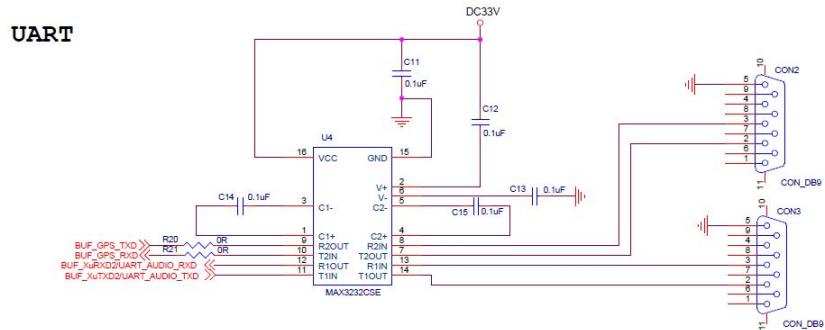
PCB 上声卡，如下图所示。



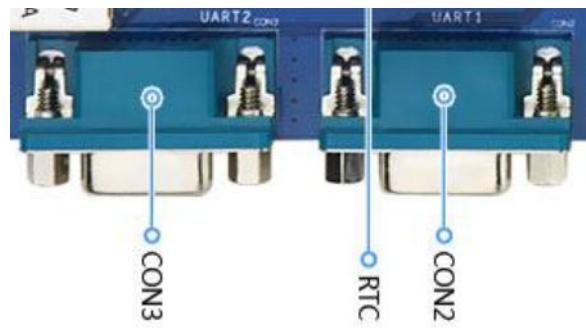
### 1.1.2.12 串口接口

CON3 口默认用来调试程序，CON2 可以直接使用。CON2 和 CON3 都是输出 RS232 电平，可以和电脑的串口直接相连，如果和电脑的 USB 接口相连，那么需要 USB 转串口。

原理图如下图所示

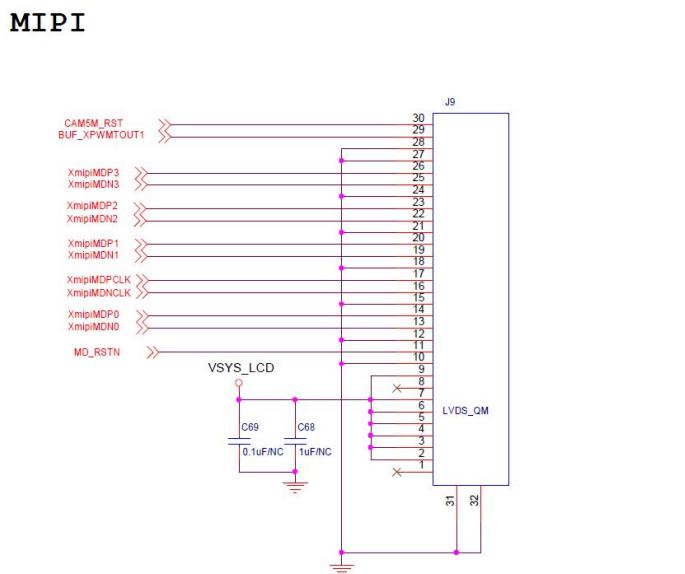


PCB 上串口，如下图所示。

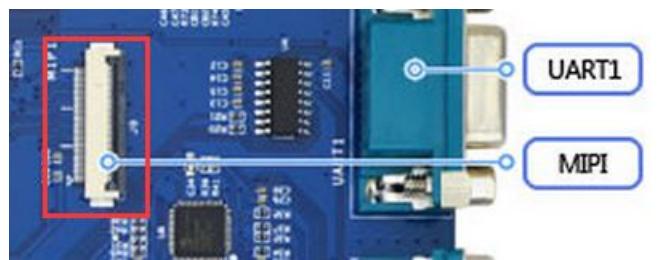


### 1.1.2.13 MIPI 接口

MIPI 接口可以接高清 MIPI 摄像头。原理图如下图所示。

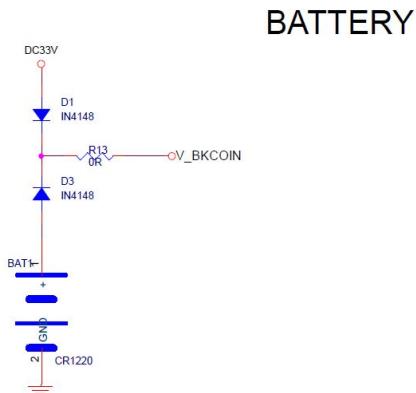


PCB 上 MIPI 接口，如下图所示。



#### 1.1.2.14 实时时钟 RTC

RTC 实时时钟，在 Android4.0.3 中，安装 CR1220 锂电池之后，完全断电之后时钟也可以工作。原理图如下图所示。



PCB 上 RTC 接口，如下图所示。



#### 1.1.2.15 以太网

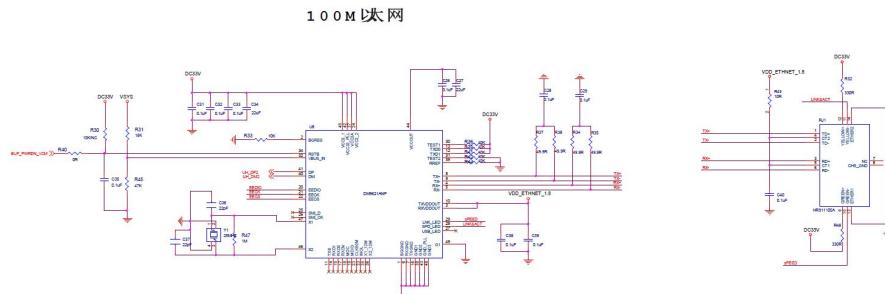
如下图所示是以太网。

在 Android4.0.3 中，需要参考 2.4.3 来设置以太网。

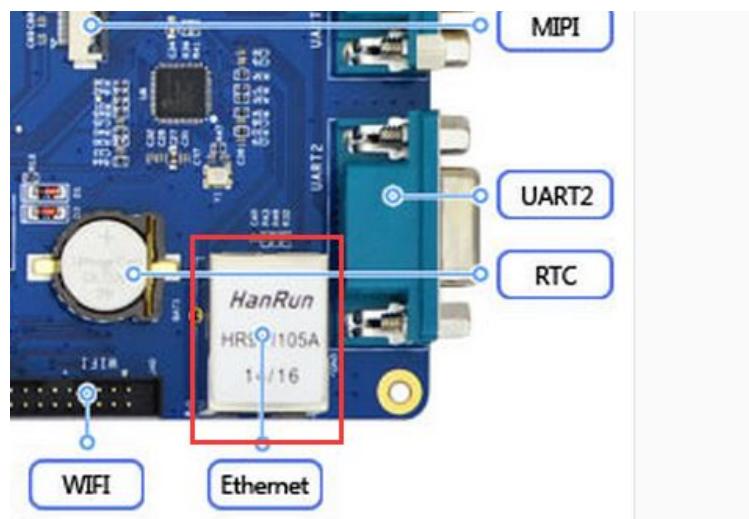
在 Android4.4.4 中，则可以在 Android 设置中，将默认 WIFI 网络设置为有线网。

设置好之后就可以上网了。

原理图如下图所示。



PCB 上以太网接口，如下图所示。

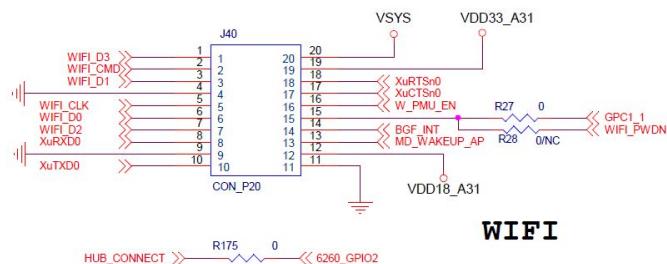


### 1.1.2.16 WIFI 扩展口

开发板接入 WIFI 模块之后即可使用 WIFI 上网。WIFI 模块连接的时候注意“三角形箭头”要和模块小箭头对应。

这里需要特别注意的是，如果在 Android 中打开了 WIFI，那么则在系统重新启动时，必须有 WIFI 模块。

原理图如下图所示。



PCB 上 WIFI 接口，如下图所示。

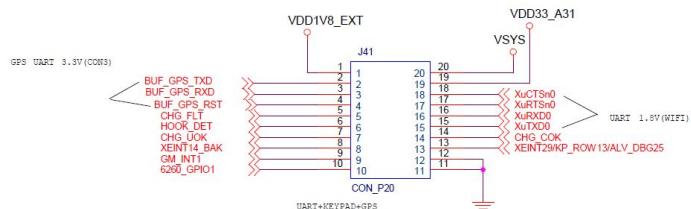


### 1.1.2.17 串口+矩阵键盘+GPS 扩展口

这里需要注意的是部分串口复用了，用户可以根据实际使用情况，参考原理图来使用串口。该接口可以用来扩展 GPS、矩阵键盘、串口模块。

原理图如下图所示。

UART+KEYPAD (4\*4)



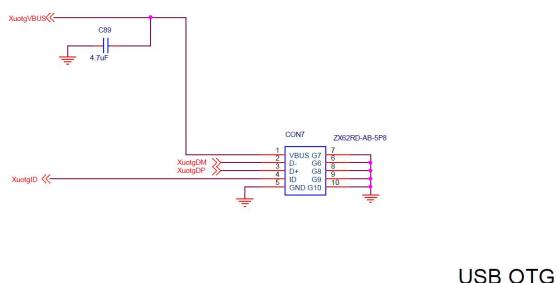
PCB 上串口+矩阵键盘+GPS 扩展口，如下图所示。



### 1.1.2.18 OTG 接口

OTG 接口用来烧写镜像，还可以用来作为 Android 应用 APP 的调试口。在 Android 系统下面可以用来上传文件和安装应用 APP（在没有网络的情况下很好用）。

原理图如下图所示。



USB OTG

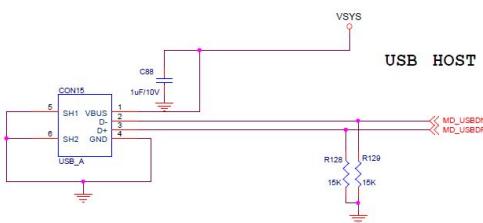
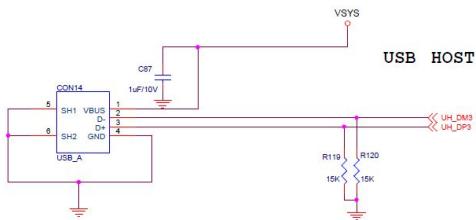
PCB 上 OTG 接口，如下图所示



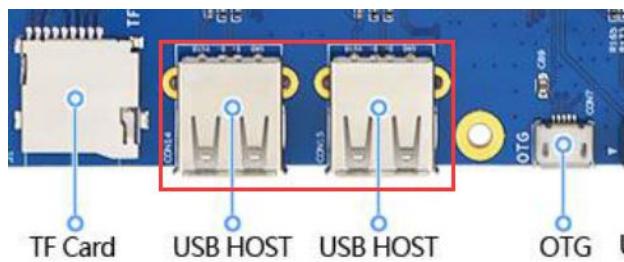
### 1.1.2.19 USB 接口

USB 接口可以用来接鼠标和键盘。还可以用来接 PL2303 转接线，用于扩展串口，不过需要加载 PL2303 的驱动，具体加载方法参考使用手册 9.4.12.

原理图如下图所示



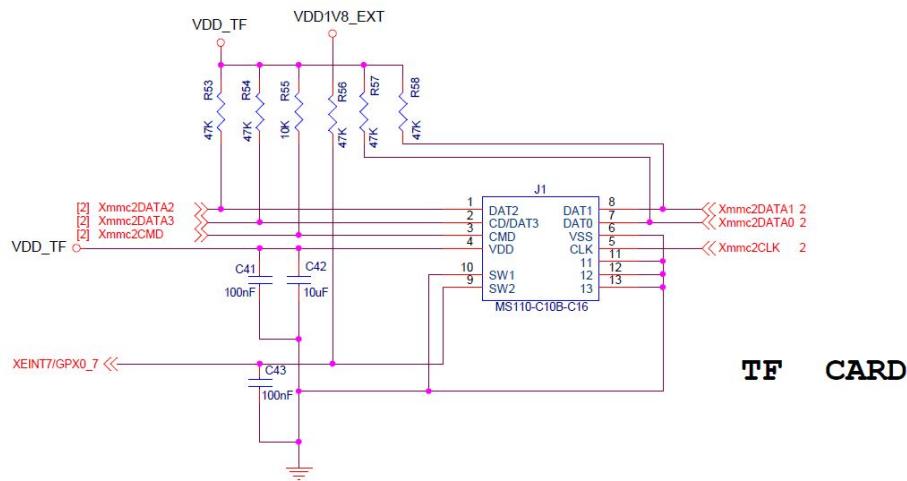
PCB 上 USB 接口，如下图所示



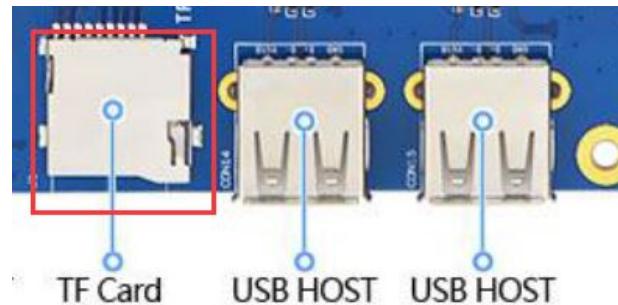
### 1.1.2.20 TF 卡接口

精英版可以使用 TF 卡，用于烧写系统或者存储数据，烧写系统在第三章有详细的介绍，存储数据的使用类似于手机中的 TF 卡。

原理图如下图所示。



PCB 上 TF Card 接口，如下图所示。



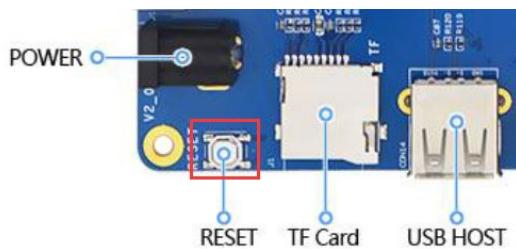
### 1.1.2.21 复位按键

火牛座和 TF Card 旁边的 RESET 是复位按键，在 Android 系统下面，按住大约 5 秒，即可重启。这个功能按键类似手机的重启按键，长按可以复位。

原理图如下图所示。



PCB 上复位按键，如下图所示。



### 1.1.3 精英版使用串口修改方法

本节供用户查阅，新手用户如果有不理解的概念，可以在学习后面的基础知识后，在用到相应串口接口时再来查阅。

精英版上引出了两个 RS232 的串口，分别是 CON2 和 CON3。

CON2 对应的串口设备节点是 /dev/ttySAC2，默认是作为系统的调试串口，输出电平是 RS232。

原理图中的网络是 “BUF\_XuRXD2/UART\_AUDIO\_RXD” 和  
“BUF\_XuTXD2/UART\_AUDIO\_TXD” 。

CON3 对应的串口设备节点是 /dev/ttySAC3，输出电平是 RS232；CON3 和 GPS 复用 ( J41，GPS 的电平是 TTL3.3V )。

原理图中的网络是 “BUF\_GPS\_RXD 和 BUF\_GPS\_RXD” 。

如果使用 CON3，不要插 GPS 模块，也不允许其它设备占用；同理如果要使用 GPS，那么 CON3 口就不要接设备。

WIFI 模块 ( J40 ) 对应的串口设备节点是 /dev/ttySAC0，输出电平是 TTL1.8V；WIFI 模块和 J41 的其中一个串口复用，J41 输出电平也是 TTL1.8V。

原理图中的网络是 “XuRXD0 和 XuTXD0” 。

无论是使用蓝牙功能还是使用 WIFI 的功能，这个串口就被占用了；同理如果 J41 上的使  
用了这个串口，也会导致 WIFI/BT 功能不正常。

485 模块 ( J38 ) 对应的串口设备节点 /dev/ttySAC1。

原理图中的网络是 “XuTXD1 和 BUF\_XuRXD1” 。电平分别是 TTL1.8V 和 TTL3.3V。

用户想测试串口，可以参考使用手册“8.5 小节 串口的测试”。

调试串口作为其它用途，可以参考使用手册“9.4.13 小节 串口虚拟控制台 console”。

串口在实际使用中可能需要修改权限，具体方法参考使用手册“10.9.8 小节 串口设备权  
限的修改”。

## 1.2 光盘资料

用户购买开发板的同时，迅为电子会给附赠一张光盘，如下图，光盘目录如下。



下面简单的做一下了解，在需要使用这些资料的时候，会针对性的做详细介绍。

将文件以及文件夹按照“名称+递增”的方式排列，如下：

01\_PCB\_SCH\_DATASHEET-----开发板的原理图，PCB 以及元件的 DATASHEET;

02\_编译器以及烧写工具-----编译工具、烧写工具、各种驱动以及其它工具；

03\_镜像Android4.0.3 文件系统-----Android4.0.3 文件系统的镜像以及对应的 uboot、kernel 镜像；

04\_镜像QT文件系统-----QtE4.7 以及 qtopia2.2.0 文件系统的镜像以及对应的 uboot、kernel 镜像；

05\_镜像\_Ubuntu 文件系统-----Ubuntu 文件系统镜像以及对应的 uboot、kernel 镜像；

06\_源码\_uboot 和 kernel-----uboot 以及 kernel 内核的源码；

07\_源码\_Android4.0.3 文件系统-----Android4.0.3 文件系统的源码；

08\_源码\_QtE 以及 qtopia2.2.0 文件系统-----QtE4.7.1 源码，qtopia2.2.0 文件系统以及对应的各种库和工具；

光盘目录说明.txt+用户手册.pdf

开发板对应的用户手册 pdf 文档以及光盘目录说明 TXT 文本。

### 1.3 网盘资料

网盘的链接在购买开发板后可以在迅为电子技术支持 QQ 群下载。如果链接有更新，会在群里贴通告。

网盘资料分为以下七个文件夹，按“文件名”方式排列，顺序如下。

#### Exynos4412 三星原厂资料

三星原厂资料

iTOP-4412 开发板所需 PC 软件（工具）

该文件夹中是和烧写相关的工具和驱动。

01-USB 转串口（PL2302 驱动）

02-超级终端（串口调试助手）

03-ADB 驱动

04-SSH 软件

05-fastboot 烧写工具

06-TF 卡测试工具

## iTOP-4412 开发板搭建编译环境所需要的工具包以及补丁包

该文件夹中是与编译相关的工具包以及补丁包。

01-虚拟机 VMware\_Workstation\_wmb 软件

02-Ubuntu 软件

03-编译 linux-QT 文件系统需要补丁包

04-编译无界面 Linux 文件系统需要工具包以及补丁包

05-Android 上层应用程序编译时需要的工具软件及插件

06-Qt\_Creator

## iTOP-4412 开发板源码（其它）

android\_4.0.3 测试 APK

android\_4.2.2 源码以及对应 Kernel 源码

android\_4.4.4 源码以及对应 Kernel 源码

QT-ARM 官方原始程序

QT-ARM 源码文件夹（2014 年 7 月前购买用户使用）

小模块的测试程序

支持 HDMI 的 Ubuntu 资料

支持以太网的 4.0 代码

最小 Linux 文件系统

## iTOP-4412 开发板相关文档（补充）

iTOP-4412 官方 QQ 群专题讨论

iTOP-4412 常见问题及解决方法

## iTOP-4412 开发板视频教程及其相关

该文件夹中是开发板配套视频以及相应视频中用到的工具以及文档。

- 01-烧写、编译以及基础知识视频
  - 02-嵌入式 Linux 视频
  - 03-iTOP-4412 开发板硬件设计指导视频
  - 04-Android 应用程序视频
  - 06-裸机程序实验文档以及工具文件
  - 07-Linux-x86-Qt5.3.2 以及 QtE4.7.1 应用视频
- 注意：裸机程序没有视频，只有手册。

### 嵌入式学习推荐书籍及软件（第三方）

Altium Designer

嵌入式 ARM 推荐书籍

于博士 Allegro 视频

## 1.4 网盘压缩包 MD5 值的使用

用户在迅为的百度网盘中，可能需要下载到几个大的压缩包，目前大的压缩文件有“source4.4.4.7z”（Android4.4.4 的源码包）和“Ubuntu12.04.2\_V2.0.7z”（搭建好的 Ubuntu 镜像），以后网盘中增加大文件不再进行特殊说明。

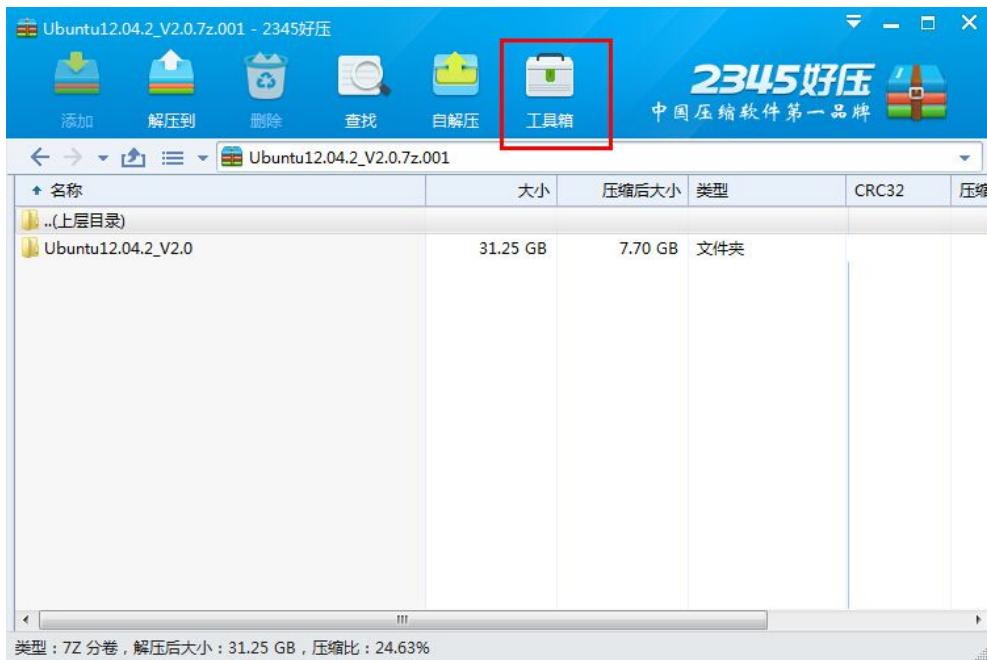
如果遇到无法解压的情况，这个时候如果全部重新下载，比较耗费时间，这里教大家一个简单的辨别方法，只需要找到 MD 值不对的压缩包，针对性的下载即可。

我们以“iTOP-4412 开发板搭建编译环境所需要的工具包以及补丁包”→“02-Ubuntu 软件”→“02-搭建好的 Ubuntu 镜像 V2.0”为例。

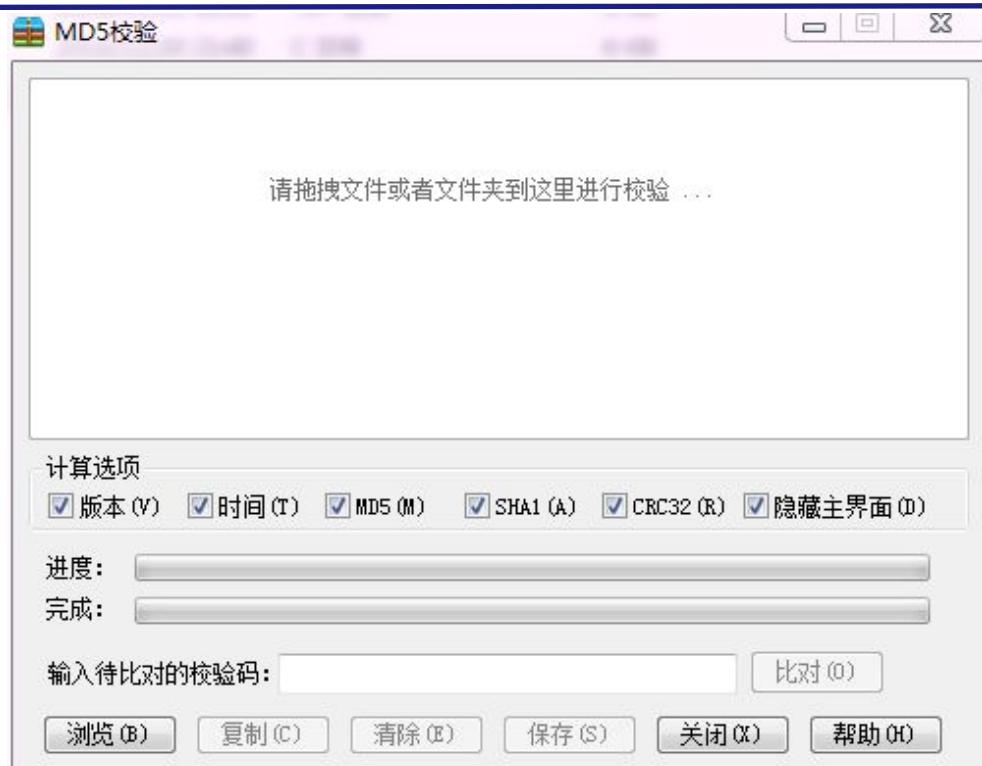
如下图所示，红框中就是包含 MD5 值的文本，蓝色框中就是大文件压缩包。

名称	修改日期	类型	大小
Read_Me.txt	2014/11/13 10:32	TXT 文件	1 KB
Ubuntu V2 MD5.c	2014/12/20 21:40	C 文件	6 KB
Ubuntu12.04.2_V2.0.7z.001	2014/11/13 1:47	好压 分卷 压缩文件	512,000 KB
Ubuntu12.04.2_V2.0.7z.002	2014/11/12 22:55	好压 分卷 压缩文件	512,000 KB
Ubuntu12.04.2_V2.0.7z.003	2014/11/13 1:47	好压 分卷 压缩文件	512,000 KB
Ubuntu12.04.2_V2.0.7z.004	2014/11/13 1:47	好压 分卷 压缩文件	512,000 KB
Ubuntu12.04.2_V2.0.7z.005	2014/11/13 1:47	好压 分卷 压缩文件	512,000 KB
Ubuntu12.04.2_V2.0.7z.006	2014/11/13 1:47	好压 分卷 压缩文件	512,000 KB
Ubuntu12.04.2_V2.0.7z.007	2014/11/13 1:47	好压 分卷 压缩文件	512,000 KB
Ubuntu12.04.2_V2.0.7z.008	2014/11/13 1:47	好压 分卷 压缩文件	512,000 KB
Ubuntu12.04.2_V2.0.7z.009	2014/11/13 1:47	好压 分卷 压缩文件	512,000 KB
Ubuntu12.04.2_V2.0.7z.010	2014/11/13 1:47	好压 分卷 压缩文件	512,000 KB
Ubuntu12.04.2_V2.0.7z.011	2014/11/13 1:47	好压 分卷 压缩文件	512,000 KB
Ubuntu12.04.2_V2.0.7z.012	2014/11/13 1:47	好压 分卷 压缩文件	512,000 KB
Ubuntu12.04.2_V2.0.7z.013	2014/11/13 1:47	好压 分卷 压缩文件	512,000 KB
Ubuntu12.04.2_V2.0.7z.014	2014/11/13 1:47	好压 分卷 压缩文件	512,000 KB
Ubuntu12.04.2_V2.0.7z.015	2014/11/13 1:47	好压 分卷 压缩文件	512,000 KB
Ubuntu12.04.2_V2.0.7z.016	2014/11/13 1:47	好压 分卷 压缩文件	394,321 KB

这里使用的是“2345 好压”软件，如果出现无法解压的情况，单击任意压缩包，右键选择用“2345 好压”打开。如下图，单击工具箱，选择“MD5 校验”。



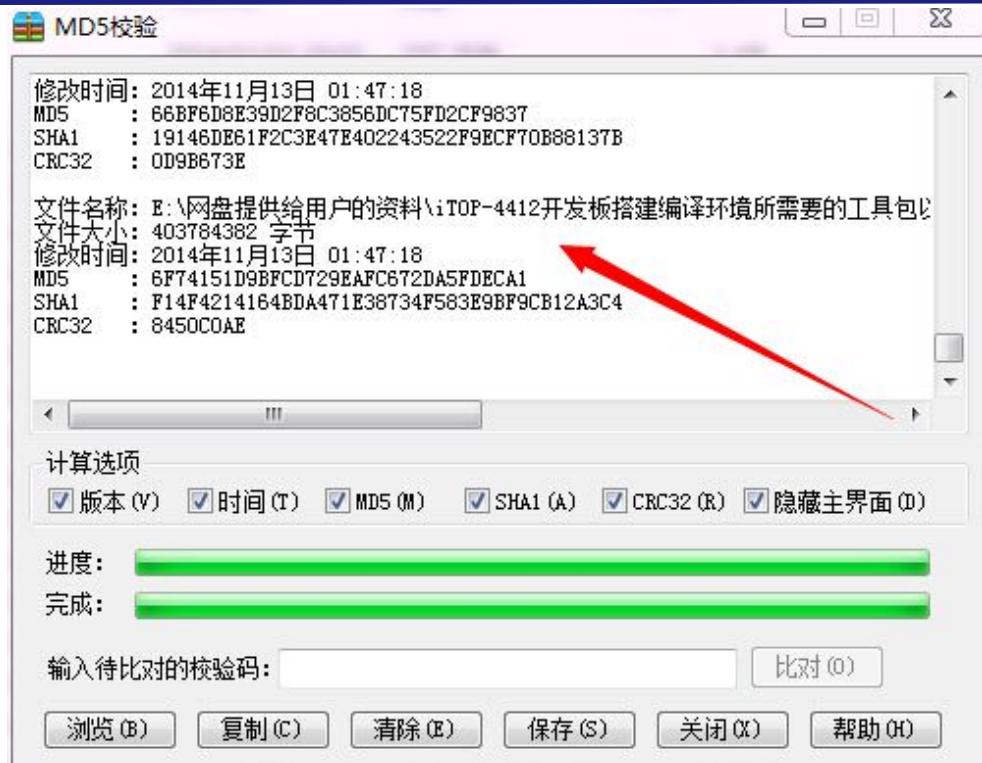
如下图，弹出 MD5 校验计算窗口，按照 2345 软件的提示，将下载的压缩包全部拖到其中。



如下图，计算中。



如下图所示，每个压缩包的 MD5 数值全部计算出来了，然后和网盘中的 MD5 对比一下，哪个压缩包的 MD5 数值不正确，则重新下载对应的压缩包。



## 1.5 使用 github 获取开发源码

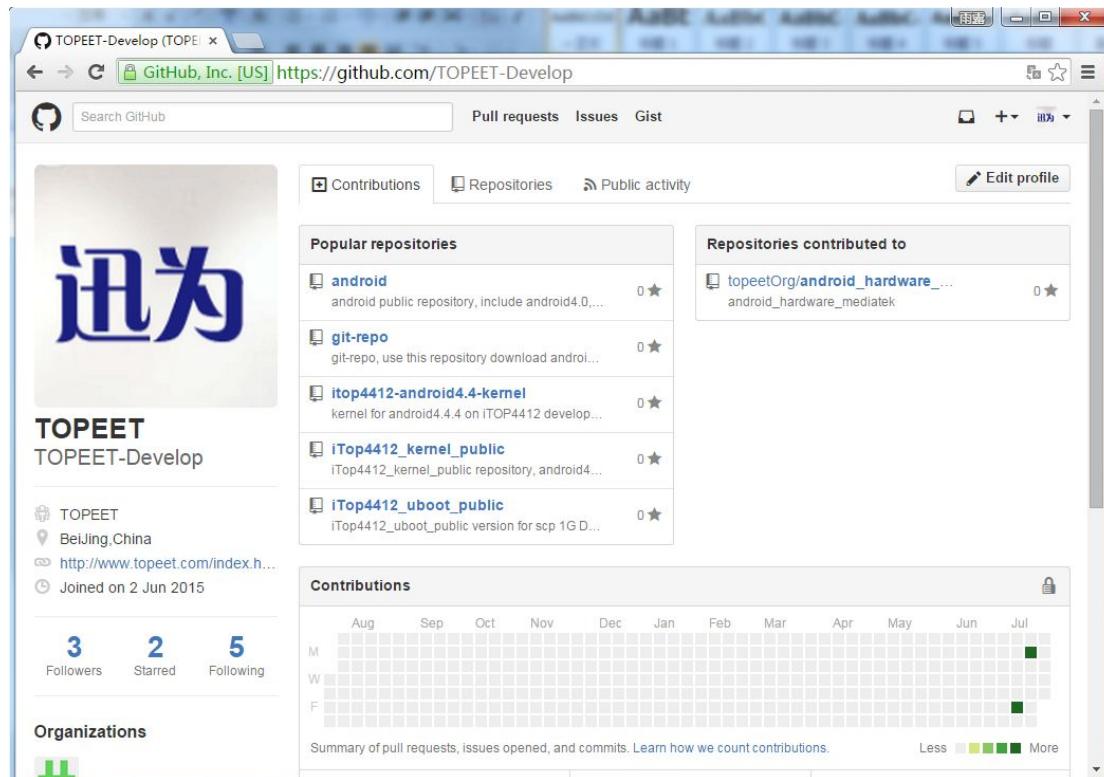
目前 iTOP4412 开发平台软件源码是通过网盘下载的方式提供给客户，每隔 3--4 个月的时间网盘内容会更新一次，采用网盘下载方式可以获取到稳定版本，但是由于发布周期较长，用户无法第一时间获取到版本最新状态，如我们解决的 Bug 和新增加的功能。故我们把 iTOP4412 的代码上传到了 GitHub 平台，通过此平台更新，同步源代码。GitHub 是我们的开发版本，里面仅提供源码下载，如需要获取二进制镜像，请在光盘或者网盘获取。

用户不需要 github 的账号既可以下载所有的源码，另外我们也会定期把项目源码发布到网盘，提供给用户下载。

**注意：**学习型用户使用光盘中的源码就足够了，等有了一定的经验之后再去下载最新的源码。

详细的 github 使用过程以及源码使用方法参考附录六

iTOP4412 项目在 GitHub 的主页地址：<https://github.com/TOPEET-Develop>



## 二 iTOP-4412 开发平台组装以及初体验

开发板是一个相对复杂的电子系统，请耐心按照本章说明组装，以免造成不必要的损失。

用户最好使用迅为提供的连接线，因为有些部件是专门定制的，可能和市场上购买的其它连接线和部件不匹配，擅自使用市场上购买的接线或者部件可能损伤开发板。

在完全弄清楚开发板接口信号定义之前，如果本文档中没有提到该部件和您在市场上所购买的部件兼容，不要擅自使用自己购买的接线和其它部件，如有疑问请咨询我司技术人员。

### 2.1 开发板的组装

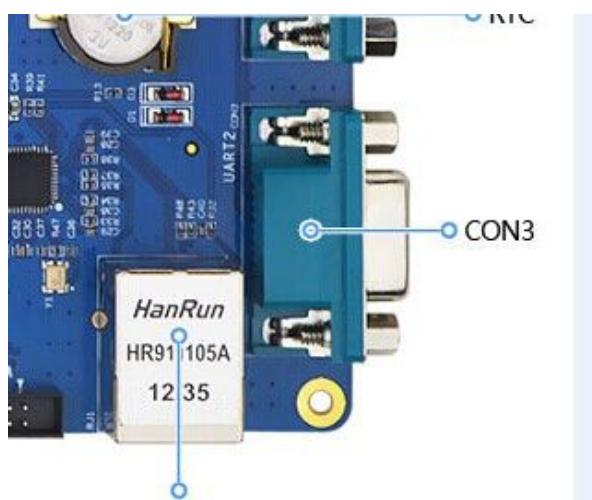
#### 2.1.1 控制台 ( Console ) 串口

使用串口线连接开发板的 COM3 到 PC 机的串口，如果 PC 或笔记本没有串口，就需要准备一条 USB 转串口的设备。

**注意：插拔串口，要在断电的情况下进行，以免带电插拔出现器件损坏。**

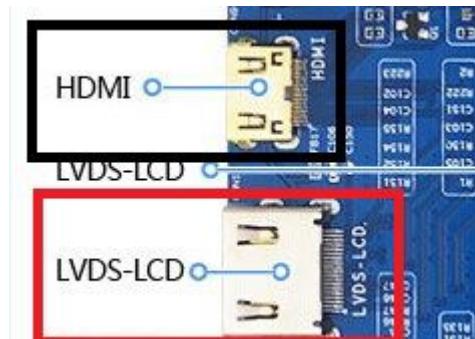
PC 上对串口的操作软件请参考“3.1 超级终端的安装和使用”。

Exynos 4412 共有四个串口，其中 CON3 是作为系统的调试串口，如下图所示：



## 2.1.2 屏幕的连接

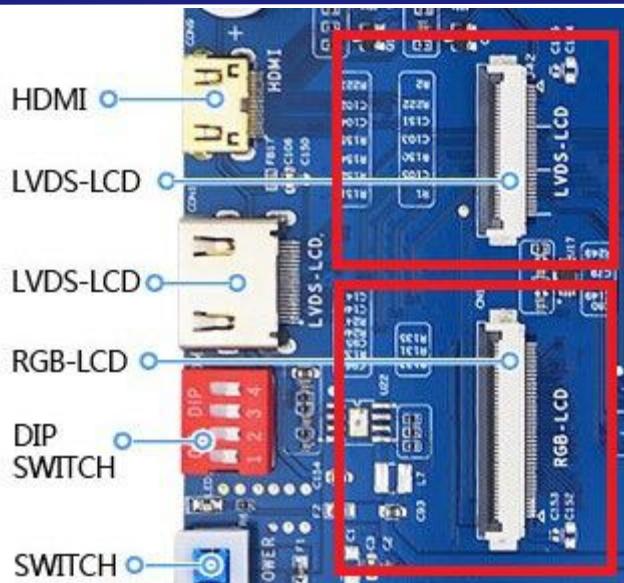
从外观上来看，开发板有 2 个 HDMI 接口，其实只有一个可以接到 HDMI 显示器上。如下图所示：



外形较大的 HDMI-A 接口（上图中红色方框内的接口），只能连接迅为提供的 7 寸屏幕或者 9.7 寸屏幕，里面有 5V（或者 3.3V）电源，绝对不能接到 HDMI 显示器上。使用迅为提供的 HDMI 线是可以防呆的，不会接错，在用户弄清楚信号之前，不要擅自使用自己购买的 HDMI 线！

外形较小的 HDMI（上图褐色方框内的接口）是标准的 HDMI-C 接口（不属于国际标准，但是在很多电器设备中都有使用，属于日本 SONY 公司定义的一种 HDMI 接口，具体可以百度），建议使用我司的 C 口转 A 口的 HDMI 线连接。

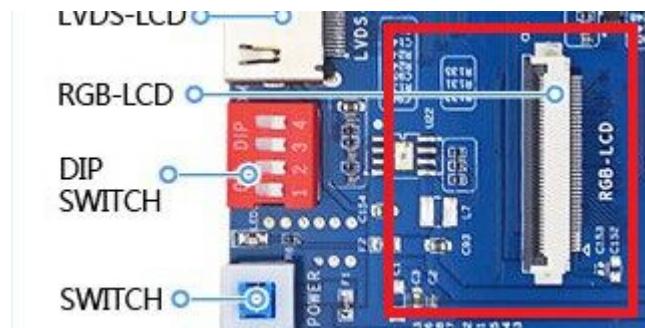
iTOP-4412 精英版除了使用 HDMI 线连接屏幕外，也可以通过用户平常使用的软排线的方式来连接，如下图所示：



上图中两个红色方框内的 LCD 接口信号是自定义的接口，只能连接迅为提供的屏幕，和其它公司的屏幕并不保证完全兼容，这点大家一定要注意！

### 2.1.2.1 电阻屏的连接 (4.3寸屏幕)

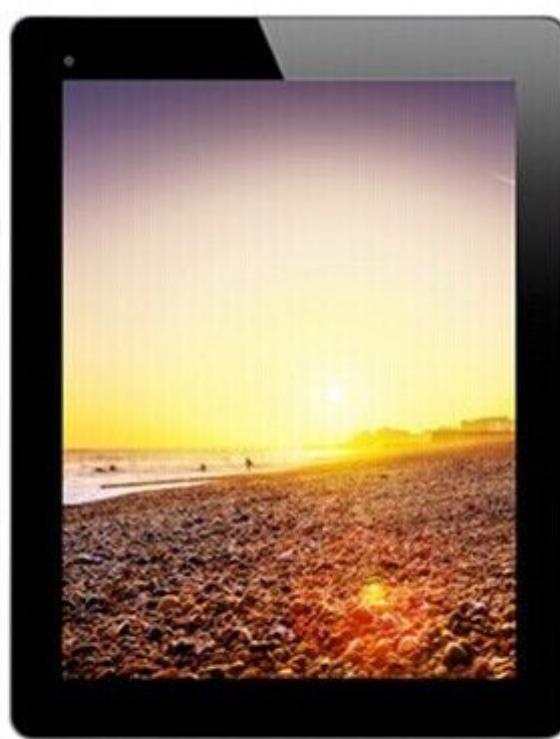
iTOP-4412 精英版可支持 4.3 寸电阻屏幕，连接方式如下图所示：



4.3 寸屏幕的接口是翻盖式的，软排线带有金属触点的一面朝下连接。

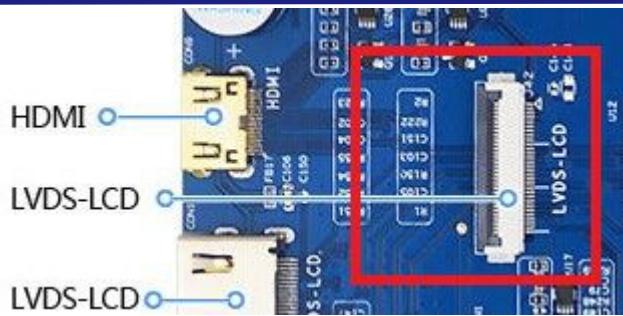
### 2.1.2.2 电容屏的连接 (7寸屏幕和9.7寸屏幕)

iTOP-4412 精英版可支持 7 寸或者 9.7 寸电容屏，如下图所示：



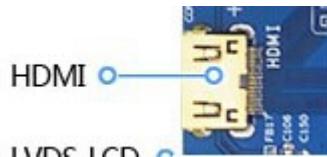
9.7寸IPS屏幕

7寸和9.7寸屏幕也可以通过软排线来连接，连接口如下图所示：



### 2.1.2.3 显示器 (HDMI) 的连接

HDMI 的接口如下图所示：



使用我司提供的 HDMI 线，将底板 HDMI-C 接口和显示器上的 HDMI-A 相连，再次提醒一定要使用我司提供的 HDMI 线！

如果使用带有 HDMI 接口的电脑显示器就没有了触摸功能，这样就需要连接 USB 鼠标和键盘。底板上的 USB 接口和电脑上一样的，内核里面自带驱动。同时，如果购买了我司的 LCD 屏幕，您也可以连接 USB 鼠标和键盘，触摸和鼠标键盘将同时有效。

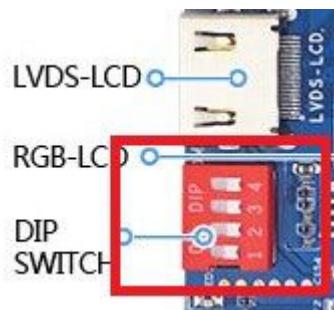
### 2.2.3 电源的连接

请使用开发板自带的 5V 电源连接开发板的 POWER 插座。如下图所示：



## 2.2 启动方式设置（拨码开关）

用户可以通过拨码开关来选择启动方式，注意，初次使用不要随意改变拨码开关设置，拨动后一定要注意恢复，如果设置不当将导致系统无法启动，拨码开关如下图所示：



精英版是 4 键的拨码开关，按照上图的放置位置，从上到下编号依次分别是 4,3,2,1。其中 3 和 4 是用来选择使用屏幕的，1 和 2 是用来选择启动方式的。按照上图的放置位置，拨码开关拨到左侧是 1，拨到右侧是 0。

选择启动方式的设置如下图所示：

拨码开关编号	1	2
EMMC 启动	0	1
TF 卡启动	1	0

选择支持屏幕的设置如下图所示：

拨码开关编号	3	4
9.7 寸屏幕 ( 1024*768 )	0	0
7 寸屏幕 ( 1280*800 )	0	1
4.3 寸屏幕 ( 480*272 )	1	0
1080P 分辨率	1	1

## 2.3 uboot 模式和文件系统模式

### 2.3.1 uboot 模式

在确认电源、串口等连接好以后，按下开发板上的 SWITCH 按键，启动开发板，这时开发板上 POWER 旁边的 LED 灯会点亮，在 PC 的串口上可以看到类似图 2-3.2 所示的系统启动的信息。图 2-3.1，是 u-boot 启动信息，读秒的过程中如果输入任何值，将进入 uboot 模式，在第四章讲解系统烧写方法时会用到该模式。

为了和 Ubuntu、Win7、XP 的命令行区分，在文档的后面，我们统一将“在超级终端的命令行里面输入命令”的过程叫做“进入超级终端的 uboot 模式，输入命令”。

```
raise: Signal # 8 caught
raise: Signal # 8 caught
MMC1: 0 MB
0 MB
*** warning - using default environment

In: serial
Out: serial
Err: serial
eMMC OPEN Success.!!
      !!!Notice!!!
!You must close eMMC boot Partition after all image writing!
!eMMC boot partition has continuity at image writing time.!
!So, Do not close boot partition, Before, all images is written.!

MMC read: dev # 0, block # 48, count 16 ...16 blocks read: OK
eMMC CLOSE Success.!!

Checking Boot Mode ... EMMC4.41
Hit any key to stop autoboot: 4
```

图 2-3.1

内核启动过程，如下图 2-3.2 所示：

```

serial-com1 - SecureCRT
File Edit View Options Transfer Script Tools Window Help
serial-com1 Enter host <Alt+R>
[ 4.931000] Registering SoC/GPSS emulation handler
[ 4.955000] DVFS : VDD_INT Voltage table set with 0 Group
[ 4.985000] Failed to init busfreq.
[ 4.995000] vdd_m1: incomplete constraints, leaving on
[ 4.990000] VDD18_CAM: incomplete constraints, leaving on
[ 5.000000] VDD42_2M: incomplete constraints, leaving on
[ 5.010000] VDD28_CAM: incomplete constraints, leaving on
[ 5.015000] DC3V_TP: incomplete constraints, leaving on
[ 5.020000] VDD10_MIP1: incomplete constraints, leaving on
[ 5.025000] VDD10_MIP1: incomplete constraints, leaving on
[ 5.030000] VDD18_2M: incomplete constraints, leaving on
[ 5.035000] VDD18_2M: range incomplete constraints, leaving on
[ 5.040000] vdd12_5m range incomplete constraints, leaving on
[ 5.045000] vdd_m12 range: incomplete constraints, leaving on
[ 5.050000] vdd_1mt range: incomplete constraints, leaving on
[ 5.060000] vdd_m12 range incomplete constraints, leaving on
[ 5.065000] vdd_m12 range incomplete constraints, leaving on
[ 5.070000] exynos-tmu exynos4210-tmu failed to get regulator:vdd_tmu
[ 5.075000] Input: gpi0-keys /devices/platform/gpi0-keys/input/input1
[ 5.075000]   ->ft5x05_ts_int1: reset-->
[ 5.075000]   ->ft5x05_ts_int2: reset-->
[ 5.335000] ft5x05_ts 3-0038: Firmware version 0x06
[ 5.335000] ft5x05_ts 3-0038: FocalTech ft5x05 TouchScreen initialized
[ 5.340000] ft5x05_ts 3-0038: FT5X05 touch screen driver vdd_tmu
[ 5.350000] platform exynos4210-tmu: Driver exynos-tmu Requests probe deferral
[ 5.350000] s3c-rtc s3c64xx-rtc: setting system clock to 2013-01-01 12:00:02 UTC (1357041602)
[ 5.350000] s3c-rtc s3c64xx-rtc: initialized with policy : DVFS_NR_BASED_HOTPLUG
[ 5.370000] ALSA device list:
[ 5.375000] #0: iTOP-4412-Audio
[ 5.380000] Framebuffer memory: 228K
[ 5.385000] intc (1) [proc/1.oom_adj] is deprecated, please use /proc/1.oom_score_adj instead.
[ 5.610000] EXT4-fs (mmcblk0p2): mounted filesystem with ordered data mode. opts: (null)
[ 5.635000] EXT4-fs (mmcblk0p3): warning: checksum reached, running e2fsck is recommended
[ 5.640000] EXT4-fs (mmcblk0p3): mounted filesystem with ordered data mode. opts: nomblk_io_submit,errors=panic
[ 5.650000] EXT4-fs (mmcblk0p3): mounted filesystem with ordered data mode. opts: nomblk_io_submit,errors=panic
[ 5.695000] EXT4-fs (mmcblk0p3): warning: checksum reached, running e2fsck is recommended
[ 5.700000] EXT4-fs (mmcblk0p3): mounted filesystem with ordered data mode. opts: nomblk_io_submit,errors=panic
[ 5.705000] EXT4-fs (mmcblk0p3): mounted filesystem with ordered data mode. opts: nomblk_io_submit,noauto_da_alloc,errors=panic
[ 5.790000] int: cannot find system/etc/install-recovery.sh, disabling 'flash_recovery'
[ 5.820000] android_usb: already disabled
[ 5.860000] android_usb: already disabled
[ 5.880000] adb_open
[ 5.885000] adb_bind_config
[ 5.885000] adb_bind_config
[ 5.885000] s3c-hsotg s3c-hsotg: PHY already ON
[ 5.905000] s3c-hsotg s3c-hsotg: phy 0: s3c-hsotg_inq: USBRST
shell@android: ~ $ [ 6.045000] warning: rfid: uses 32-bit capabilities (legacy support in use)
[ 7.605000] Mail: :::exynos4_result_of_asv : 2
[ 7.610000] Mail: :::exynos4_result_of_asv : 2
[ 7.615000] Mail: mail_dvfs[1].vol = 92500
[ 7.620000] Mail: mail_dvfs[2].vol = 100000
[ 7.625000] Mail: mail_dvfs[2].vol = 100000
[ 7.625000] Mail: mail_dvfs[3].vol = 105000
[ 7.630000] Mail: mail_dvfs[3].vol = 105000

```

图 2-3.2

### 2.3.2 文件系统模式

启动开发板，在 PC 机的超级终端上可以看到类似图 2-5.1 所示的系统启动的信息。

在开发板启动过程中，如果用户不做任何操作和控制，最后超级终端会出现如下图 2-3.3 的界面。如果看到下图红色方框中的内容，就表明超级终端进入了 Android 的文件系统模式。

```

[ 4.945000] Registering SMP/SMPB emulation handler
[ 4.950000] [!] Memory type not determined.
[ 4.950000] vdd_int: vdd_int voltage constraint set with 0 group
[ 4.985000] Failed to init busfreq
[ 4.985000] DVDD1: incomplete constraints, leaving on
[ 4.985000] DVDD2: incomplete constraints, leaving on
[ 5.000000] DVDA2_2M: incomplete constraints, leaving on
[ 5.005000] DVDA2_AF: incomplete constraints, leaving on
[ 5.020000] DVDB1: incomplete constraints, leaving on
[ 5.020000] DVDB2: incomplete constraints, leaving on
[ 5.030000] DVDB2_2M: incomplete constraints, leaving on
[ 5.035000] DVDT2S: range: incomplete constraints, leaving on
[ 5.035000] DVDT2S_2M: range: incomplete constraints, leaving on
[ 5.045000] vdd_micr2 range: incomplete constraints, leaving on
[ 5.055000] vdd_int range: incomplete constraints, leaving on
[ 5.065000] vdd_arm range: incomplete constraints, leaving on
[ 5.085000] vdd_sd range: incomplete constraints, leaving on
[ 5.070000] exynos-tmu exynos4210-tmu: Failed to get regulator:vdd_tmu
[ 5.075000] platform exynos4210-tmu: driver exynos-tmu requests probe deferral
[ 5.075000] ftsx0t2s: ftsx0t2s: detected gpios gpio-keys/input/input1
[ 5.075000] =>ftsx0t2s_init: reset=-
[ 5.255000] Input: ftsx0t2s at /device/virtual/input/input2
[ 5.255000] ftsx0t2s: FocalTech ftsx0t2s TouchScreen initialized
[ 5.335000] ftsx0t2s: 3-0383: FocalTech ftsx0t2s TouchScreen initialized
[ 5.345000] exynos-tmu exynos4210-tmu: Failed to get regulator:vdd_tmu
[ 5.355000] platform exynos4210-tmu: driver exynos-tmu requests probe deferral
[ 5.355000] s3c-rtc s3c84xx-rtc: setting system clock to 2013-01-01 12:00:02 UTC (1357041602)
[ 5.365000] hotplug_policy_intt: initialised with policy : DVS_NR_BASED_HOTPLUG
[ 5.375000] s3c-usb-ehci: #0: ITOP-4412-Audio
[ 5.380000] Freeing init memory: 228k
[ 5.385000] android_usb: already disabled
[ 5.385000] android_usb: already disabled
[ 5.880000] android_usb: already disabled
[ 5.880000] adb open

```

图 2-3.3

当超级终端进入了文件系统模式，向串口中输入回车键，就会出现下图 2-3.4 中红色方框中的命令行终端。

```

[ 5.800715] ADB open:/system/bin/sh: No controlling tty (open /dev/tty: No such device or address)
/system
[ 5.920324] [WMT-DEV][I]WMT_open:major 190 minor 0 (pid 1254)
[ 5.926232] [WMT-DEV][I]WMT_open:1st call (400)
[ 5.929888] [mtk_wcn_stp_set_if_tx_type] set STP_IP_TX to UART.
[ 5.936609] [WMT-LIB][I]iwmt lib_set_hif:new hifType:0, fcCtrl:0, baud:921600, fm:2
[ 5.943501] [WMT-C][I]opfunc_hif_conf:WMI HIF info added
m/bin/sh: warning: won't have full job control
root@androi1: # [ 6.068542] warning: 'rild' uses 32-bit capabilities (legacy support in use)
[ 6.538917] s3c-fimc3: FIMC3 1 opened.
[ 7.072696] s3cfb s3cfb.0: [fb0] dma: 0x8eff4000, cpu: 0xf0dff000, size: 0x00600000
[ 7.086620] s3cfb s3cfb.0: [fb1] dma: 0x694f4000, cpu: 0xf1400000, size: 0x00600000
[ 7.101428] s3cfb s3cfb.0: [fb2] dma: 0x69504000, cpu: 0xf1400000, size: 0x00600000
[ 7.101428] s3c-fimc3: FIMC3 2 opened.
[ 8.240051] Mali: :::exynos_result_of_asv : 2
[ 8.242928] Mali: mail_dvfs[0].vol = 900000
[ 8.247116] Mali: :::exynos_result_of_asv : 2
[ 8.251452] Mali: mail_dvfs[1].vol = 925000
[ 8.255618] Mali: :::exynos_result_of_asv : 2
[ 8.259941] Mali: mail_dvfs[2].vol = 1000000
[ 8.264194] Mali: :::exynos_result_of_asv : 2
[ 8.268663] Mali: mail_dvfs[3].vol = 1050000
[ 14.121172] request_suspend_state: wakeup (3->0) at 14121163049 (2012-01-01 00:00:11.217349421 UTC)
[ 14.333316] acc_open
[ 14.334028] acc_release
[ 21.469575] CPU2: shutdown
[ 21.967264] CPU3: shutdown

```

图 2-3.4

为了和 Ubuntu、Win7、XP 以及 2.3.1 小节的 uboot 模式的命令行区分，在文档的后面，我们统一将“在超级终端的命令行里面输入命令”的过程叫做“进入超级终端的文件系统模式，输入命令”。在文件系统模式中，可以支持一部分常见的的 Linux 命令。

另外，在“Linux-QT 系统”、“Ubuntu 系统”以及“最小 Linux 文件系统”中，都可以进入文件系统模式，输入命令，对开发板进行操作。

## 2.4 iTOP-4412 开发平台初体验

iTOP-4412 开发板预装 Android4.0.3 系统，采用 9.7 寸（或者 7 寸或者 4.3 寸）IPS 屏幕，至少 5 点以上触控，操作流畅，无论是高清视频、游戏等都会有上佳的表现，实际操作感受超过市面多数平板电脑。

本章主要介绍 Android4.0.3 系统操作应用实例。部分功能需要相应配套硬件，烧写对应的镜像才能测试。第四章会讲解系统镜像的烧写方法。

### 2.4.1 系统基本功能

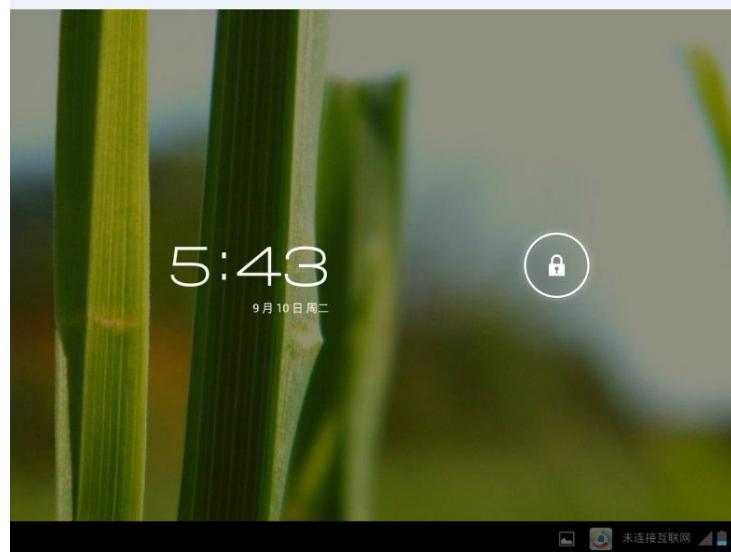
#### 2.4.1.1 开机

开发板接通电源，并按下电源开关，系统即启动，在启动过程中，系统会依次显示下图中的开机画面，它们分别是 Linux 内核和 Android 系统启动时的 Logo 画面：





最后会显示如下解锁画面：

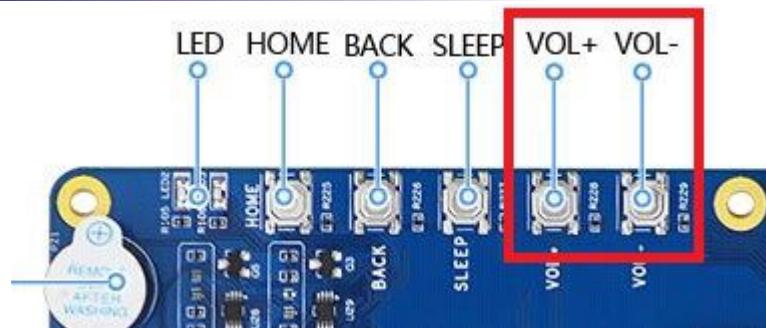


#### 2.4.1.2 音量调节

同样在系统‘设置’里进行操作，如下图所示：



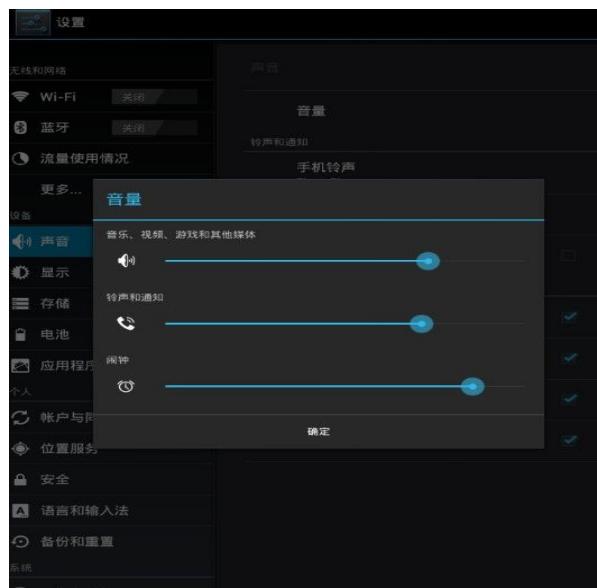
用户也可以通过开发板上的按键，Vol+和Vol-来调节音量。如下图所示：



### 2.4.1.3 亮度调节

操作方法和您使用手机或者平板的方法基本一样，通过按键来控制，如上图。

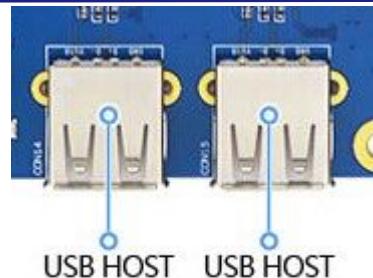
对于‘暴风影音’等视频播放软件，可以通过在屏幕上滑动等方式调节亮度；除此之外，在Android中可以通过系统‘设置’来调节亮度，如下图所示：



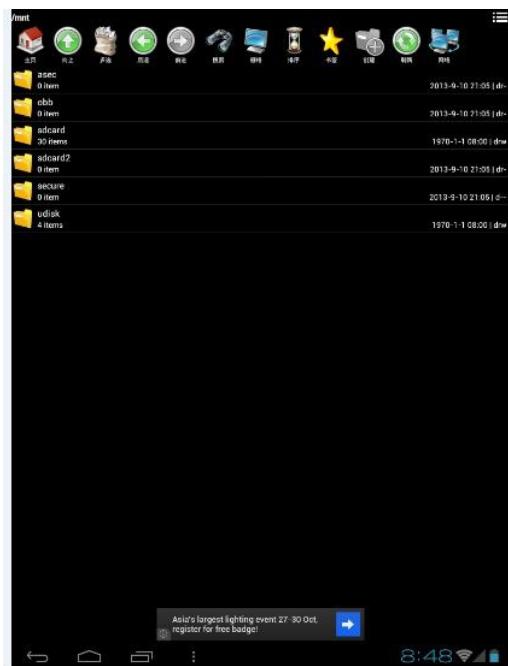
### 2.4.2 USB 和 OTG 功能

#### 2.4.2.1 连接 U 盘

当插入 U 盘以后（底板上 2 个 USB 接口都可以插入 U 盘），如下图所示：



打开桌面上的文件管理器，在“目录”“/mnt/udisk\*”下面将会出现U盘文件，如图所示：注意文件管理器是一个第三方软件，并没有直接安装，用户可以到网盘下载。



#### 2.4.2.2 鼠标及键盘

USB2.0 鼠标和键盘都可以支持。

在 USB HOST 接口连接鼠标或键盘，Android 系统会自动识别，对于没有购买触摸屏，使用 HDMI 显示器的用户，这是个很好的选择。

### 2.4.2.3 将开发板当做平板与 PC 相连

将开发板的 OTG 接口和 PC 机的 USB 接口相连，在 Android 桌面的右下部分状态栏将出现相应提示，如下图所示：



点击该提示后将会出现如下图所示：



继续点击按钮 ‘打开 USB 存储设备’ ，在电脑 PC 上就会提示有存储设备插入，这时用户就可以把平板当做 U 盘来操作了。

后面的章节将要讲到的 USB 方式烧写，也会用到 OTG 接口。

## 2.4.3 网络设置和连接

### 2.4.3.1 WIFI 连接 (选配)

默认不支持。需要连接开发板配套 WIFI 模块才能使用。

WIFI 功能需要在系统 ‘设置’ 里打开，如下图所示：

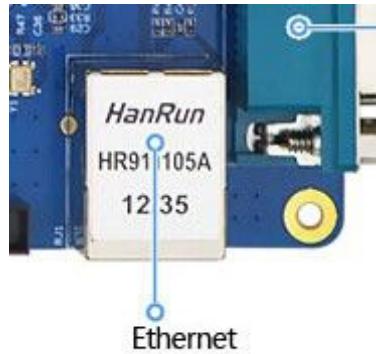


打开 WIFI 后，会在屏幕右边搜索到附近的无线网络，点击属于自己的无线网络，并输入密码，如下图所示：



### 2.4.3.2 有线网 ( RJ45 )

默认支持。需要通过一根网线与 PC 或路由器相连。



通过一根网线，连接这个 RJ45 接口到 PC 机的网口或者路由器，或者到另外一个开发板，可实现有线以太网络通讯。网线连接以后，两台机器的 IP 地址要设置在同一个网段，即同一个子网，这样双方就可以通讯了：

至此，上层应用可以通过 socket 协议实现端到端的数据通讯，完成各种网络应用。

网络接口芯片采用 DM9621, 其中已经包含了 MAC 和 PHY 部分。

下面是设置 PC 机和开发板以太网的例子：

- 1)设置 PC 主机 IPV4 地址为 192.168.1.2 ( IP 地址可以根据实际情况选择 )
- 2)连接开发板和主机的串口 , 网口 , 启动开发板 , 系统启动后 , 设置 Android 的 IP 地址和主机在同一个网段 , 例如 , 在串口中输入 :

```
ifconfig eth0 192.168.1.130 ( IP 地址可以根据实际情况自行确定 )
```

测试以太网连接 , 在串口中输入 :

```
ping 192.168.1.2 ( 开发板 ping PC )
```

或者在 PC 的 cmd 窗口中输入 :

```
ping 192.168.1.130 ( PC ping 开发板 )
```

然后可以看到 ping 命令执行的结果。

开发板连接路由器例子 :

```
ifconfig eth0 192.168.1.130 netmask 255.255.255.0 up  
route add default gw 192.168.1.1 dev eth0  
setprop net.dns1 192.168.1.1
```

#### 2.4.3.3 浏览网页

参考本章使用 RJ45 或者 WIFI 连接网络后就可以上网了。

如下图所示 :

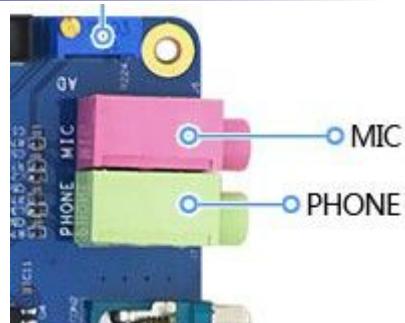


## 2.4.4 多媒体

### 2.4.4.1 电影音乐

默认支持。需要使用一个 TF 卡或者通过网络在线视频。

拷贝视频文件到 TF 卡后，将卡插入开发板的 TF 卡插座，使用暴风影音播放器，就可以观看影片（具体方法参考您智能手机 TF 卡或者平板 TF 卡使用方法）。这时需要连接耳机，注意不同颜色的接口分别对应耳机的听筒和麦克，如下图所示：



在线影音需要上网，参考本章 2.4.3.1 小节中的 WIFI 连接。

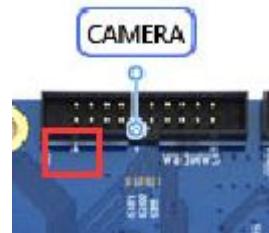
播放视频，如下图所示：



#### 2.4.4.2 摄像头（选配）

开发板默认支持摄像头接口。

摄像头接口是一个 20PIN 的双排插针，注意防呆箭头，可以和迅为提供的 500M 摄像头模块相连接，如下图所示：



摄像头模块外观，如下图所示：



#### 2.4.4.3 声卡的内外放设置

这实际上就是耳机和喇叭输出的选择切换，可以通过命令行来实现。用户在超级终端中输入以下命令即可实现内放和外放的切换。（如果对超级终端的命令不熟悉，可以在了解了第三章之后再来手动测试）

##### 1 ) 使用耳机输出

现在程序默认使用的耳机，使用耳机输入下面的命令：

tinymix 4 127

tinymix 5 1

tinymix 39 1

tinymix 46 1

2 ) 使用外置的喇叭 :

在串口输入以下命令 :

tinymix 6 127

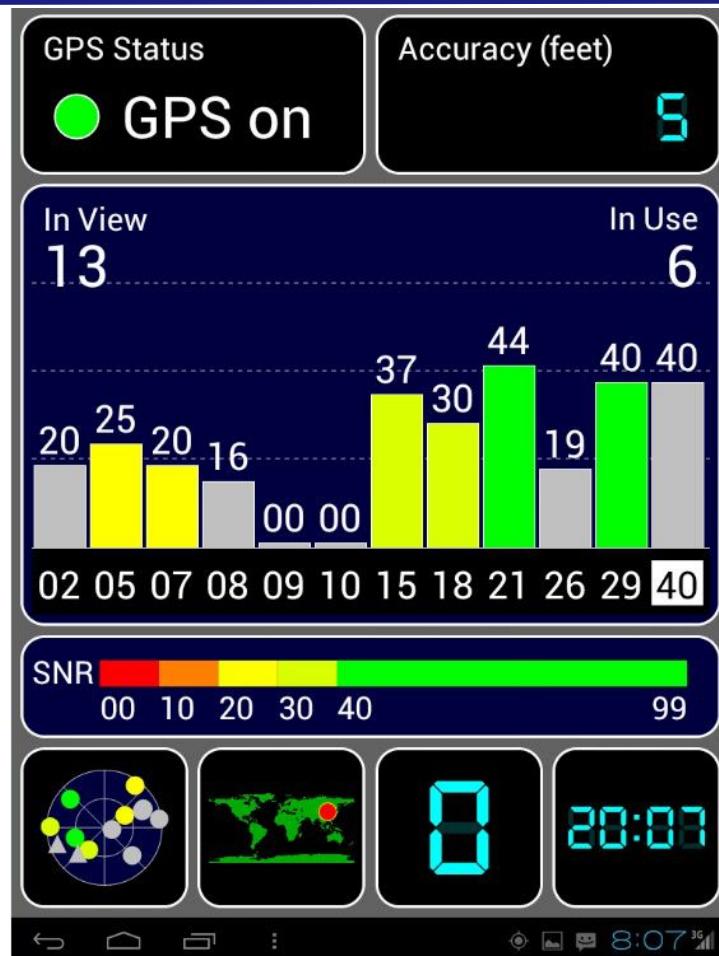
tinymix 7 1

tinymix 39 1

tinymix 44 1

## 2.4.5 GPS 功能 ( 选配 )

GPS 测试需要到室外空旷的地段 , 测试前应装上对应的 GPS 天线 , 需要安装一个第三方的 GpsTest.apk , 然后点击运行 , 查看搜星结果 ; 如果安装凯立德等软件 , 可实现 GPS 导航等功能 , 搜星速度也足够快 , 十分好用 , 如下图所示 :



以上为室内靠窗测试效果图，室外可以达到 10 颗星以上。

## 2.4.6 游戏 3D 性能

用户可以使用 TF 卡或者 WIFI 安装相应 3D 测试软件。

Exynos 4412 配备四核 GPU，即 ‘Mali-400’，性能强劲，各种游戏测试无压力，运行游戏效果，如下图所示：



#### 2.4.7 蓝牙功能 (选配)

迅为的 wifi 模块和蓝牙模块是一体的，因为该模块同时支持 WIFI 和蓝牙，可以使用蓝牙功能和其他支持蓝牙的设备进行文件传送。

## 三 iTOP-4412 平台基础软件的安装和学习

为了方便大家学习和开发，迅为电子给用户提供了一套完整的开发环境搭建方法。其中，包括超级终端、虚拟机、Ubuntu、Vim 编辑器、Source Insight 以及 Android ADB。

如果用户以前没有接触过嵌入式 Linux 开发，最好使用迅为电子提供的整套软件和方法，这些软件都可以在网盘中找到。

### 3.1 超级终端的安装和使用

#### 3.1.1 安装 USB 转串口驱动

如果用户将开发板的串口 COM3 和 PC 机的串口直接相接，那么只需要装超级终端软件，不需要安装 USB 转串口驱动。

如果用户使用的是笔记本电脑，或者是没有串口的 PC 机，那么就需要使用 USB 转串口线来连接开发板和 PC 了，则需要安装 USB 转串口驱动。

在网盘中，用户可以下载 USB 转串口驱动的压缩包"迅为 usb 转串口驱动.zip "（针对迅为电子的 USB 转串口线驱动），该驱动是 PL2303 的驱动，在 Win7-64 位操作系统下测试无误。驱动软件在 “iTOP-4412 开发板所需 PC 软件（工具）” → “01-USB 转串口（PL2302 驱动）” 目录。

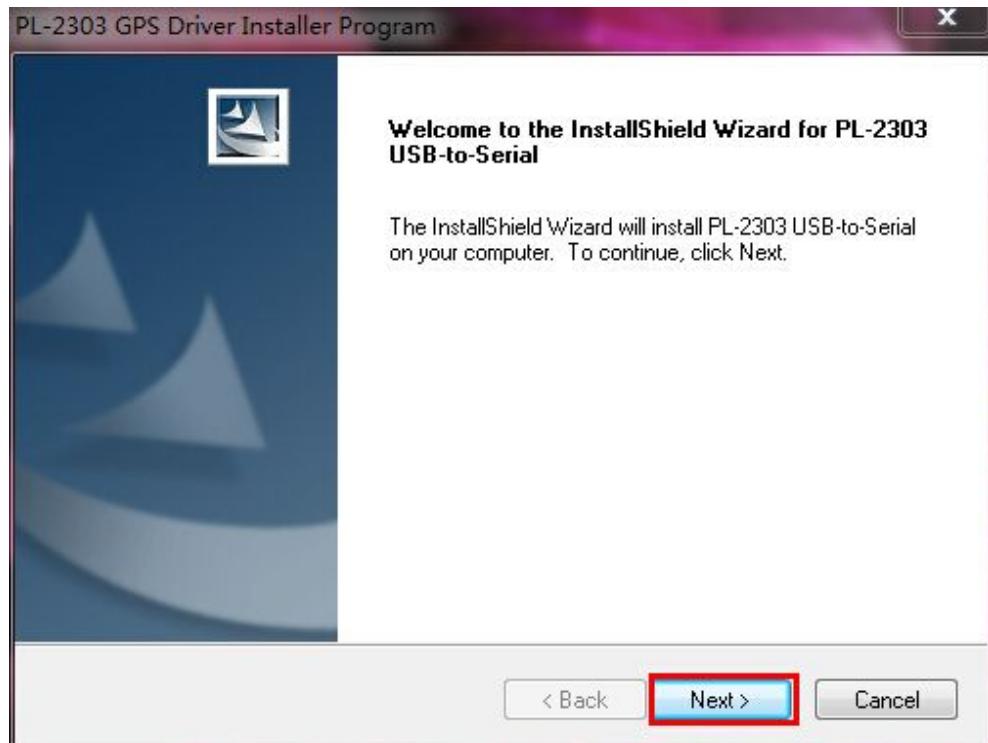
下面详细的讲解一下，在 PC 机上，USB 转串口驱动的安装。

1 ) 解压 “迅为 usb 转串口驱动.zip” 压缩包得到文件

“PL2303\_Prolific\_xunwei.exe” , 如下图 , 然后双击文 “PL2303\_Prolific\_xunwei.exe” ,  
开始安装 USB 转串口驱动。



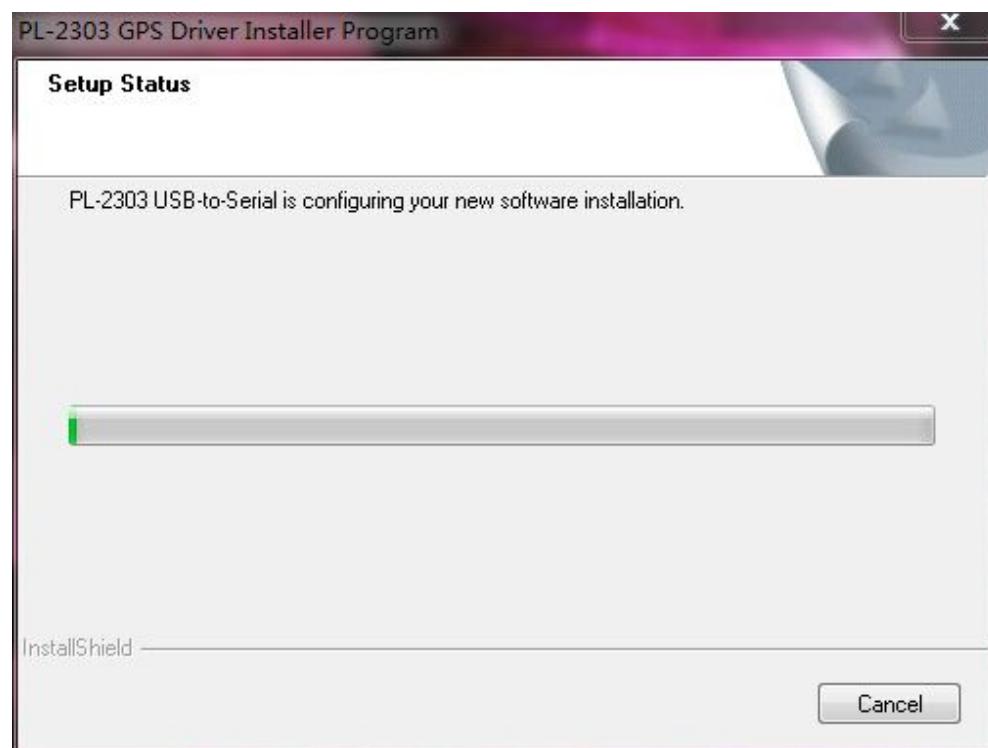
2 ) 如下图 , 单击 “Next” , 继续安装。



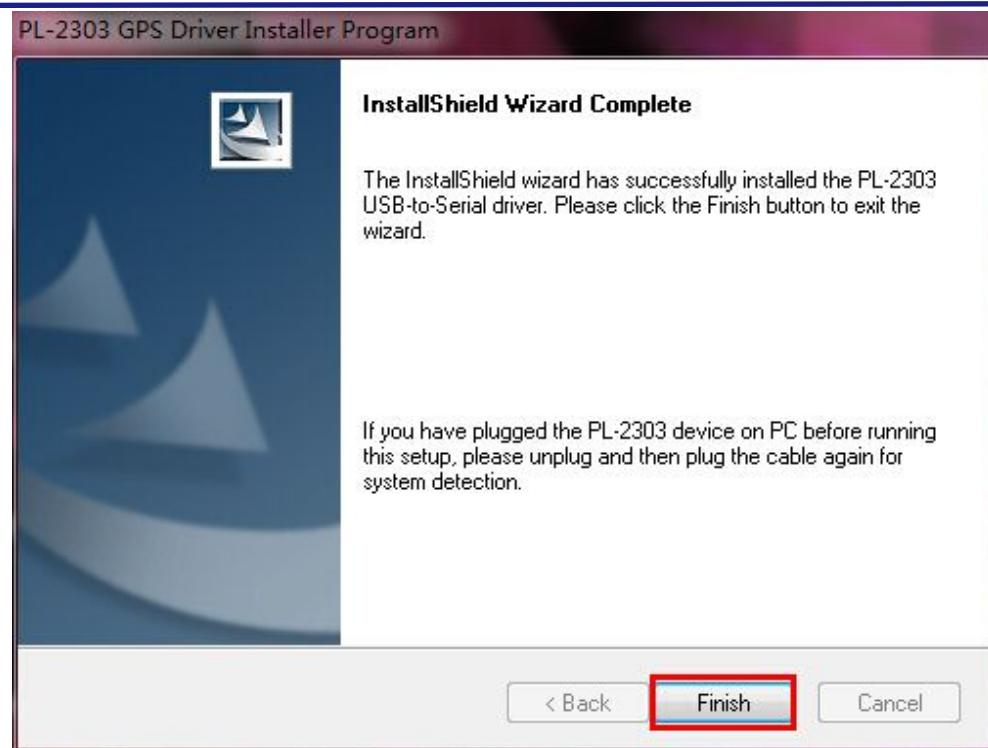
3 ) 如下图 , 选择接受安装协议 , 然后单击 “Next” , 继续安装。



4 ) 如下图 , USB 转串口的驱动在安装中。



5 ) 如下图 , 提示驱动的安装已经完成 , 单击 “Finish” , 结束 USB 转串口驱动安装。

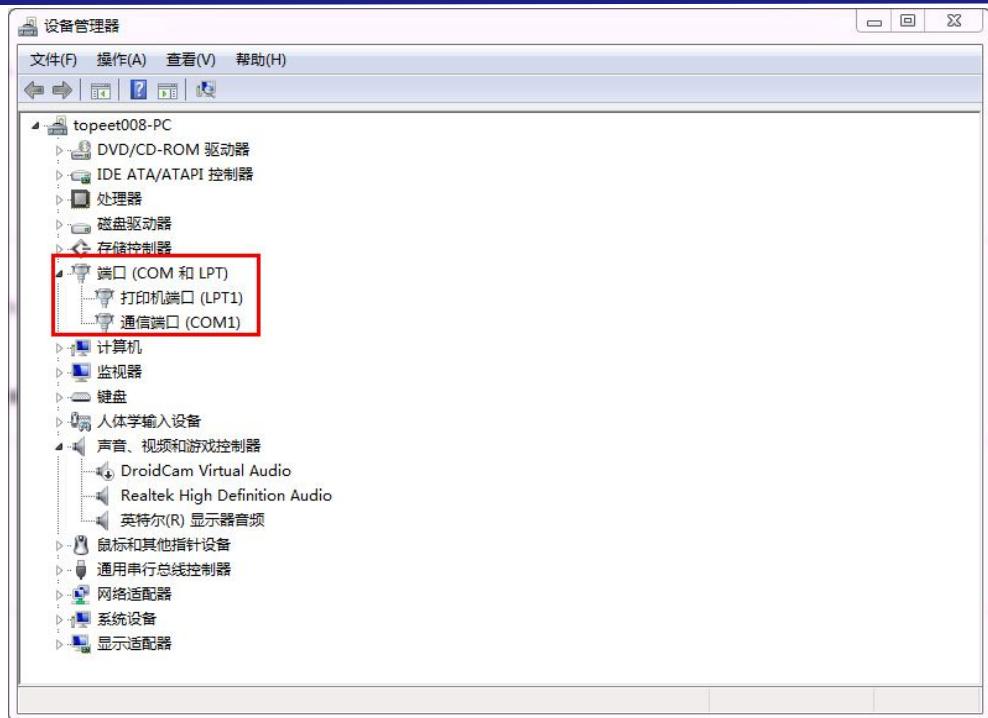


然后检查一下 USB 转串口的驱动安装是否正确。

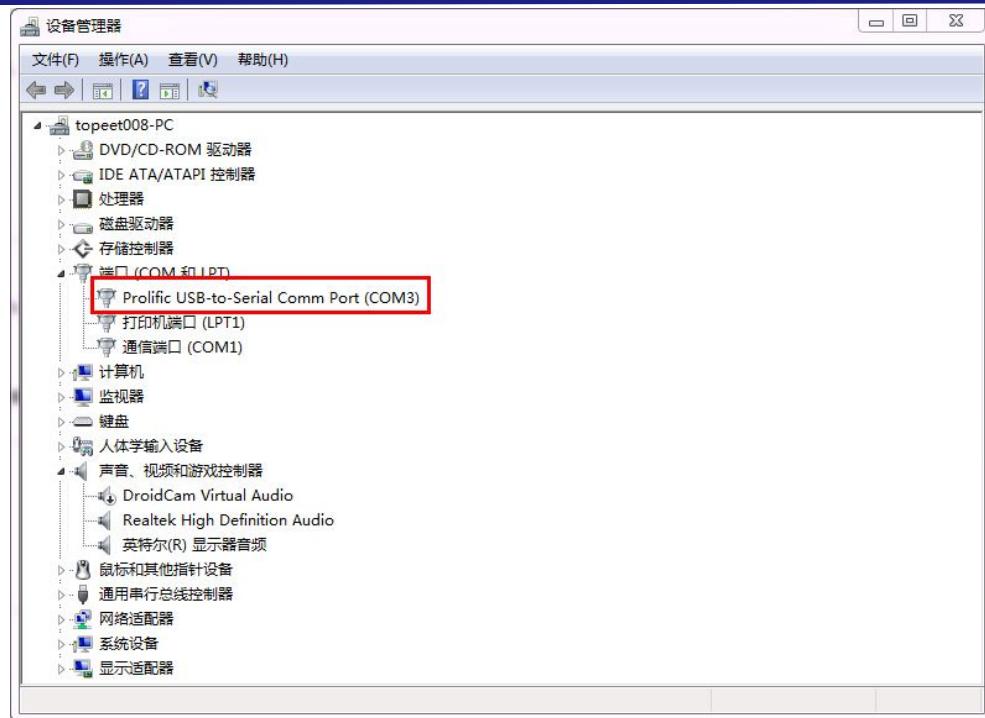
以作者的机器为例，下面的截图可能和用户的界面稍微有点不同。

6 ) 用户先不要把 USB 转串口线和 PC 机相连，直接进入 Win7-64 位操作系统的设备管理器，如下图，打开“端口”。

如下图，红色矩形框中显示机器只有一个串口 COM1, 作者的机器自带串口，所以有 COM1。



7 ) 然后将 USB 转串口线连接电脑的 USB 接口。设备管理器显示如下图所示，将下图和上图对比，可以发现下图中多出了红色矩形框中的内容，其中 COM3 编号就是接到电脑 USB 后，系统给你分配的串口号。 ( 这里需要提醒一下，用户插入的 USB 接口不一样，显示的端口号也会不一样，大家只需要关注 USB 转串口插入后，增加的那个串口号，这个增加的串口号需要大家记住，在超级终端的设置中会用到。 )



到这里，USB 转串口线的驱动就安装完成了。

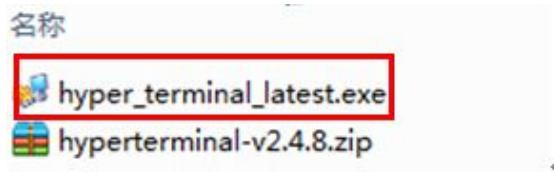
### 3.1.2 超级终端的安装

超级终端，也叫串口软件或者串口助手。迅为提供的超级终端安装包为“hyperterminal-v2.4.8.zip”，该超级终端在Win7-64位操作系统测试可用。

用户可以在网盘目录“iTOP-4412 开发板所需PC软件（工具）”→“02-超级终端（串口调试助手）”下载该安装包。

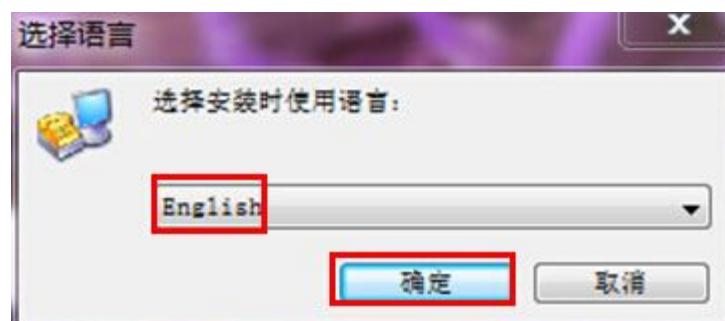
下面详细讲解一下在PC机上超级终端的安装。

1) 如下图所示，解压“hyperterminal-v2.4.8.zip”安装包后会生成软件“hyper\_terminal\_latest.exe”，然后双击软件“hyper\_terminal\_latest.exe”，开始安装

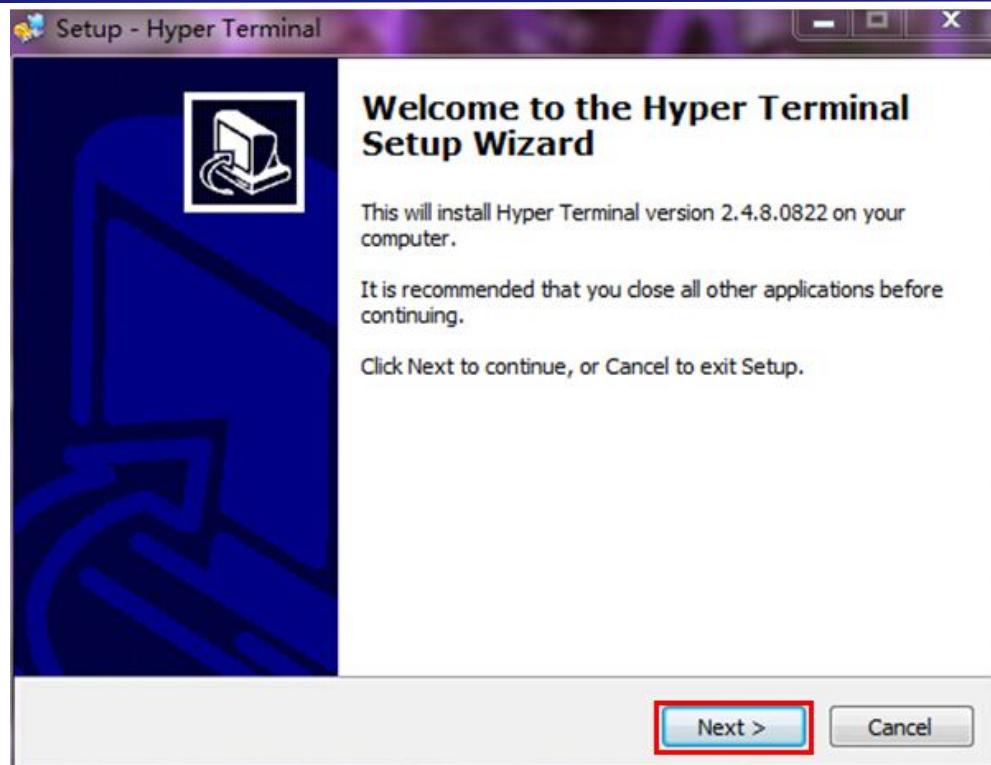


2) 如下图所示，语言版本，选择“English”，这里需要注意的是，语言选择为“English”，在超级终端安装完毕后，菜单选项可能仍然是中文，这并不影响使用。另外，在所有嵌入式软件的安装过程中，在选择语言的时候尽量全部选择 English。

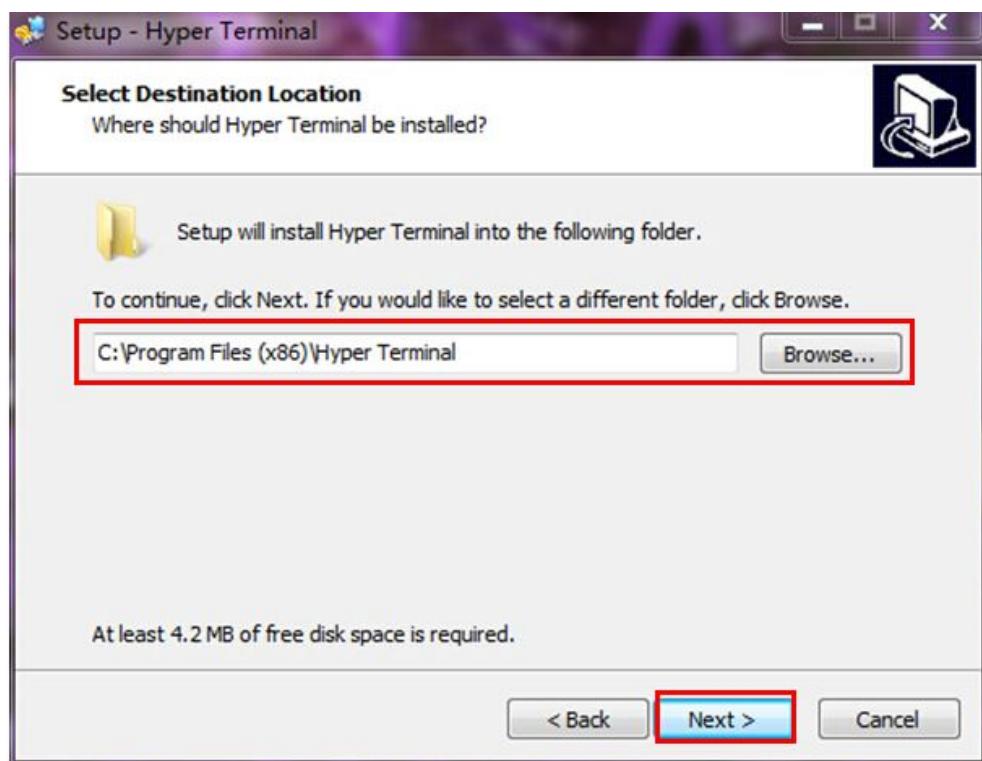
单击“确定”，继续安装。



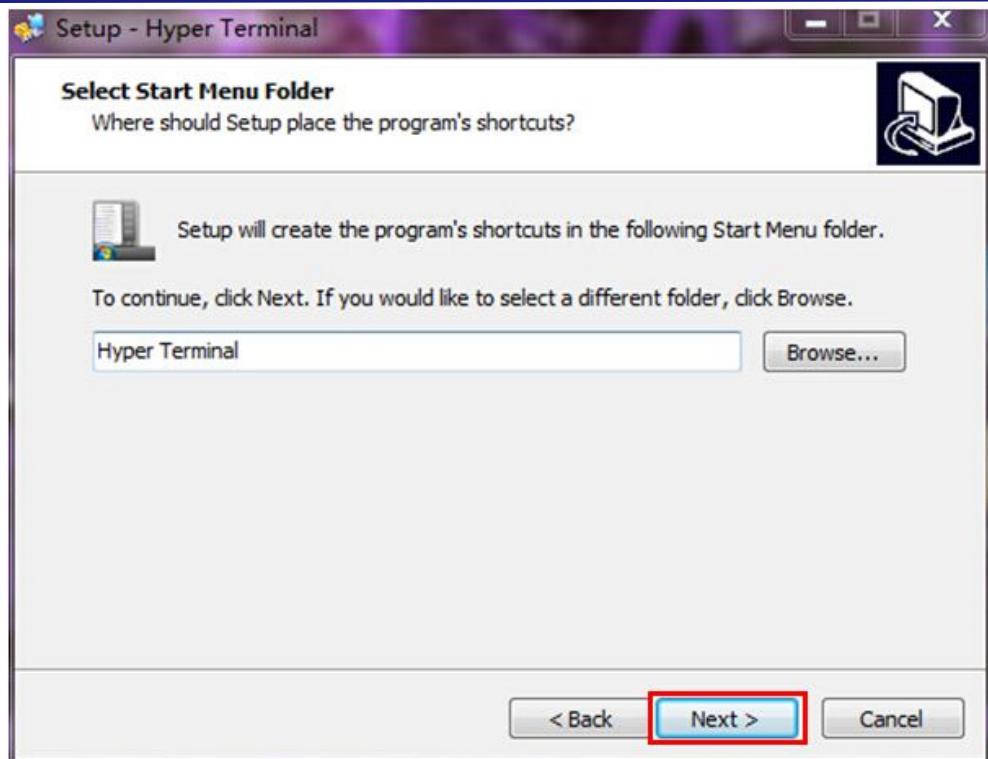
3) 如下图，单击 Next，继续安装。



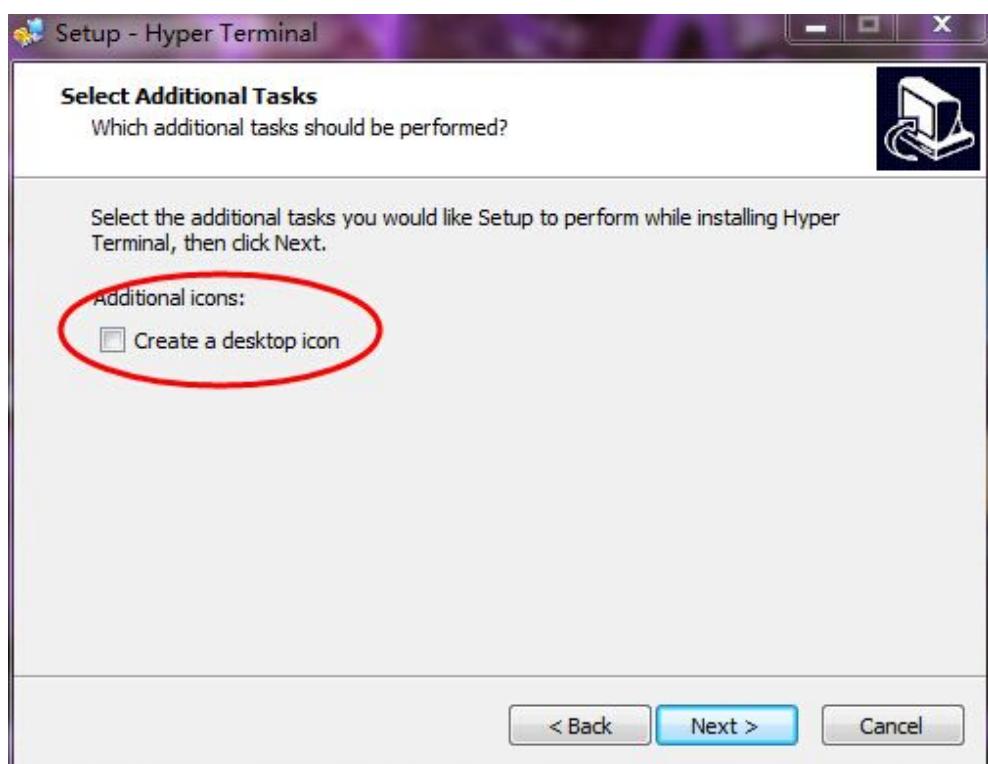
4 ) 如下图 , 选择安装路径 , 可以选择默认 , 单击 “Next” , 继续安装。



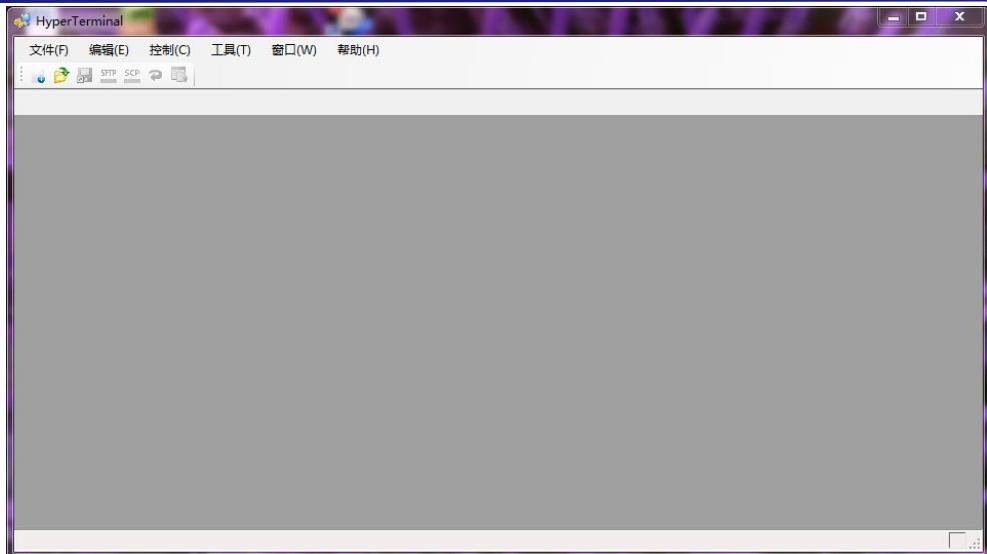
5 ) 如下图 , 选择默认设置 , 然后单击 “Next” , 继续安装。



6 ) 如下图 , 椭圆框中的选项 , 可选可不选 , 如果选择了 , 安装完毕之后 , 会在在桌面创建快捷方式 , 单击 “Next” , 继续安装。



8 ) 如下图 , 出现下图所示界面 , 表明安装完成。

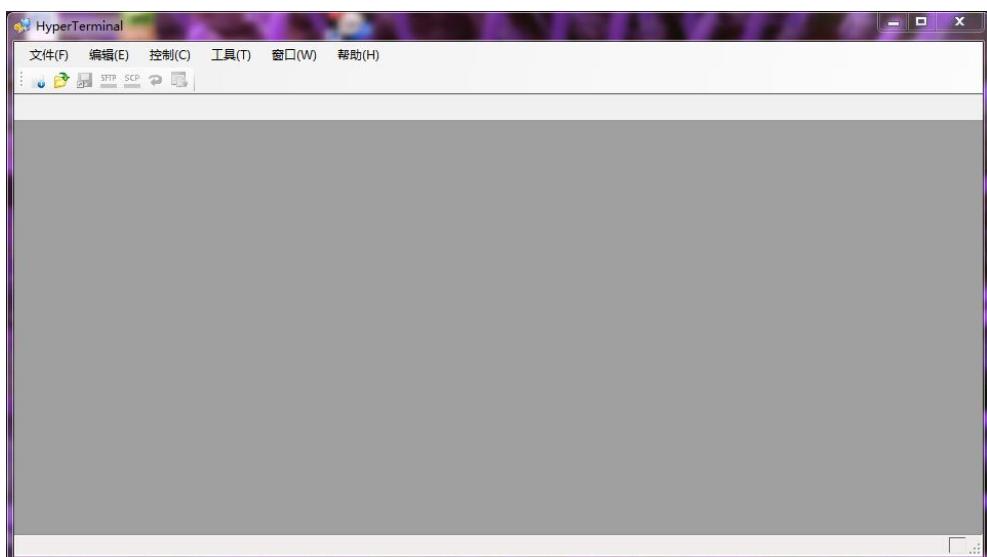


### 3.1.3 超级终端的设置

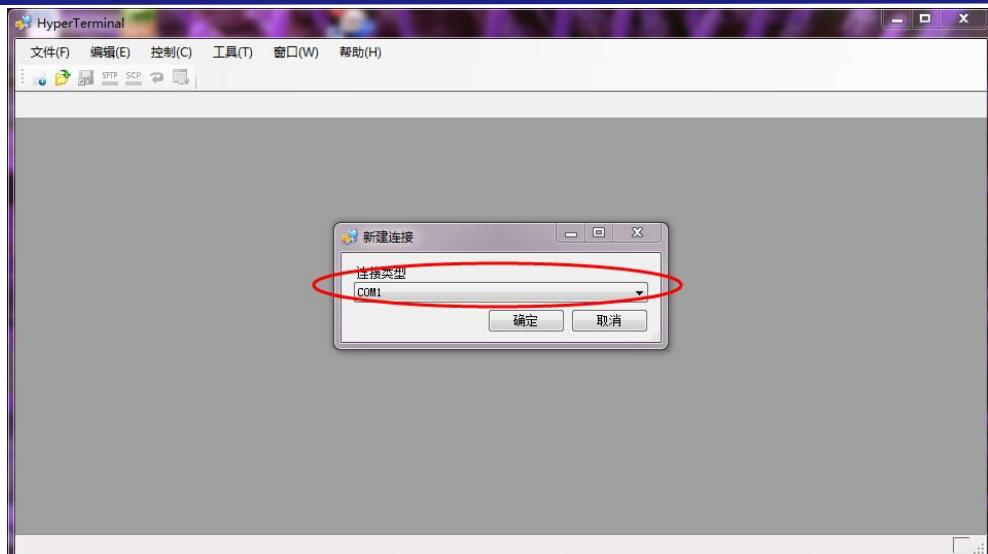
超级终端和开发板连接前，需要进行设置。

下面详细讲解一下在 PC 机上超级终端的设置。

1 ) 打开前一小节安装的软件 “Hyper Terminal ” ，如下图所示。



2 ) 然后执行菜单命令 “文件(F)” --> “新建连接(N)” ，出现下图界面。这里需要注意的是，下面的 COM1，用户应该根据自己机器的实际情况选用串口号，如果不清楚可以参考 3.1.1 小节。单击 “确认” ，继续设置。



3 ) 出现下图所示的配置界面 , 用户参考下图进行设置。



4 ) 如下图 , 单击 “确认” 按钮 , 超级终端设置完成。



### 3.1.4 超级终端的系统配置

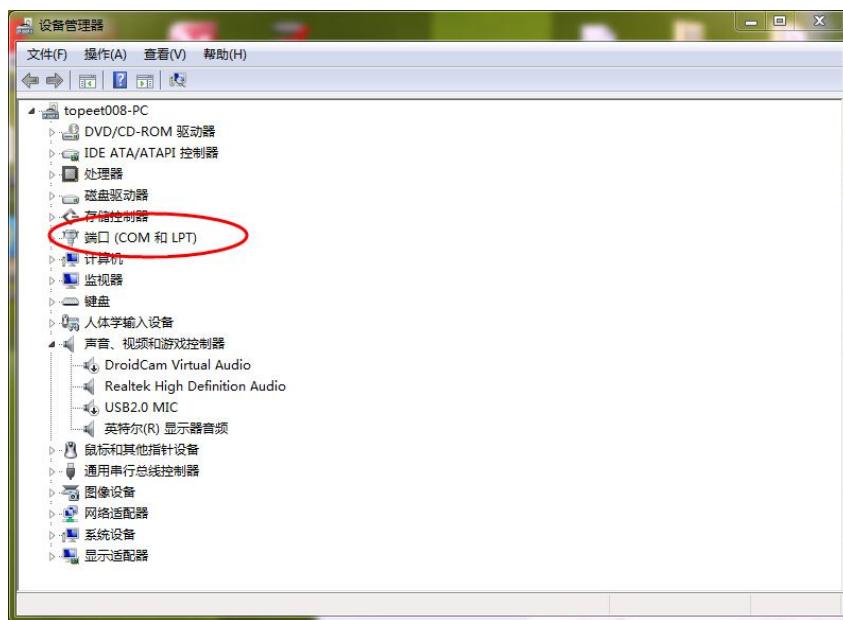
在 PC 机上，用户也需要针对操作系统进行设置。

下面详细讲解一下，以 Win7-64 位为例，讲解一下如何设置。

1 ) 如下图，进入操作系统的“控制面板” --> “系统和安全” --> “系统”

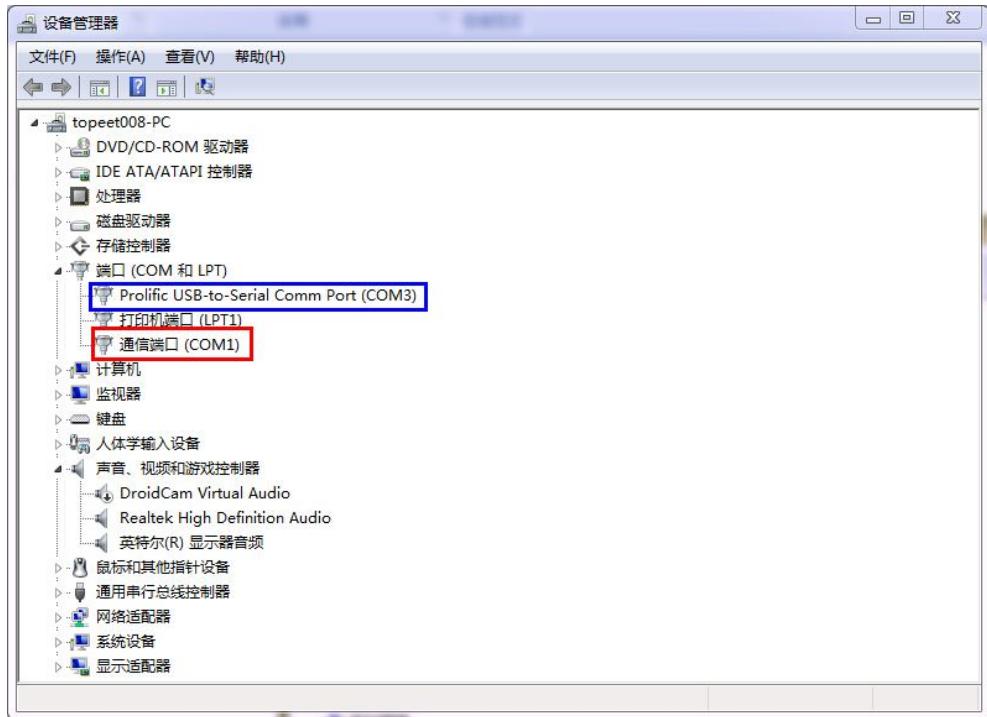


2 ) 如下图，进入设备管理器，单击菜单“端口”。

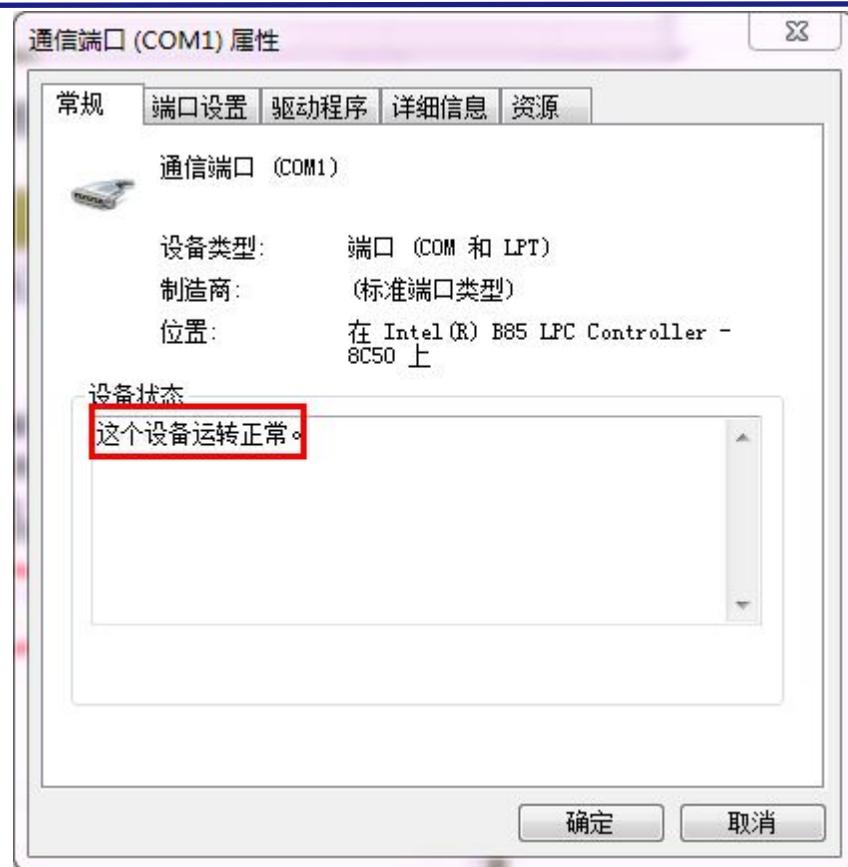


3 ) 如下图 , 选择你要用到的串口。蓝色方框中 , 是 USB 转串口的端口 ; 红色方框中 , 是 PC 机自带的串口。后面以电脑自带串口为例 , 进行讲解 , USB 转串口的设置也是一样。

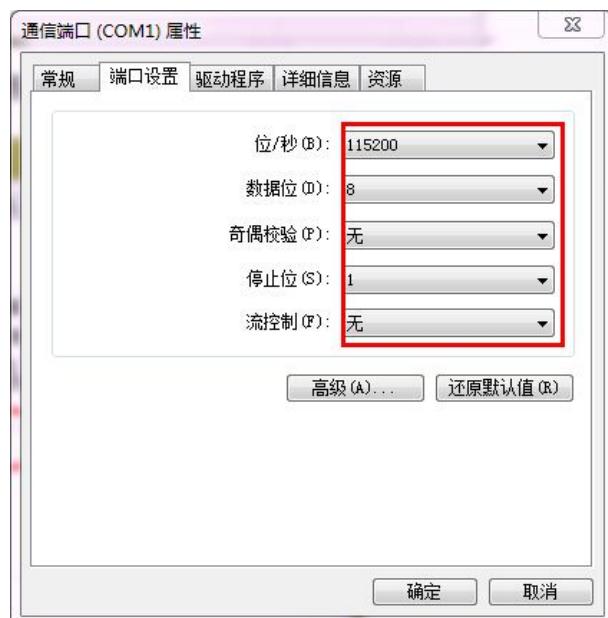
单击红色方框 “通信端口 ( COM1 ) ” 。



4 ) 如下图 , 可以看到 “这个设备运转正常” ; 然后单击菜单 “端口设置” 。



5 ) 如下图 , 用户参考红色框中的参数进行设置。



6 ) 设置完之后 , 如下图 , 单击 “确定” , 退出设置。这样超级终端的系统配置就完成了。



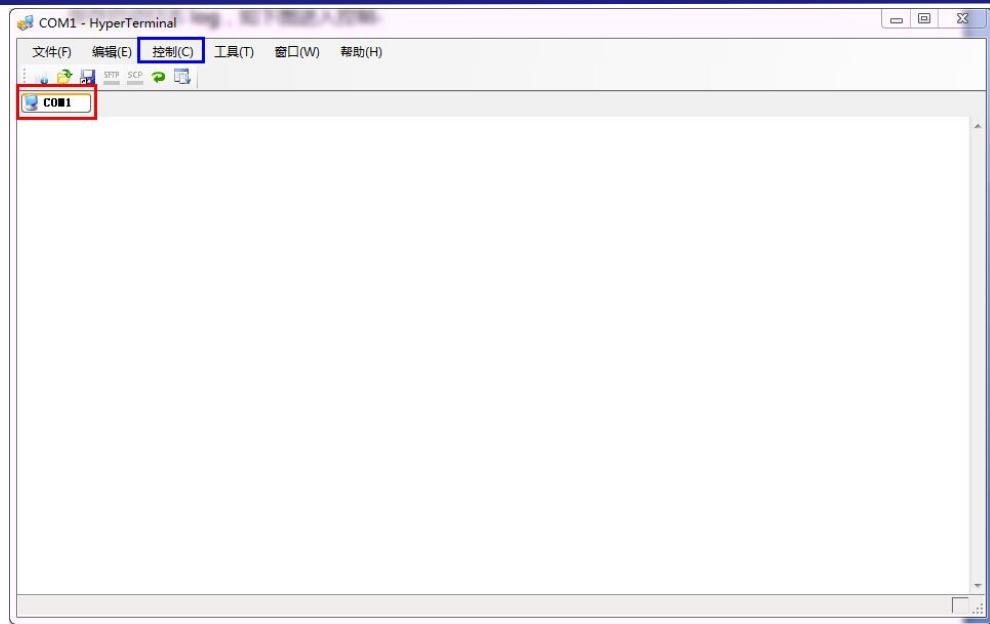
### 3.1.5 超级终端保存日志

开发板启动的时候，通过超级终端会看到串口打印的启动信息，通过这些信息可以用来分析和调试程序。

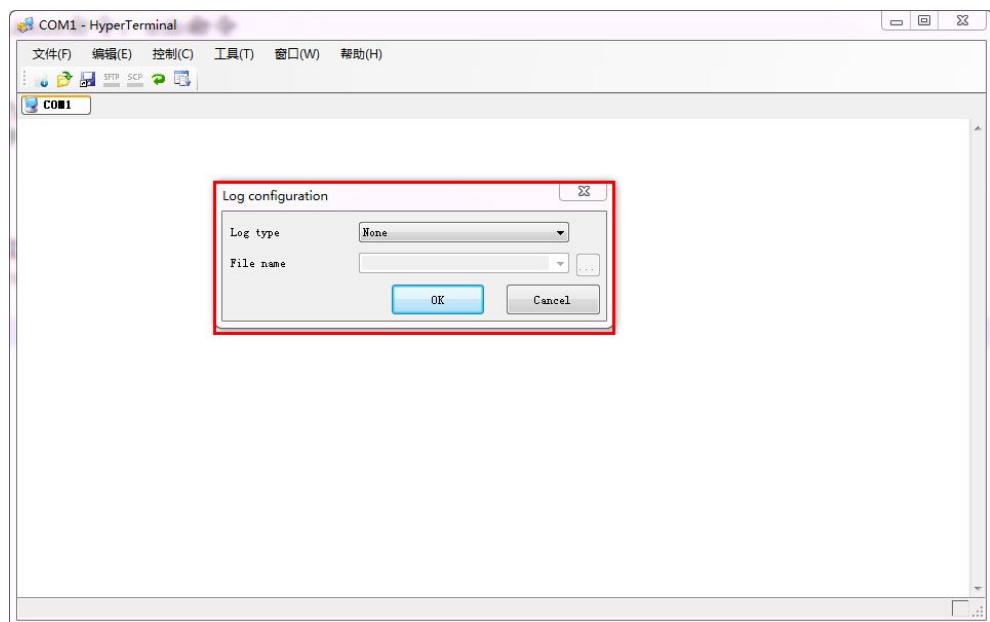
开发板启动的时候，超级终端打印的信息太快，无法进行细致分析，所以在这一小节里，我将教给大家如何把启动信息保存成文本。保存为文本之后，系统的启动信息就会被全部保存下来了，将有助于我们分析。

下面详细讲解一下，在PC机上，如何使用超级终端将系统的启动信息保存为文本。

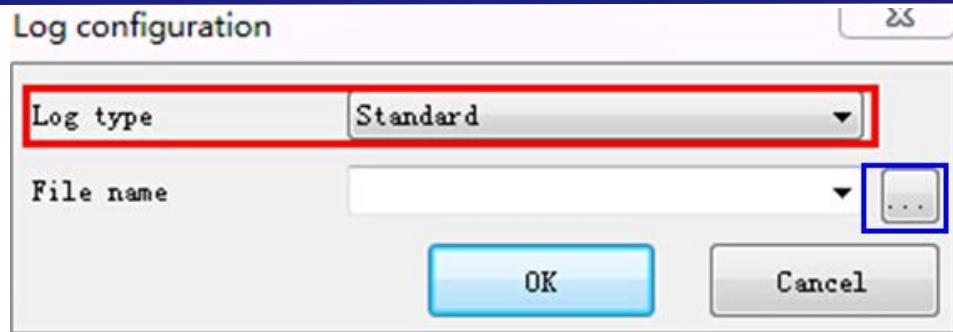
1) 如下图，先设置要保存日志的端口，我这里以 COM1 为例；执行菜单命令“控制” --> “日志设置 (L) ”。



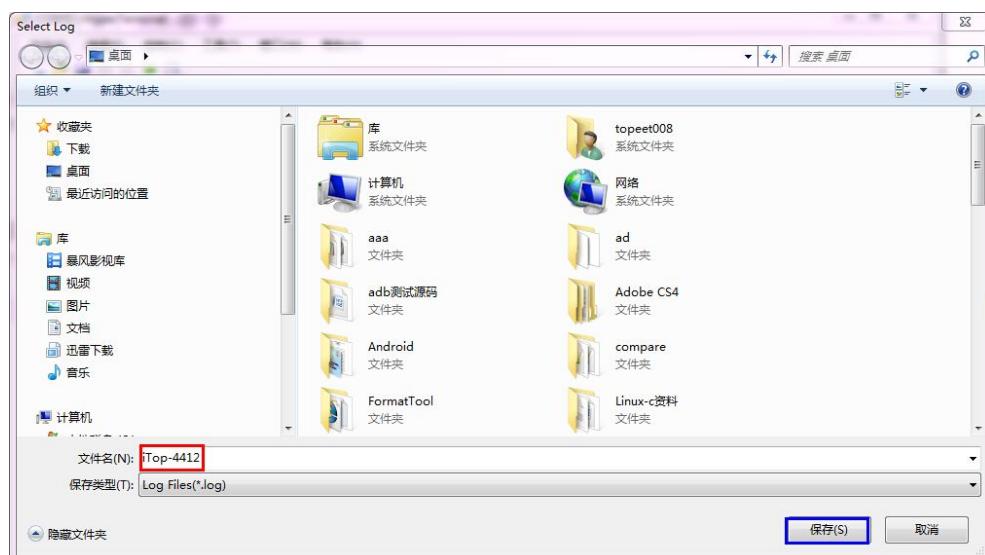
2) 如下图，弹出“Log configuration”配置框。



3) 如下图，在下拉菜单“Log type”中选择“Standard”，单击蓝色框中路径选择按钮“...”。



4 ) 如下图 , 选择文本存储路径 , 路径可以随意选择 , 这里选择保存在桌面 , 文件命名为 “iT0p-4412” 。单击 “保存” , 设置完成。



5 ) 如下图 , 单击 “OK” 。设置完成。



6 ) 设置完成以后 , 所有的启动信息和调试信息都可以保存在这个文本中。这样大家就可以通过分析文本中的信息去查找 BUG 或者调试程序。

## 3.2 安装虚拟机以及 Ubuntu12.04.2 等软件

在嵌入式开发和学习过程中，我们需要使用 Ubuntu 64 位操作系统，迅为电子使用是“Ubuntu12.04.2”。我们这里使用的是在 Windows 操作系统上安装虚拟机，然后在虚拟机上安装 Ubuntu 操作系统。虚拟机版本是“VMware-workstation-full-8.0.3”或者是“VMware-workstation-full-10.0.1”。

这里大家要注意的是，这一章节中讲的方法，都有特定的版本要求。

针对版本的特殊要求，归纳总结一下：

Win7-64 位操作系统下安装虚拟机" VMware-workstation-8.0.3"或者“VMware-workstation-full-10.0.1”

虚拟机上面安装“Ubuntu 12.04.2”

Ubuntu 操作系统上面搭建“uboot，内核，文件系统等”的编译环境

因为涉及开源软件，导致版本众多，所以无法一一进行测试以及提供使用方法，敬请见谅。

如下图所示，我们提供了两个版本的虚拟机，“VMware-workstation-full-8.0.3” 和“VMware-workstation-full-10.0.1-1379776.exe”。如果用户电脑主板的 USB 接口是 3.0 的，那么就需要用 10.0.1 版本的虚拟机，否则虚拟机无法识别到部分 usb 设备，两个版本的虚拟机安装方法类似，这里只介绍 8.0.3 的。

文件名	大小	修改时间
VMware-workstation-full-10.0.1-1379...	490.28MB	2015-07-01 16:46
VMware_Workstation_wmb.zip	483.04MB	2015-04-28 09:25

### 3.2.1 虚拟机 VMware-workstation 的安装

声明：本小节安装的虚拟机，所有资源来源于网络，仅仅作为学习参考，不得进行任何商业用途，否则产生的一切后果将由使用者本人承担。

这里需要注意的是，迅为电子给用户提供的虚拟机安装包是“VMware-workstation-full-8.0.3-703057”和“VMware-workstation-full-10.0.1-1379776.exe”。

可以在网盘目录“iTOP-4412 开发板搭建编译环境所需要的工具包以及补丁包”→“01-虚拟机 VMware\_Workstation\_wmb 软件”

安装方法基于 Win7-64 位操作系统，VMware 软件 8.0.3 和 10.0.1 安装方法一模一样，安装虚拟机小节中图片是 VMware8.0.3 的截图。

### 3.2.1.1 安装虚拟机

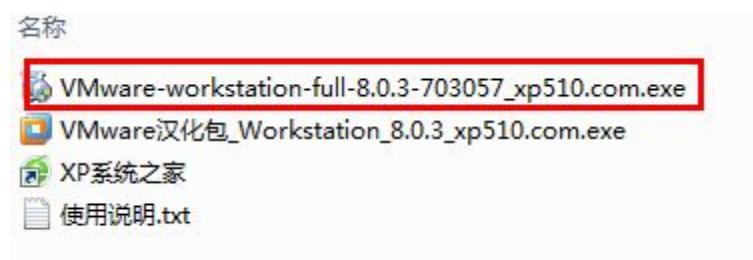
下面详细讲解一下，在 PC 机上，如何安装虚拟机。

1 ) 在网盘中下载虚拟机安装包，如下图。

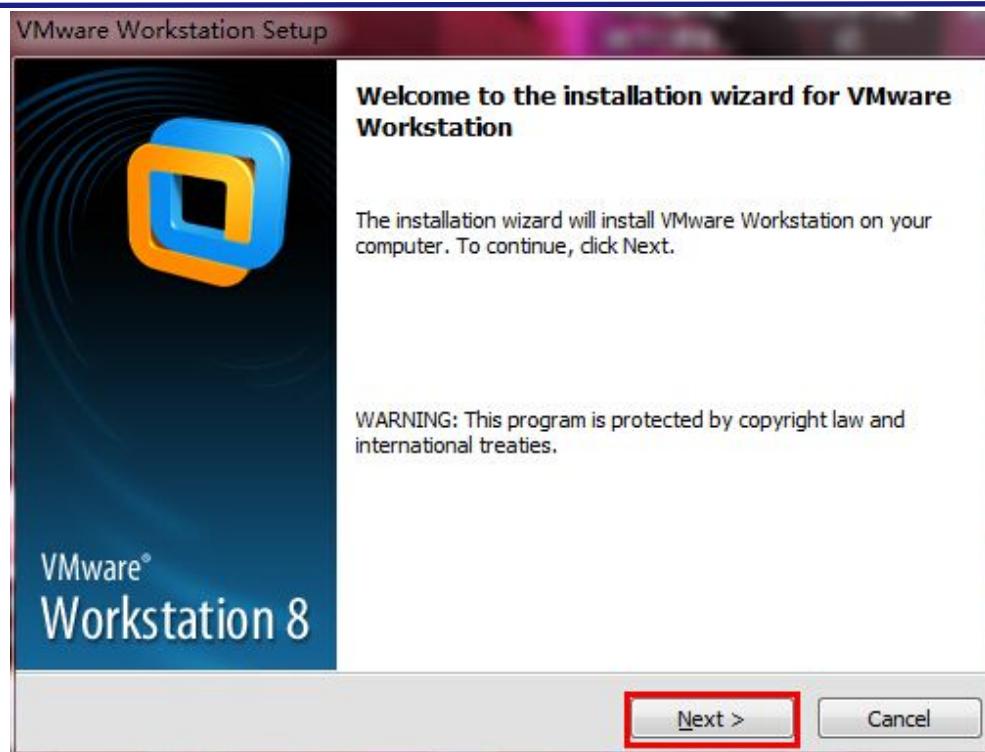


2 ) “VMware\_Workstation\_wmb.zip”解压后得到下面一个文件夹：

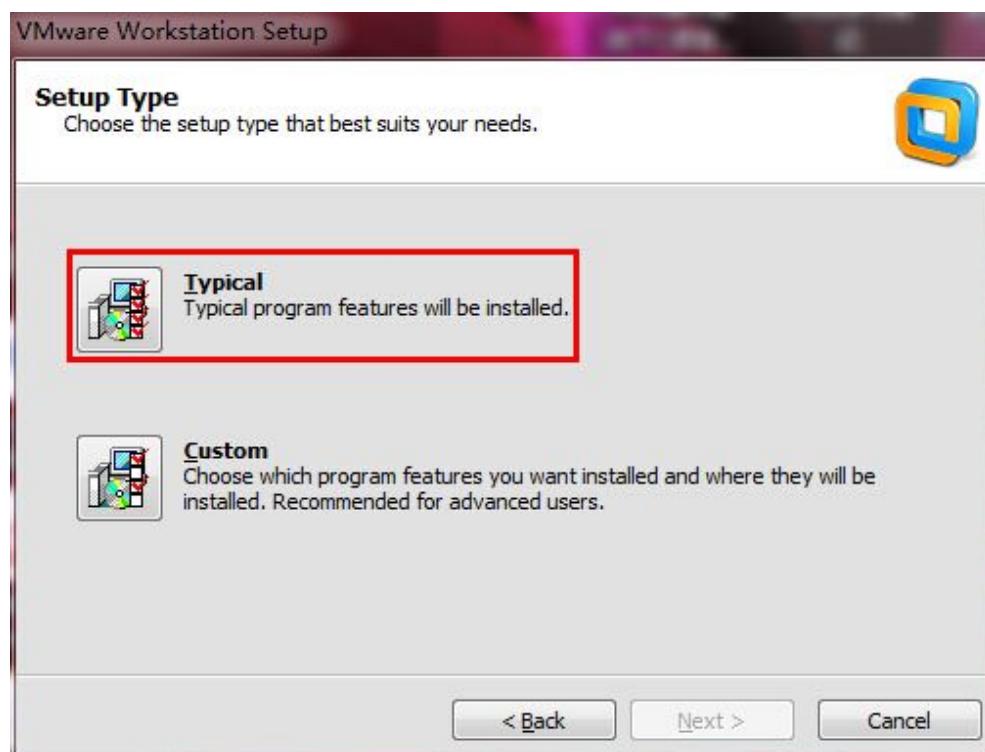
“VMware\_Workstation\_wmb” , 文件夹下面有一个文件“VMware-workstation-full-8.0.3-703057\_xp510.com.exe” , 这个文件是用户唯一要用到的文件，其它文件都不需要使用，也不需要汉化。



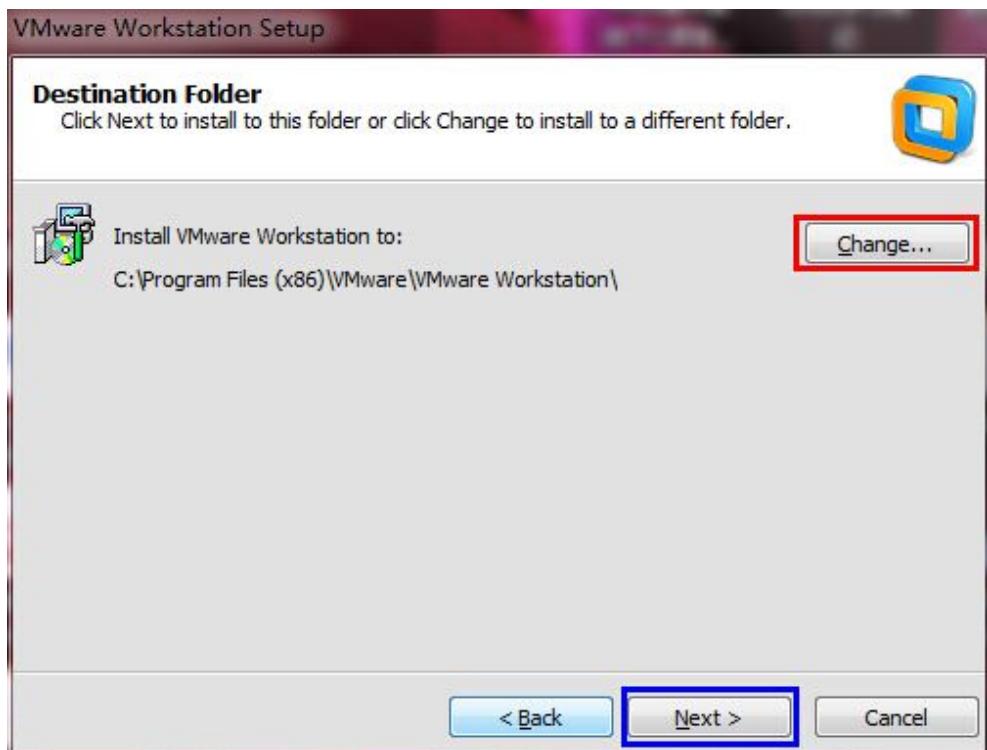
3 ) 单击上图红框中的可执行文件，开始安装，如下图，单击按钮“Next”，继续安装。



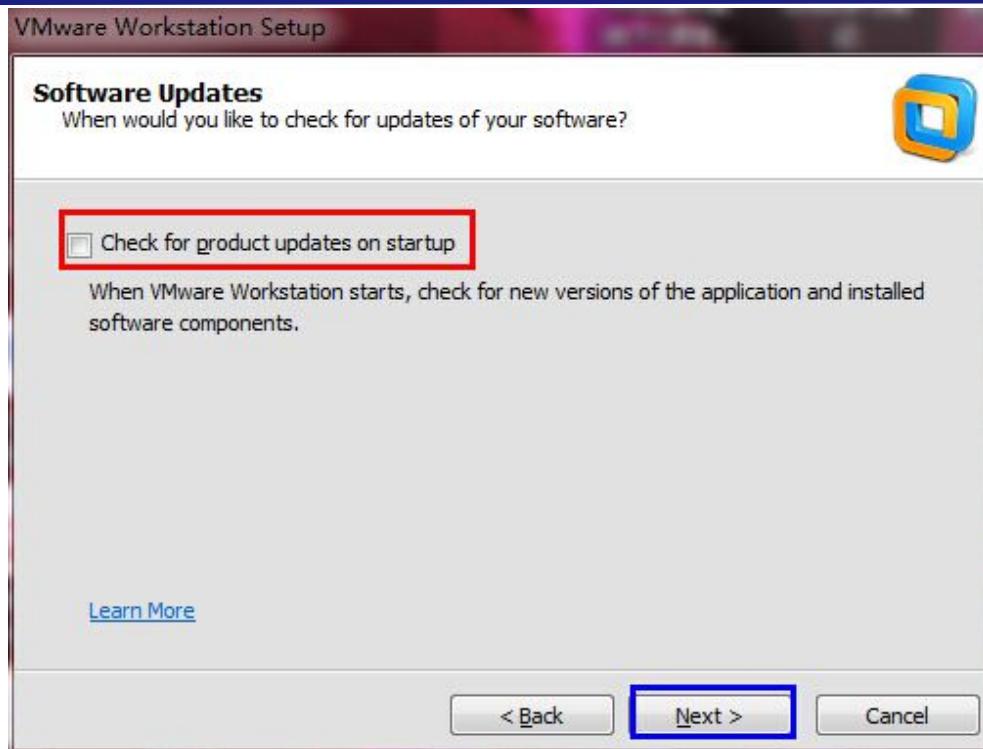
4 ) 如下图 , 单击选择 “Typical” 安装方式 , 继续安装。



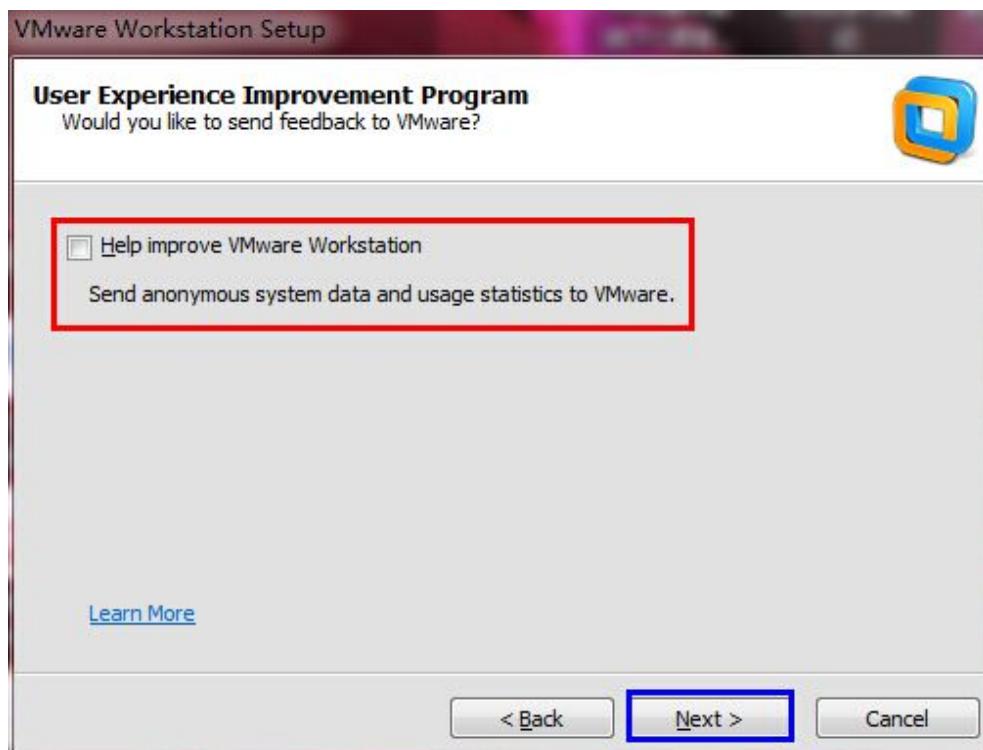
5 ) 如下图 , 红色框中可以选择安装路径 , 图中选择的是默认路径 , 用户可以根据实际情况选择软件的安装路径。软件的安装路径选择完成后 , 单击蓝色框中的按钮 “Next” , 继续安装。



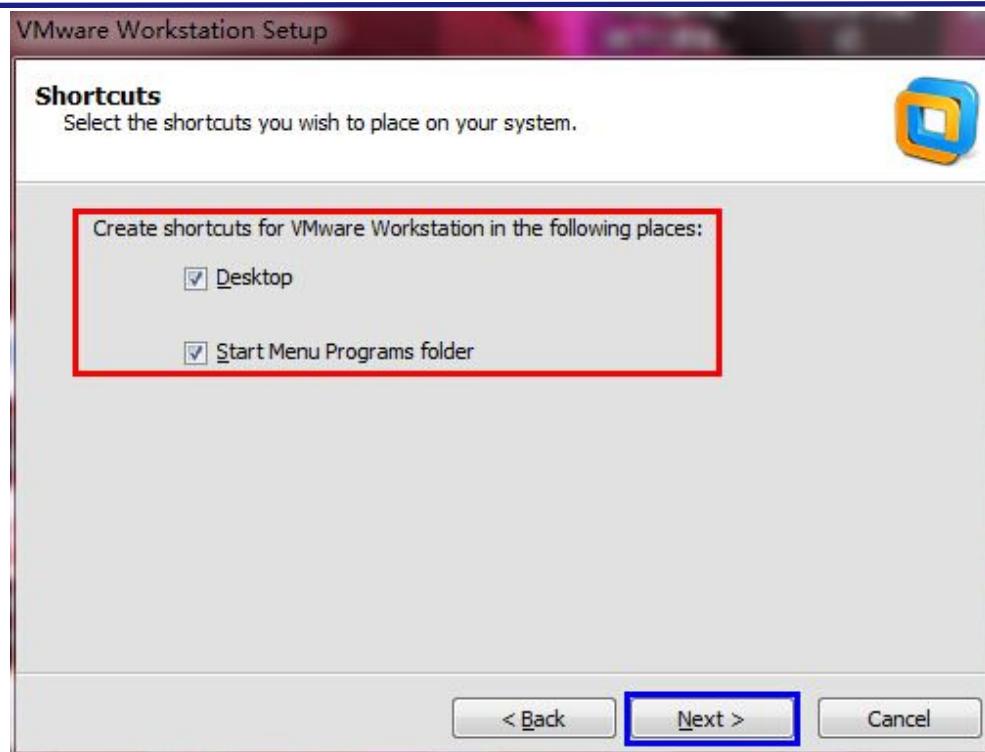
6 ) 如下图 , 这里要特别注意的是 , 红色框中 “√” 选项一定要去掉 , 不然虚拟机会自动升级到最高版本。如果虚拟机升级到最高版本了 , 用户在后面将无法使用迅为电子提供的 “搭建好的编译环境” , 虚拟机版本高了之后 , 显卡驱动会出现不兼容的情况 , 最后导致无法进入 Ubuntu 桌面系统。单击蓝色框中的按钮 “Next” , 继续安装。



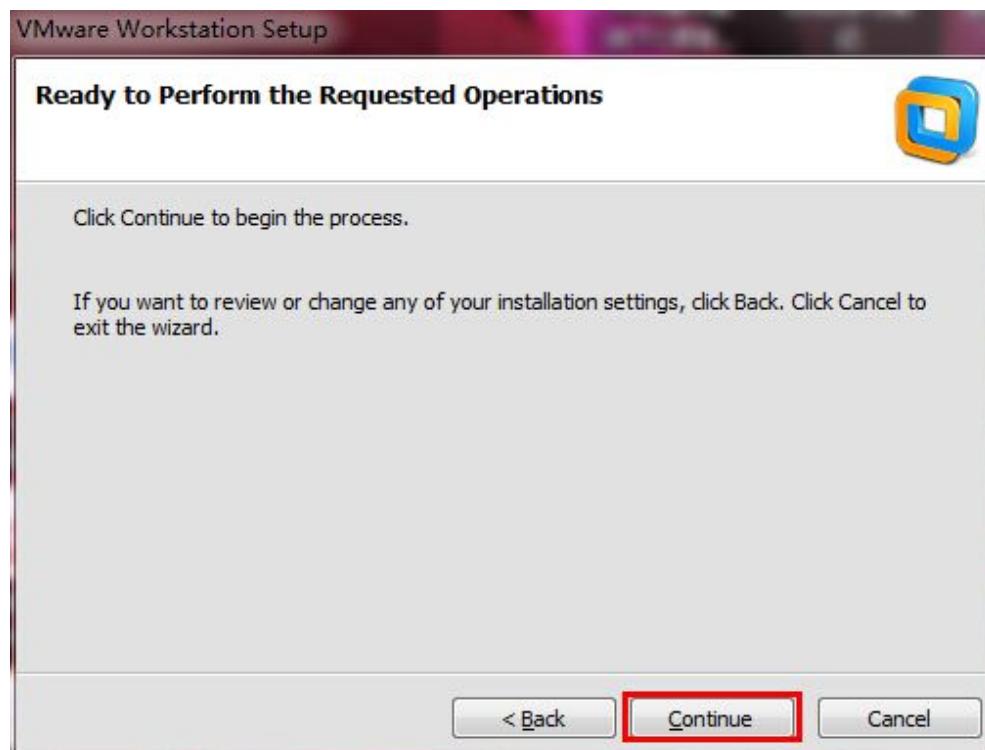
7) 如下图，红色框中的“√”去掉，单击按钮“Next” ，继续安装。



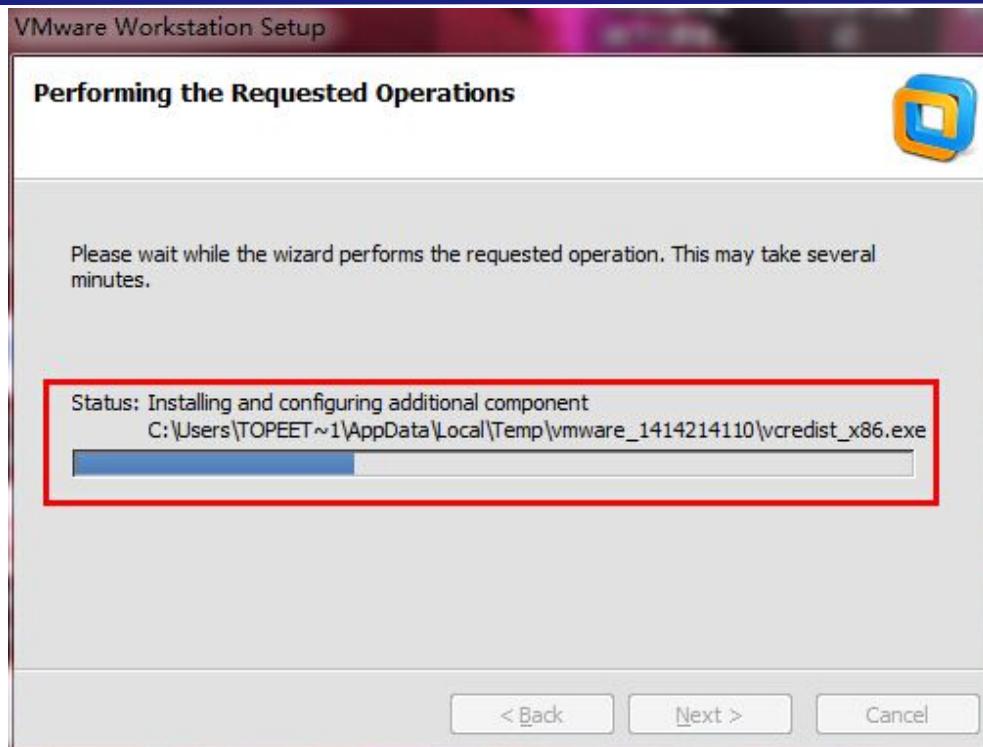
8) 如下图，红色框中的“√” ，用户自己选择，选了后，可以生成桌面图标以及在开始菜单中生成菜单。单击蓝色框中的按钮“Next” ，继续安装。



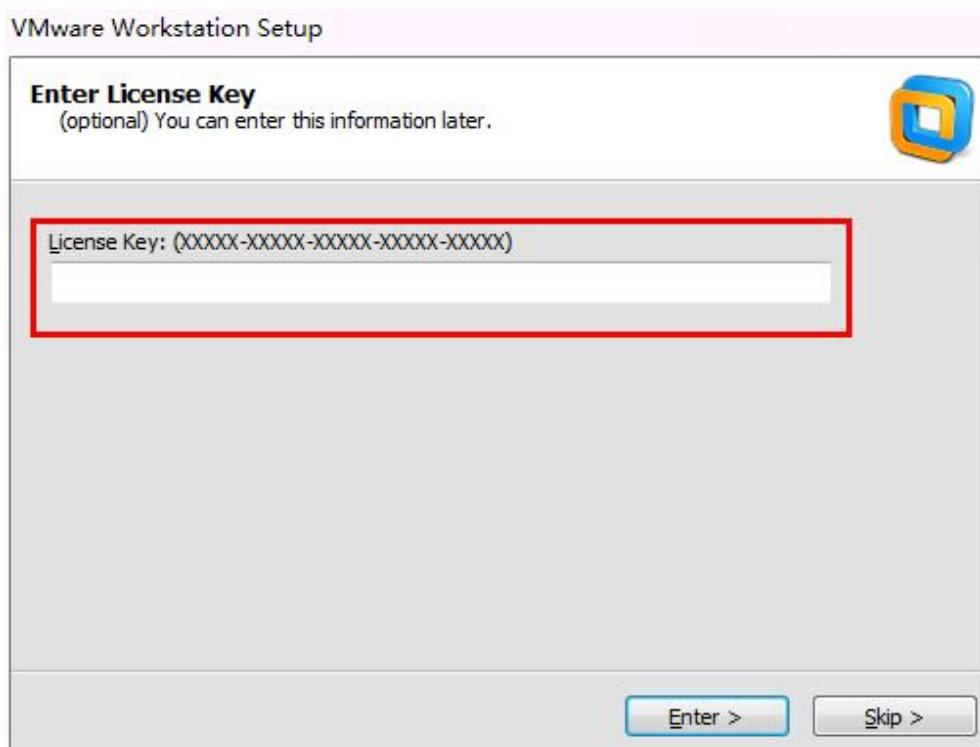
9 ) 如下图 , 单击按钮 “Continue” , 继续安装。



10 ) 如下图 , 软件在安装中 , 等待软件自动安装。



11 ) 如下图 , 需要输入 “ License Key ” 。



12 ) 如下图 , 使用浏览器 , “ 百度一下 ” 关键词 “ vmware8.0.3 序列号 ” , 找一个能用的 , 粘贴复制到上图的红色框中。

Baidu 百度  百度一下

网页 新闻 贴吧 知道 音乐 图片 视频 地图 文库 更多»

百度为您找到相关结果约10,100个

[求vmware workstation 8.0.3有效注册码\\_百度知道](#)  
2个回答 - 提问时间: 2012年05月23日  
vmware workstation 8.0.3 注册码: UV3NK-25Y41-081CZ-0YP7C-PQ89A  
[zhidao.baidu.com/link?...](#) 2012-05-23 - 百度快照 - 86%好评

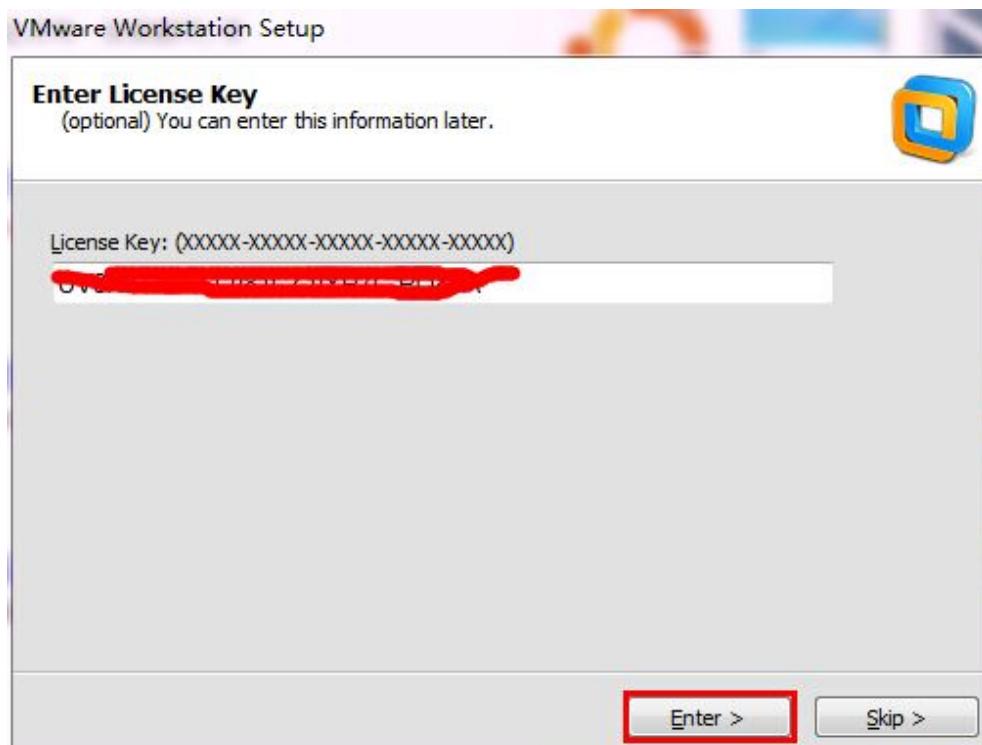
[VMware Workstation 8.0.3 build 703057 简体中文破解版\(序列号\)](#)  
3条评论  
VMware Workstation 8.0.3 build 703057 简体中文破解版(序列号),0x37的网易博客,这里是用  
来备份[http://blog.ruqin.in](#).  
[blog.163.com/bug/blog/...](#) 2012-05-14 - 百度快照 - 89%好评

[vmware workstation 8.0.3许可密匙 这个我在网上找的注...\\_百度知道](#)  
3个回答 - 提问时间: 2012年05月13日  
网友都在找: vmware8.0.3注册机 vmware8.0.4序列号 虚拟机8.0密钥 vmware8.0序列号 vmware  
workstation 9 序列号 vmware的相关知识...  
[zhidao.baidu.com/link?...](#) 2012-05-13 - 百度快照 - 86%好评

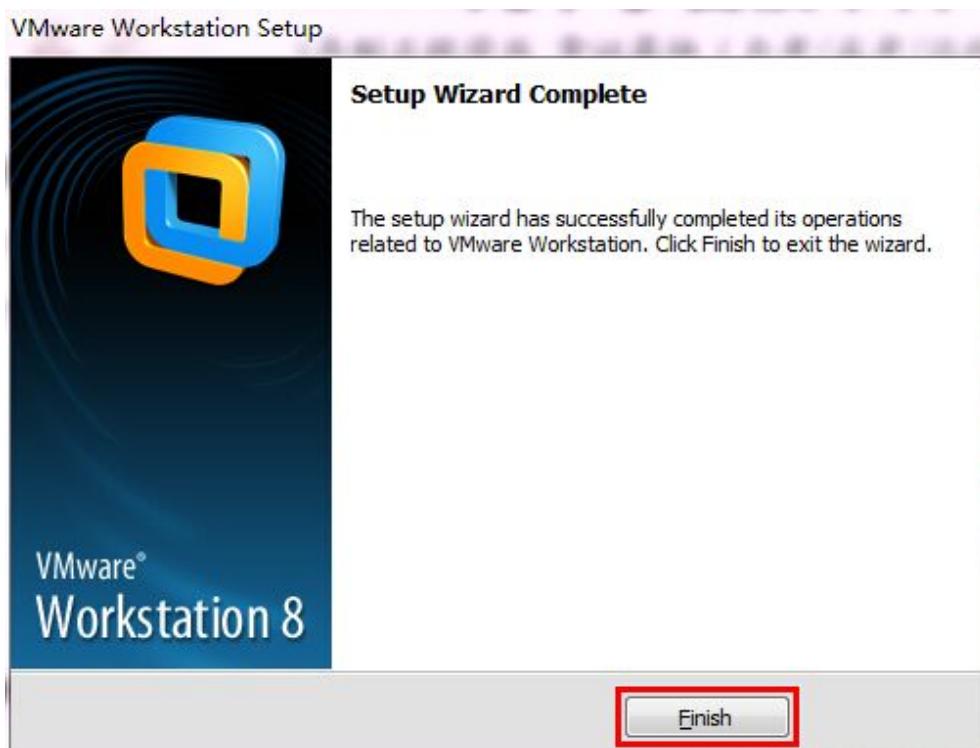
[vmware workstation 8.0.3 build-703057 破解补丁序列号](#)  
vmware workstation 8.0.3 build-703057 破解补丁序列号双击直接导入就可以了资源积分:3分  
下载次数:532 资源类型:工具类 资源大小:13KB 资源得分: (114位用户...  
[download.csdn.net/deta...](#) 2012-05-19 - 百度快照 - 80%好评

[vmware 8.0.3序列号](#)  
vmware 8.0.3序列号 vmware 8.0.3序列号 0.4序列号 vmware8.0 0.4序列号 vmware8.0 vmware  
8.0.3序列号 vmware 8.0.3序列号 vmware 8.0.3...  
[www.003c.com/html/tvmw...](#) 2014-10-04 - 百度快照 - 62%好评

13 ) 然后 , 如下图 , 单击按钮 “Enter” 。下图中 , 破解需要的 Key , 用户可以很容易的通过网络找到。



13 ) 如下图 , 单击按钮 “Finish” , 完成安装。



14 ) 安装完成后 , 桌面生成快捷方式 , 如下图所示 , 双击快捷方式就可以打开虚拟机。这里大家要注意的是 , 如果对虚拟机不是很熟悉 , 不要汉化虚拟机 , 也不要升级虚拟机。



### 3.2.1.2 安装虚拟机常见错误

安装完虚拟机后 , 加载或者安装镜像后会报错 , 报错一般是由下面几种情况造成的。

1 ) 当出现无法加载迅为提供镜像的情况 , 一般原因是没有破解成功。没有破解成功 , 很有可能是用户以前安装过虚拟机 , 在重新安装迅为电子提供的虚拟机的时候 , 需要先卸载以前的虚拟机 , 在卸载的时候 , 没有将虚拟机的产品信息以及注册信息删除掉。参考本小节中 “ 卸载后重装虚拟机需要注意的问题 ” , 可以解决这个问题。

2 ) 用户可以加载镜像，但是打开 Ubuntu 的时候，发现 Ubuntu 没有进入图形界面，只有“黑漆漆”的一片。这种情况下，一般有以下两种原因。

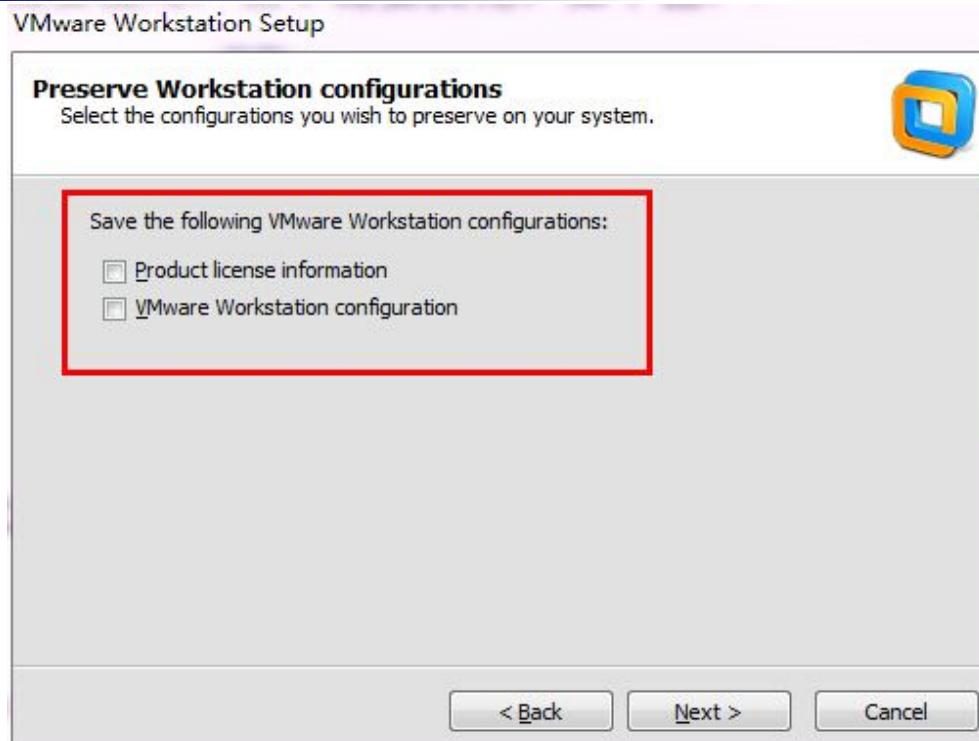
第一，用户装的虚拟机版本和迅为电子提供的不一样，或者进行了版本升级，这样 Ubuntu 的显卡就无法启动，只能进入 Ubuntu 终端而无法进入 Ubuntu 的图形界面。

第二，用户打开虚拟机之后，刚开始可以使用，但是用了几次后出现无法进入图形界面的情况，这种状况是由于用户使用 Ubuntu 系统不当造成的，例如关机方式不对或者在 Ubuntu 系统里面升级了显卡程序。这种情况发生后，就只能使用 Ubuntu 的终端了，无法进入图形界面。当然如果能够在 Ubuntu 终端重装一下显卡，也是可以的，但是这种方法和用户具体的机器有关系，很难有一个通用的教程。所以，用户只能自学重装 Ubuntu 显卡或者重新装一下 Ubuntu。

### 3.2.1.3 卸载后重装虚拟机需要注意的问题

部分用户反馈，卸载后再重装虚拟机，有时候会出现无法破解的情况，这里迅为电子根据实际使用过程中积累的经验，提醒大家注意几点。

1 ) 卸载的时候，出现如下图，有关是否保留虚拟机安装信息的提示框，全部去掉“√”。



2 ) 虚拟机卸载完成后，还需要清除一下注册表信息，下图中，以 360 的为例。单击红色框中的“电脑清理”。



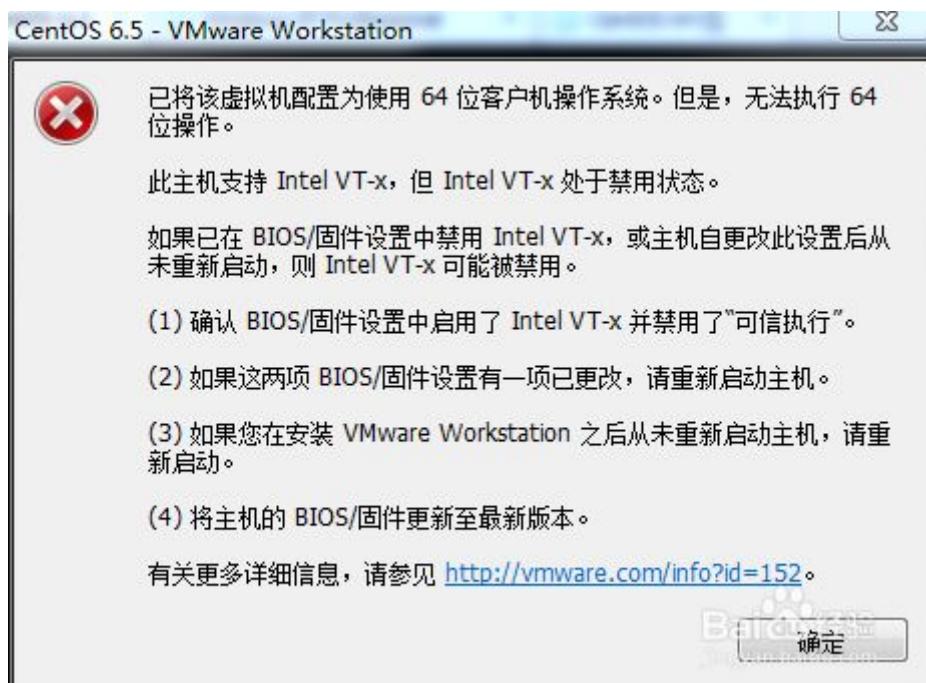
3 ) 如下图，选上蓝色框中“残留的软件信息”，然后“一键清理”。



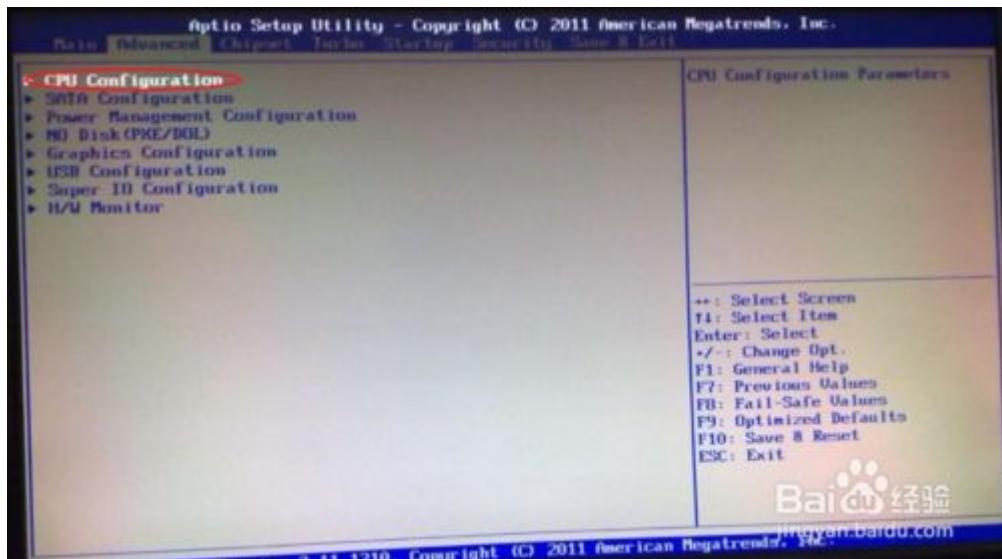
4 ) 再次安装虚拟机，一般不会出现问题。

### 3.2.1.4 虚拟机安装 Ubuntu 常见问题之 64 位虚拟化

VMware Workstation 虚拟机安装使用 64 位操作系统时涉及到一个 CPU 虚拟化的问题，如果 BIOS 没有开启 CPU 的虚拟化选项，创建和打开 64 位虚拟机就会报错。如下图所示



如果你的电脑没有打开虚拟化，需要重启电脑，开机之后按 Delete 键（笔记本按 F2 或者其他键，具体根据电脑型号，有的不是常规按键需要查看说明书或者打电话给客服），进入 BIOS 模式，找到【CPU Configuration】选项。



找到 CPU 虚拟化配置选项，这里是【Intel HT Technology】选项，改为“Enable”启用虚拟化，改完之后 F10 保存 BIOS 配置，重启电脑之后再次打开虚拟机正常启动。

如果用户不知道怎么设置自己机器的 64 位虚拟化，那么则可以百度关键词“自己电脑主板型号+64 位虚拟化”很容易找到如何设置虚拟化。

### 3.2.2 虚拟机加载 Ubuntu12.04.2 镜像

安装好虚拟机之后，用户就可以加载 Ubuntu12.04.2 镜像。用户可以在网盘中下载“编译好的镜像”，该镜像已经安装好了编译 Android4.0.3 所需要的软件了。用户加载镜像后，直接可以用于编译 uboot、Kernel，Android4.0.3。

下面详细讲解一下，在 PC 机上，如何用虚拟机加载 Ubuntu12.04.2 编译好的镜像。

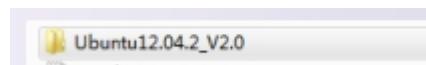
编译好的 Ubuntu12.04.2 镜像在网盘“iTOP-4412 开发板搭建编译环境所需要的工具包以及补丁包” → “02-Ubuntu 系统安装包” → “02-搭建好开发环境的 Ubuntu 虚拟机镜像 V2.0” 目录下，如下图所示。

Read_Me.txt	872B	2015-04-28 09:25
Ubuntu12.04.2_V2.0.7z.001	500.00MB	2015-04-28 09:25
Ubuntu12.04.2_V2.0.7z.002	500.00MB	2015-04-28 09:25
Ubuntu12.04.2_V2.0.7z.003	500.00MB	2015-04-28 09:25
Ubuntu12.04.2_V2.0.7z.004	500.00MB	2015-04-28 09:25
Ubuntu12.04.2_V2.0.7z.005	500.00MB	2015-04-28 09:25
Ubuntu12.04.2_V2.0.7z.006	500.00MB	2015-04-28 09:25
Ubuntu12.04.2_V2.0.7z.007	500.00MB	2015-04-28 09:25
Ubuntu12.04.2_V2.0.7z.008	500.00MB	2015-04-28 09:25
Ubuntu12.04.2_V2.0.7z.009	500.00MB	2015-04-28 09:25
Ubuntu12.04.2_V2.0.7z.010	500.00MB	2015-04-28 09:25
Ubuntu12.04.2_V2.0.7z.011	500.00MB	2015-04-28 09:25
Ubuntu12.04.2_V2.0.7z.012	500.00MB	2015-04-28 09:25
Ubuntu12.04.2_V2.0.7z.013	500.00MB	2015-04-28 09:25
Ubuntu12.04.2_V2.0.7z.014	500.00MB	2015-04-28 09:25
Ubuntu12.04.2_V2.0.7z.015	500.00MB	2015-04-28 09:25
Ubuntu12.04.2_V2.0.7z.016	385.08MB	2015-04-28 09:25

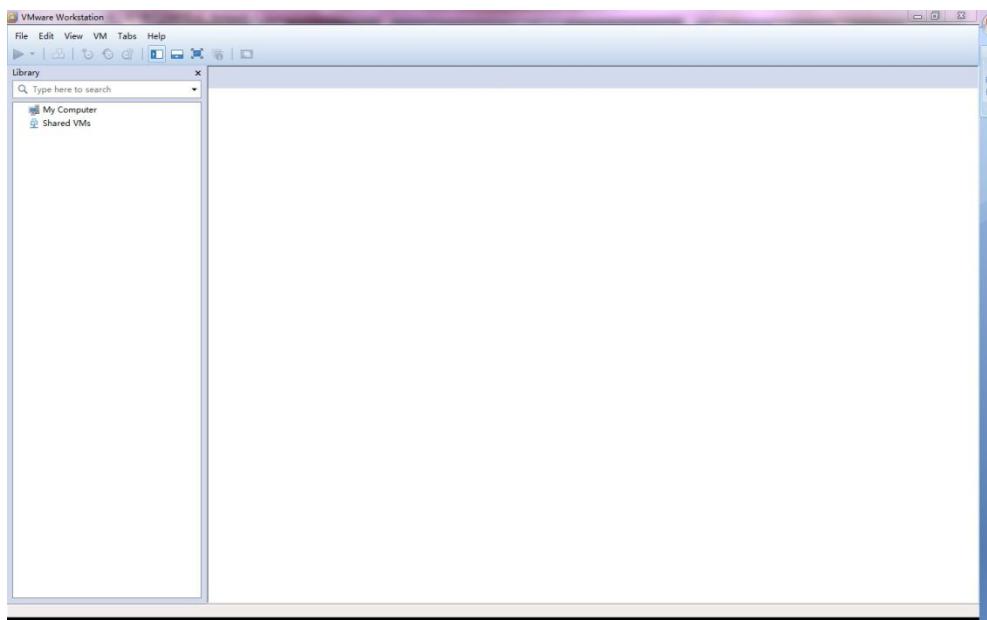
1 ) 如下图 , 编译好的镜像很大 , 而且要全部下载完之后才能开始解压 , 压缩软件是 “2345 好压” 。

Read_Me.txt	872B	
Ubuntu12.04.2_V2.0.7z.001	500.00MB	
Ubuntu12.04.2_V2.0.7z.002	500.00MB	
Ubuntu12.04.2_V2.0.7z.003	500.00MB	
Ubuntu12.04.2_V2.0.7z.004	500.00MB	
Ubuntu12.04.2_V2.0.7z.005	500.00MB	
Ubuntu12.04.2_V2.0.7z.006	500.00MB	
Ubuntu12.04.2_V2.0.7z.007	500.00MB	
Ubuntu12.04.2_V2.0.7z.008	500.00MB	
Ubuntu12.04.2_V2.0.7z.009	500.00MB	
Ubuntu12.04.2_V2.0.7z.010	500.00MB	
Ubuntu12.04.2_V2.0.7z.011	500.00MB	
Ubuntu12.04.2_V2.0.7z.012	500.00MB	
Ubuntu12.04.2_V2.0.7z.013	500.00MB	
Ubuntu12.04.2_V2.0.7z.014	500.00MB	
Ubuntu12.04.2_V2.0.7z.015	500.00MB	
Ubuntu12.04.2_V2.0.7z.016	385.08MB	
Ubuntu_V2_MD5.c	5KB	

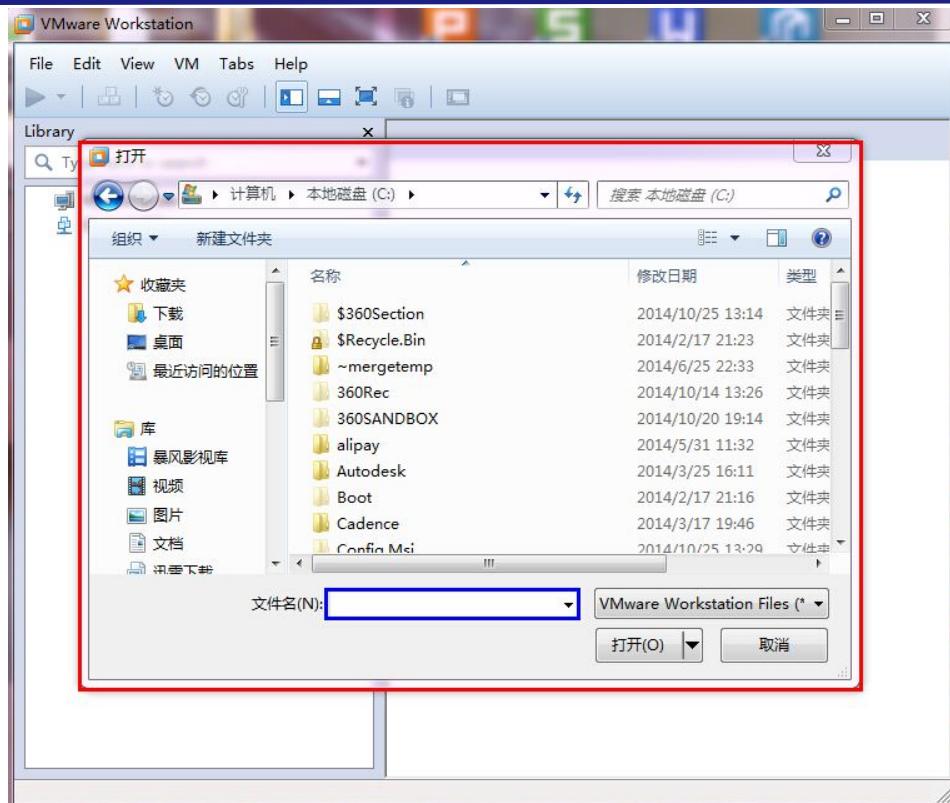
2 ) 安装好解压软件后，右键单击上图中的压缩文件 “Ubuntu12.04.2\_V2.0.7z.001” ，开始解压。解压完成后到文件夹 “Ubuntu12.04.2\_V2.0” ，如下图所示。



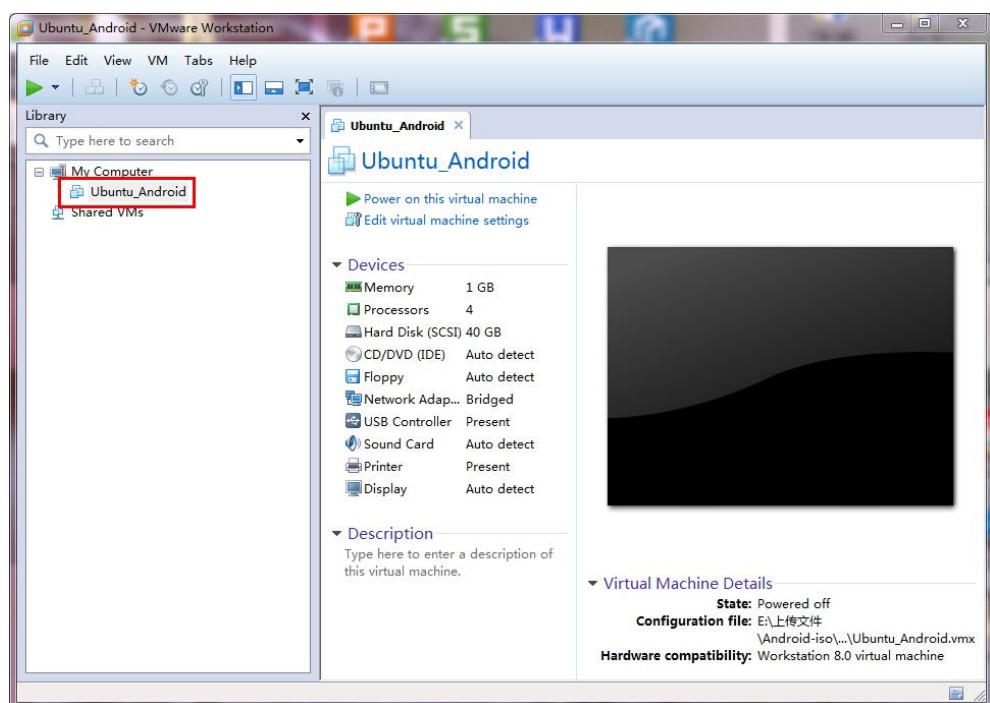
3 ) 如下图，打开虚拟机，如果有提示升级虚拟机，跳过就行，不需要进行升级。



4 ) 执行菜单命令 “File” --> “Open...” ，弹出如下图所示对话框。



5 ) 进入前面解压生成的文件夹 “Ubuntu12.04.2\_V2.0” , 单击文件 “iTOP-4412\_V2.0.vmx” , 然后单击按钮 “打开” , 如下图 , 虚拟机加载 Ubuntu12.04.2 “编译好的镜像” 完成。



### 3.2.3 虚拟机安装 Ubuntu12.04.2 初始系统

用户也可以自己安装“Ubuntu12.04.2”。

这里要注意的是，自己安装的 Ubuntu 系统中，很多基本软件都没有安装，需要用户参考“Android4.0.3 开发环境搭建以及编译”章节，安装基本软件，搭建开发环境。

另外，系统需要安装一些插件，所以用户的 PC 机需要联网，而且要尽量保证网络的通畅，这样安装起来会节省不少时间。

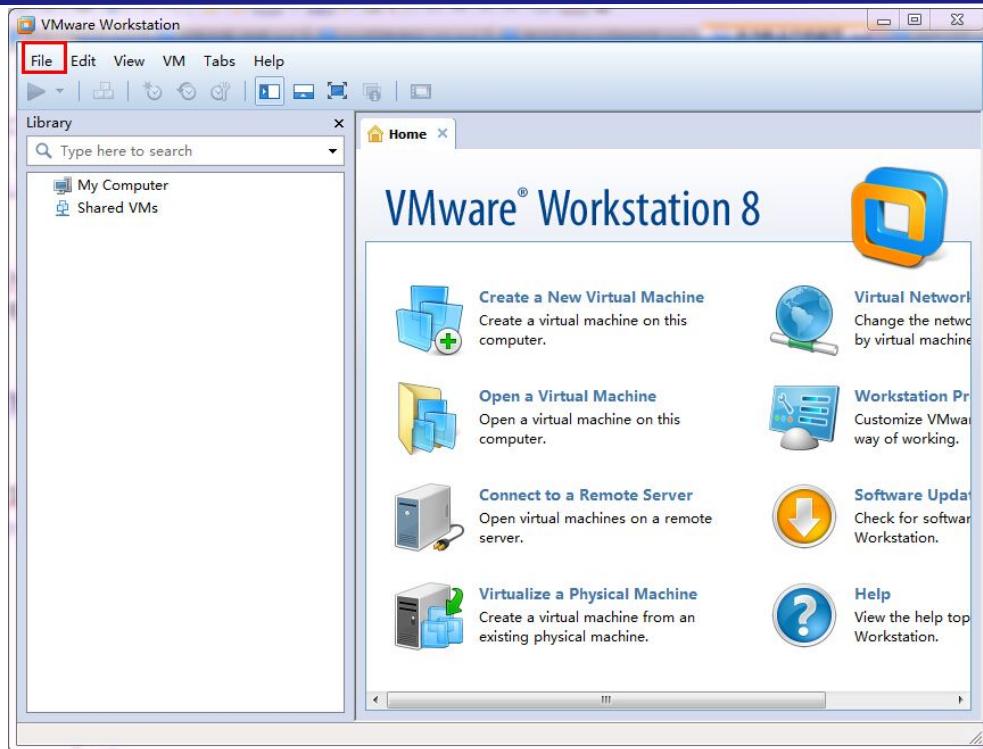
在网盘“iTOP-4412 开发板搭建编译环境所需要的工具包以及补丁包”→“02-Ubuntu 系统安装包”→“01-Ubuntu 12.04.2 安装包”可以下载初始系统“ubuntu-12.04.2-desktop-amd64.iso”。

下面详细讲解一下，在 PC 机上，如何使用虚拟机安装 Ubuntu 系统。

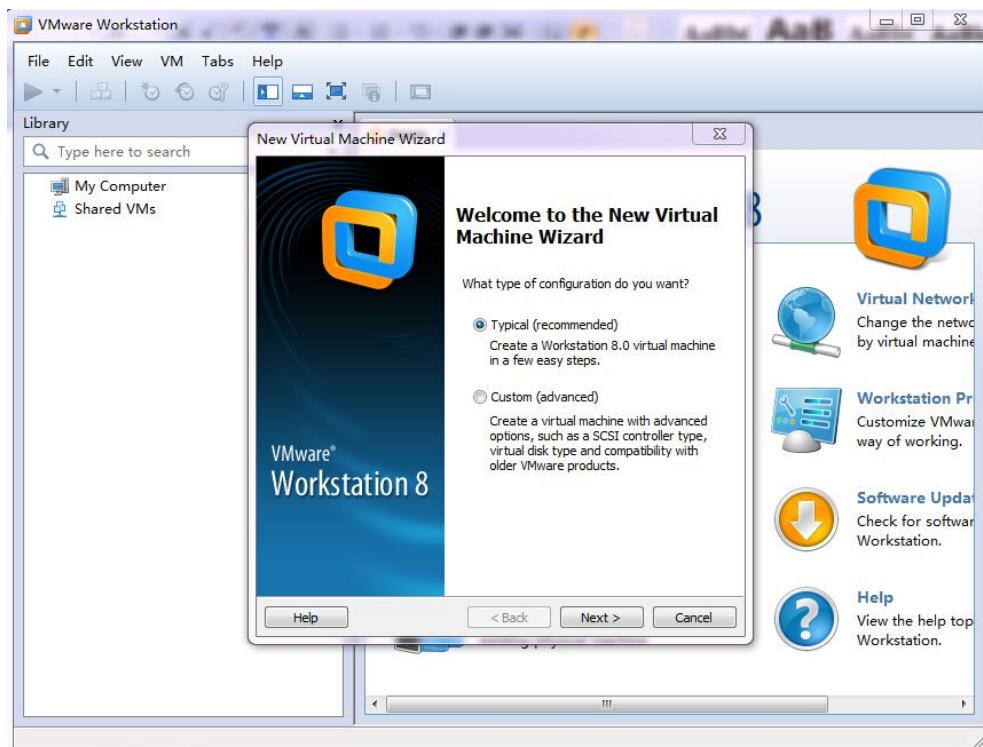
1) “Ubuntu12.04.2”的系统安装镜像可以到网盘下载，文件“ubuntu-12.04.2-desktop-amd64.iso”即为 Ubuntu 系统的安装镜像，如下图。



2) 打开虚拟机，如下图。



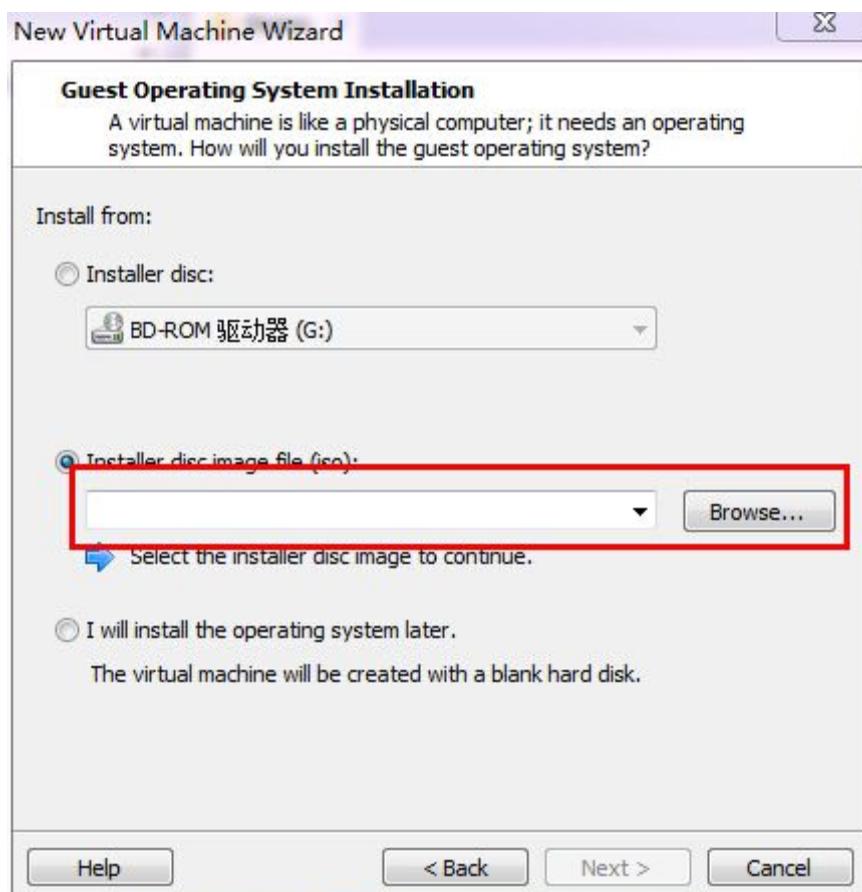
3 ) 执行菜单命令 “File” --> “New Virtual .....” , 弹出安装引导的对话框 , 如下图



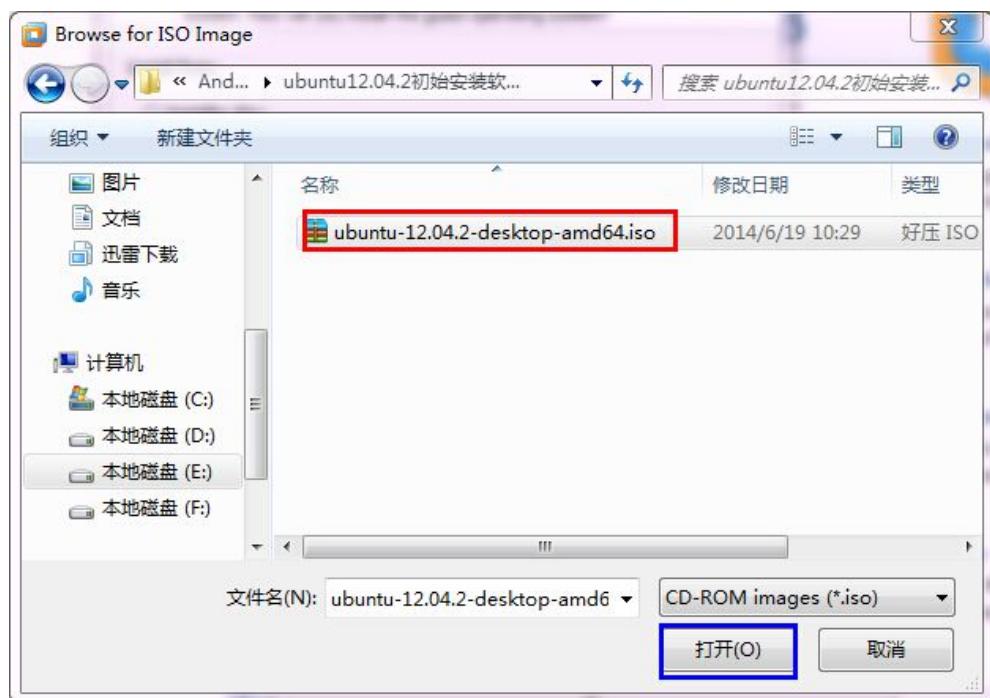
4 ) 如下图 , 选择默认设置 , 单击按钮 “Next” 。



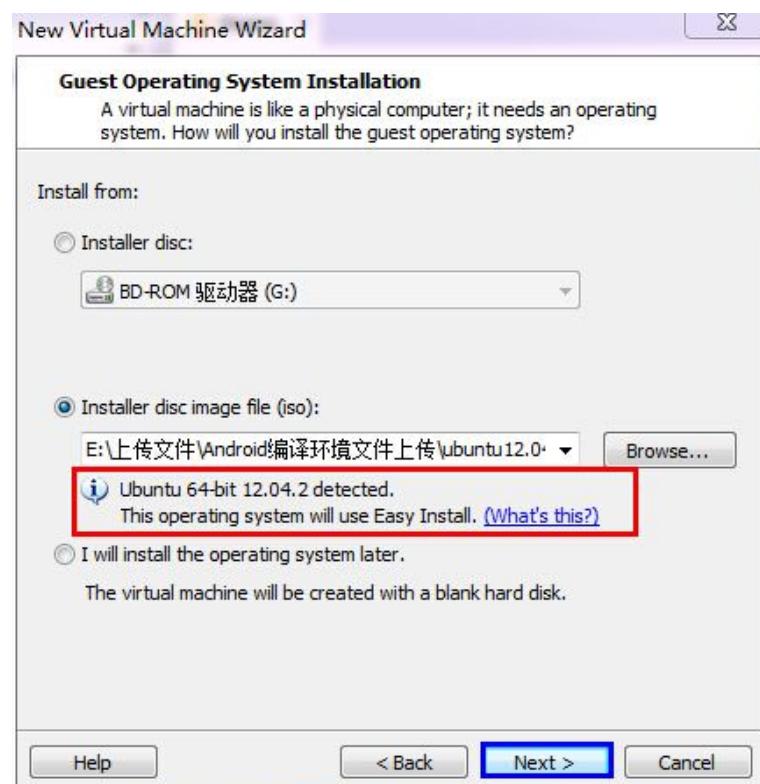
5 ) 如下图 , 红色框中选择 Ubuntu 的安装镜像路径。



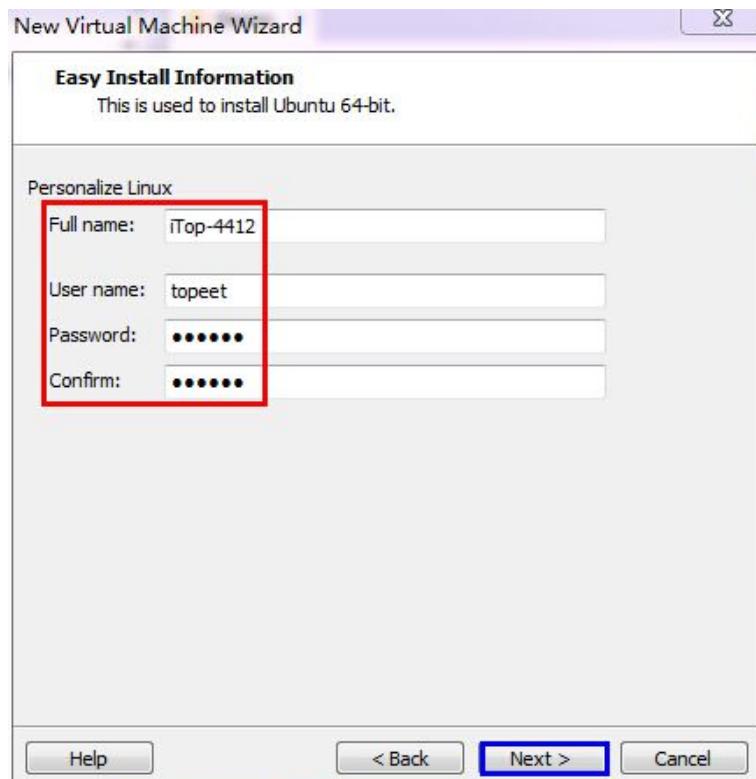
6 ) 如下图 , 选择红色框中的文件 “ubuntu-12.04.2-desktop-amd64.iso” , 然后 , 单击按钮 “打开” 。



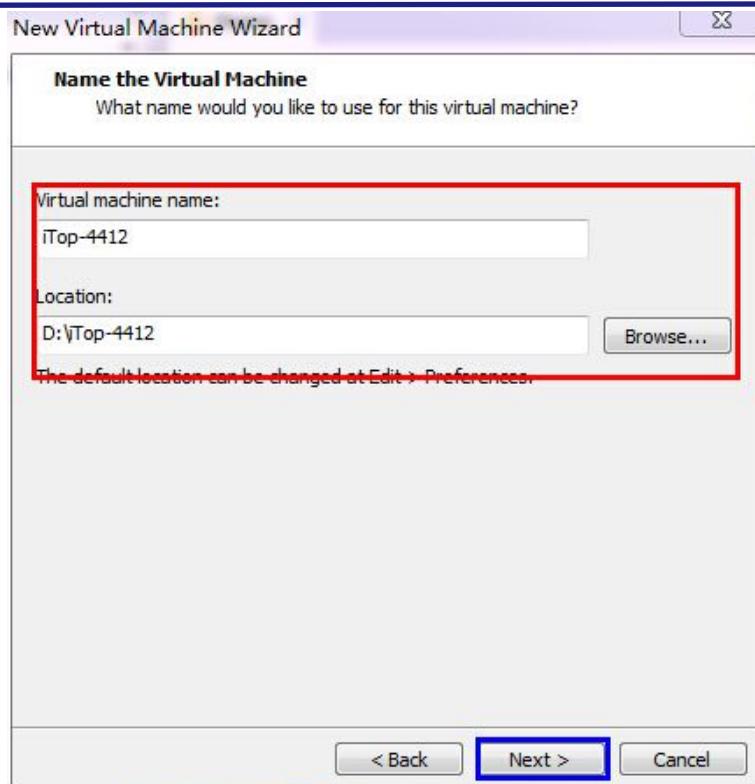
7 ) 如下图 , 设置好镜像的路径 , 红色框中提示虚拟机检测到 “Ubuntu 64-bit 12.04.2” 。单击蓝色框中的按钮 “Next” , 开始安装。



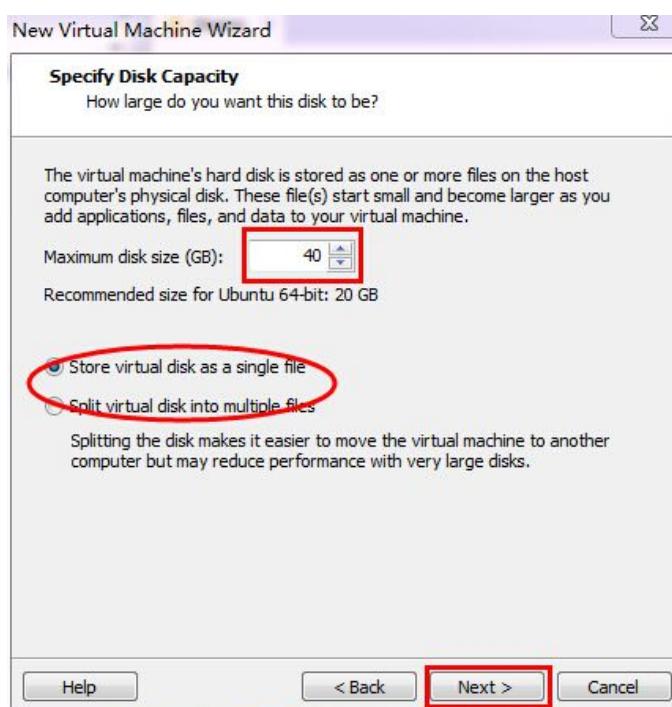
8 ) 如下图，在红色框中，分别设置名称，用户名，用户密码等，用户密码在登陆 Ubuntu 系统的时候会用到，这里用户可以根据实际情况自己设定，这里建议不要使用中文字符，然后点击蓝色框中的按钮 “Next” 。



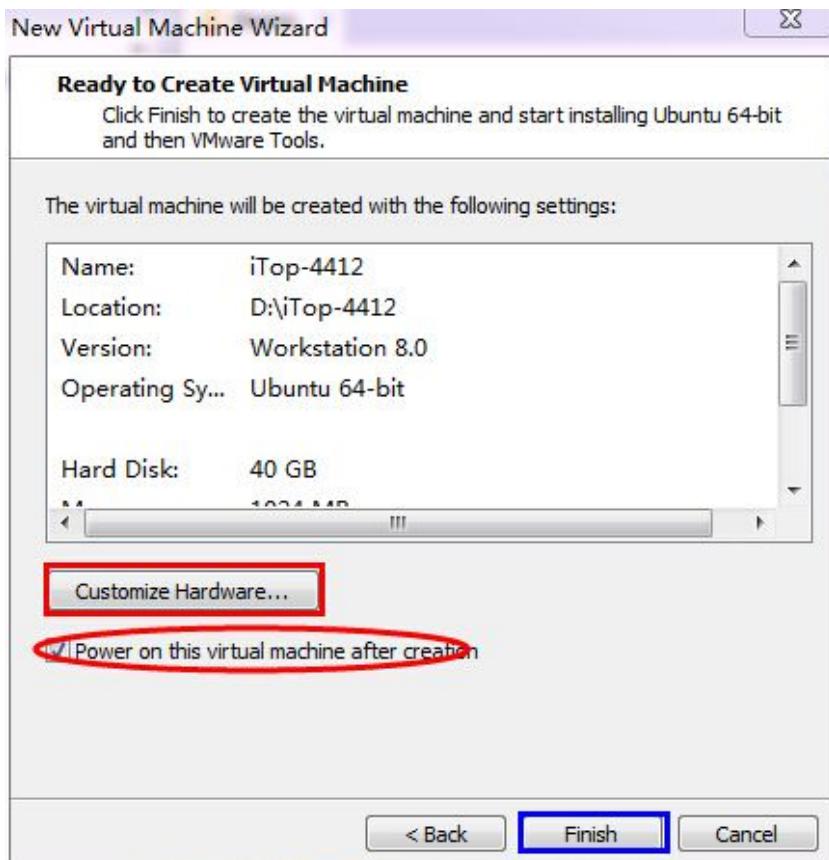
9 ) 如下图，在红色框中，设置虚拟系统的名称，虚拟机文件存储的路径。设置路径的时候，要注意盘符的剩余空间，推荐的盘符空间是 40G，不要少于 30G，也就是安装路径的盘符最少要有 30G 的剩余空间。单击蓝色框中的按钮 “Next” 。



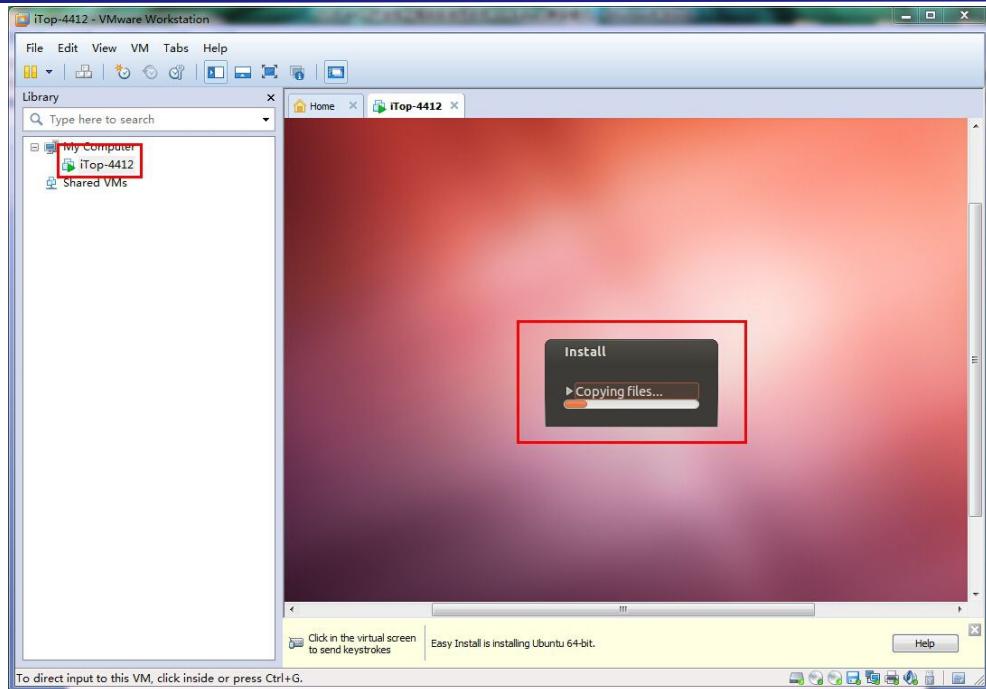
10)如下图，红色矩形框中设置虚拟机系统的硬盘大小，硬盘设置为 40G，椭圆框中选择是将系统装成一个文件，还是多个文件。“多个文件”便于移动位置，例如迅为电子给大家提供的“编译好的镜像”就是采用这种方式安装的。单击蓝色框中的按钮 “” Next。



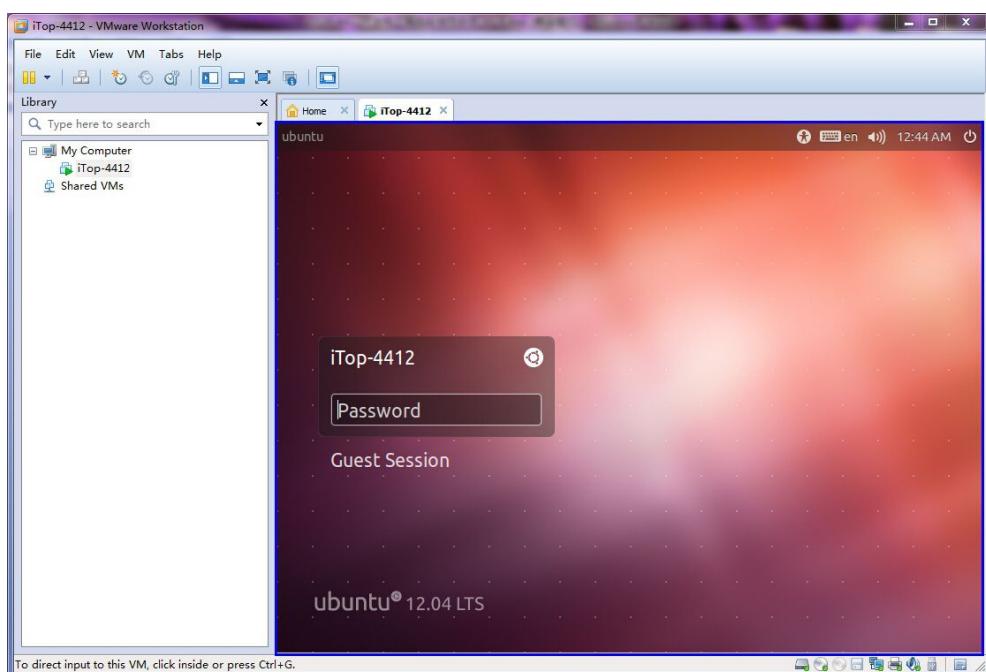
11 ) 如下图 , 红色矩形框中 , 是关于硬件设置的选项 , 在后面 “ 虚拟机联网以及其它设置 ” 里面会讲到 , 这里就直接忽略。红色椭圆框中 , 表示安装完成后运行 Ubuntu 系统 , 下图中是默认打开。单击蓝色框中的按钮 “ Finish ” , 开始安装。



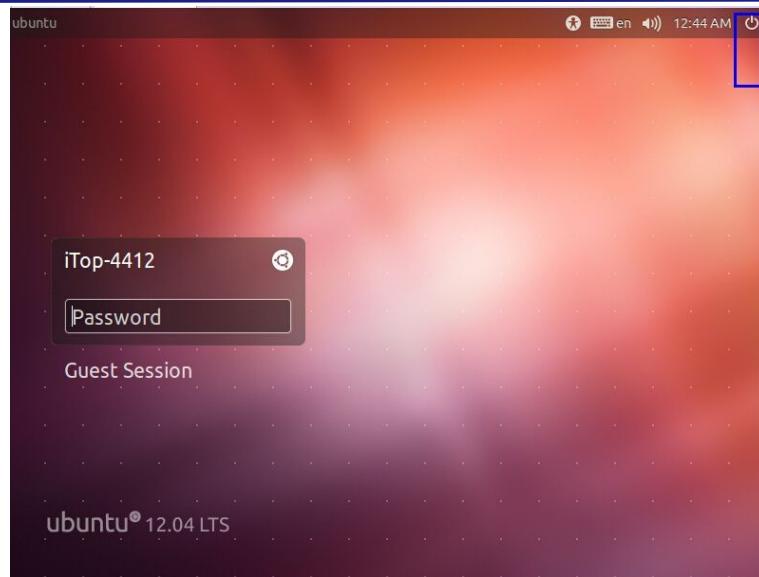
12 ) 如下图 , 开始安装。左上的红色框中就是前面设置的系统名字 “iTop-4412” , 中间的红色框显示系统开始安装。安装时间和机器配置相关 , 较低的配置 , 一个小时之内也可以装完。



13 ) 安装过程中不需要用户操作 , 当出现如下图所示的界面 , Ubuntu12.04.2 系统就安装完成了。



14 ) 如下图 , 点击右上角的按钮 , 执行菜单命令 “Shut Down” , 可以关机。



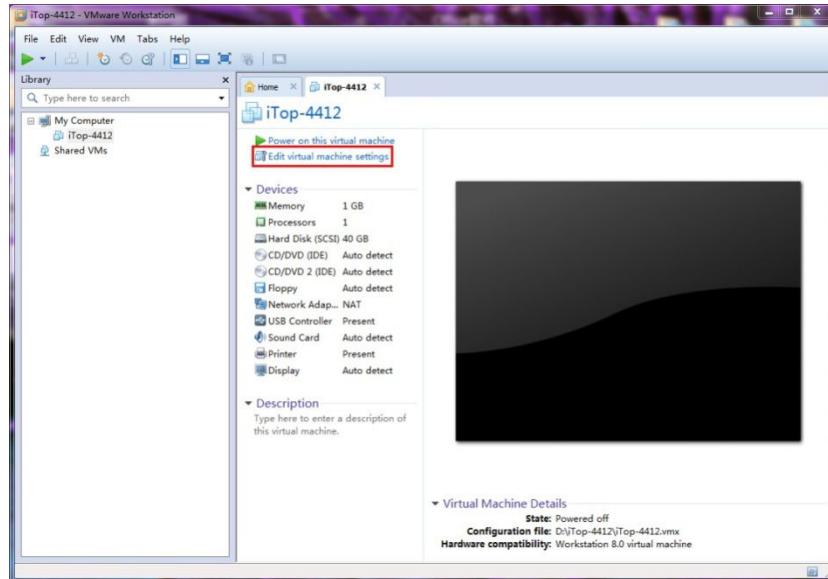
### 3.2.4 虚拟机 VMware-workstation8.0.3 联网以及基本设置

虚拟机需要根据用户的实际情况，进行网络设置以及其他一些基本的设置。

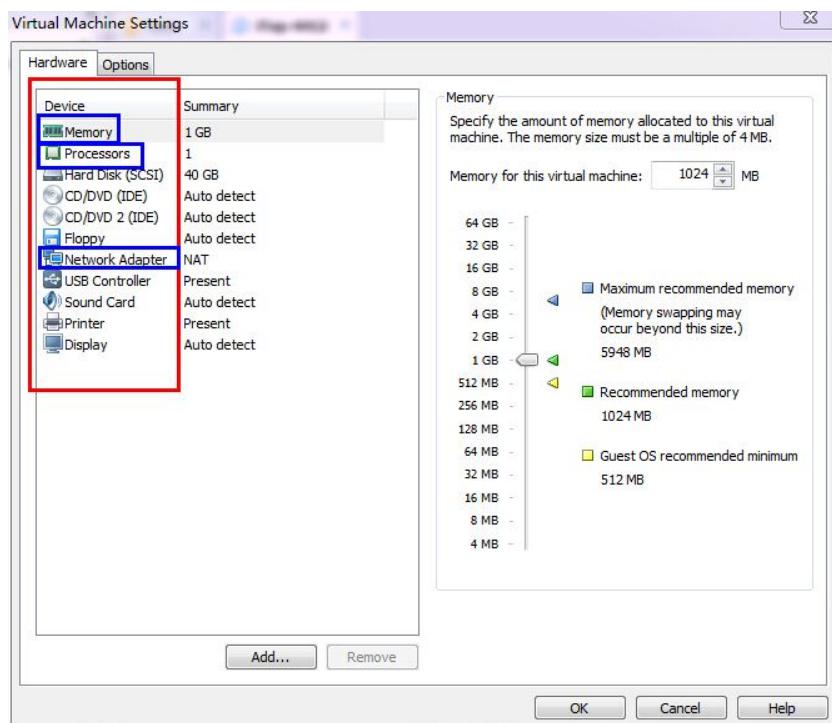
VMware10.0.1 和 Vmware8.0.3 联网和基本设置类似。

下面先详细讲解一下，虚拟机的一些基本的设置。

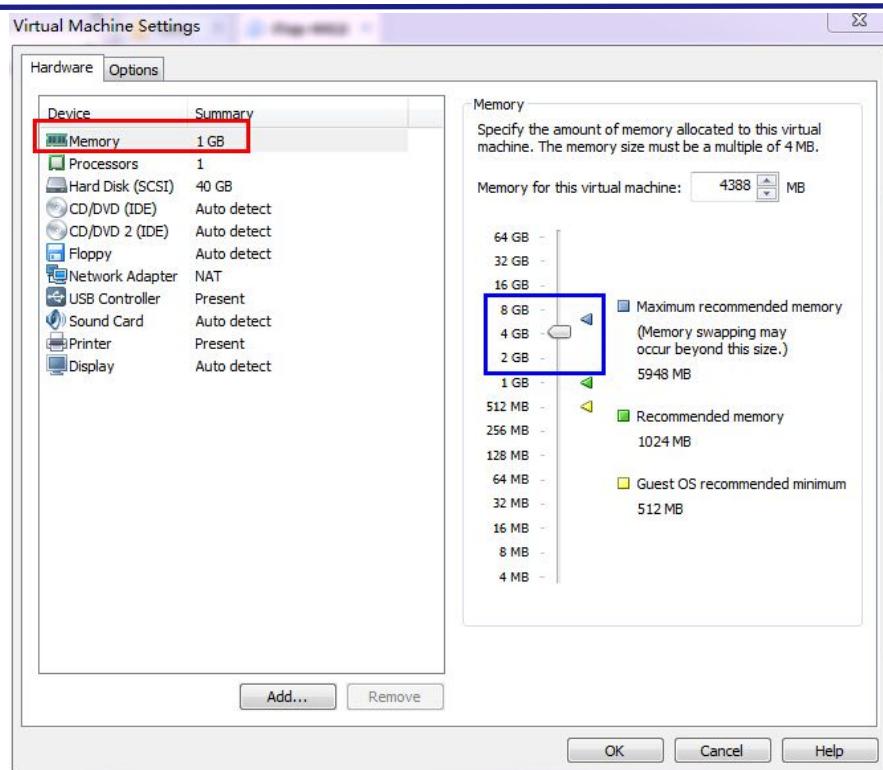
1 ) 打开虚拟机，如下图。单击红色框中的“Edit virtual .....”。



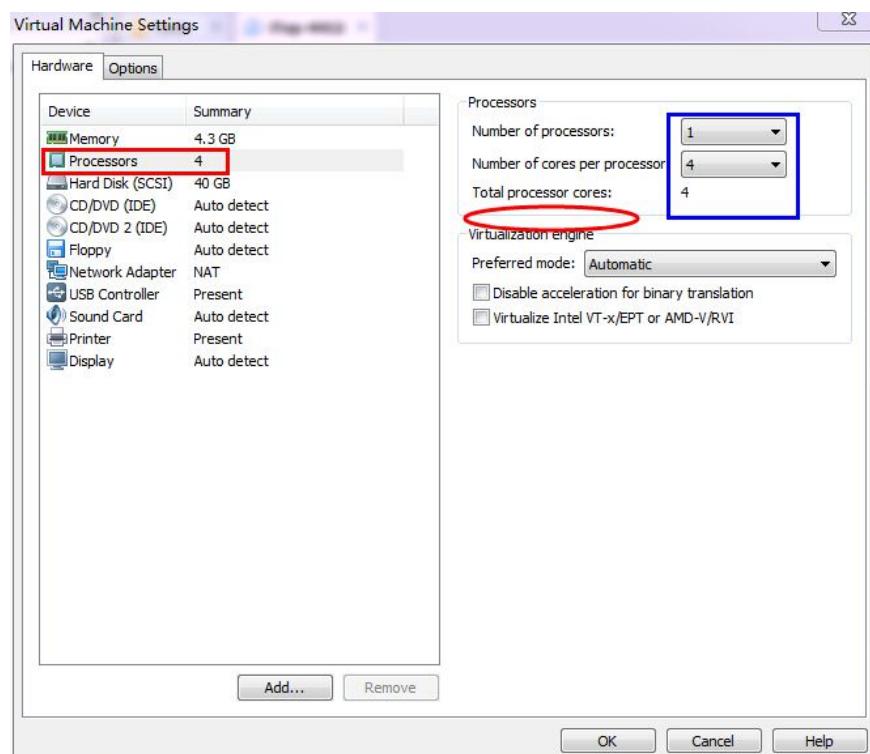
2) 如下图，弹出虚拟机设置窗口“Virtual Machine Settings”。红色框中有三个基础的“Device”需要去设置，蓝色框中的内存“Memory”，CPU核“Processor”，网桥“Network Adapter”。这里先讲内存和CPU核的配置。



3) 如下图，配置内存，根据用户根据实际情况进行配置。下面这台机器中，显示最多可以设置 6G，这里分配 4 个多 G 的内存，内存可以不为整数。



4 ) 如下图 , 配置 CPU 核。举例说明一下 , 下面这台机器中 , 有 4 个核 , 全部分配。如果配置不对 , 椭圆区域会出现带有 “ ! ” 的提示。

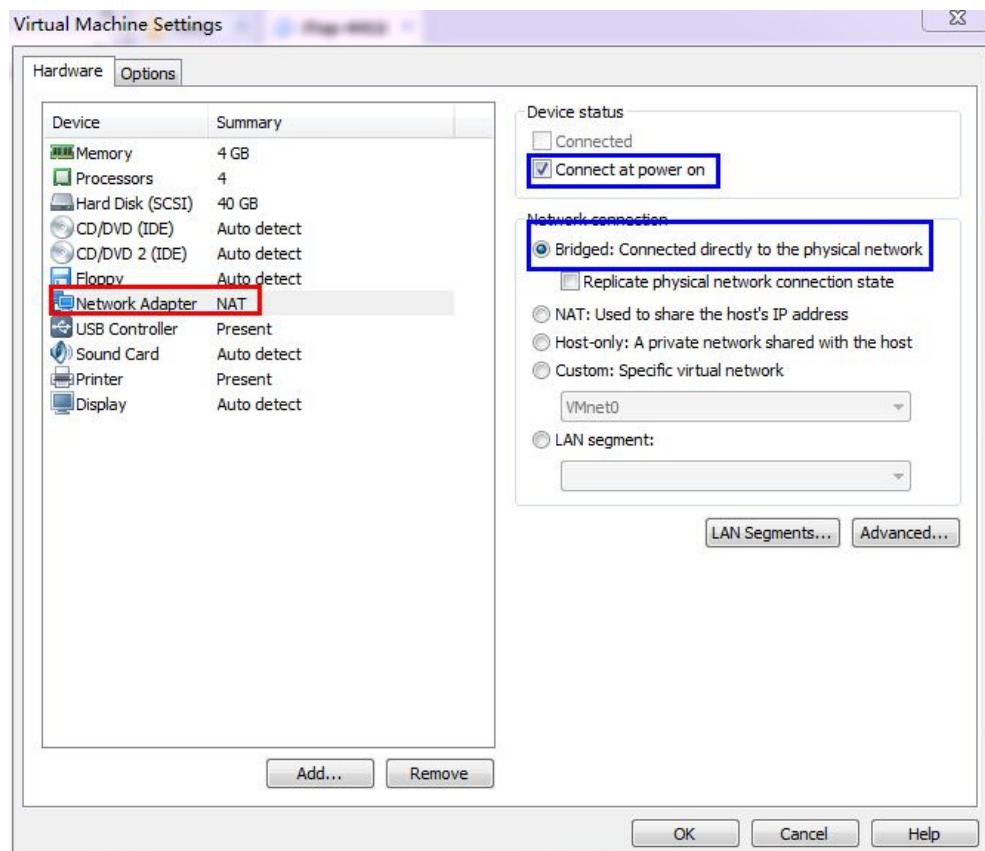


5 ) 基本设置就完成了

下面介绍一下，虚拟机的联网设置。

1) 如下图，这是一般的配置方法，对于带有路由器，能够自动分配 IP 的网络适用。小蓝色框中设置为开机启动网络服务，这个选项要选上；大蓝色框中，设置为“桥接模式”。

这种模式需要路由器能够动态分配 IP，对于很多采用固定 IP 的公司或者部分校园网用户，这种方式是无法联网的。

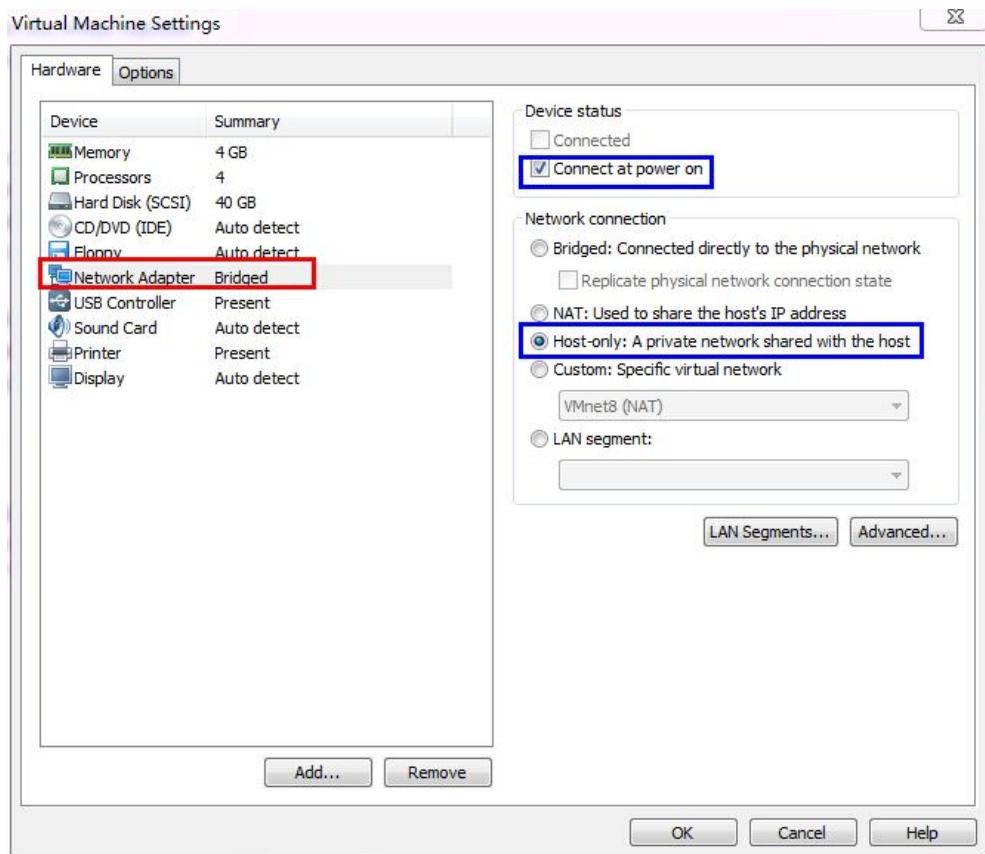


配置成桥接网络连接模式的虚拟机，会被当作主机所在以太网的一部分。虚拟系统和宿主机的关系，就像连接在同一个 Hub 上的两台电脑，可以像主机一样访问以太网中的所有共享资源和网络连接，也可以直接访问互联网。

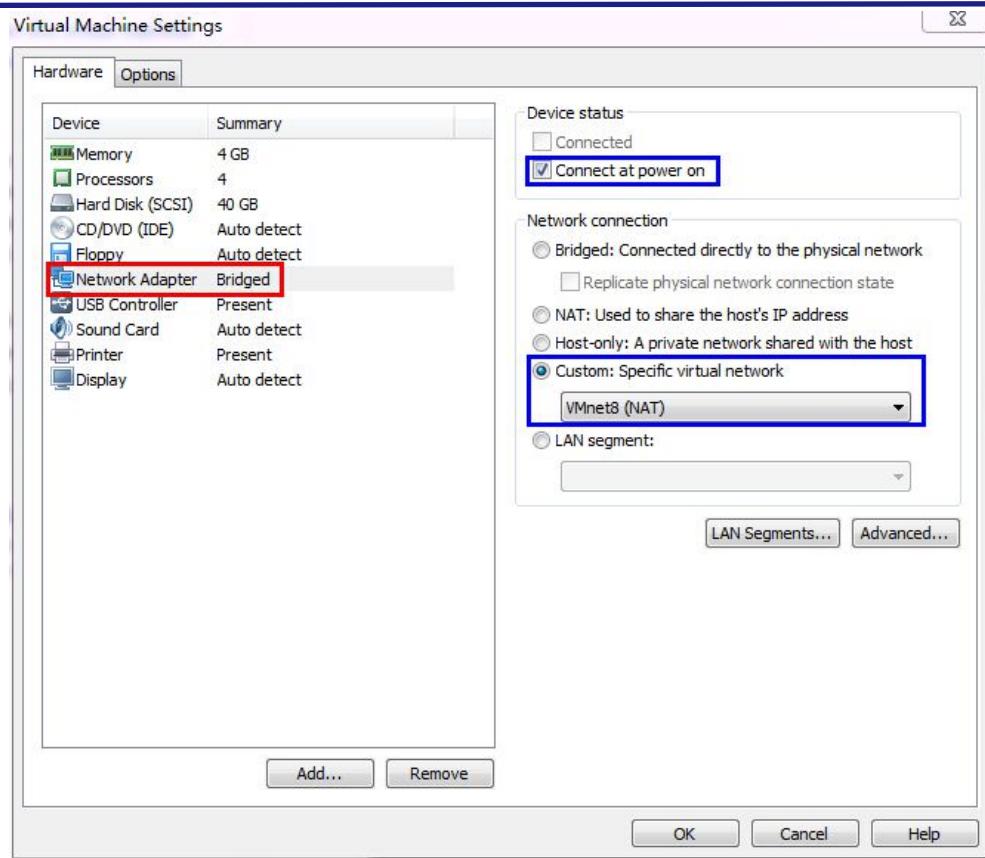
主机与虚拟机之间，以及各虚拟机之间都可以互访。对应虚拟机就被当成主机所在以太网上的一个独立物理机来看待，各虚拟机通过默认的 VMnet0 网卡与主机以太网连接，虚拟机

间的虚拟网络为 VMnet0。这时你的虚拟机就像局域网中的一个独立的物理机一样。虚拟机中的操作系统可以 PING 别的主机，反过来别的主机也可以 PING 虚拟机。

2 ) 如下图 , 选择蓝色框中 “Host-only” 模式 , 这是一种封闭的模式。在这种模式下只能用于主机和 Ubuntu 的通信 , 虚拟机无法上网。在没有网络的情况下 , 主机无法上网 , 为了实现主机和虚拟机的通信 , 可以采用这种模式来设置。



3 ) 如下图 , 选择蓝色框中 “Custom” 模式 , 选择网络 “VMnet8 ( NAT ) ” 。这种模式适用于部分固定 IP 的公司和校园网。当安装虚拟机的时候 , 它会给 PC 机装额外的两个虚拟网卡 , 其中一个就是 “VMnet8” , 相当于 PC 机建了一个虚拟的局域网。虚拟机和 PC 机进行网络连接的时候 , 连入了局域网 , 然后 PC 机连的是外网。



更详细的虚拟机网络配置说明，用户可以去网上查资料。一般情况下，掌握了上面三种设置方法，虚拟机的网络设置就没有问题了。

### 3.2.5 安装和使用 SSH 软件

用户可以安装“SSH 软件”，通过 SSH 软件可以很容易的实现主机和虚拟机之间的通信。SSH 软件不仅要在 Windows 上面安装，还需要在 Ubuntu 系统上面安装。这里讲解的是 SSH 软件在 Windows 上面的安装。

用户可以在网盘“iTOP-4412 开发板所需 PC 软件（工具）”→“04-SSH 软件”目录中下载压缩包“SSH.zip”。

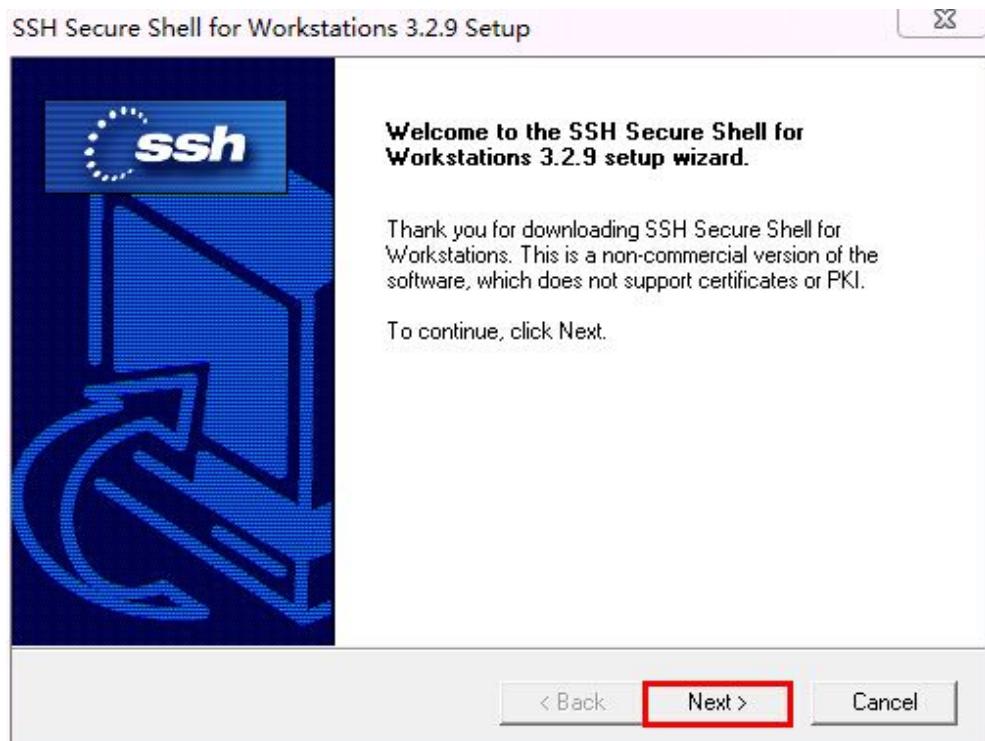
下面详细讲解一下，在 PC 机上如何安装 SSH 软件。

1 ) 解压 SSH 软件“SSH.zip”，得到文件夹“SSH”，进入文件夹，如下图。

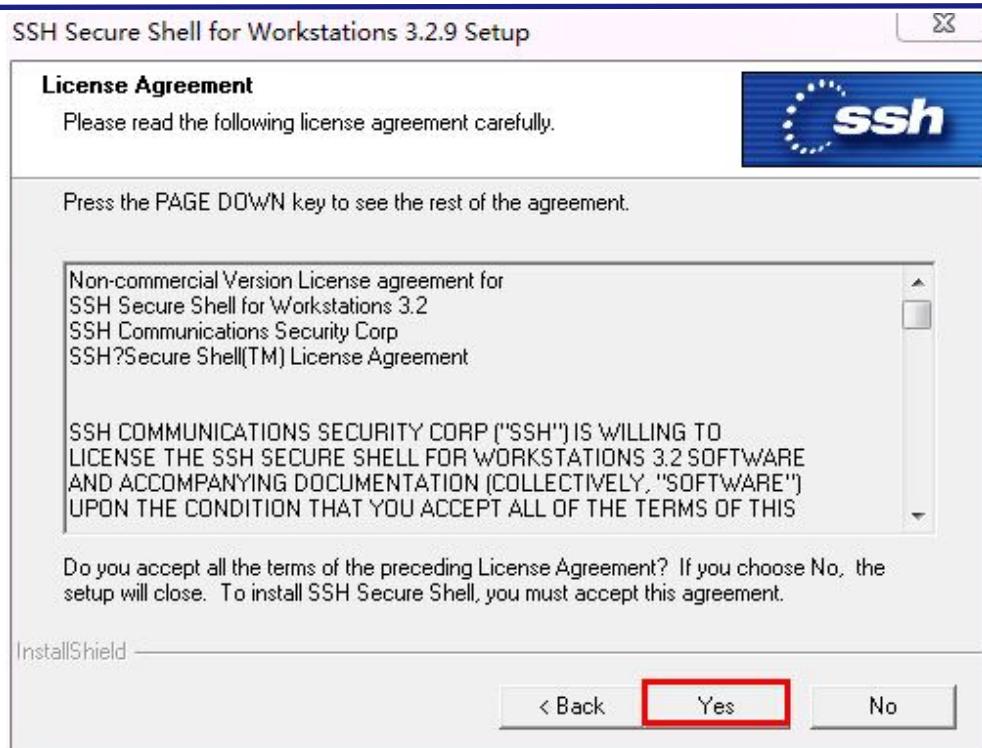
名称

-  SSH Secure Shell
-  SSHSecureShellClient-3.2.9.exe

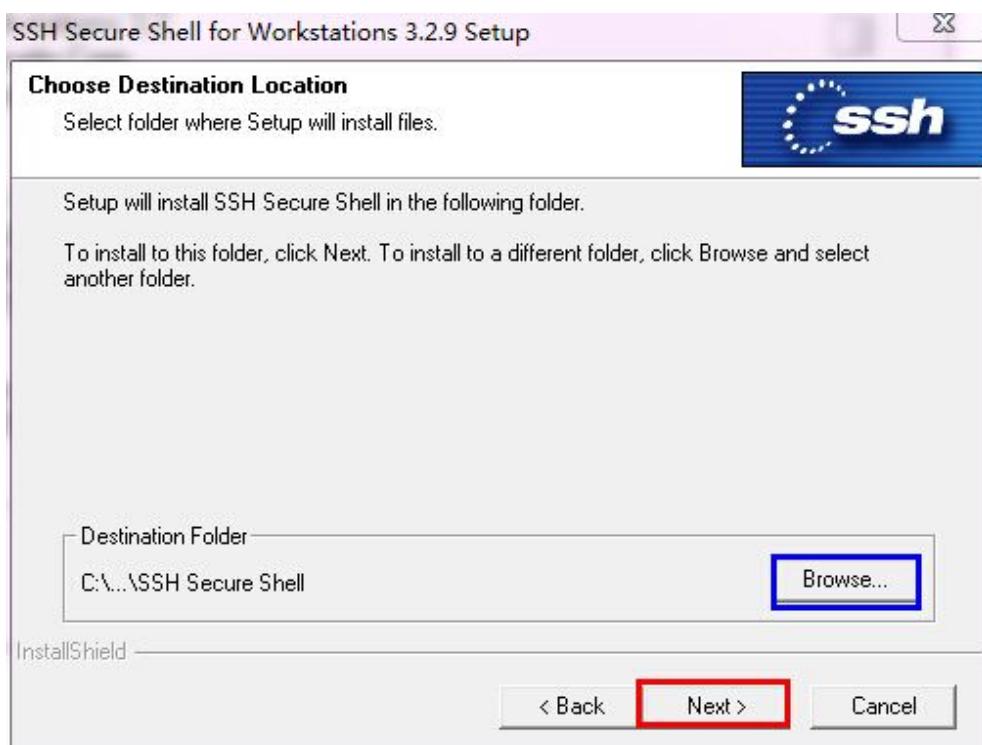
2 ) 双击上图中的软件 “SSHSecureShellClient-3.2.9.exe” , 如下图开始安装 , 单击按钮 “Next” , 继续安装。



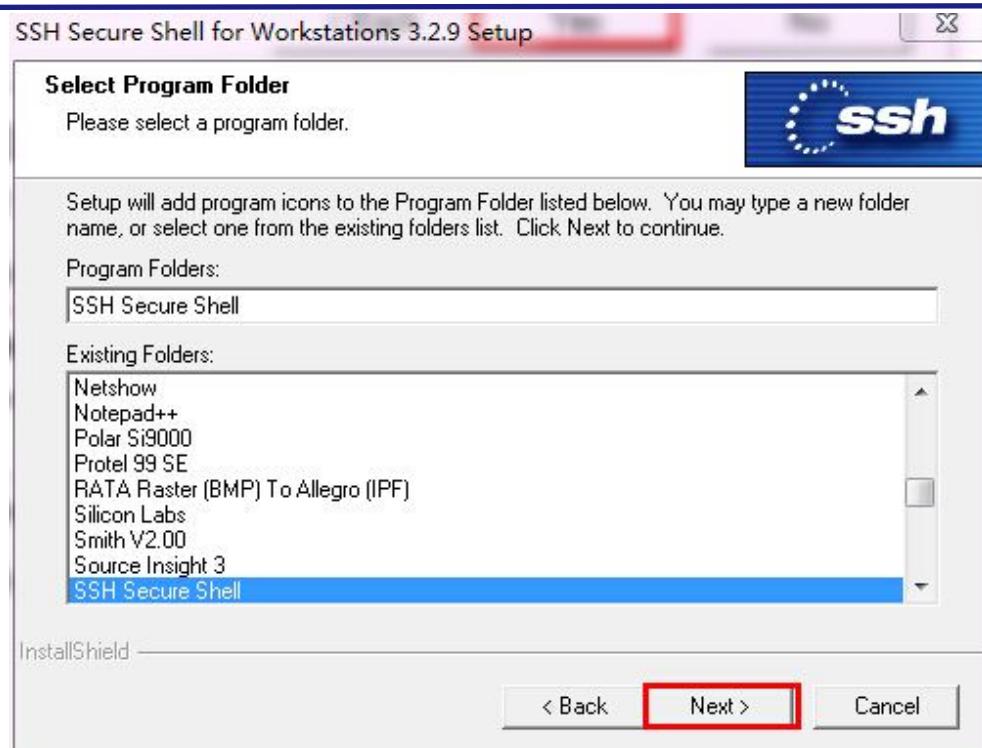
3 ) 如下图 , 单击按钮 “Yes” , 继续安装。



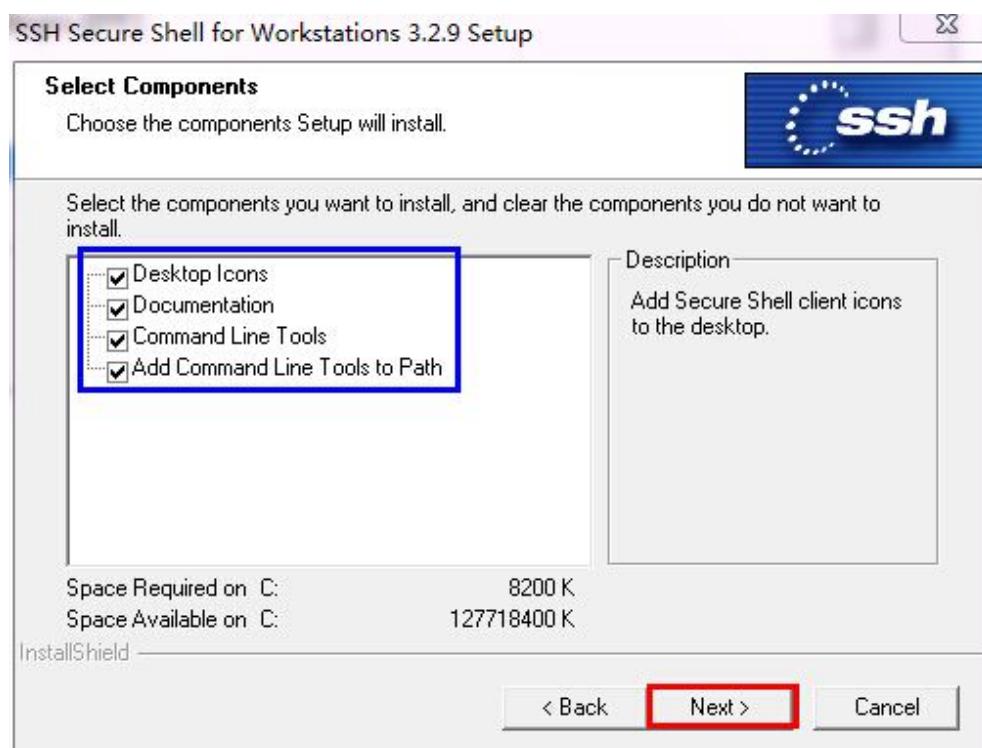
4) 如下图，单击蓝色框中的按钮，用户可以自定义安装路径，这里选择默认，单击红色框中的按钮“Next”。



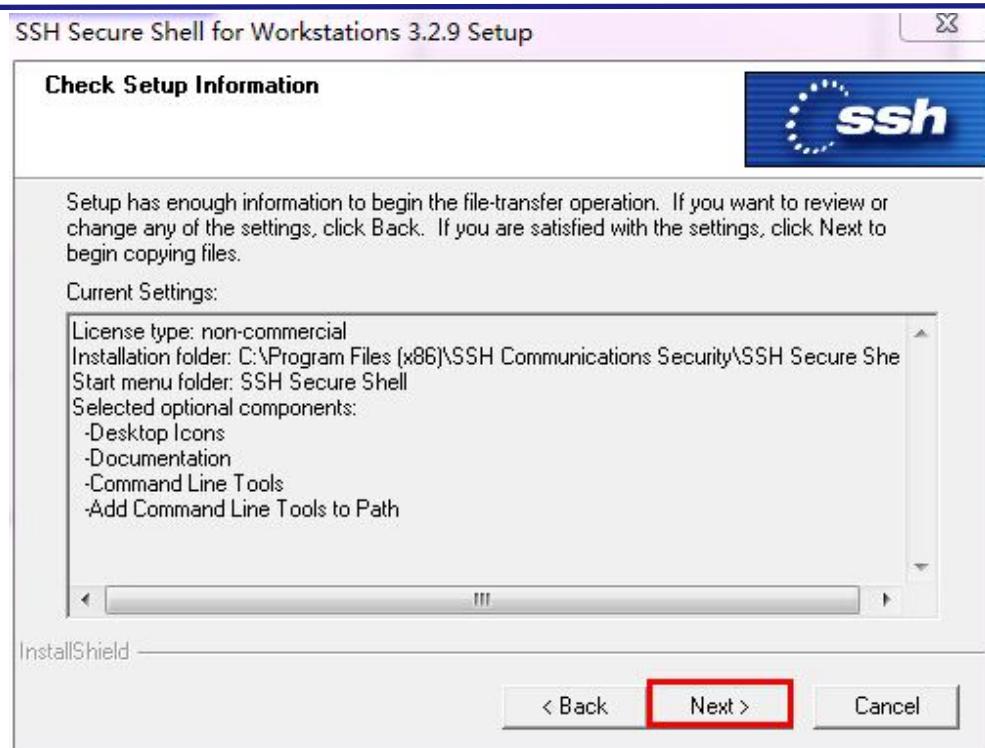
5) 如下图，选择默认配置，单击按钮“Next”继续。



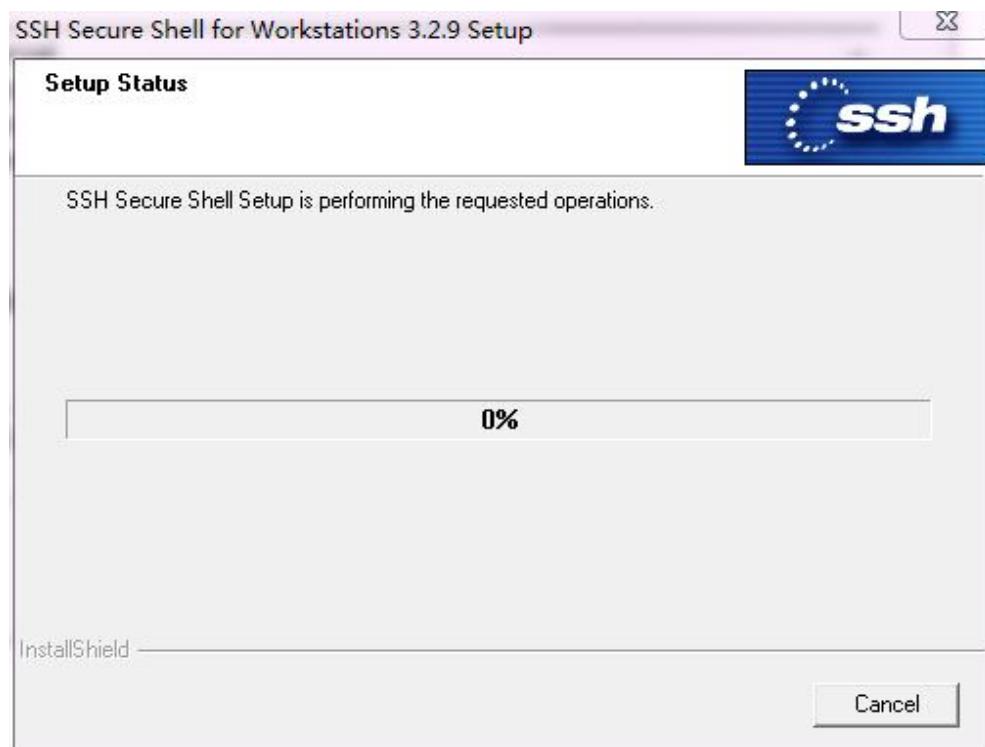
6 ) 如下图 , 按照蓝色框中的配置 , 单击红色框中的按钮 “Next” , 继续。



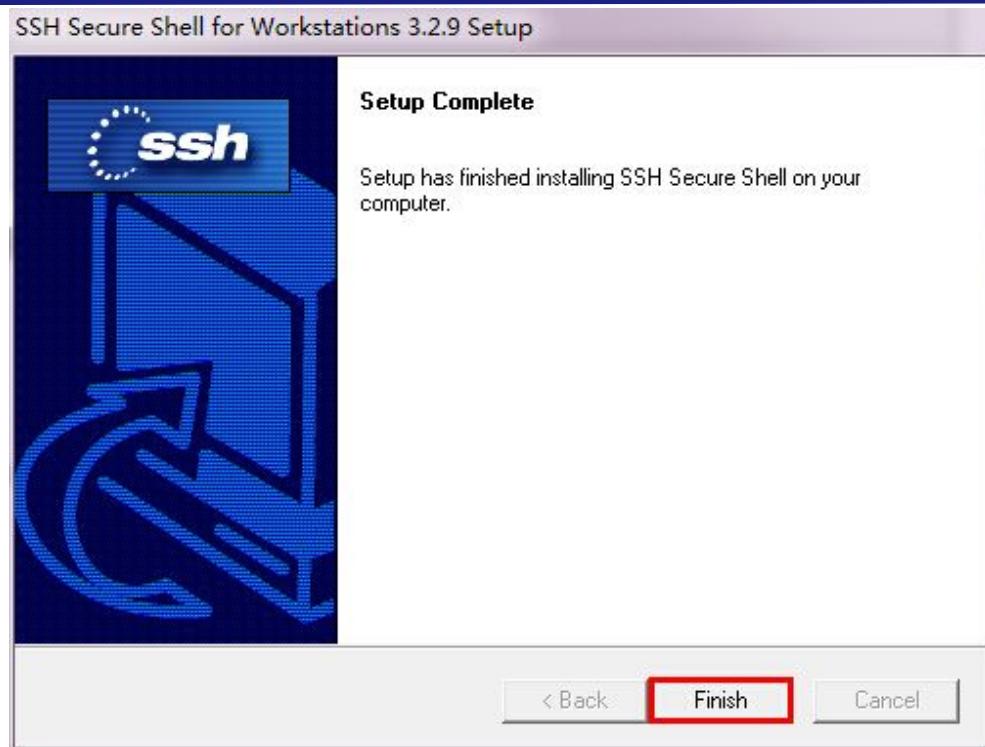
7 ) 如下图 , 选择默认配置 , 单击红色框中的按钮 “Next” , 继续。



8 ) 如下图 , 软件在安装过程中。



9 ) 如下图 , 单击按钮 “Finish” , 完成安装。



10 ) 安装完成后，如下图，有两个图标，上面一个可以用来用做 Ubuntu 系统的终端，下面一个是用来传文件的。



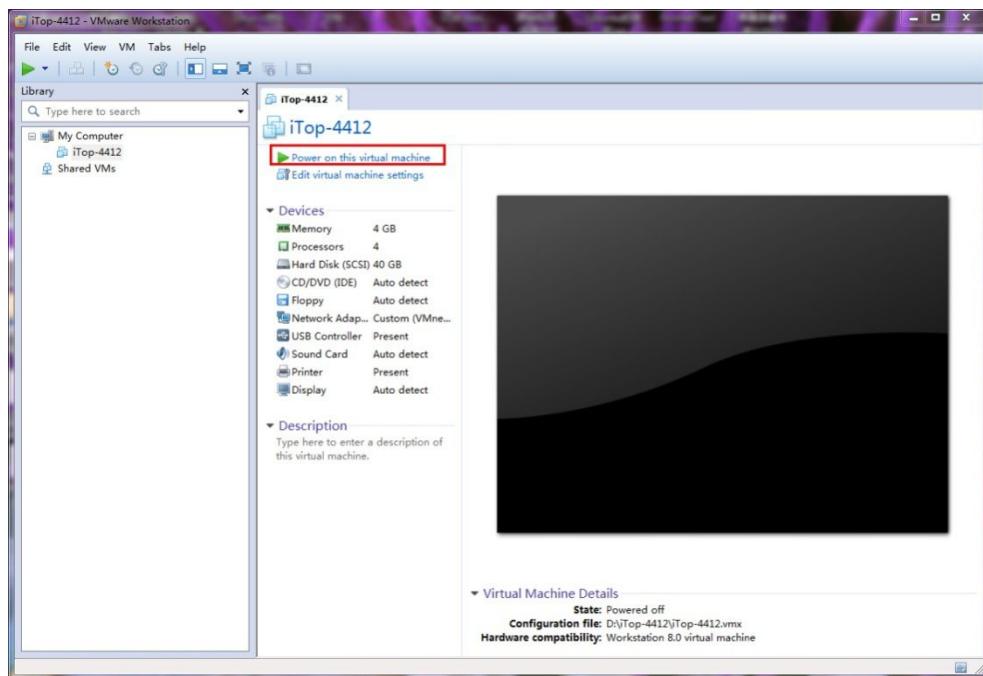
### 3.3 Ubuntu 的基本操作

如果用户以前完全没有接触过 Linux, 在使用开发板之前则需要学会在 Ubuntu 下的一些基本操作。

### 3.3.1 初识 Ubuntu12.04.2 以及 Ubuntu 命令行

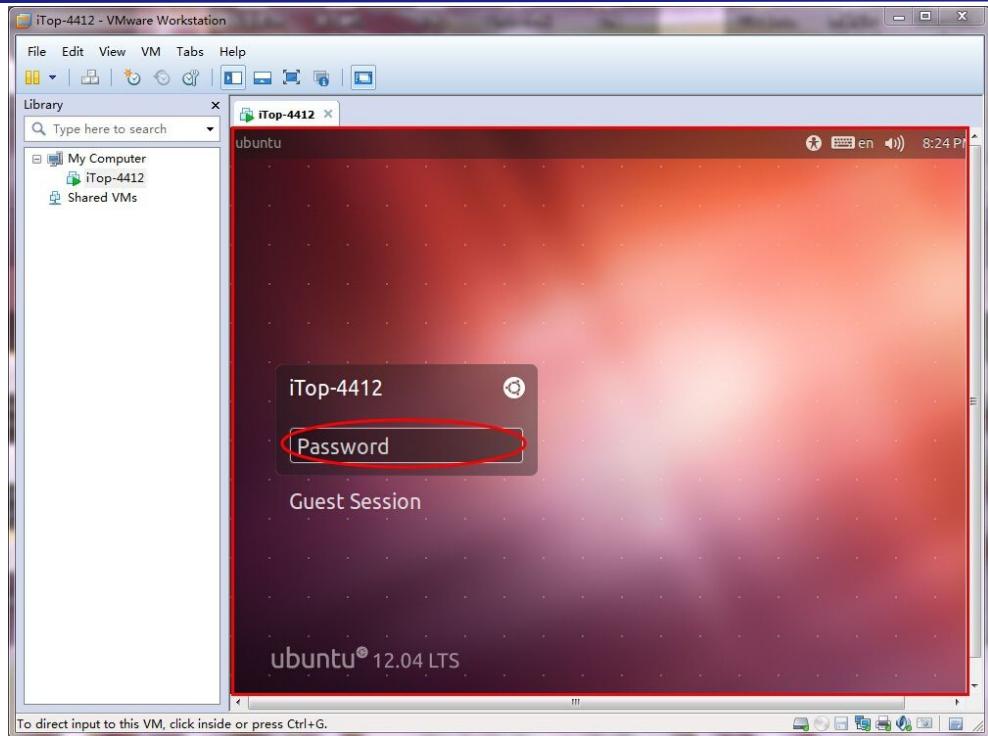
本小节介绍一下 Ubuntu，让用户熟悉一下 Ubuntu 的操作界面，特别是学会使用 Ubuntu 的命令行。在 Ubuntu 下操作，尽可能的去使用命令行，虽然刚开始会觉得很别扭，而且 Ubuntu 带了界面，用鼠标也可以进行部分操作，但是这里还是强烈建议新手直接使用 Ubuntu 命令行。

1)装好虚拟机以及 Ubuntu 之后，进入下图所示的界面。

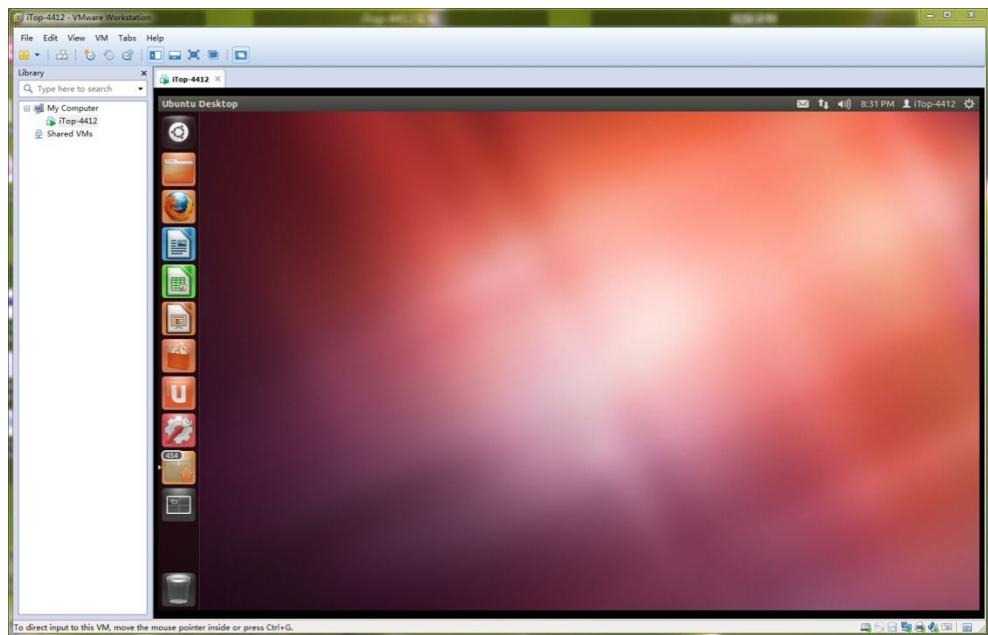


2)单击上图红色框中的“Power on this virtual machine”，开启虚拟机，进入 Ubuntu。如下图，进入“iTop-4412”登录界面，这个用户名“iTop-4412”是前面安装 Ubuntu 时输入用户密码的。在“Passwd”中输入密码“topeet”，密码也是安装 Ubuntu 的时候设置的。输入密码，回车。

需要注意的是，这里还有一个来宾用户，不要使用来宾用户，直接使用如下图所示的“iTOP-4412”用户，这是一个 admin 用户。



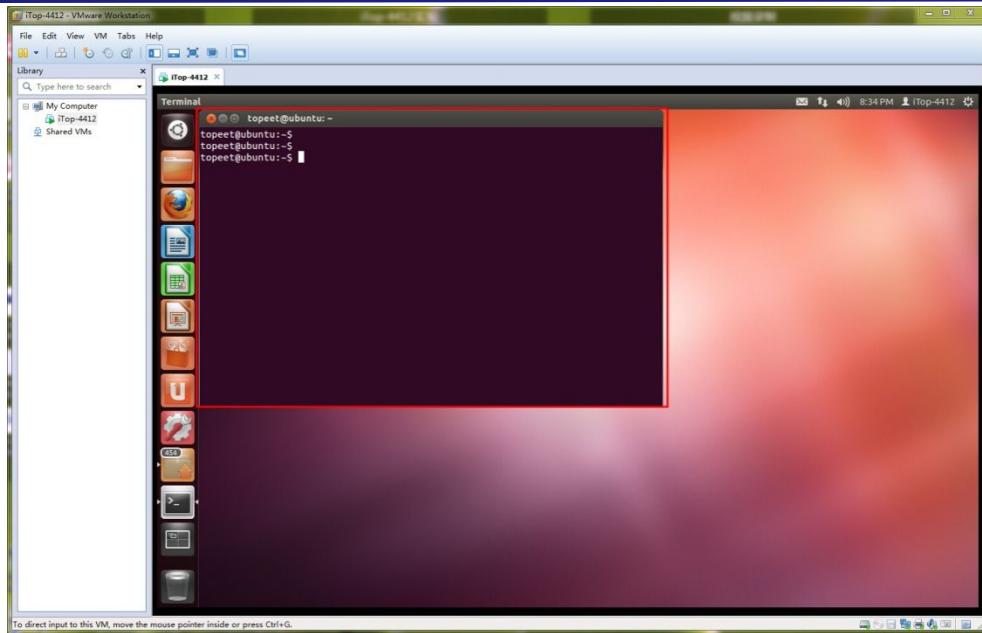
3)如下图，进入 Ubuntu 图形界面，同时按住按键 “Ctrl” + “Alt” + “t”。



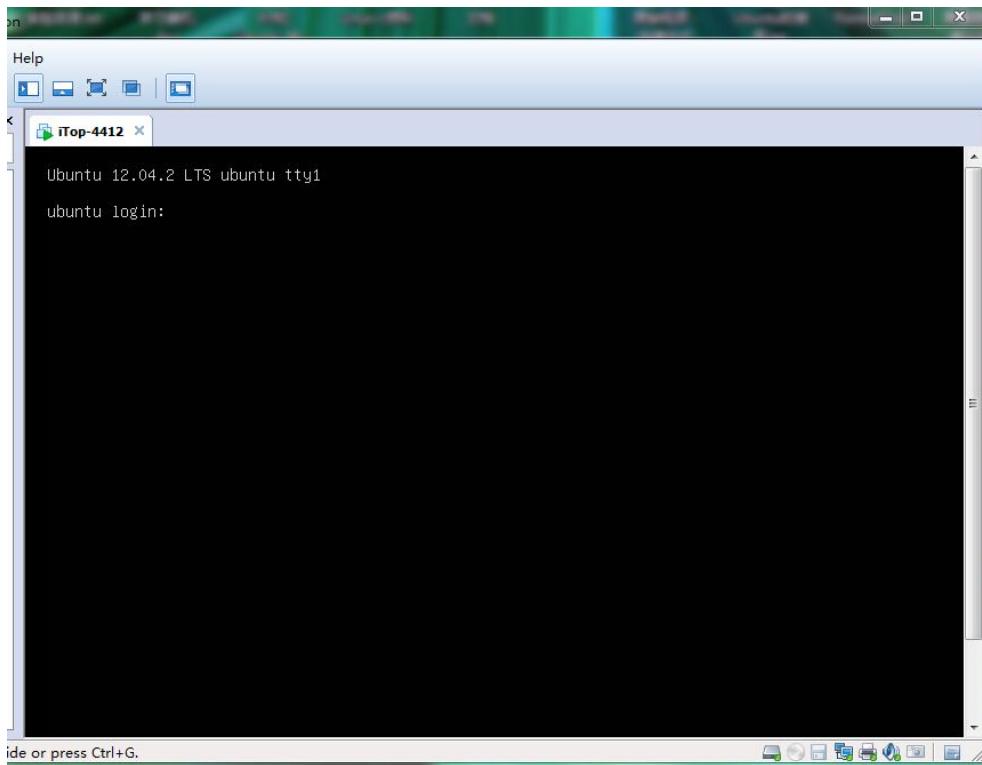
4)如下图，弹出 Ubuntu 命令行终端，终端里面可以输入 Linux 命令。

需要注意的是，这里 Linux 命令，和 2.5.1 小节中的超级终端的“文件系统模式”中，输入的命令是类似的。

按住按键 “Ctrl” + “d” , 就可以退出 Ubuntu 命令行终端。



5)如上图，Ubuntu 命令行终端是属于 Ubuntu 图形界面系统的终端，用户可以尝试进入原始的类似 DOS 系统的终端。按住按键 “Ctrl” + “Alt” + “F1” ，如下图，出现类 Dos 系统的终端。



6)需要返回 Ubuntu 的图形界面，按住按键 “Ctrl” + “Alt” + “F7” 。

类 DOS 界面的终端，一共可以打开六个，命令分别是，“Ctrl+Alt+F1”，“Ctrl+Alt+F2”依此类推。

### 3.3.2 Ubuntu 中启用 root 用户

Ubuntu 中的 root 用户，在初装系统时，是被禁用的。

在安装过程中，提示创建的用户是被分到 admin 组的，使用 admin 组的用户，可以启用并设置 root 帐户的密码。前面提到的用户 topeet 就是属于 admin 组的用户。

如果用户使用的是“编译好的镜像”，则不需要启动 root 用户，在编译好的镜像中，已经启用了 root 用户。

下面给大家介绍启用 root 账号的方法。

先打开 Ubuntu 终端。

在 Ubuntu 命令行中，输入命令：

sudo passwd

接着根据提示，输入当前用户的密码

接着输入超级用户的密码 2 次。

然后就会提示 root 账户密码启动成功。

然后如下图，在 Ubuntu 命令行中，输入登录命令：

su root

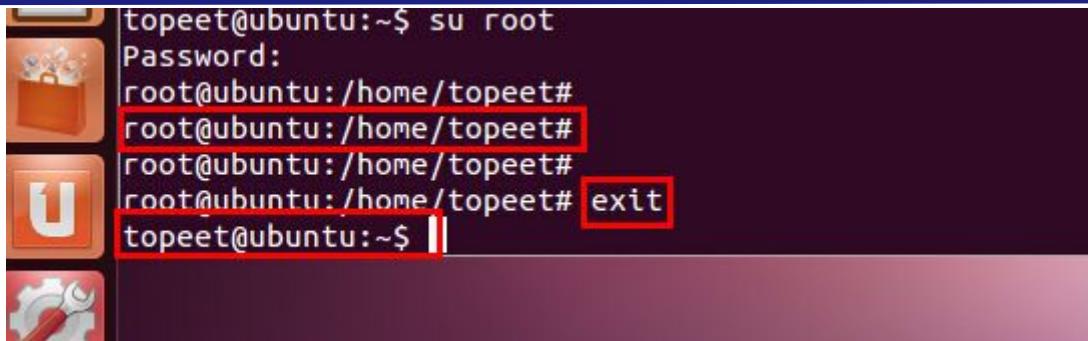
xxx        xxx 表示用户设置的 root 账户密码

A screenshot of a terminal window on an Ubuntu system. The terminal shows the following sequence of commands:

```
topeet@ubuntu:~$  
topeet@ubuntu:~$  
topeet@ubuntu:~$  
topeet@ubuntu:~$ su root  
Password:  
root@ubuntu:/home/topeet#
```

The command "su root" is highlighted with a red box.

如下图，输入“Ctrl” + “d”，可以退出 root 用户，进入 admin 用户。

A screenshot of an Ubuntu desktop environment. On the left, there's a dock with three icons: a shopping bag, a letter 'U', and a gear wrench. The main area shows a terminal window with the following text:

```
topeet@ubuntu:~$ su root
Password:
root@ubuntu:/home/topeet#
root@ubuntu:/home/topeet#
root@ubuntu:/home/topeet#
root@ubuntu:/home/topeet# exit
topeet@ubuntu:~$
```

The lines starting with 'root@ubuntu:' are highlighted with red boxes.

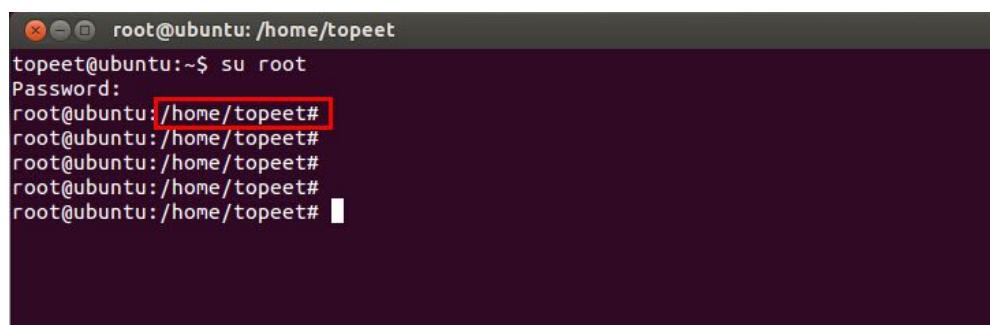
每次重新打开 Ubuntu，如果想要执行最高权限的命令，都需要先登录 root 用户。例如后面的小节要讲的下载命令等都需要 root 用户。

### 3.3.3 Linux 常用 shell 命令

本节介绍 Linux 的基本概念和命令，如果以前没有接触过 Linux 命令，可以参考群共享的文档“华为的内部 linux 教程”来做一下基础练习。在后面开发板的使用过程中，几乎所有的操作都需要用到 Linux 命令。

## 用户文件夹

如下图，这个文件夹是用户文件夹，所有和程序相关的代码、压缩包等等，如果没有特殊说明，都是放到这个文件夹中。

A screenshot of a terminal window with the title bar "root@ubuntu: /home/topeet". The window contains the following text:

```
root@ubuntu:~$ su root
Password:
root@ubuntu:/home/topeet#
root@ubuntu:/home/topeet#
root@ubuntu:/home/topeet#
root@ubuntu:/home/topeet#
root@ubuntu:/home/topeet#
```

The lines starting with 'root@ubuntu:' are highlighted with red boxes.

## 显示命令

语法：ls [选项] [路径]

功能：显示指定工作目录下的内容

主要参数举例：

-a 显示所有文件以及目录。

-l 列出文件名称外，还将文件形态、权限、拥有者、文件大小等详细信息列出。

例如：

ls -l

```
root@ubuntu:/home/topeet# ls -l
total 6192
drwxr-xr-x 4 root root 4096 Oct 30 05:23 Android_JDK
drwxr-xr-x 5 root root 4096 Oct 29 20:01 APP-android
-rw----- 1 topeet topeet 25874432 Oct 30 06:54 core
drwxr-xr-x 2 topeet topeet 4096 Oct 25 01:16 Desktop
drwxr-xr-x 2 topeet topeet 4096 Oct 25 01:16 Documents
drwxr-xr-x 2 topeet topeet 4096 Oct 25 01:16 Downloads
-rw-r--r-- 1 topeet topeet 8445 Oct 25 00:22 examples.desktop
drwxr-xr-x 4 root root 4096 Oct 31 00:40 Linux+QT
drwxr-xr-x 6 root root 4096 Nov 2 23:44 miniLinux
drwxr-xr-x 2 topeet topeet 4096 Oct 25 01:16 Music
drwxr-xr-x 2 topeet topeet 4096 Oct 25 01:16 Pictures
drwxr-xr-x 2 topeet topeet 4096 Oct 25 01:16 Public
drwxr-xr-x 2 root root 4096 Oct 30 05:33 qt
drwxr-xr-x 4 root root 4096 Oct 30 06:47 qt-example
drwxr-xr-x 8 root root 4096 Oct 30 23:11 Qtsoft
drwxr-xr-x 2 topeet topeet 4096 Oct 25 01:16 Templates
drwxr-xr-x 2 topeet topeet 4096 Oct 25 01:16 Videos
drwxr-xr-x 4 root root 4096 Nov 1 02:34 zImage
drwxr-xr-x 3 root root 4096 Nov 3 01:24 zImageM
```

## 查看当前工作路径

语法：pwd

功能：显示当前工作目录的绝对路径

主要参数：无

例如：

pwd

```
root@ubuntu:/home/topeet# pwd
/home/topeet
```

## 切换目录

语法 : cd [路径]

功能 : 切换到指定路径下

主要参数 : 无

例如 : 切换到/home 文件夹下

cd /home

```
root@ubuntu:/home/topeet# cd /home
```

## 清屏

语法 : clear

功能 : 清楚屏幕上的所有内容 , 只保留当前提示符 , 并显示在新屏幕的第 1 行

## 显示和配置网络属性

语法 : ifconfig [interface]

功能 : 查看或设置网络设备属性

主要参数 :

interface : 网络接口的名称 , 如 eth0 ( 网卡 ) ;

up : 激活网络设备 ;

down : 关闭网络设备 ;

add : IP 地址 , 即设置网络设备地址 ;

netmask add : 子网掩码。

例如 :

ifconfig

显示网络配置 , 下图中红色方框中是 IP 地址 , 在后面会用到。

```
root@ubuntu:~# ifconfig
eth0      Link encap:Ethernet HWaddr 00:0c:29:3f:6f:45
          inet addr:192.168.1.117 Bcast:192.168.1.255 Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe3f:6f45/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:5256 errors:0 dropped:0 overruns:0 frame:0
          TX packets:376 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:635024 (635.0 KB) TX bytes:36076 (36.0 KB)

lo       Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING MTU:16436 Metric:1
          RX packets:152 errors:0 dropped:0 overruns:0 frame:0
          TX packets:152 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:41868 (41.8 KB) TX bytes:41868 (41.8 KB)
```

## 新建文件夹

语法 : mkdir [文件夹名]

功能 : 新建一个目录

主要参数 :

-p 如果给出的路径父目录不存在，则同时创建父目录

例如 :

mkdir iTop

```
root@ubuntu:/home/topeet# mkdir iTop
root@ubuntu:/home/topeet#
```

## 删除命令

语法 : rm [选项] [文件以及文件夹]

功能 : 删除档案及目录

主要参数 :

-i 删除前逐一询问确认 ;

-f 即使原档案属性设为唯读，亦直接删除，无需逐一确认 ;

-r 将目录及以下之档案亦逐一删除

例如：

```
rm -rf iTop
```

```
root@ubuntu:/home/topeet# rm -rf iTop
root@ubuntu:/home/topeet#
```

## 压缩和解压命令

语法：tar [选项] [文件目录列表]

功能：对文件目录进行打包备份

主要参数：

-c 建立新的归档文件

-r 向归档文件末尾追加文件

-x 从归档文件中解出文件

可以这样记忆，创建新的文件是 c，追加在原有文件上用 r，从文件中解压出用 x

-O 将文件解开到标准输出

-v 处理过程中输出相关信息

-f 对普通文件操作 - - - 似乎一直都要用 f，不然的话，可能会不显示

-z 调用 gzip 来压缩归档文件，与-x 联用时调用 gzip 完成解压缩

-Z 调用 compress 来压缩归档文件，与-x 联用时调用 compress 完成解压缩

例如：

```
tar -vcf iTop.tar.gz iTop
```

```
root@ubuntu:/home/topeet#
root@ubuntu:/home/topeet# tar -vcf iTop.tar.gz iTop
iTop/
root@ubuntu:/home/topeet# ls
Android_JDK  Documents          iTop.tar.gz  Pictures      Qtsoft       zImageM
APP-android   Downloads          Linux+QT    Public        Templates
core          examples.desktop    miniLinux   qt           Videos
Desktop      iTop               Music      qt-example  zImage
root@ubuntu:/home/topeet#
```

```
tar -vxf iTop.tar.gz
```

```
root@ubuntu:/home/topeet# rm -rf iTop
root@ubuntu:/home/topeet# tar -vxf iTop.tar.gz
iTop/
root@ubuntu:/home/topeet# ls
Android_JDK  Documents          iTop.tar.gz  Pictures    Qtsoft      zImageM
APP-android   Downloads          Linux+QT    Public     Templates
core          examples.desktop   miniLinux   qt         Videos
Desktop      iTop               Music       qt-example zImage
root@ubuntu:/home/topeet#
```

## 拷贝命令

语法 : cp [ 选项 ] 源文件或目录 目标文件或目录

功能 :

主要参数 :

- a 该选项保留链接、文件属性，并递归地拷贝目录，其作用等于 dR 选项的组合。
- d 拷贝时保留链接。
- f 删除已经存在的目标文件而不提示。
- i 与 f 命令相反，在覆盖目标文件之前将给出提示要求用户确认。回答 y 时目标文件将被覆盖，是交互式拷贝。

- p 此时 cp 除复制源文件的内容外，还将把其修改时间和访问权限也复制到新文件中。
- r 若给出的源文件是一目录文件，此时 cp 将递归复制该目录下所有的子目录和文件。

此时目标文件必须为一个目录名。

- l 不作拷贝，只是链接文件。

例如：

先新建一个文件夹 iToptest，然后拷贝

cp -r iToptest iTop

```
root@ubuntu:/home/topeet# cp -r iToptest iTop
```

## 帮助命令

语法 : man

功能 : 阅读参考手册

例如 :

man -ls

```
ls(1)                               User Commands                         ls(1)

NAME
    ls - list directory contents

SYNOPSIS
    ls [OPTION]... [FILE]...

DESCRIPTION
    List information about the FILEs (the current directory by default).
    Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

    Mandatory arguments to long options are mandatory for short options too.

    -a, --all
        do not ignore entries starting with .

    -A, --almost-all
        do not list implied . and ..

    --author
    Manual page ls(1) line 1 (press h for help or q to quit)
```

输入 “Q” 可以退出阅读模式

大写命令的输入方法为按键 “Shift+字母”

这里需要注意一下，man 命令是最难用的，这个命令学会了，所有的其他命令都不是问题。在 man 命令中，都是英文解释，包括很多帮助和说明文档都是英文，这也是为什么给大家强调，学习 Linux 的时候，开发软件尽量用英文的原因。

另外，Linux 基本命令就只能讲到这里了，更多的基本命令，在需要使用的时候再去利用网络去学习对应的命令。

### 3.3.4 Linux 的重要命令 apt-get

安装软件需要使用 “apt-get” 命令，在使用这个命令的时候，需要 Ubuntu 系统联网。在使用这个命令的时候，系统根据文件 “sources.list” 来下载和安装软件，文件

“sources.list” 包含了数据源的地址，数据源放在世界各地的服务器中，给大家推荐使用的数据源是国内的 163 源，更新速度比较快。

另外，执行 apt-get 命令，需要 root 权限。

下面就具体给大家讲一讲，与 “apt-get 命令” 相关的基本操作和命令。

### 3.3.4.1 查看数据源文件

数据源文件 " sources.list "，这个文件中贮存了下载源的地址，用户知道有这个文件就可以了，不用深入研究。

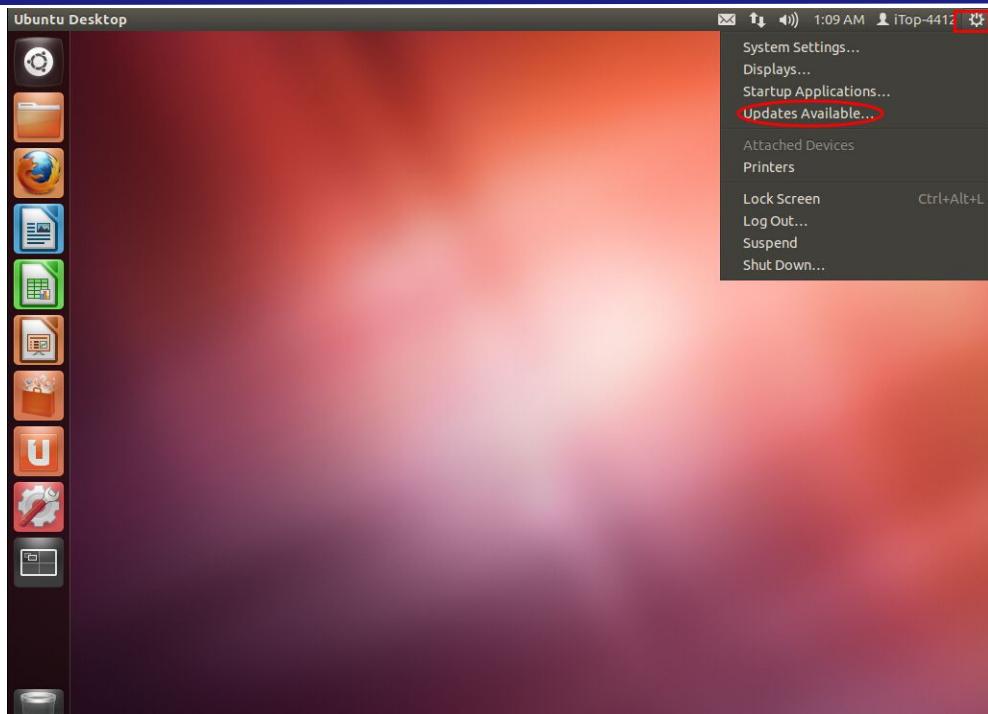
```
ls /etc/apt/
```

```
root@ubuntu:~# ls /etc/apt/
apt.conf.d      sources.list      sources.list.save  trusted.gpg
preferences.d   sources.list.d    trustdb.gpg       trusted.gpg.d
```

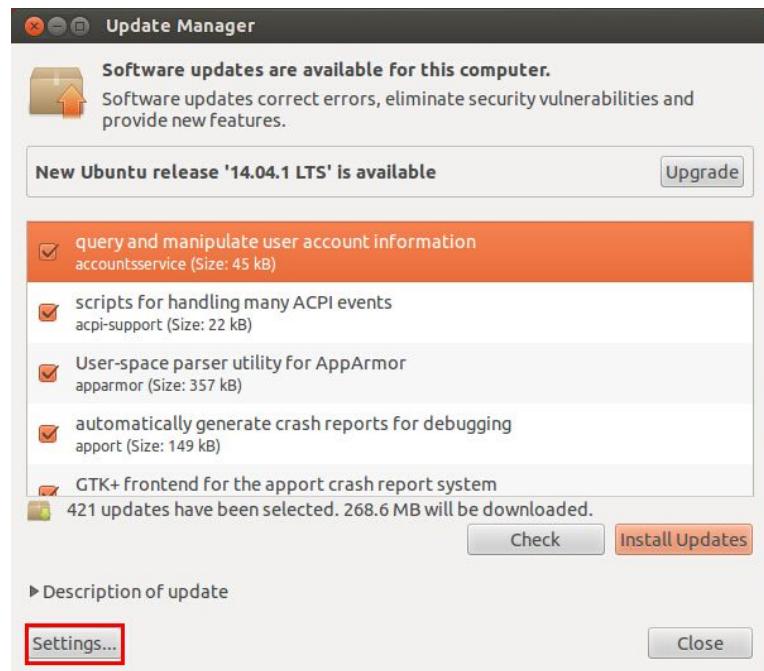
### 3.3.4.2 修改数据源地址

数据源的服务器分布在世界各地，默认安装的 Ubuntu 数据源应该是美国的地址，为了后面下载更新速度更快，需要修改一下数据源地址。

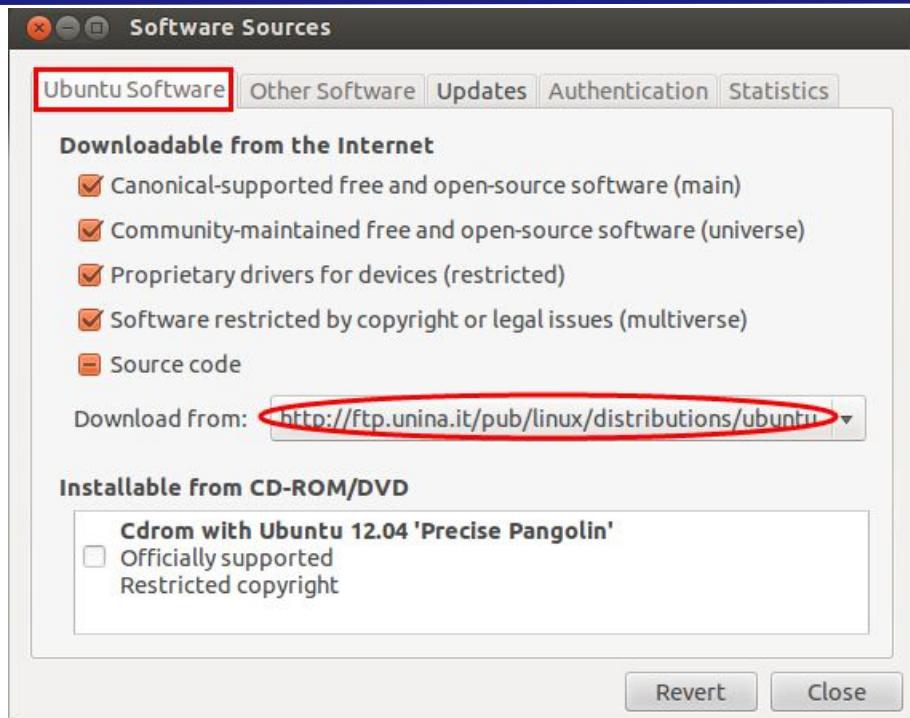
1 ) 如下图，单击红色框中的“齿轮”按钮，然后单击椭圆框中的选项“Updates Available.....”。



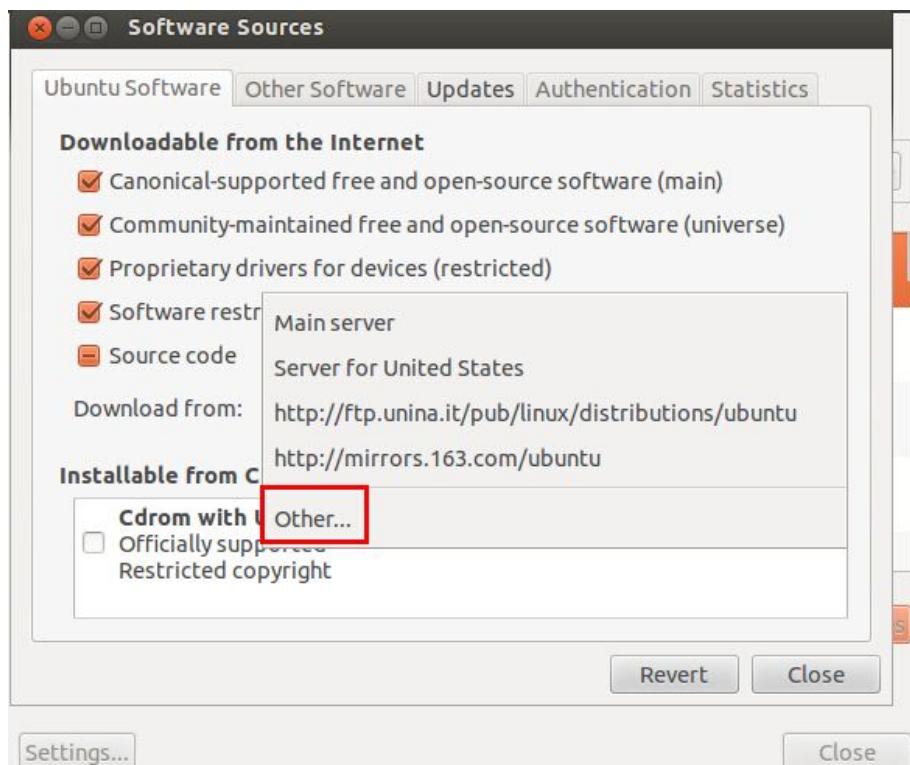
2 ) 系统弹出如下图界面 , 需要注意的是 , 在下图的界面中 , 不要轻易完全更新 , 更新后 Ubuntu 的图形界面 , 一般就不能直接使用了 , 需要重新装显卡驱动。这里单击红色矩形框中的按钮 “Setting” 。



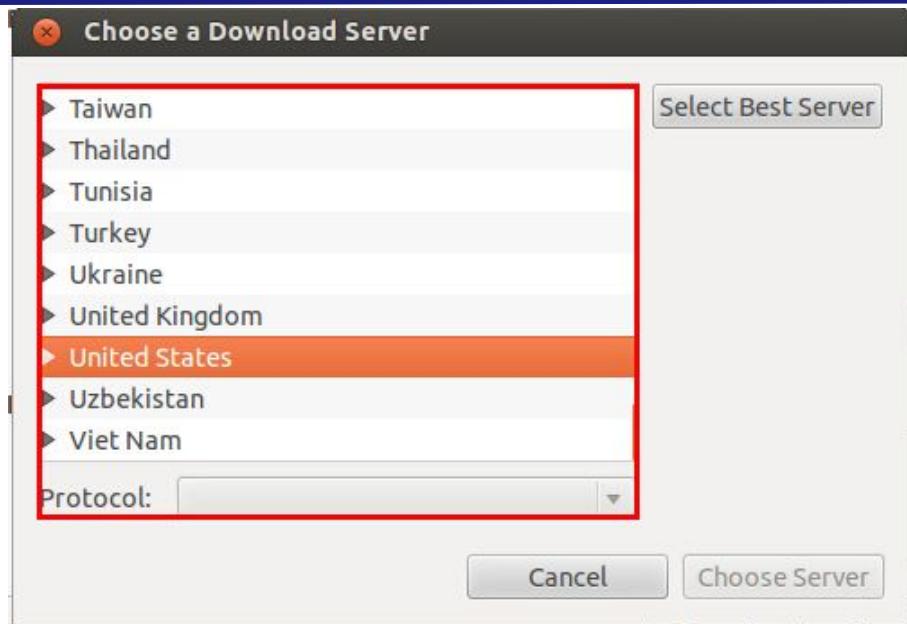
3 ) 如下图 , 选上 “Ubuntu Software” , 然后单击椭圆框中的复选框。



4) 如下图，单击“Other”。



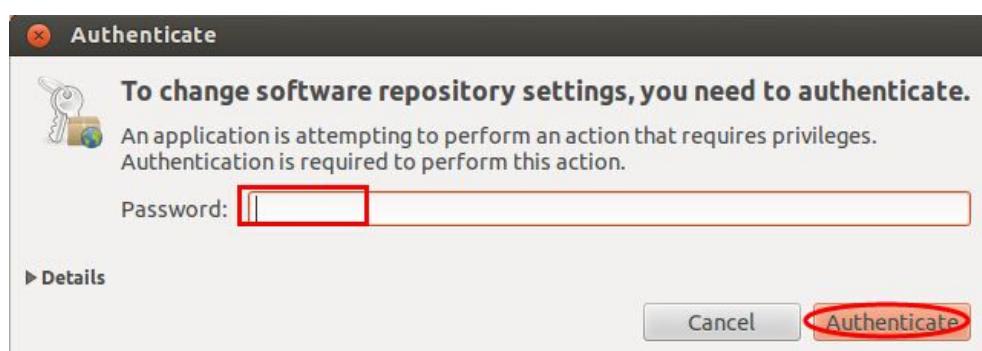
5) 如下图，在红色框中选择国内的服务器。



6 ) 如下图 , 选择 163 服务器 , 单击按钮 “Choose Server” 。



7 ) 如下图 , 输入用户密码 , 单击 “Authenticate” , 数据源地址修改完毕。



### 3.3.4.3 “apt-get update” 命令

修改了数据源的地址后，然后需要使用命令“apt-get update”用来更新数据源列表，数据源会不定期的更新，用户需要及时更新，不然部分软件可能无法安装，如下图，在Ubuntu命令行中，输入命令

apt-get update

```
root@ubuntu:~# apt-get update
Ign http://mirrors.163.com precise InRelease
Ign http://mirrors.163.com precise-updates InRelease
Ign http://mirrors.163.com precise-backports InRelease
Ign http://mirrors.163.com precise-security InRelease
Hit http://mirrors.163.com precise Release.gpg
Hit http://mirrors.163.com precise-updates Release.gpg
Hit http://mirrors.163.com precise-backports Release.gpg
Hit http://mirrors.163.com precise-security Release.gpg
Hit http://mirrors.163.com precise Release
Hit http://mirrors.163.com precise-updates Release
```

如下图，更新完毕。

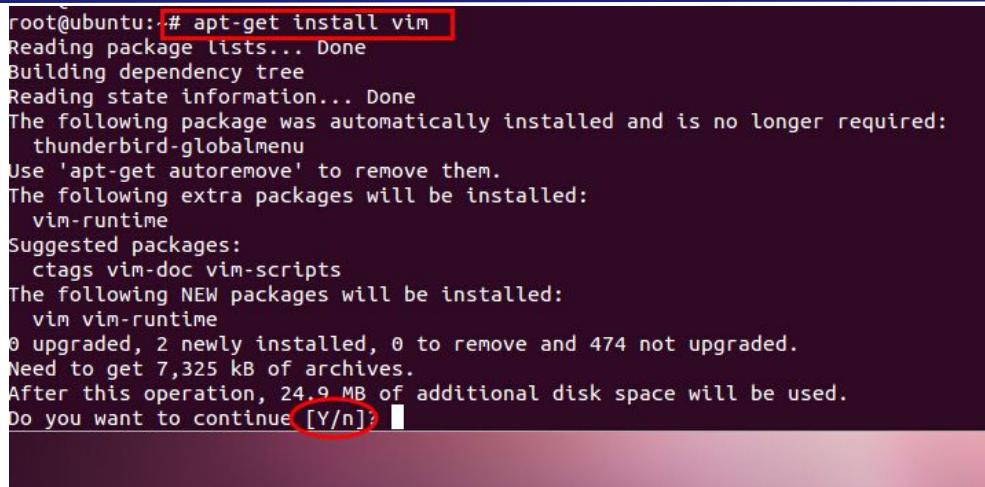
```
Hit http://mirrors.163.com precise-backports/universe Translation-en
Hit http://mirrors.163.com precise-security/main Translation-en
Hit http://extras.ubuntu.com precise/main amd64 Packages
Hit http://extras.ubuntu.com precise/main i386 Packages
Ign http://extras.ubuntu.com precise/main TranslationIndex
Hit http://mirrors.163.com precise-security/multiverse Translation-en
Hit http://mirrors.163.com precise-security/restricted Translation-en
Hit http://mirrors.163.com precise-security/universe Translation-en
Ign http://extras.ubuntu.com precise/main Translation-en_US
Ign http://extras.ubuntu.com precise/main Translation-en
Reading package lists... Done
root@ubuntu:~#
```

### 3.3.4.4 “apt-get install” 命令

“apt-get install”命令是安装和更新软件的命令。如果用户自己安装的Ubuntu系统，那么就有几个基本软件需要安装。

如下图，安装“vim 编辑器”。在Ubuntu命令行中，输入命令

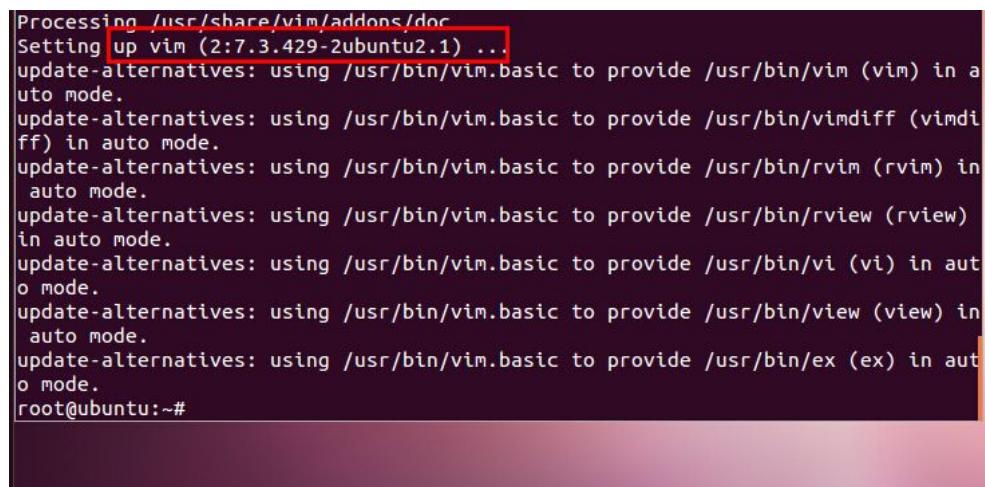
apt-get install vim



```
root@ubuntu:~# apt-get install vim
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  thunderbird-globalmenu
Use 'apt-get autoremove' to remove them.
The following extra packages will be installed:
  vim-runtime
Suggested packages:
  ctags vim-doc vim-scripts
The following NEW packages will be installed:
  vim vim-runtime
0 upgraded, 2 newly installed, 0 to remove and 474 not upgraded.
Need to get 7,325 kB of archives.
After this operation, 24.9 MB of additional disk space will be used.
Do you want to continue [Y/n]? █
```

如上图，在使用命令" apt-get install "安装和更新软件过程中，可能需要用户选择一下，常见的选择包括如下图 "Y/n" ，回车按键 "Enter" 等，用户在使用" apt-get install "中，直接选择确定就可以了。

如下图，更新完成



```
Processing /usr/share/vim/addons/doc
Setting up vim (2:7.3.429-2ubuntu2.1) ...
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/vim (vim) in auto mode.
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/vimdiff (vimdiff) in auto mode.
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/rvim (rvim) in auto mode.
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/rview (rview) in auto mode.
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/vi (vi) in auto mode.
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/view (view) in auto mode.
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/ex (ex) in auto mode.
root@ubuntu:~#
```

如下图，安装 "SSH" 软件。在 Ubuntu 命令行中，输入命令

apt- get install ssh

```
root@ubuntu:~# apt-get install ssh
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  thunderbird-globalmenu
Use 'apt-get autoremove' to remove them.
The following extra packages will be installed:
  openssh-client openssh-server ssh-import-id
Suggested packages:
  libpam-ssh keychain monkeysphere openssh-blacklist openssh-blacklist-extra
  rssh molly-guard
The following NEW packages will be installed:
  openssh-server ssh ssh-import-id
The following packages will be upgraded:
  openssh-client
1 upgraded, 3 newly installed, 0 to remove and 473 not upgraded.
Need to get 1,289 kB of archives.
After this operation, 910 kB of additional disk space will be used.
Do you want to continue [Y/n]? 
```

如下图，安装完成。

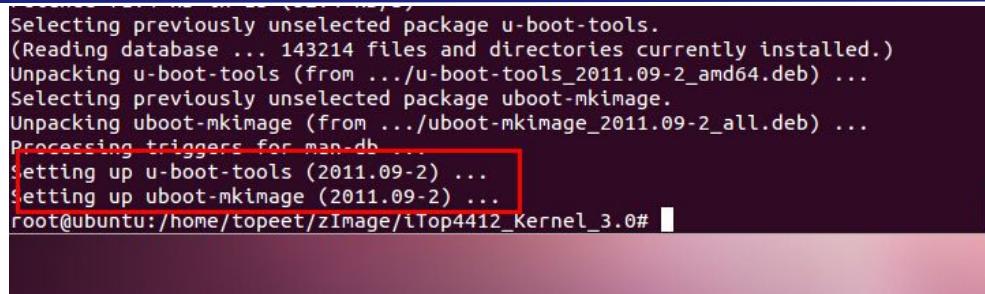
```
... ...
Selecting previously unselected package ssh.
Unpacking ssh (from .../ssh_1%3a5.9p1-5ubuntu1.4_all.deb) ...
Selecting previously unselected package ssh-import-id.
Unpacking ssh-import-id (from .../ssh-import-id_2.10-0ubuntu1_all.deb) ...
Processing triggers for man-db ...
Processing triggers for ureadahead ...
ureadahead will be reprofiled on next reboot
Processing triggers for ufw ...
Setting up openssh-client (1:5.9p1-5ubuntu1.4) ...
Setting up openssh-server (1:5.9p1-5ubuntu1.4) ...
Creating SSH2 RSA key; this may take some time ...
Creating SSH2 DSA key; this may take some time ...
Creating SSH2 ECDSA key; this may take some time ...
ssh start/running, process 3979
Setting up ssh (1:5.9p1-5ubuntu1.4) ...
Setting up ssh-import-id (2.10-0ubuntu1) ...
root@ubuntu:~# 
```

安装工具“uboot-mkimage”。在 Ubuntu 命令行中，输入命令

apt-get install uboot-mkimage

```
root@ubuntu:/home/topeet/zImage/iTop4412_Kernel_3.0# apt-get install uboot-mkimage
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  thunderbird-globalmenu 
```

如下图，更新完成。



```
Selecting previously unselected package u-boot-tools.  
(Reading database ... 143214 files and directories currently installed.)  
Unpacking u-boot-tools (from .../u-boot-tools_2011.09-2_amd64.deb) ...  
Selecting previously unselected package uboot-mkimage.  
Unpacking uboot-mkimage (from .../uboot-mkimage_2011.09-2_all.deb) ...  
Processing triggers for man-db ...  
Setting up u-boot-tools (2011.09-2) ...  
Setting up uboot-mkimage (2011.09-2) ...  
root@ubuntu:/home/topeet/zImage/iTop4412_Kernel_3.0#
```

### 3.3.5 安装和使用 SSH 软件

Ubuntu 系统和 Windows 主机之间经常需要传文件，在以往的 XP 系统中，通常做法是使用“共享文件夹”的方式来解决这个问题。但是在 Win7 系统下，使用“共享文件夹”的方式，需要进行系统设置的地方非常多，而且效率很低。这里推荐大家使用“SSH 软件”在主机和虚拟机之间传文件。

#### 3.3.5.1 安装 SSH 软件

在前面讲解“apt-get”命令的时候，已经给 Ubuntu 安装了 SSH 软件，这里还需要给 Win7 安装 SSH 软件。

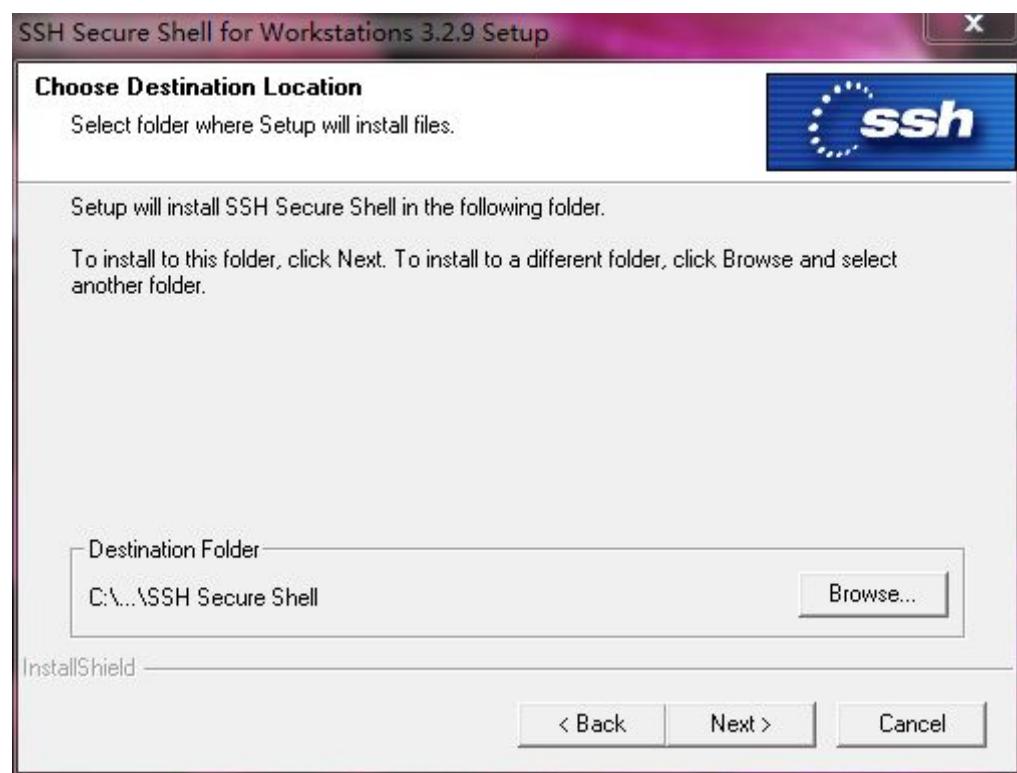
1 ) SSH 软件压缩包可以在网盘下载，下载后解压，进入解压出来的文件夹，如下图。

名称	修改日期	类型	大小
SSH Secure Shell	2014/6/18 12:39	文件夹	
SSHSecureShellClient-3.2.9.exe	2014/6/18 12:39	应用程序	5,388 KB

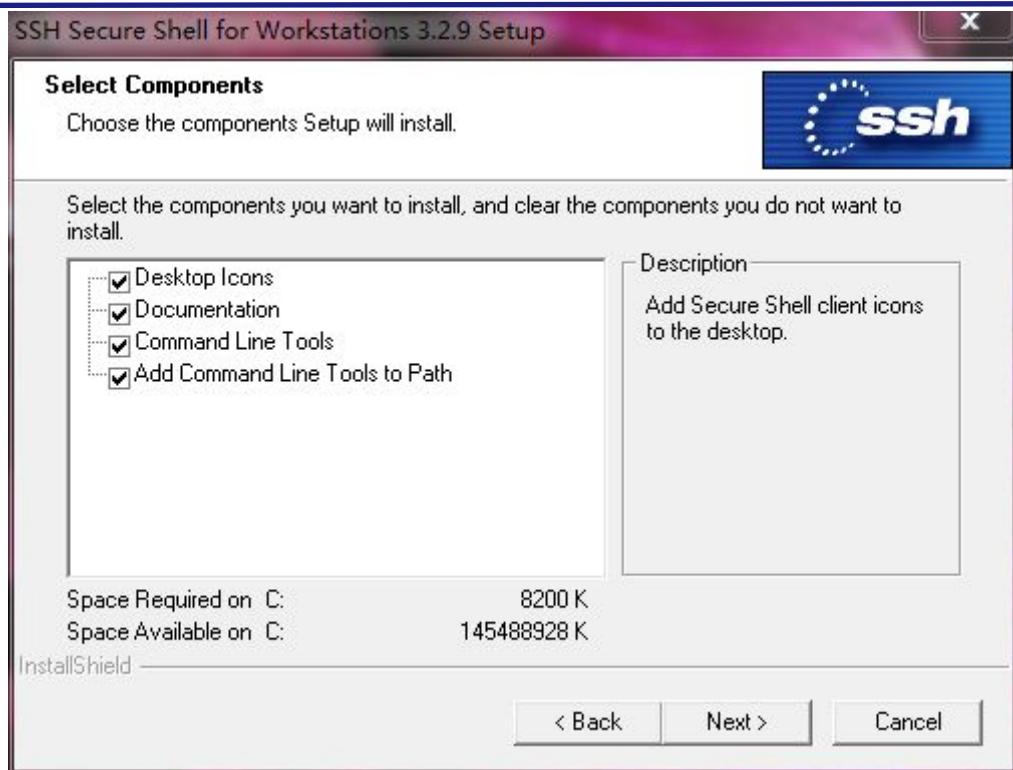
2 ) 单击上图中的“SSHSecureShellClient-3.2.9.exe”，开始安装，如下图。



3 ) 单击按钮 “Next” 两次，进入如下图所示界面，选择 SSH 软件的安装目录，下图中选择默认。单击按钮 “Next” ，继续安装。



4 ) 如下图，选择默认，单击按钮 “Next” ，继续安装。



5 ) 连续单击按钮 “Next” , 自动安装 , 直到出现如下图界面 , 单击按钮 “Finish” , 安装结束。



6 ) 如下图 , 生成桌面图标。



### 3.3.5.2 使用 SSH 软件传文件

下面介绍“SSH Secure File Transfer Client”的使用方法。

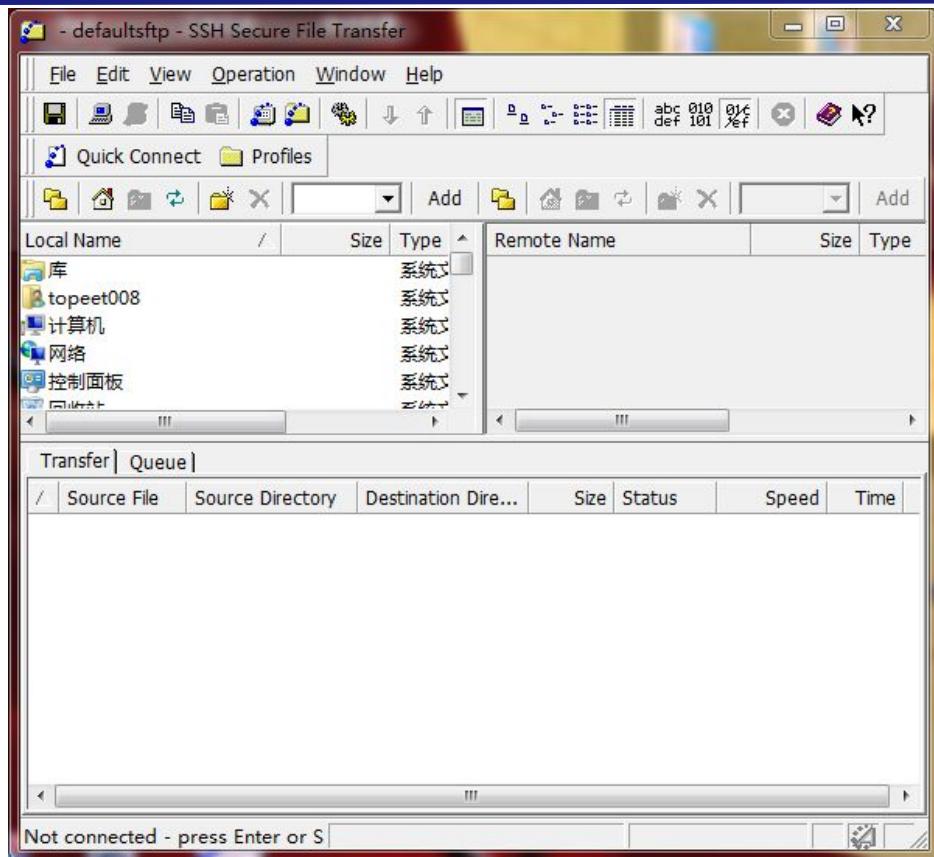
1) 进入虚拟机的 Ubuntu 系统，使用 ifconfig 命令，确定 Ubuntu 的 IP 地址。如下图所示，用户的 IP 地址，需要自己确认，这里只是以作者机器的 IP 为例。

```
root@ubuntu:/home/topeet
Creating SSH2 ECDSA key; this may take some time ...
ssh start/running, process 12277
Setting up ssh (1:5.9p1-5ubuntu1.4) ...
Setting up ssh-import-id (2.10-0ubuntu1) ...
root@ubuntu:/home/topeet# ifconfig
eth0      Link encap:Ethernet HWaddr 00:0c:29:0d:e1:02
          inet addr:192.168.198.133 bcast:192.168.198.255 Mask:255.255.255.0
                      inet6 addr: fe80::20c:29ff:fe0d:e102/64 Scope:Link
                        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
                        RX packets:7231 errors:0 dropped:0 overruns:0 frame:0
                        TX packets:3887 errors:0 dropped:0 overruns:0 carrier:0
                        collisions:0 txqueuelen:1000
                        RX bytes:9063557 (9.0 MB) TX bytes:271745 (271.7 KB)

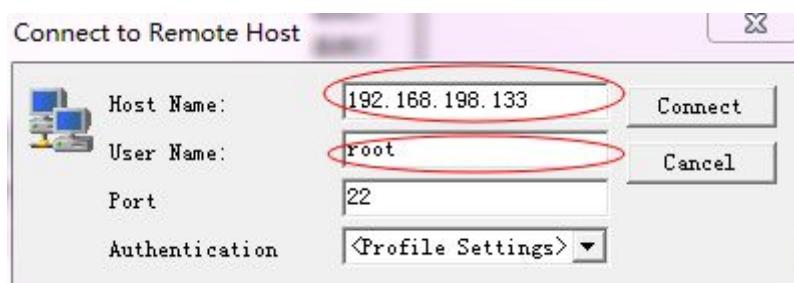
lo        Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK RUNNING MTU:16436 Metric:1
            RX packets:98 errors:0 dropped:0 overruns:0 frame:0
            TX packets:98 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:8425 (8.4 KB) TX bytes:8425 (8.4 KB)

root@ubuntu:/home/topeet#
```

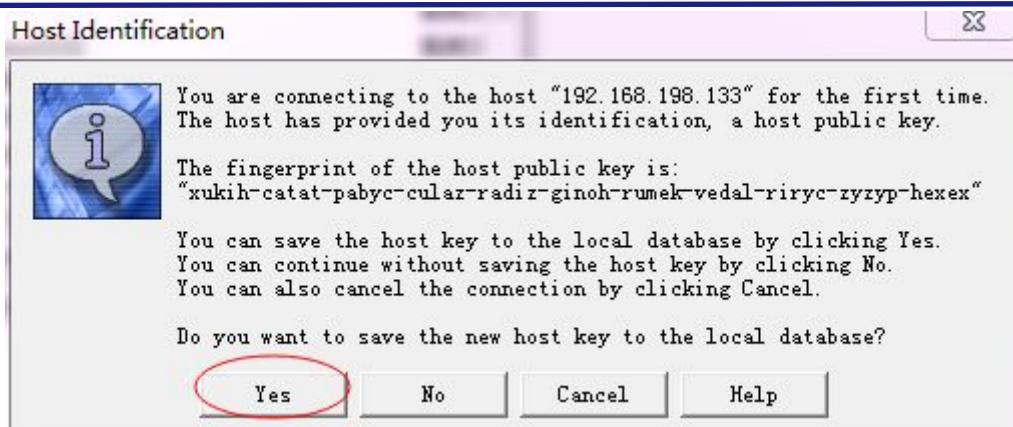
2) 双击快捷方式“SSH Secure File Transfer Client”，打开 SSH 软件，如下图。



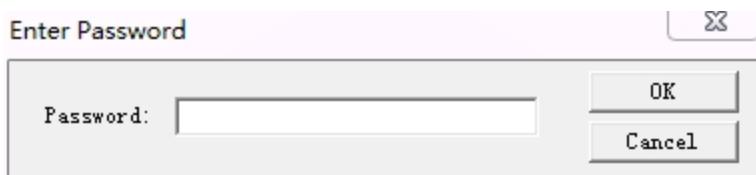
3 ) 在上图的界面中 , 单击菜单 “File” --> “Quick Connect” , 然后 , 会弹出如下图所示的对话框 , 在对话框中的 “User Name” 里面输入 “root” 。 “Host Name” 中输入的是前面查看到的 Ubuntu 的 IP 地址 , “root” 用户需要在安装 Ubuntu 之后新建。



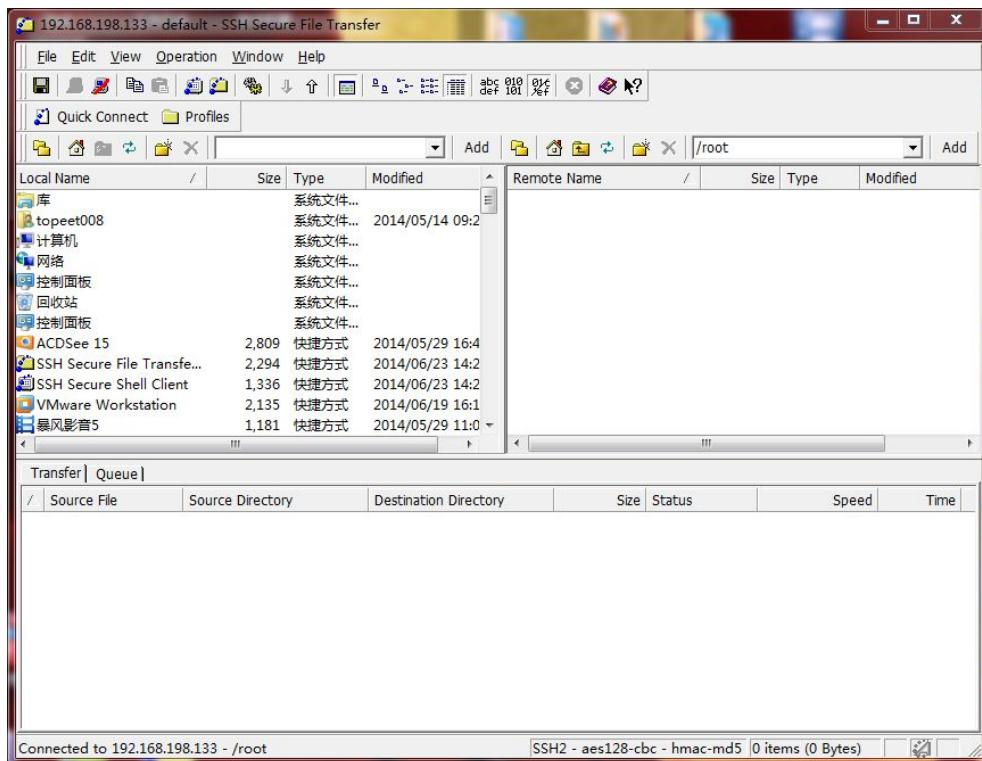
4 ) 如下图 , 在弹出的窗口中 , 单击按钮 “Yes” 。



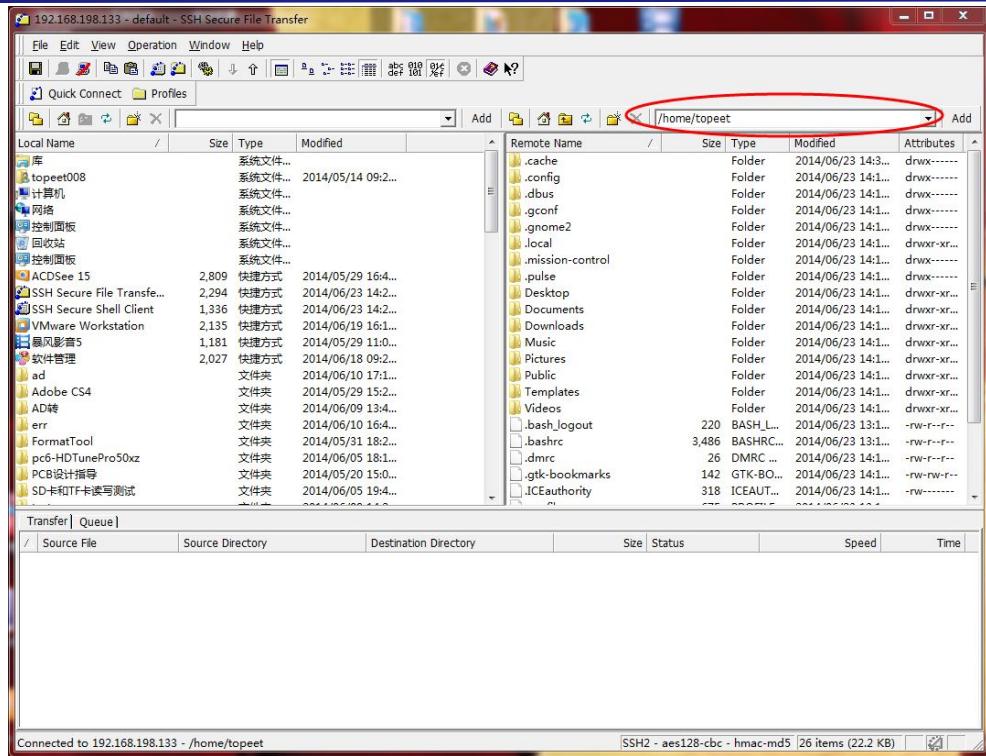
5 ) 如下图，在弹出的窗口中，输入用户密码。如果用户使用的是“搭建好的镜像”，密码是“topeet”。



6 ) 如下图显示，Windows 的 SSH 软件已经连接到 Ubuntu 了。

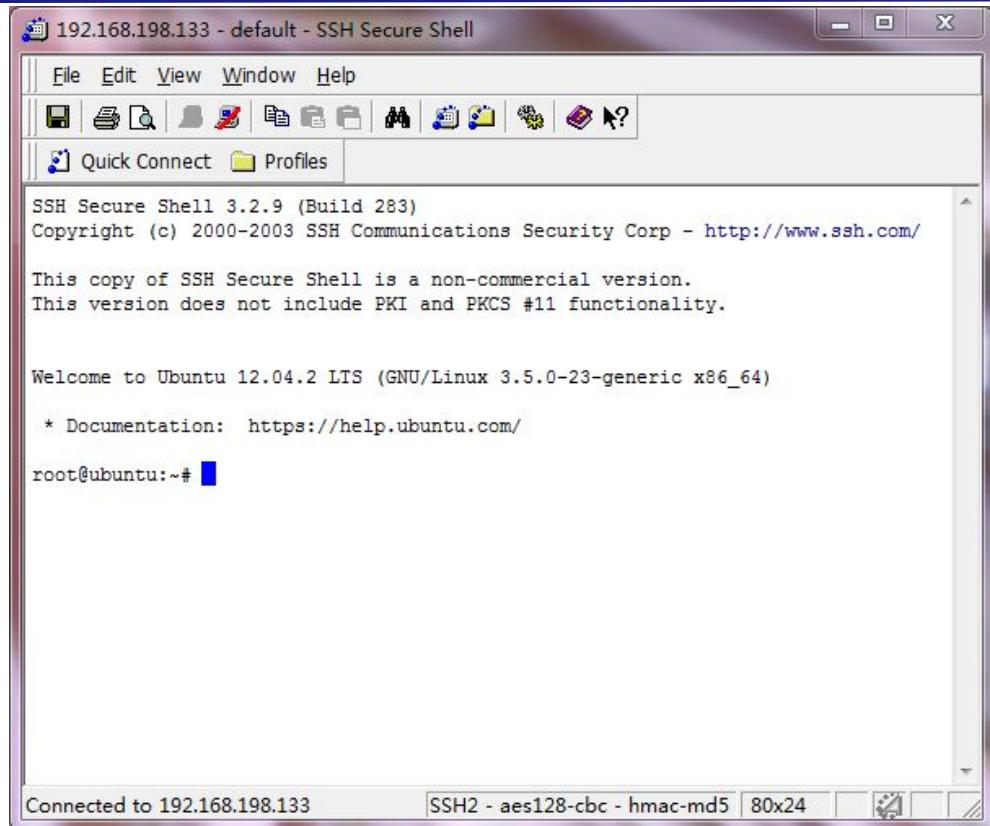


7 ) 如下图，左边部分是 Windows 主机的文件目录，右边为 Ubuntu 系统的目录，可以通过鼠标拖动来传文件，非常方便。



### 3.3.5.3 SSH 控制台

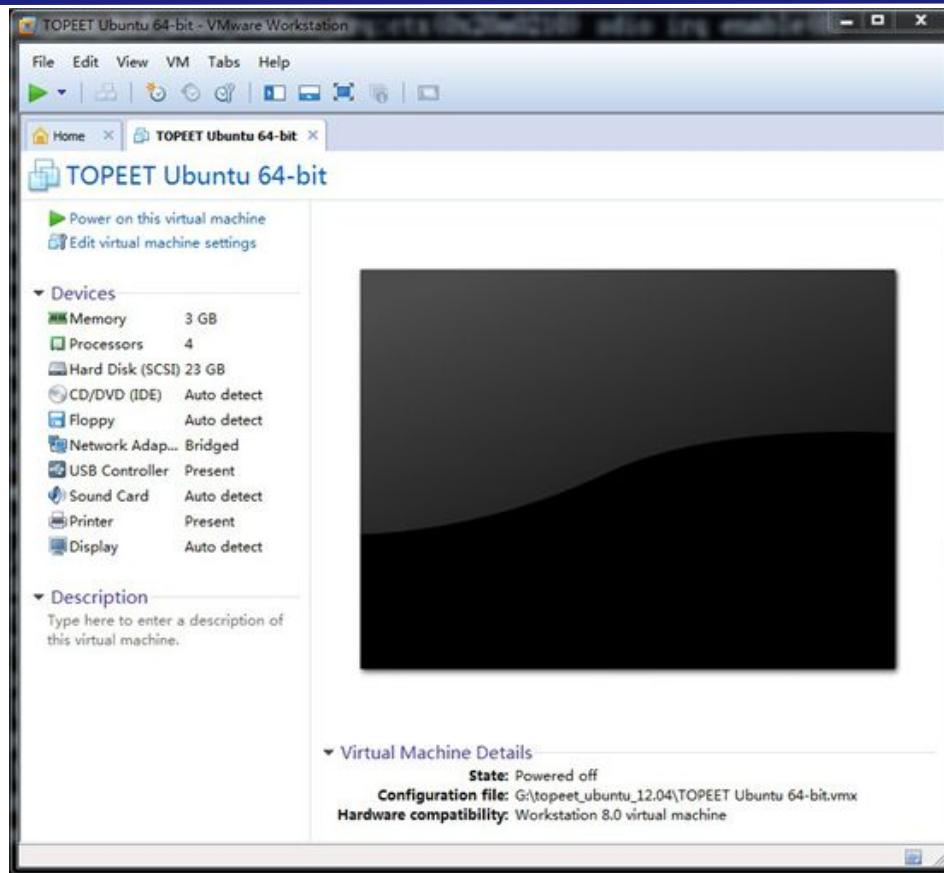
参照上一小节，将 SSH 控制台和 Ubuntu 连接。连接后如下图，等同于 Ubuntu 的命令行，可以在其中输入 Linux 命令。



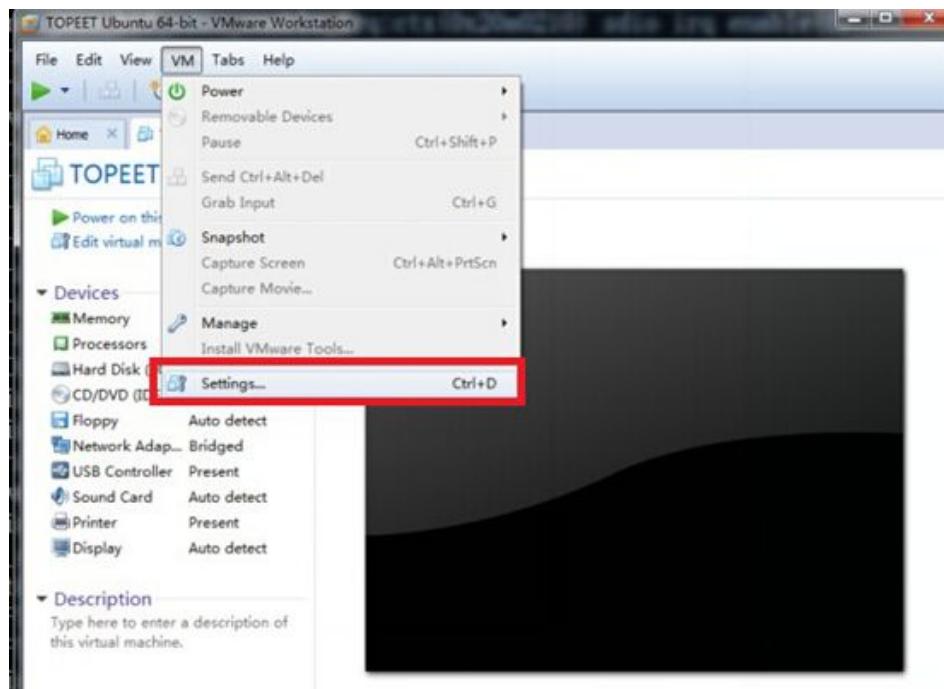
### 3.3.6 虚拟机 Ubuntu 扩展硬盘空间

在虚拟机 Ubuntu 的使用过程中，有时候会发现 Ubuntu 初始的硬盘空间分配的不够，这样就需要重新扩展。本小节介绍一下扩展的方法。

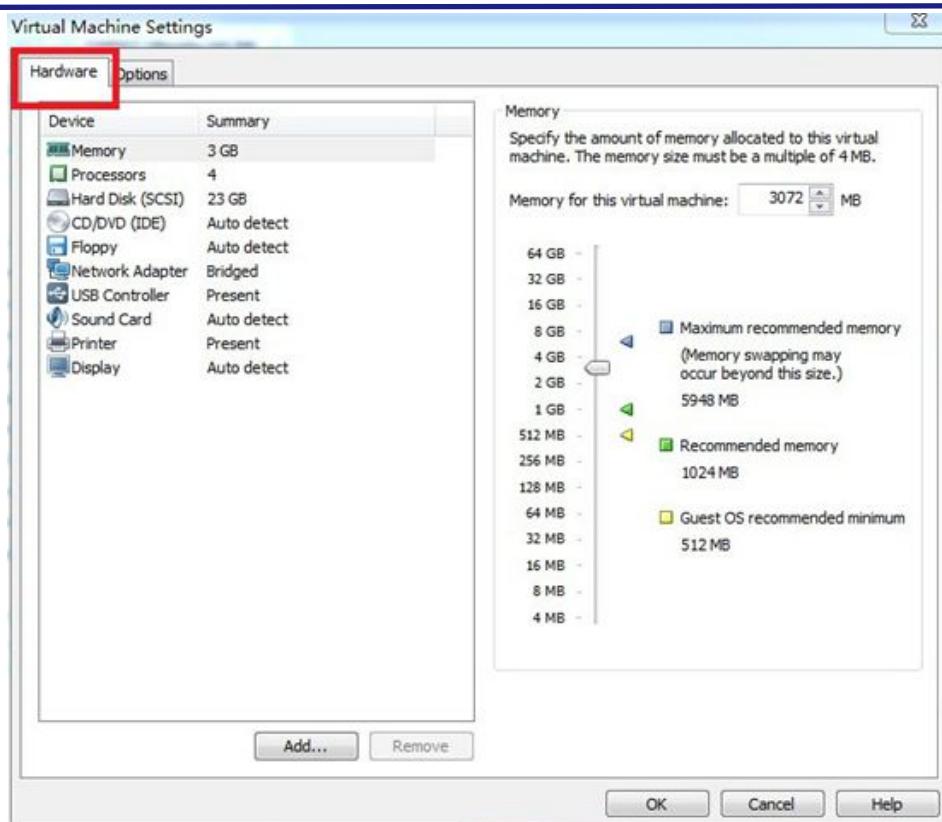
首先在虚拟机里面关闭 Ubuntu 系统，如下图所示：



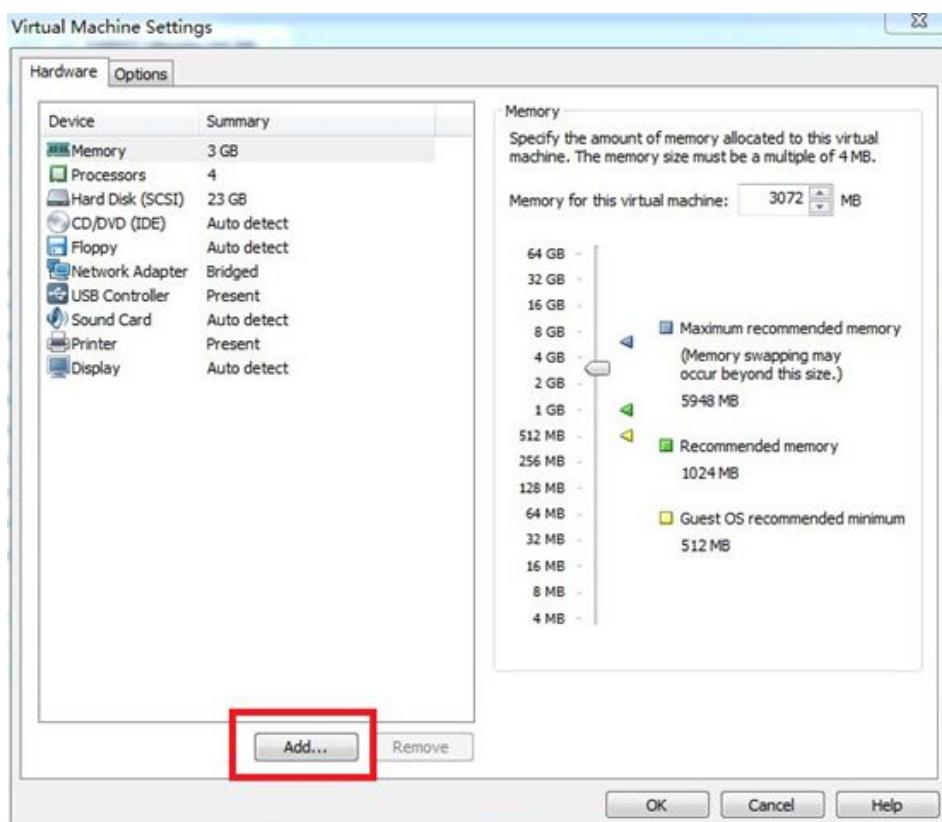
在 VMware 里点击菜单 “VM→Settings...” , 如下图 :



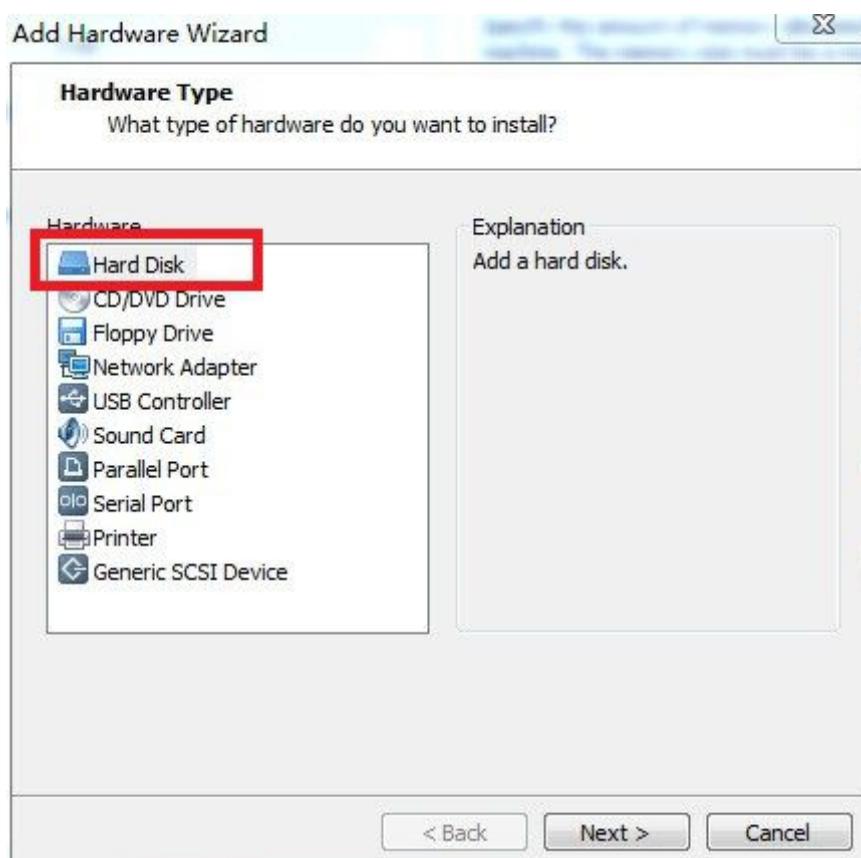
然后弹出 “Virtual Machine Settings” 对话框 , 如下图 :



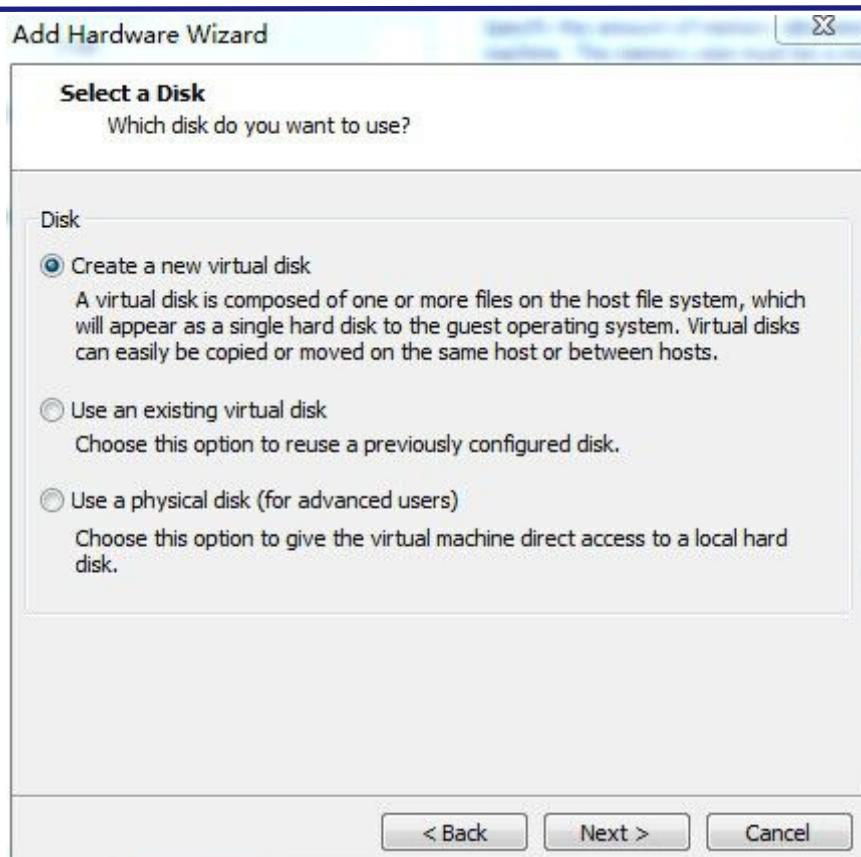
在上图的"Hardware"里面单击底下的 Add 按钮，如下图：



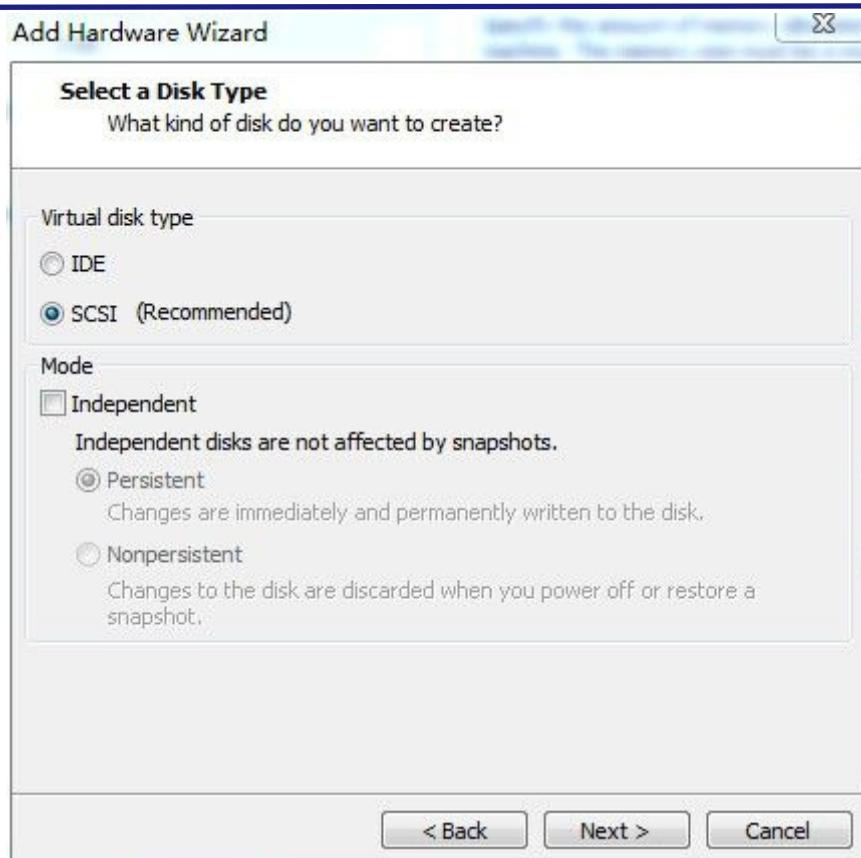
弹出“Add Hardware Wizard”对话框，如下图：



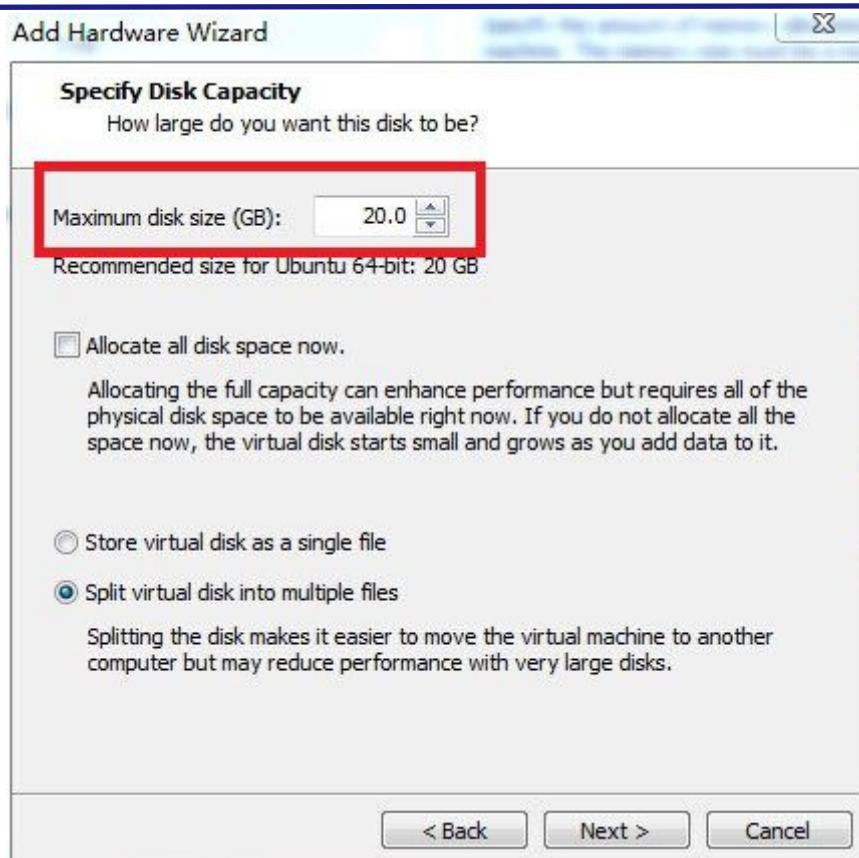
选中上图的“Hard Disk”，然后单击“Next”按钮，出现下面的对话框：



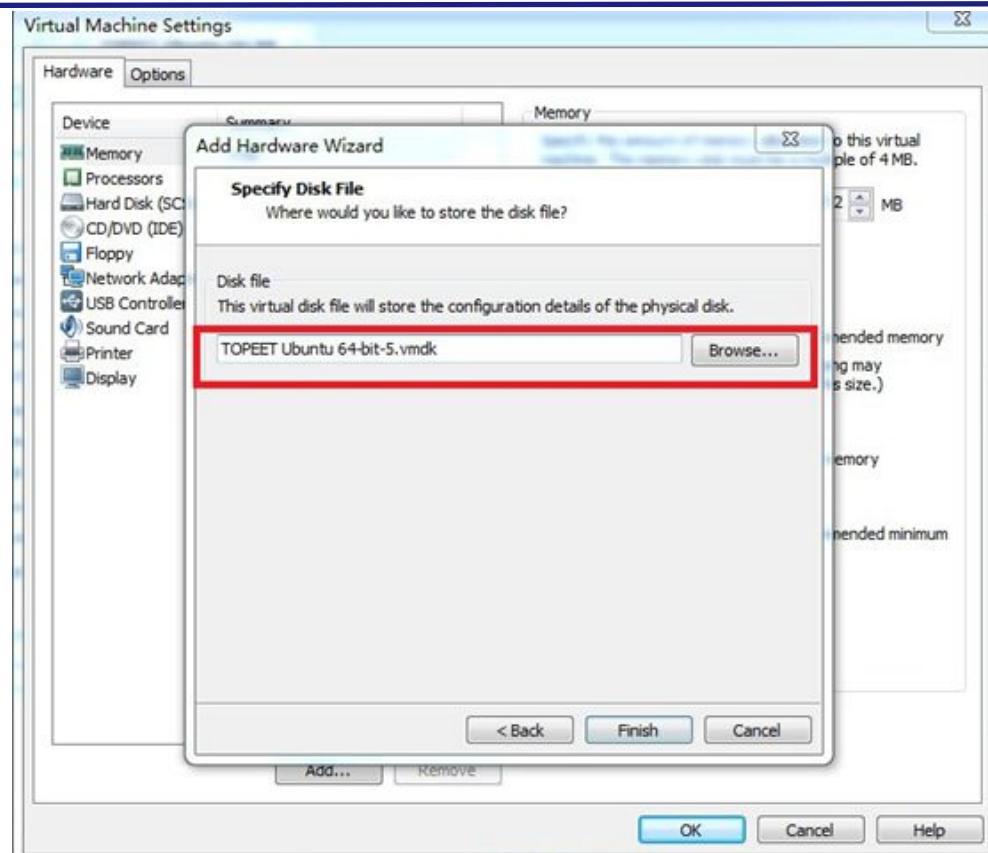
按照上图的设置，然后单击“Next”按钮，出现下面的对话框：



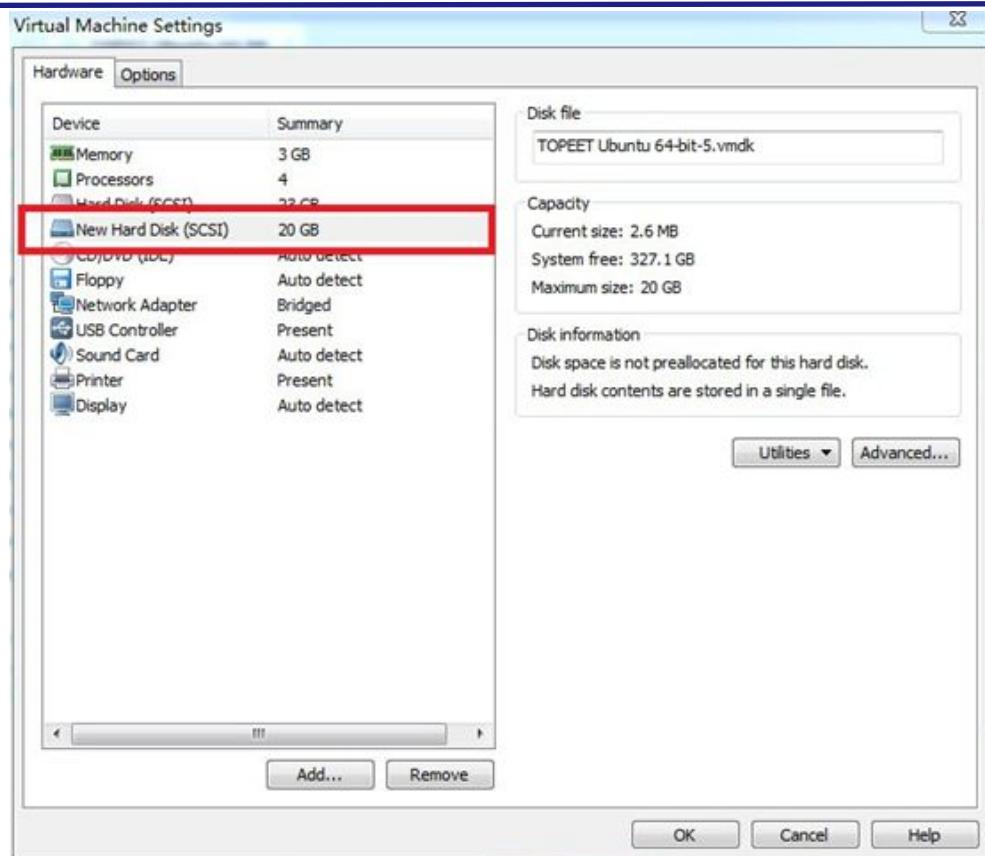
按照上图的设置，然后单击“Next”按钮，出现下面的对话框：



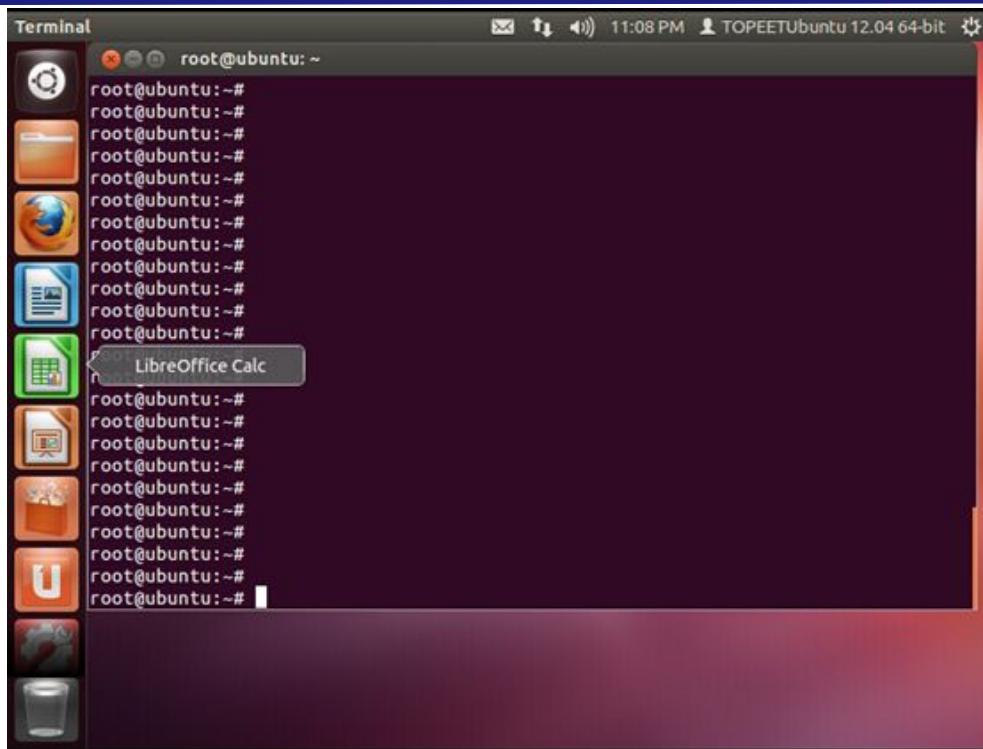
按照上图的设置，其中红色方框内是设置需要扩展的硬盘的大小，这里我们选择 20G（可以根据需要来修改这个值），然后单击“Next”按钮，出现下面的对话框：



上图中红色框内的文件是用于保存扩展的硬盘的信息，文件的名字和保存的路径可以自己定义，确定好文件名字和保存路径后，点击“Finish”按钮，然后回到“Virtual Machine Settings”对话框，在这个对话框会看到我们添加的硬盘的信息，如下图：

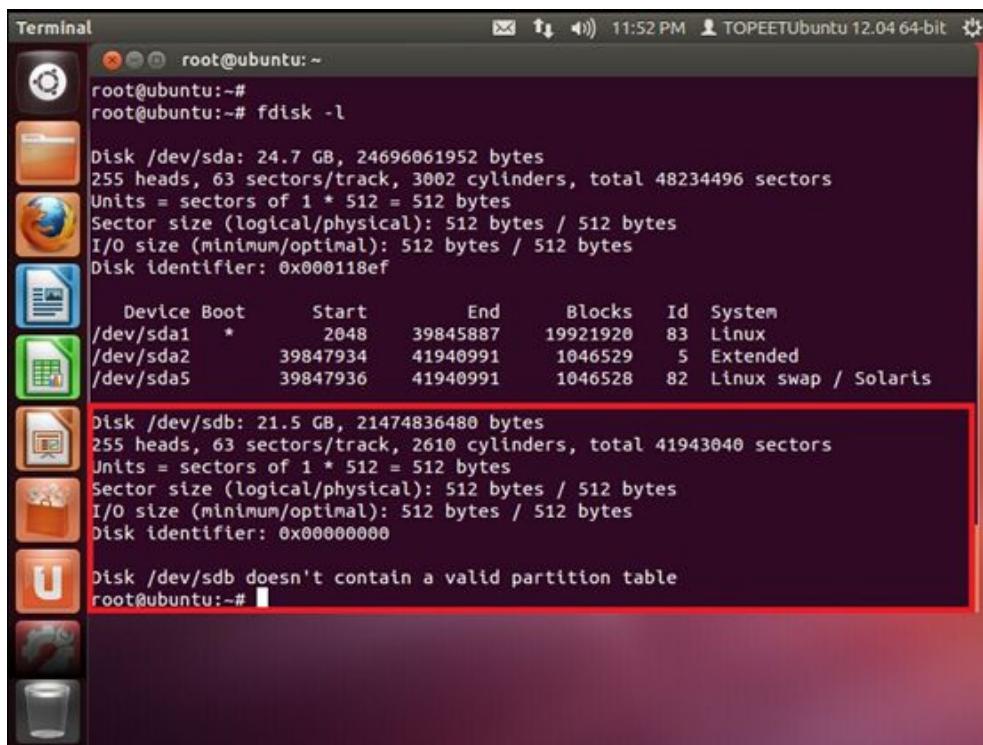


点击上图中的“OK”按钮，这样经过前面的操作，在VMwar的Device里就多了一个Hard Disk，接下来要进入Ubuntu，把新的硬盘mount进去，启动Ubuntu系统，以root用户登录，如下图：

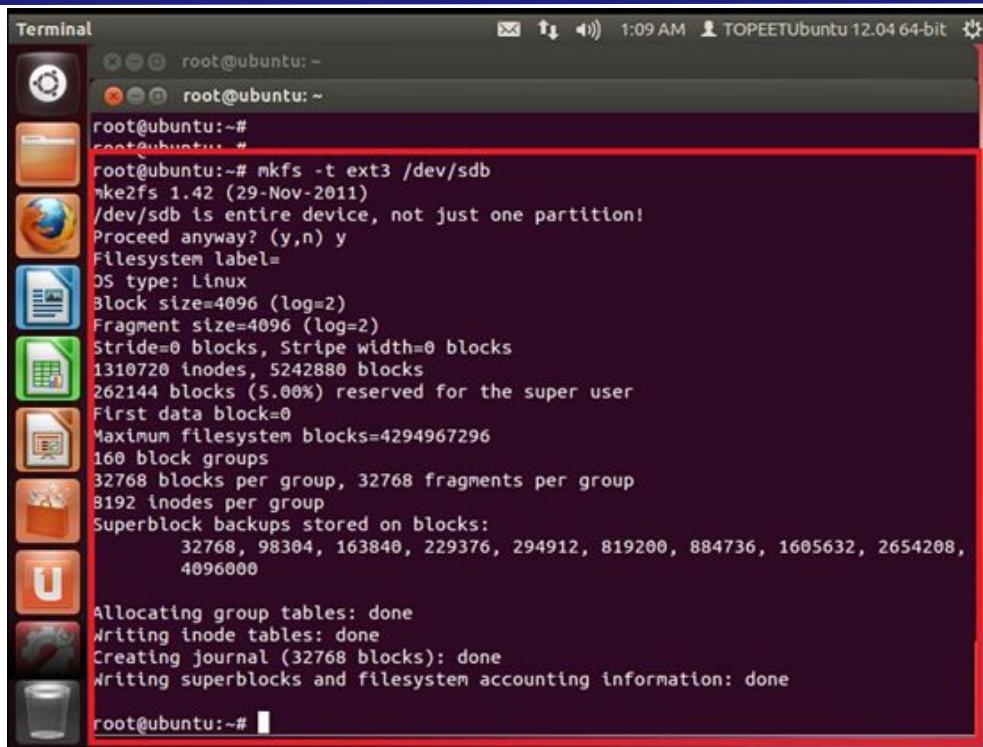


在上图中的超级终端里输入命令 “fdisk -l” , 我们将会看到添加的新的硬盘

“/dev/sdb” 的信息,如下图 :



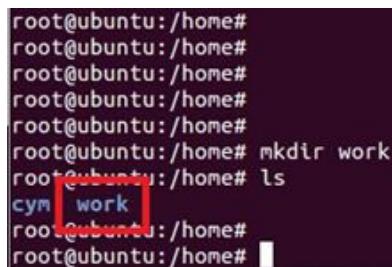
在上图的超级终端里输入命令 “mkfs -t ext3 /dev/sdb” , 把新添加的硬盘格式化成 ext3 格式 , 如下图 :



```
root@ubuntu:~# mkfs -t ext3 /dev/sdb
mke2fs 1.42 (29-Nov-2011)
/dev/sdb is entire device, not just one partition!
Proceed anyway? (y,n) y
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
1310720 inodes, 5242880 blocks
262144 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=4294967296
160 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
    4096000
Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

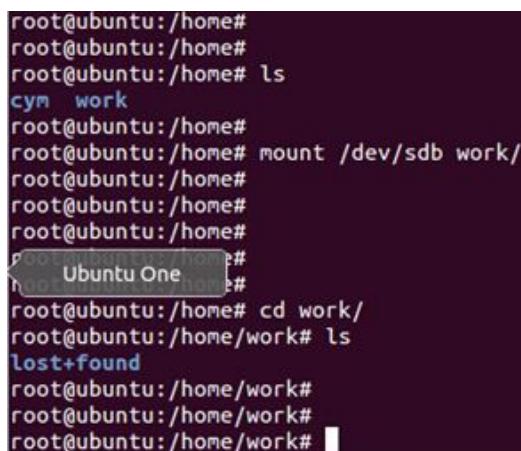
root@ubuntu:~#
```

在/home 目录下建立 work 文件夹 , 如下图 :



```
root@ubuntu:/home#
root@ubuntu:/home#
root@ubuntu:/home#
root@ubuntu:/home#
root@ubuntu:/home#
root@ubuntu:/home# mkdir work
root@ubuntu:/home# ls
cym  work
root@ubuntu:/home#
root@ubuntu:/home#
```

使用命令 “mount /dev/sdb work/” 挂载硬盘到 work 目录 , 如下图 :



```
root@ubuntu:/home#
root@ubuntu:/home#
root@ubuntu:/home# ls
cym  work
root@ubuntu:/home#
root@ubuntu:/home# mount /dev/sdb work/
root@ubuntu:/home#
root@ubuntu:/home#
root@ubuntu:/home#
root@ubuntu:/home# cd work/
root@ubuntu:/home/work# ls
lost+found
root@ubuntu:/home/work#
root@ubuntu:/home/work#
root@ubuntu:/home/work#
```

使用 “df -l” 命令查看一下挂载情况 , 如下图 :

```
root@ubuntu:/home/work#
root@ubuntu:/home/work# df -l
Filesystem      1K-blocks   Used   Available  Use% Mounted on
/dev/sda1        19609276 16894500   1718680  91% /
udev             1530680      4   1530676   1% /dev
tmpfs            615980     816   615164   1% /run
none              5120       0    5120   0% /run/lock
none             1539944    200   1539744   1% /run/shm
/dev/sdb         20642428 176196   19417656   1% /home/work
root@ubuntu:/home/work#
root@ubuntu:/home/work#
```

通过上图可以看到新扩展的硬盘已经挂载到了/home/work 目录下面了，下面修改 Ubuntu 系统的配置文件，实现开机自动挂载，使用命令"vi /etc/fstab "打开 fstab 文件，参照里面的格式，添加挂载新硬盘的命令"/dev/sdb /home/work ext3 defaults 0 0"，如下图：

```
root@ubuntu:/home/work
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point> <type> <options> <dump> <pass>
proc          /proc      proc    nodev,noexec,nosuid  0      0
# / was on /dev/sda1 during installation
UUID=614a24d5-f542-4d80-b758-c90a6ad402d0  /          ext4    errors=remount
-ro 0 1
# swap was on /dev/sda5 during installation
UUID=b3c9e1f1-d0f1-4478-84fb-b2701dfad7be none      swap    sw
0 0
/dev/fde     /media/floppy0 auto    rw,user,noauto,exec,utf8 0      0
/dev/sdb /home/work ext3 defaults 0 0
```

保存上图修改的文件，开机启动自动挂载设置完成。

### 3.3.7 虚拟机无法识别 USB3.0 的解决方法

我们经过测试发现 VMware-workstation8.0.3 对 USB3.0 的支持不是很好，甚至有时候用户的主板支持 USB2.0，有时候也会因为驱动支持不完全而导致 Ubuntu 系统无法识别到 USB。

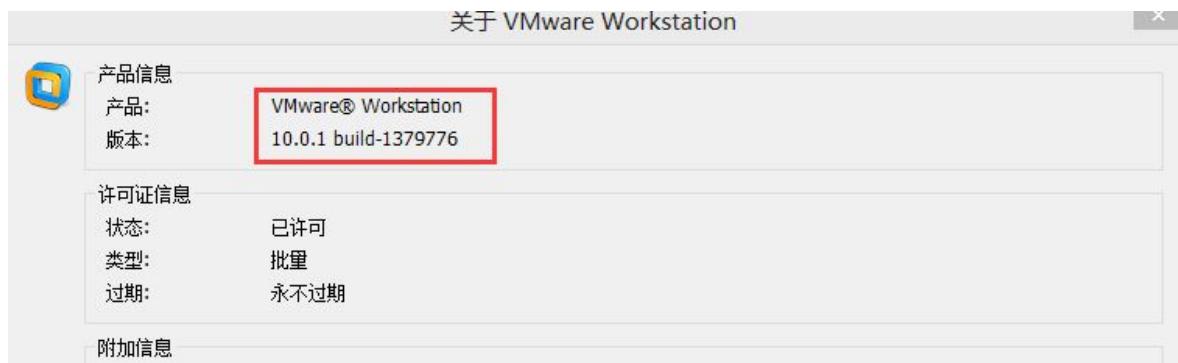
如果用户的主板有任意一个 USB 接口可以支持 USB3.0，这里建议用户使用我们提供的新版本虚拟机 VMware-workstation-full-10.0.1-1379776.exe

如下图，该文件和原来低版本的虚拟机放在网盘的同一个目录中。

文件名	大小	修改时间
VMware-workstation-full-10.0.1-1379776.exe	490.28MB	2015-07-01 16:46
VMware_Workstation_wmb.zip	483.04MB	2015-04-28 09:25

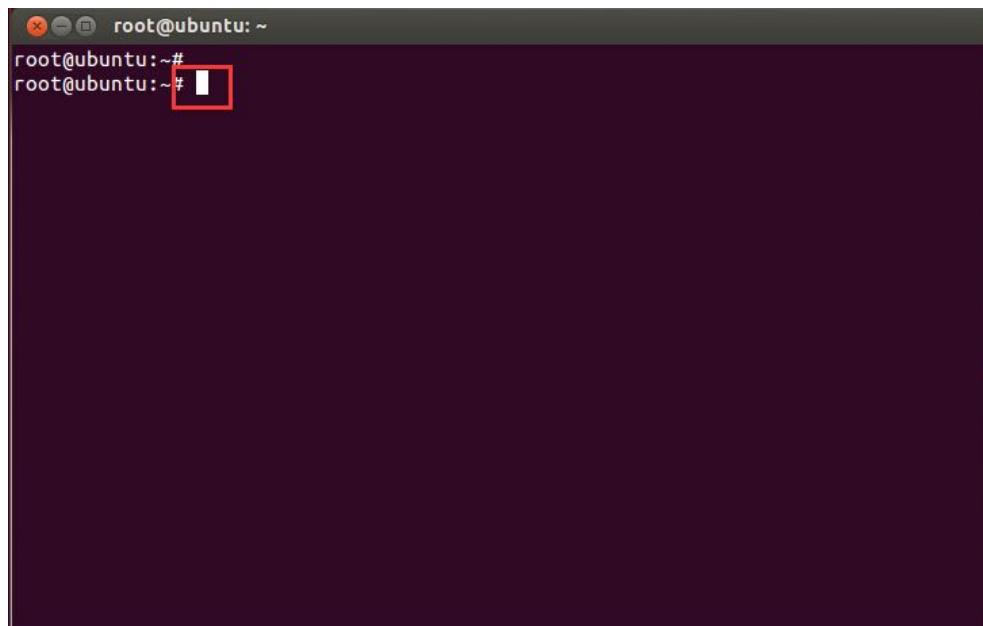
安装方法参考低版本的虚拟机，破解码可以百度。

新版本虚拟机安装后破解之后，产品信息如下。

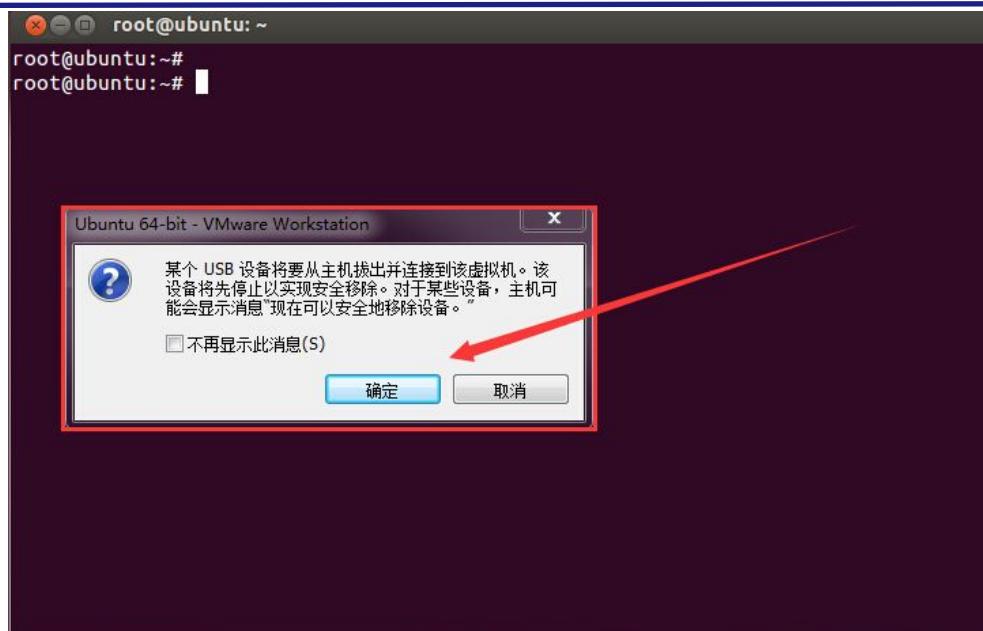


### 3.3.8 U 盘、TF 卡与虚拟机连接

如下图所示，将 Ubuntu 的界面激活（简单的理解就是，你在键盘输入，显示在 Ubuntu 界面）



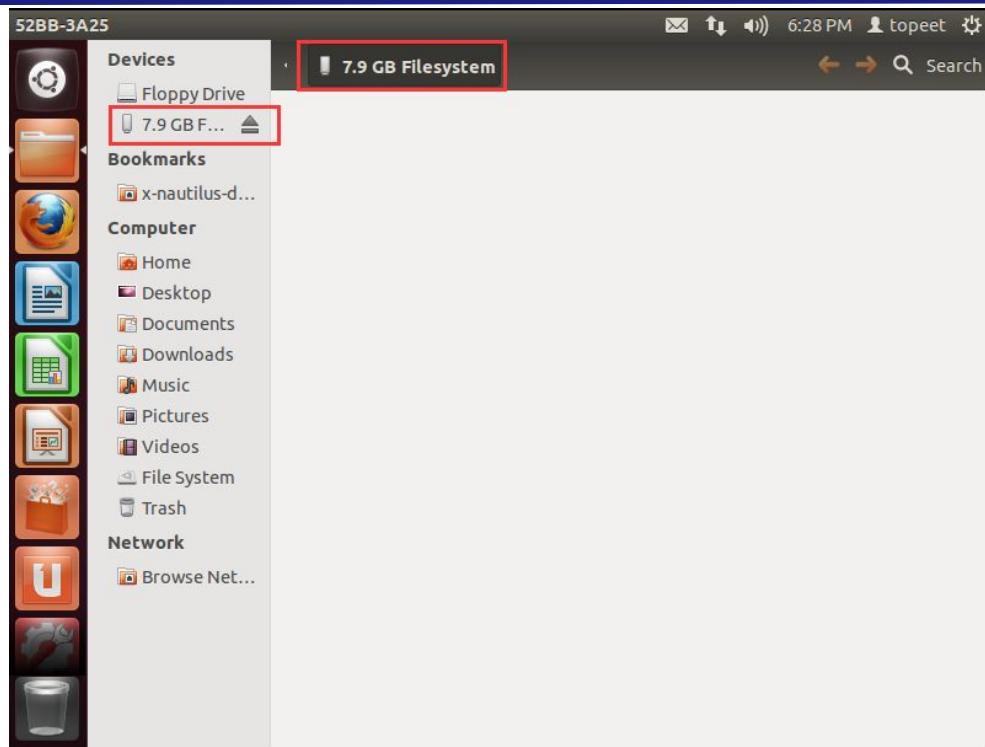
插入 U 盘或者 TF 卡，如下图所示，可能弹出这个界面，单击确定。



如下图所示，也可能弹出这个界面。



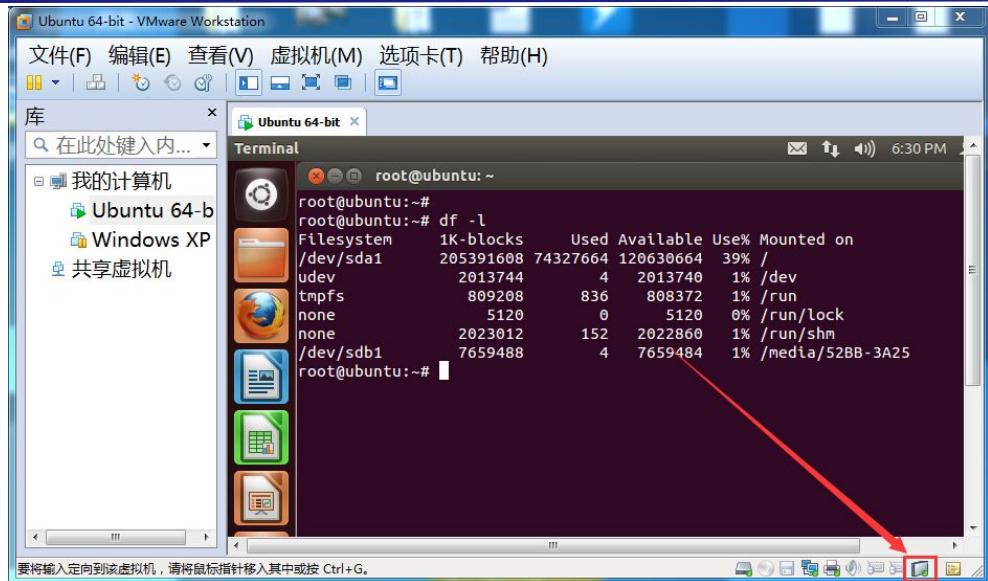
如下图所示，8G 的 TF 卡直接被连接到了 Ubuntu 系统。



上图的这个图形界面比较简单，这里不再啰嗦。如下图所示，在命令行中输入“df -l”，会发现多了一个盘符。

Filesystem	1K-blocks	Used	Available	Use%	Mounted on
/dev/sda1	205391608	74327664	120630664	39%	/
udev	2013744	4	2013740	1%	/dev
tmpfs	809208	836	808372	1%	/run
none	5120	0	5120	0%	/run/lock
none	2023012	152	2022860	1%	/run/shm
<b>/dev/sdb1</b>	<b>7659488</b>	<b>4</b>	<b>7659484</b>	<b>1%</b>	<b>/media/52BB-3A25</b>

有时候虚拟机直接无法识别加载 USB 设备，那么注意虚拟机右下方的 USB 设备盘符的小图标。



右键或者左键单击上图的小图标，就可以弹出下面三个小选项

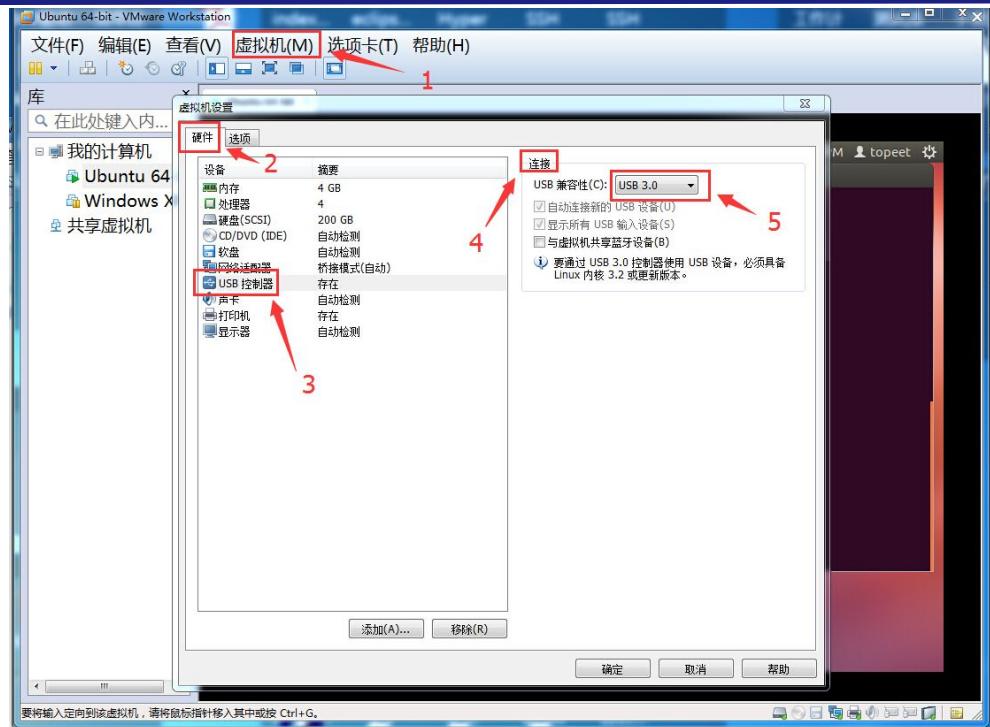
断开连接（连接主机）

更改图标

隐藏图标

这里我们选择“断开连接（连接主机）”就可以在主机和虚拟机系统之间切换 USB 盘符设备了。

如下图所示，在“虚拟机(M)”→“硬件”→“USB 控制器”→“连接”可以选择 USB 设备的版本。



如果用户 PC 主板的 USB 设备是 3.0 的，则虚拟机最好安装“VMware-workstation-full-10.0.1”，这个版本对于 USB3.0 的设备支持比较好一点。

### 3.4 Vim 编辑器

在使用 Ubuntu 的过程中，会涉及文档和文件的编辑。Linux 下的编辑工具非常多，这里给大家介绍一下 Vim 编辑器。

Vim 的操作属于命令组合，是一种指令式的编辑器，有不同的工作模式，不需要使用鼠标，也没有菜单，仅仅用键盘就能完成所有的工作。

因为它有几种不同的工作模式，这会让刚开始接触的用户有点厌烦。但是它的优点在于，只需要使用键盘就可以完成所有的编辑工作，不需要在键盘和鼠标之间来回切换，因而可以大大的提高工作效率。

下面就介绍一下 Vim 编辑器基本的操作，帮助大家入手这款编辑器，与 Vim 编辑器相关的命令非常多，在这里只能做简单的入门介绍。用户只要入门了，跨过了最基本的第一步，就可

以自行研究和学习了。在网盘中，有关于 Vim 编辑器命令的文档“linux 下 vim 使用详解.pdf”，如果用户以前没有接触过，学习了本章节的内容后，可以参照文档练习一下。

### 3.4.1 安装 Vim 编辑器

虚拟机联网后，使用命令“apt-get install”安装。

在 root 用户下，在 Ubuntu 命令行中，输入命令：

```
apt-get install vim
```

### 3.4.2 Vim 打开文件以及新建文件

Vim 打开一个不存在的文件，就可以创建文件该文件，如下图，在 Ubuntu 命令行中，输入命令

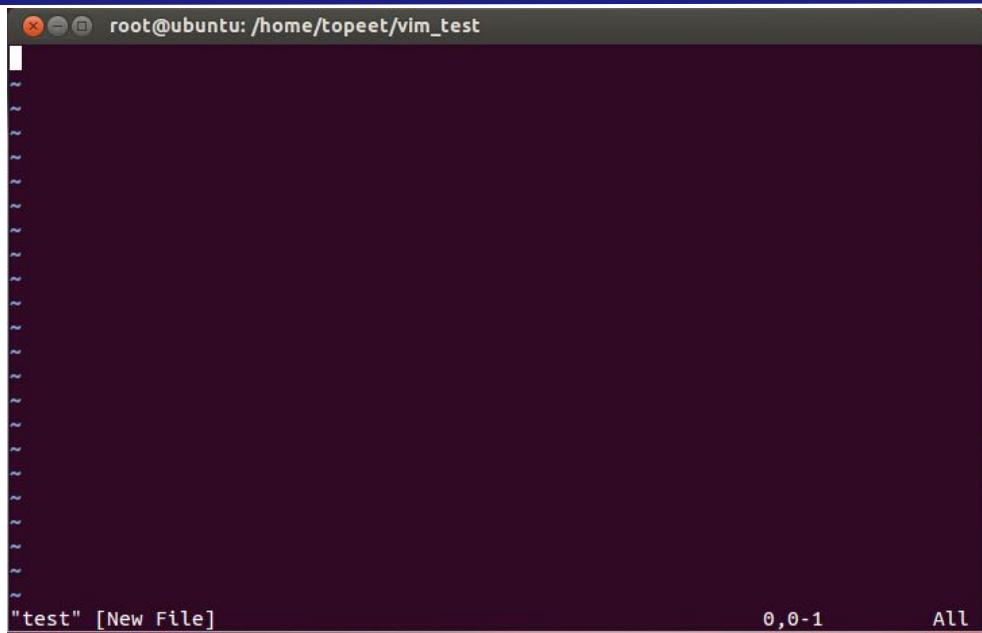
vim test 新建一个名为 test 的文件

```
root@ubuntu:/home/topeet/vim_test#  
root@ubuntu:/home/topeet/vim_test#  
root@ubuntu:/home/topeet/vim_test# vim test■
```

如下图，因为本来不存在这个文件，所以新建了一个空的文件。

接着讲一下如何退出文件。

首先按键盘的“shift+冒号”，然后在文件的底行可以输入命令，如果在文件的最底行输入“q+！+回车”，则不保存文件退出；如果保存文件则输入“wq+回车”。保存后退出，可以看到文件夹下面有一个新建的文件“test”；不保存退出，则没有“test”文件。



Vim 打开文件的命令还有：

vim n filename      打开文件，将光标置于第 n 行首。

如下图，输入帮助命令会有 vim 命令组合的详细说明

vim --help

```
root@ubuntu:~# vim --help
VIM - Vi IMproved 7.3 (2010 Aug 15, compiled May 4 2012 04:25:35)

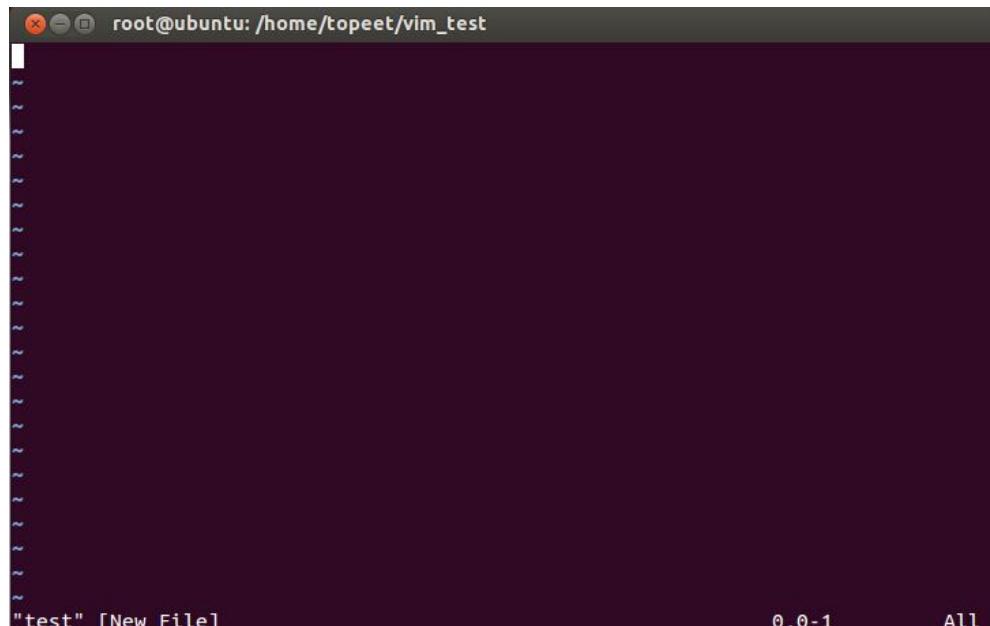
usage: vim [arguments] [file ..]      edit specified file(s)
      or: vim [arguments] -       read text from stdin
      or: vim [arguments] -t tag    edit file where tag is defined
      or: vim [arguments] -q [errorfile] edit file with first error

Arguments:
  --          Only file names after this
  -v          Vi mode (like "vi")
  -e          Ex mode (like "ex")
  -s          Silent (batch) mode (only for "ex")
  -d          Diff mode (like "vimdiff")
  -y          Easy mode (like "evim", modeless)
  -R          Readonly mode (like "view")
  -Z          Restricted mode (like "rvim")
  -m          Modifications (writing files) not allowed
  -M          Modifications in text not allowed
  -b          Binary mode
  -l          Lisp mode
  -c          Compatible with vi: 'compatible'
```

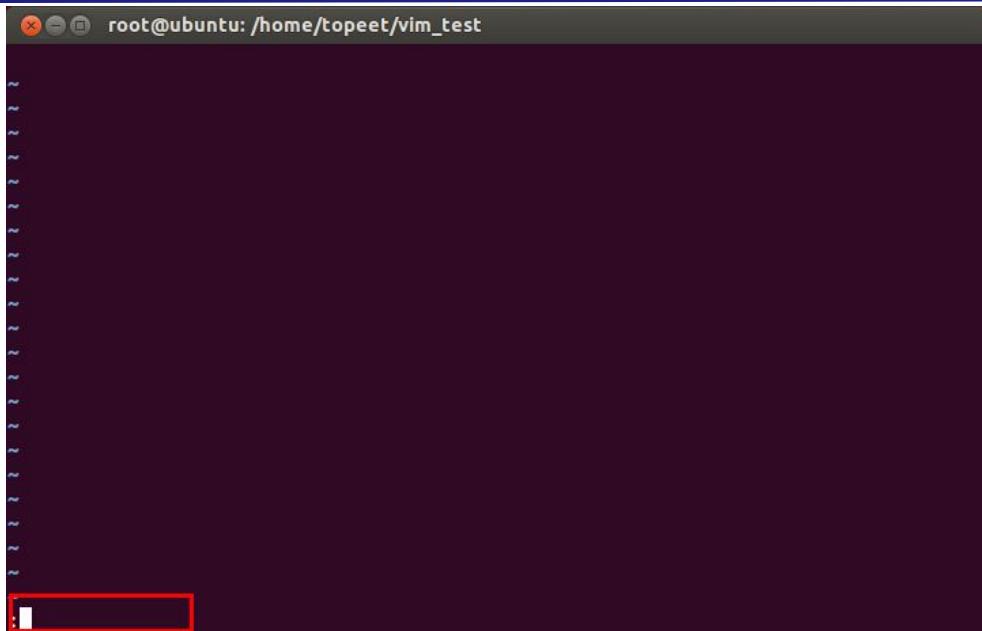
### 3.4.3 三种模式的切换

学习 Vim 编辑器，首先要掌握它的三种模式。这三种模式分别是：指令模式，输入模式，底行模式。Vim 编辑器有大量的命令，不同的命令只能在相对应的模式下使用。

如下图，用户新建一个文件 “vim\_test” ，进入文件之后就是指令模式，注意因为这个文件是空的，所以很多操作无效。在了解后面输入模式之后，向文件里面添加了部分内容之后，再进行指令操作

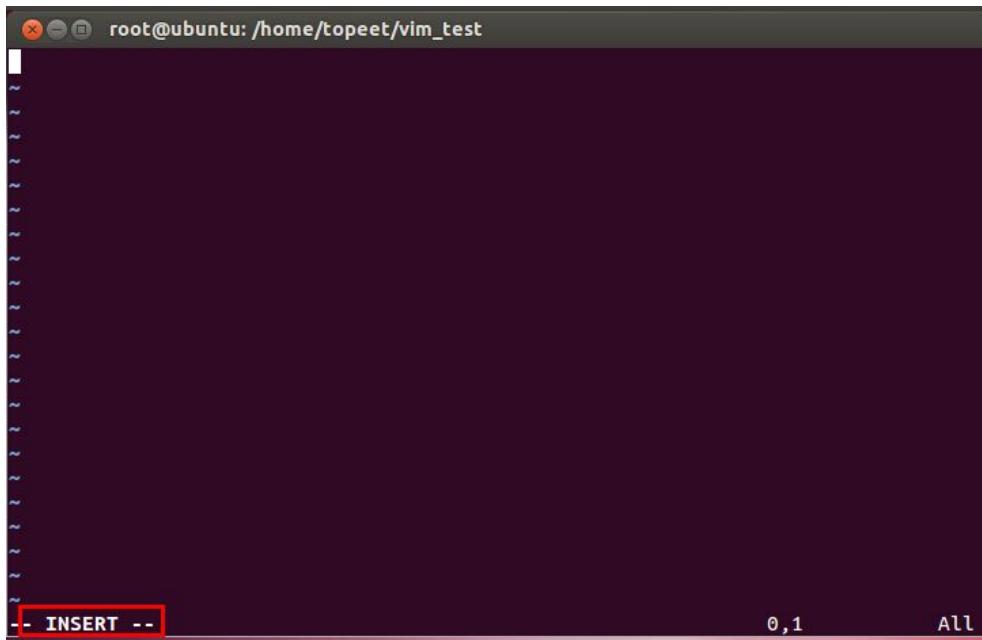


指令模式可以切换到底行模式，按键盘 “shift+冒号” ，进入底行模式，如下图，底行模式可以输入查找命令，退出等。



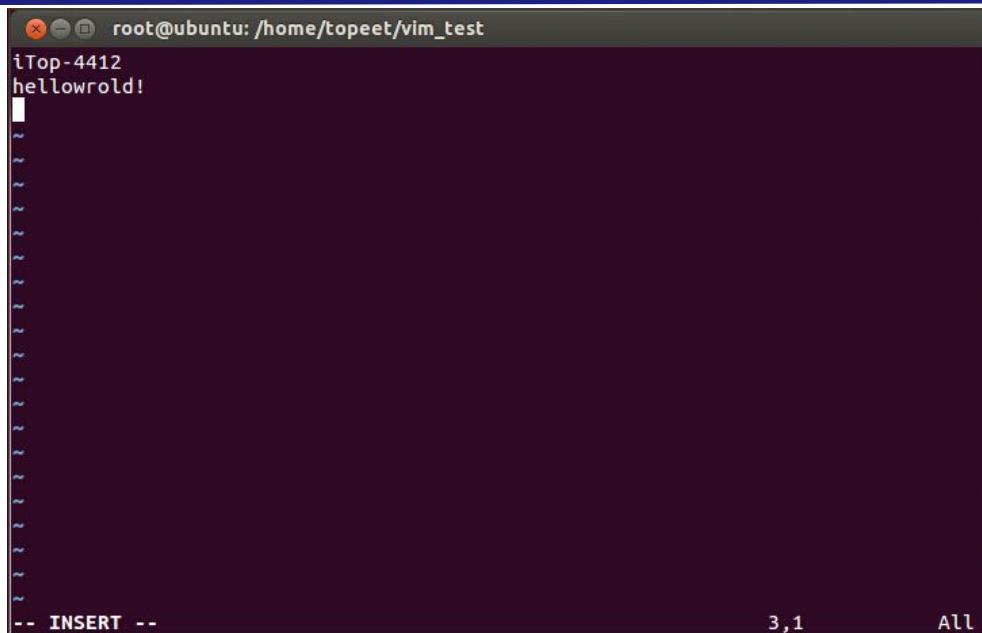
然后退出底行模式，切换到指令模式，按键盘“Esc”。

切换到指令模式后，再切换到输入模式，按键盘“Insert”，如下图，在文件的最下面一行出现“Insert”，高亮光标出现在第一行第一列。



在输入模式中，才可以给文件添加内容，如下图，输入模式中的基本操作比较容易掌握。例如：回车代表换行，方向键代表上下翻动文本等。

如下图，在文件中，输入“iT0p-4412”，“回车”，“helloworld!”，“回车”。



```
iTop-4412
helloworld!
```

在输入模式中，按键盘“Esc”，切换到命令模式，然后按键盘“shift+冒号”，进入底行模式。在底行模式中，才输入退出编辑器的命令。这里需要注意的是，如果仅仅输入“q”是无法退出的，在修改了文件之后，如果不保存退出，必须输入“q！”；如果要保存退出则输入“wq”。

### 3.4.4 Vim 编辑器常用基本命令

这里给大家介绍三种模式中最基本的，最常用的几个命令。如果用户以前没有接触过 Vim 编辑器，下面给大家介绍的命令可以都敲一敲，掌握了下面几个命令，一般的操作都没有问题了。

#### 3.4.4.1 命令行模式

四个方向按键，移动高亮显示的光标

G 光标置于文本最后一行，最后一列。大写输入为按键“Shift+g”

gg 光标置于文本第一行，第一类。连接两次按键 “g”

dd 删除光标所处行的所有内容。连接两次按键 “d”

按键 delete 删除光标高亮显示的字符。键盘 “Delete”

#### 3.4.4.2 输入模式

四个方向按键，移动高亮显示的光标

按键 delete 删除光标高亮显示的字符。按键盘 “Delete”

按键 Backspace 删除光标高亮显示字符的前一个字符。按键盘 “Backspace”

#### 3.4.4.3 底行模式

q 文档没有修改，退出不保存

q! 文档修改过，退出不保存

w 保存文档

wq 保存退出

按键 “/” + “字符” 查找 “字符”

底行模式中，输入的命令可以通过方向按键上翻下翻命令

Vim 编辑器就介绍到这里了，如果用户想进一步学习 Vim 编辑器的使用，可以在网盘下载文档“linux 下 vim 使用详解.pdf”来学习。

## 3.5 Source Insight 的安装和使用

Source Insight 是一个面向项目开发的程序编辑器和代码浏览器，它拥有内置的对 C/C++, C# 和 Java 等程序的分析工具。能分析源代码并在工作的同时动态维护它自己的符号数据库，并自动显示有用的上下文信息。

这个是一个比较复杂的软件，用户只需要学会基本的操作，能够使用软件查看 Linux 内核源码就足够了。

### 3.5.1 Source Insight 的安装

Source Insight 的安装比较简单，下面简要的介绍一下。

百度搜“source insight”，直接在搜索栏上的百度软件中心下载安装之。在安装好后会弹出一些组件的选择，您可以全选也可以选择性的选择您所需要的功能。



Source Insight 的安装很简单，教程很多，方法都通用，需要重点介绍的是下一小节的查看内核代码。

### 3.5.2 使用 Source Insight 查看内核代码

本小节介绍如何使用 Source Insight 查看代码。

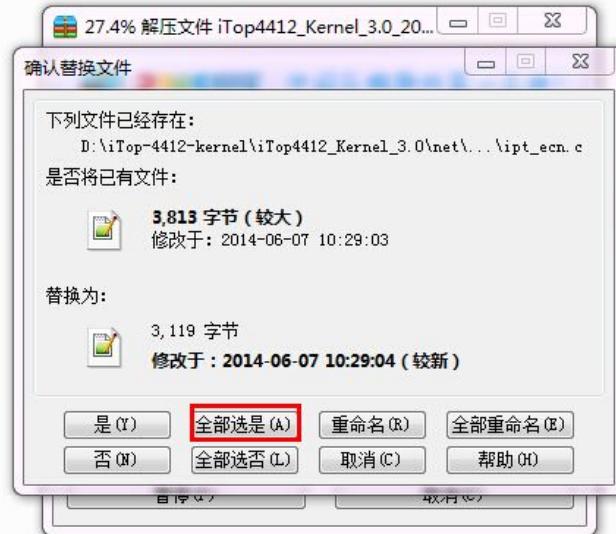
1 ) 首先需要在 Windows 操作系统下面解压 Linux 内核源码。内核源码是光盘 “06\_源码\_uboot 和 kernel” 文件夹中的压缩包 “iT0p4412\_Kernel\_3.0\_xxx.tar.gz” , “xxx” 代表压缩日期。

这里有一点要注意的是，内核源码解压的路径最好是英文路径。



2 ) 如下图，解压压缩包，解压的时候会弹出对话框，单击按钮“全部选是”。

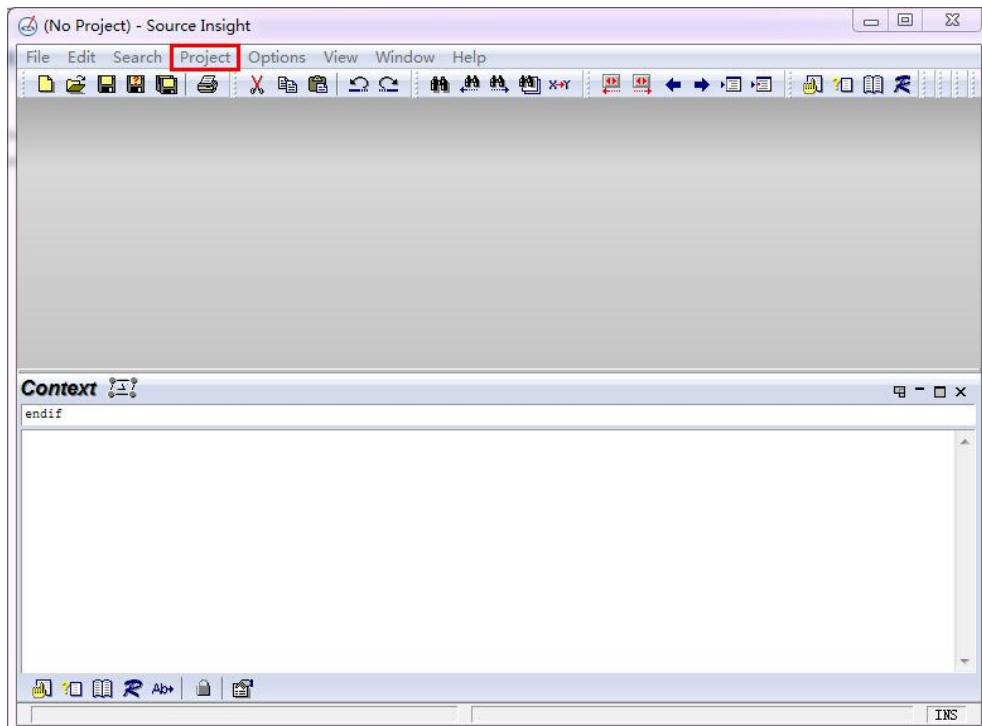
这里有一点需要注意的是，内核源码在 Windows 下面解压后，很多文件都丢失了，不能够将文件夹直接拷贝到 Ubuntu 中编译。



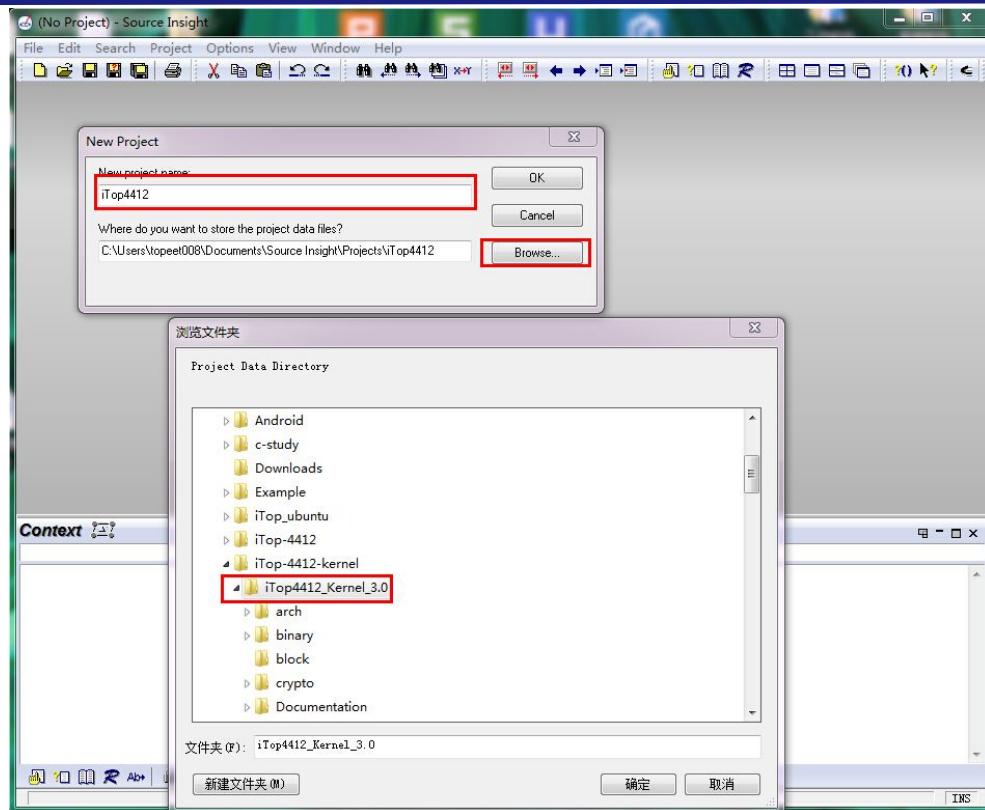
3 ) 解压完成后得到文件夹 “iT0p4412\_Kernel\_3.0” , 如下图所示。

名称	修改日期	类型	大小
iTop4412_Kernel_3.0	2014/9/19 11:38	文件夹	
iTop4412_Kernel_3.0_20140919.tar.gz	2014/9/30 11:42	好压 GZ 压缩文件	130,864 KB

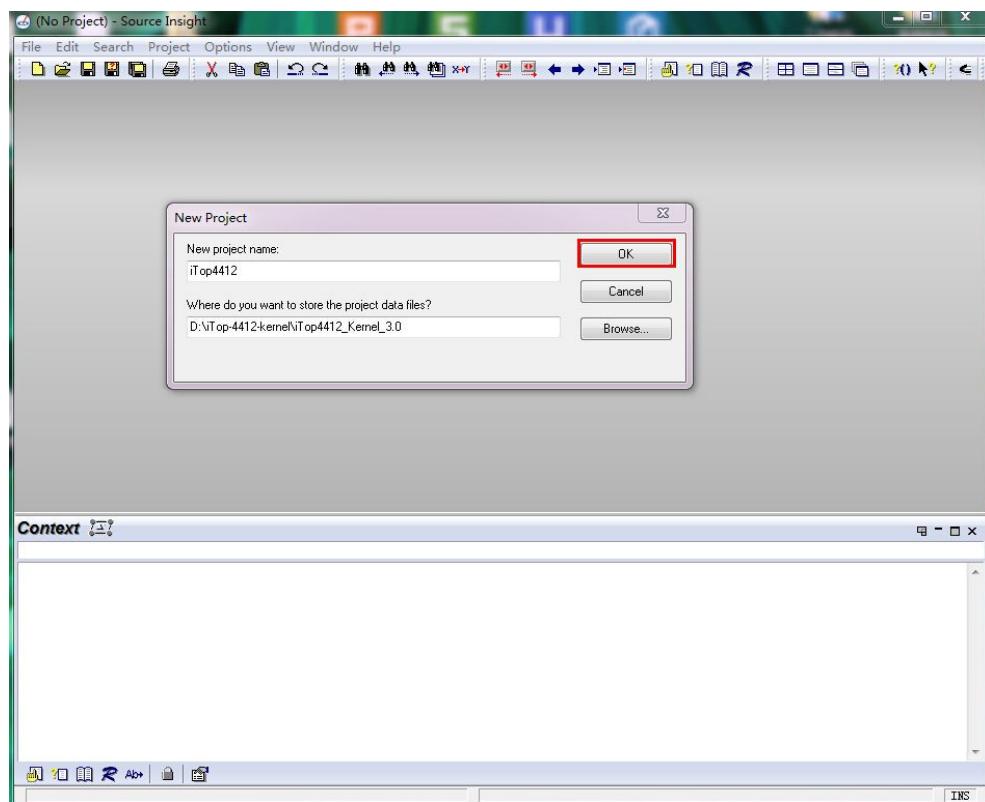
4 ) 打开 Source Insight 软件 , 如下图 , 单击菜单 “Project” --> “New Project” 。



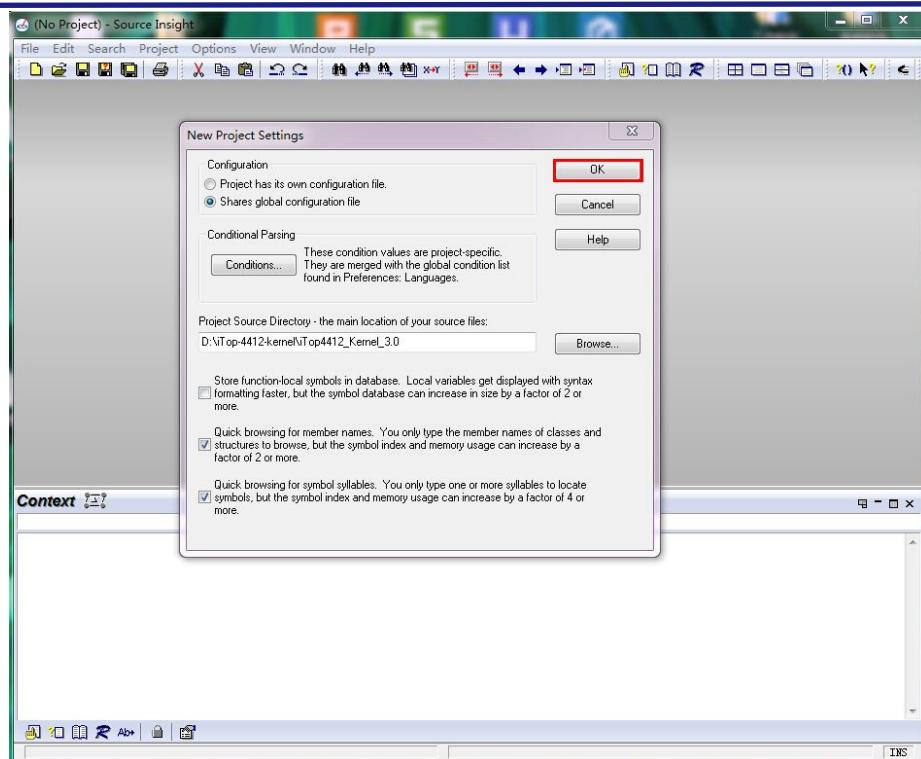
5 ) 如下图 , 输入工程名和前面解压的内核源码文件夹的路径。



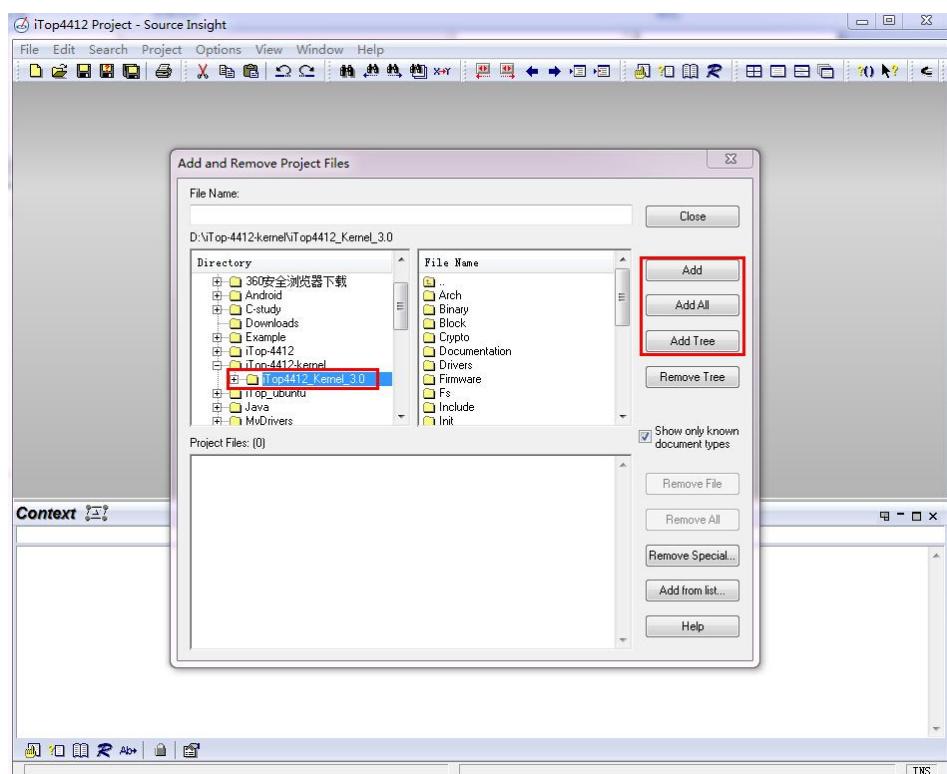
6 ) 如上图 , 单击按钮 “确定” , 如下图 , 单击按钮 “OK” 。



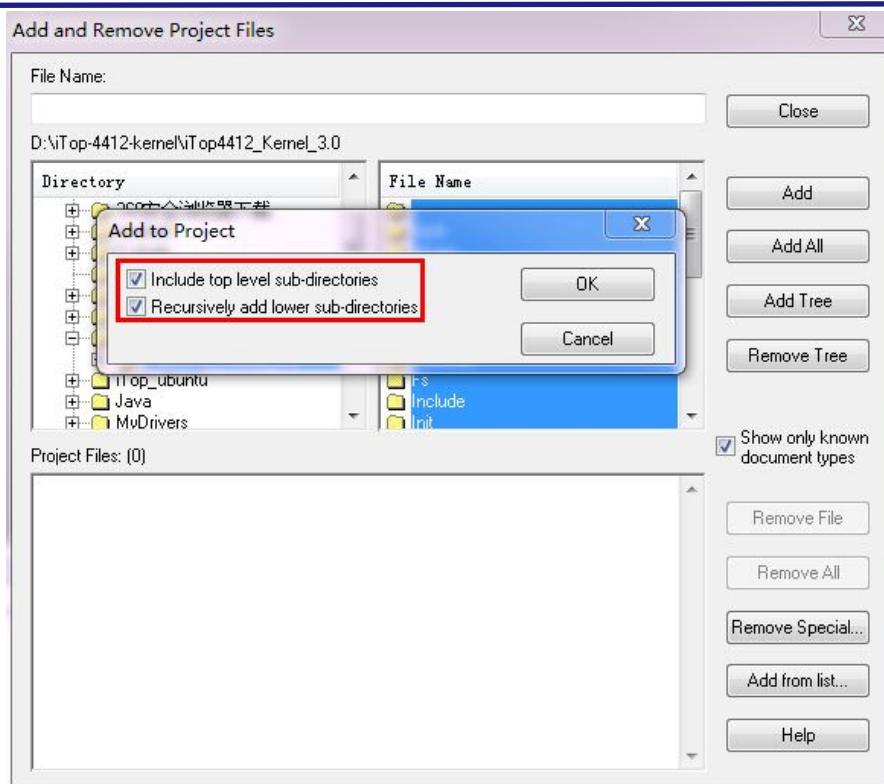
7 ) 如下图 , 单击按钮 “OK” 。



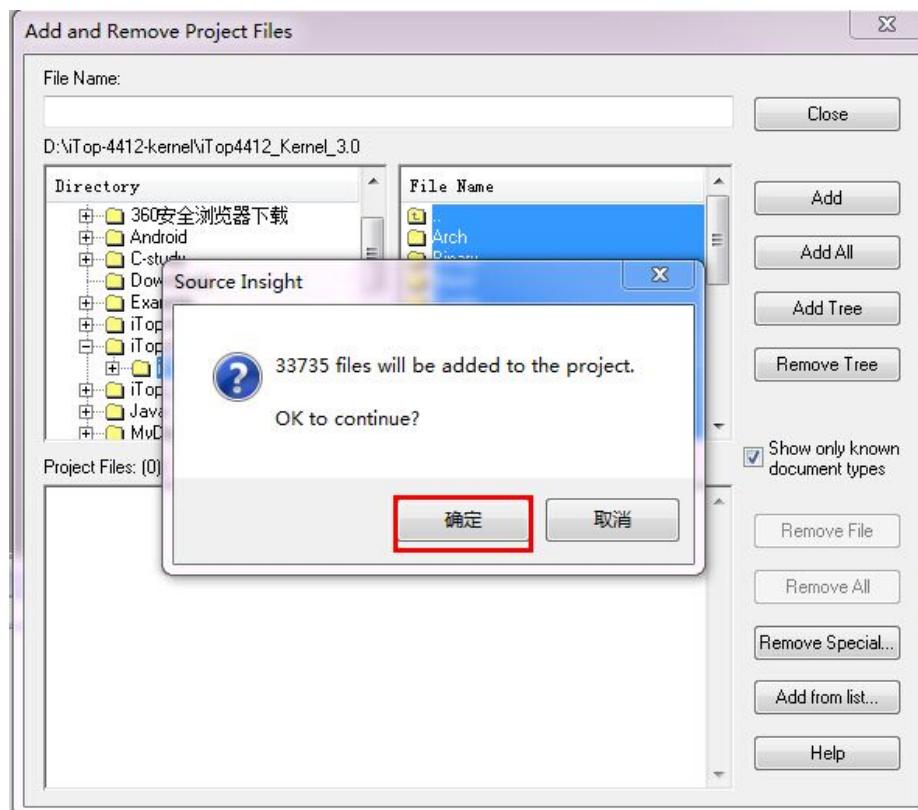
8 ) 如下图 , 同步所有内核源码文件。左边矩形框中的文件夹一定都要选上 , 右边矩形中单击按钮 “Add All” 。



9 ) 如下图 , 在弹出的对话框中 , 选上红色矩形框中的两个选项 , 单击按钮 “OK” 。

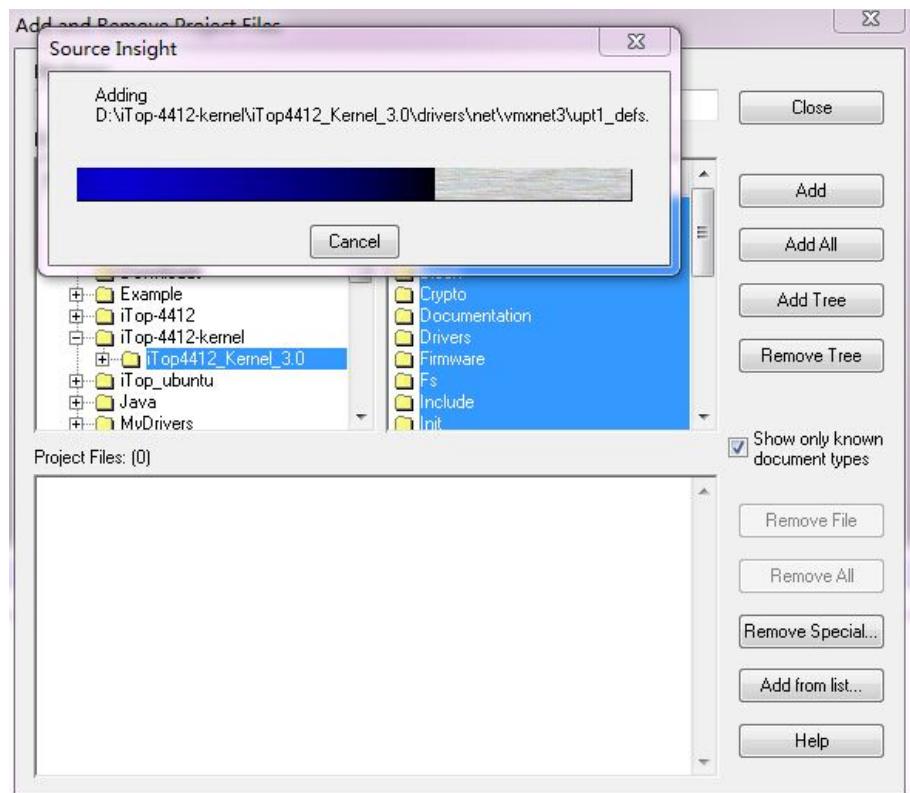


10 ) 如下图 , 弹出的对话框中 , 单击按钮 “ 确定 ” 。



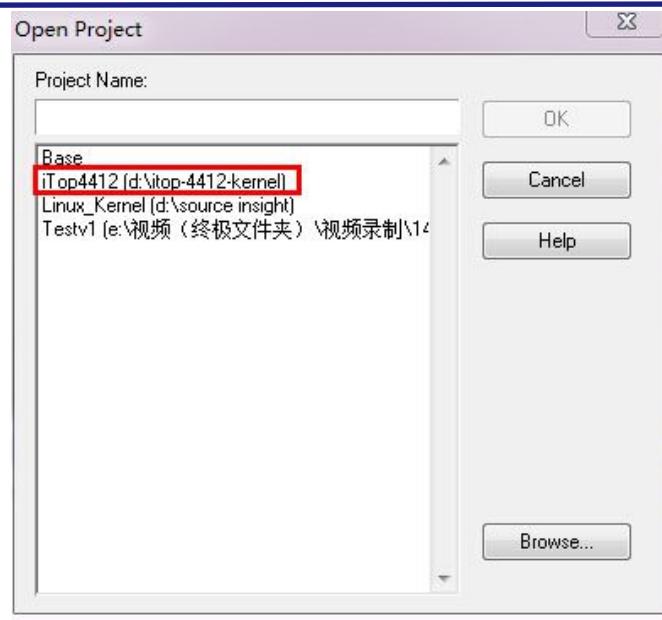
11 ) 接着要用比较久的时间来同步整个内核文件 , 如下图。这里有可能无法完全同步 , 甚至不会弹出同步 , 不过没有关系。如果没有进行同步 , 可以执行后面的步骤 , 来给内核文件同步。

文件同步需要十分钟左右。

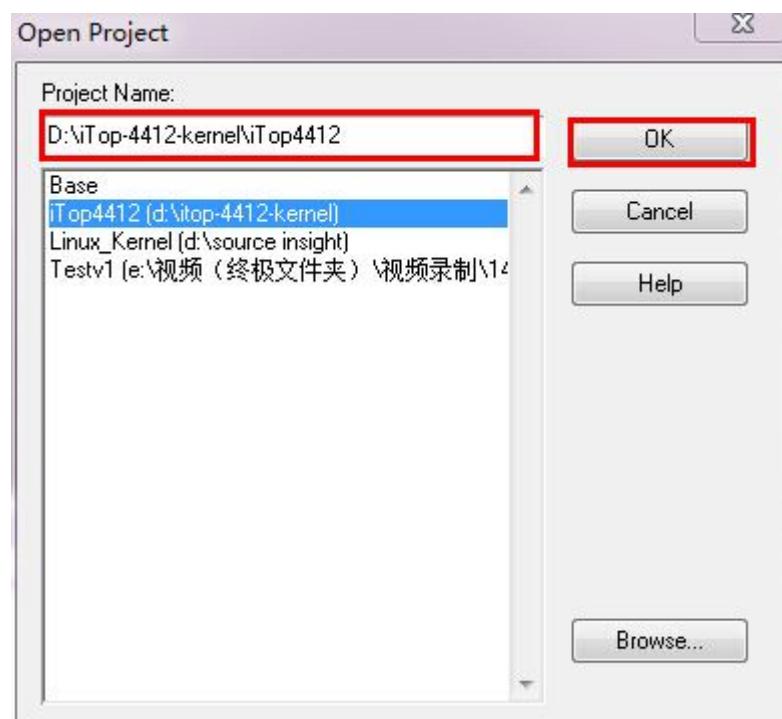


12 ) 有时候工程的文件在第一次加载的时候同步不了 , 还需要进一步的处理。如下图 , 打开前面新建立的内核工程文件。单击菜单 “Project” --> “Open Project” , 如下图 , 选上红色矩形框中的工程 , 这个工程就是用户前面新建的工程。

这里需要注意一点 , 用户新建的工程只有一个 , 另外两个是作者正在使用的工程 , 大家直接忽视掉。



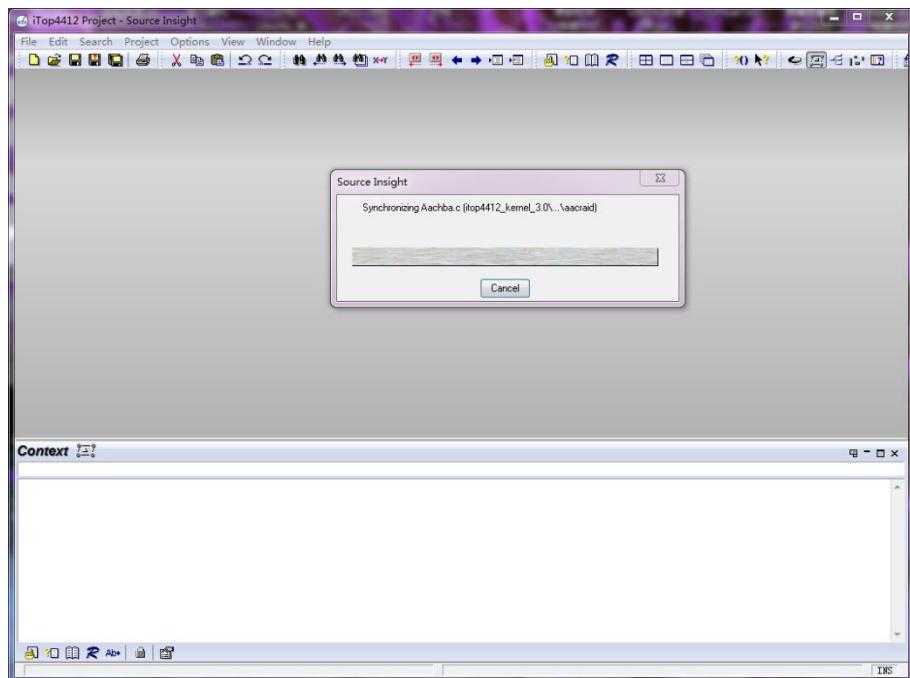
13 ) 如下图 , 单击按钮 “OK” 。



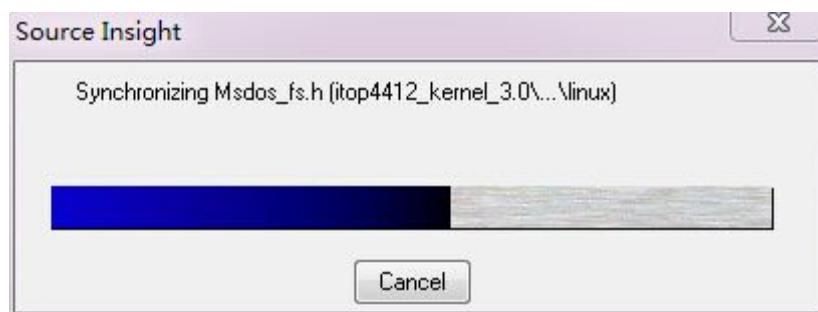
14 ) 如下图 , 它这一步也有可能弹出同步的提示。单击按钮 “是 Y” 。



15 ) 如下图 , 开始同步 , 在这一步中 , 也有可能不弹出同步的对话框 , 但是只要能打开工程 , 也可以使用后面的方法来同步。

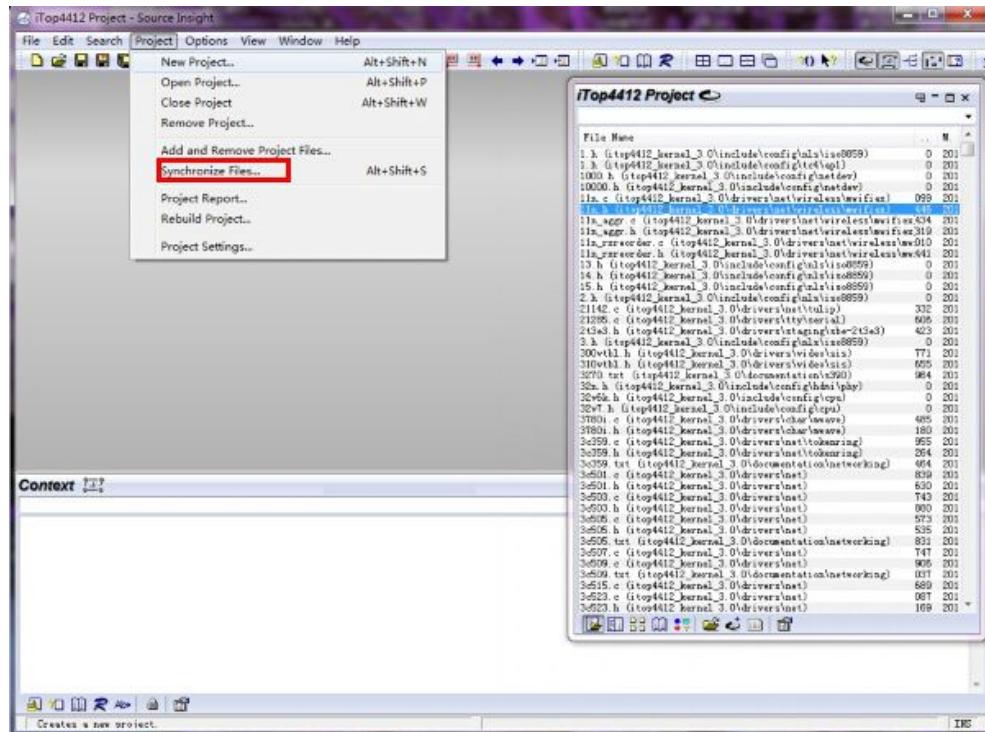


16 ) 如下图 , 内核文件同步中。

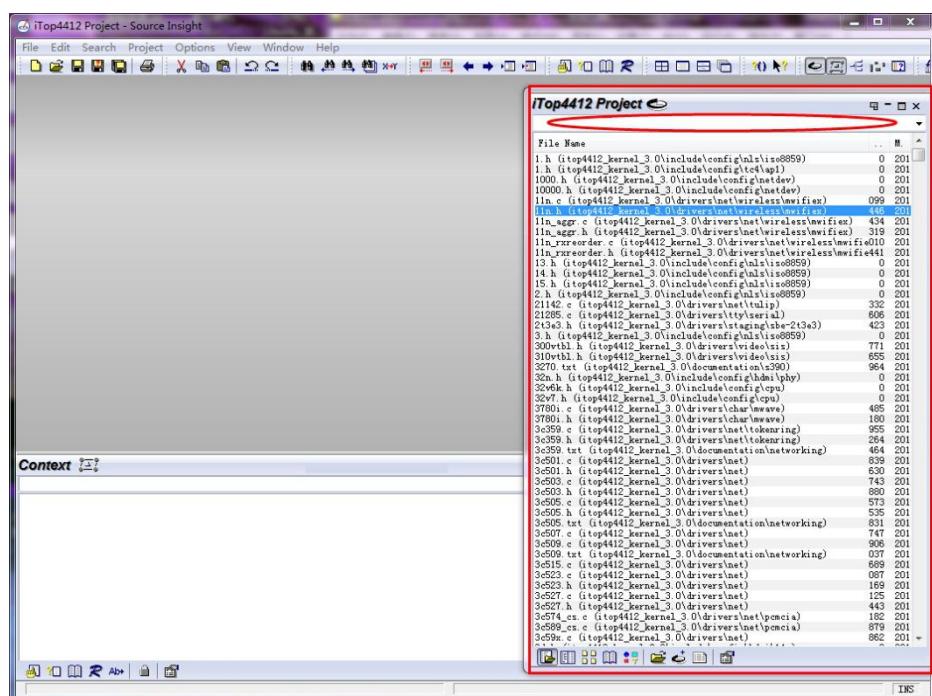


17 ) 如果到了打开工程文件的时候 , 仍旧没有弹出上面的对话框 , 提示同步 , 那么 , 在打开工程后 , 如下图 , 单击菜单 “Project” --> “Synchronize Files.....” , 如果前面提示了

同步，也有可能同步不完整，这里为了保险起见，也同步一下内核文件。操作方法和前面的类似。



18 ) 如下图，在右方的工程窗口中，可以通过输入文件的名称，搜索文件，也可以通过目录查找。



19 ) 这个软件非常强大 , 它可以帮助用户在庞大的内核文件中快速定位 “函数” “变量” 等等 , 比较麻烦的操作就是前面讲的同步 , 用户容易忽视。因为这个软件有友好的窗口界面 , 和其它的 Windows 软件都是类似的 , 例如查找函数以及变量的时候 , 大部分功能都可以通过右键实现 , 使用起来不会有太多的难度 , 用户花一点时间 , 就可以上手了。

## 3.6 安卓 ADB 功能介绍

这一章中 , 会给大家介绍 ADB 驱动的安装和 ADB 命令的使用。这里 ADB 驱动不仅在烧写的时候会用到 , 在用户编译调试安卓应用的时候也会用到。在烧写章节中 , OTG 方式烧写 (或者叫 fastboot 烧写方式) 使用的是这个 ADB 驱动。在本章节中 , 介绍的 ADB 命令 , 也是使用这个驱动。

### 3.6.1 安卓 ADB 驱动的安装

在用户光盘中 , 文件夹 “02\_编译器以及烧写工具” → “tools” → “usb\_otg\_driver” 中 , 有两个驱动软件。一个是 “winxp\_32 位” 文件夹 , 它是 Windows-XP-32 位操作系统的驱动 ; 另外一个是 “win7\_64 位” 软件 , 它是 Win7-64 位操作系统的驱动。用户需要根据具体情况选择安装哪个驱动软件。如果用户使用其他版本的 windows 系统 , 可以通过驱动精灵在网下载合适的 USB 驱动。

下面给大家具体说一下如何安装 , 这里以 Win7-64 位操作系统为例 , Windows-XP-32 位操作系统安装方法类似。

1 ) 如下图 , 单击软件 “android\_drv\_90000\_64.exe” , 开始安装。



2 ) 如下图 , 在弹出的对话框中 , 单击按钮 “下一步 ( N )” , 继续安装。



3 ) 如下图 , 要是装有杀毒软件 , 会弹出对话框 , 直接选择信任或者安装之类的选项。



4 ) 如下图 , 继续安装。

## 设备驱动程序安装向导

正在安装驱动程序...



驱动程序正在安装，请等待。这可能需要一段时间才能完成。

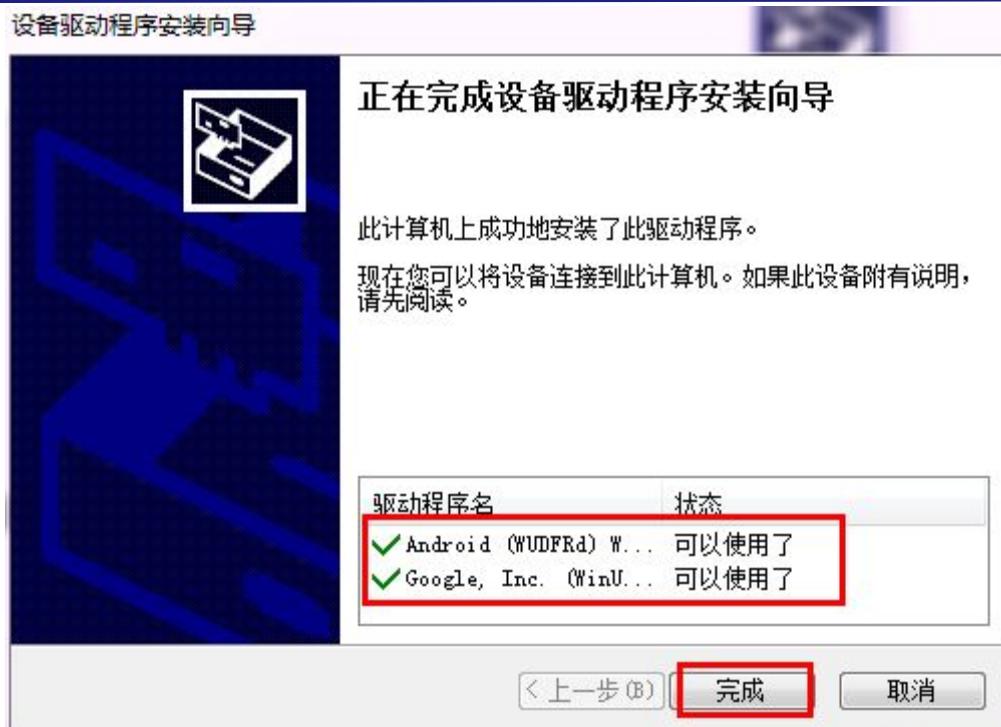
&lt; 上一步(B) 下一步(N) &gt;

取消

5 ) 如下图，可能会再次弹出警告，继续选择信任或者安装之类的选项。



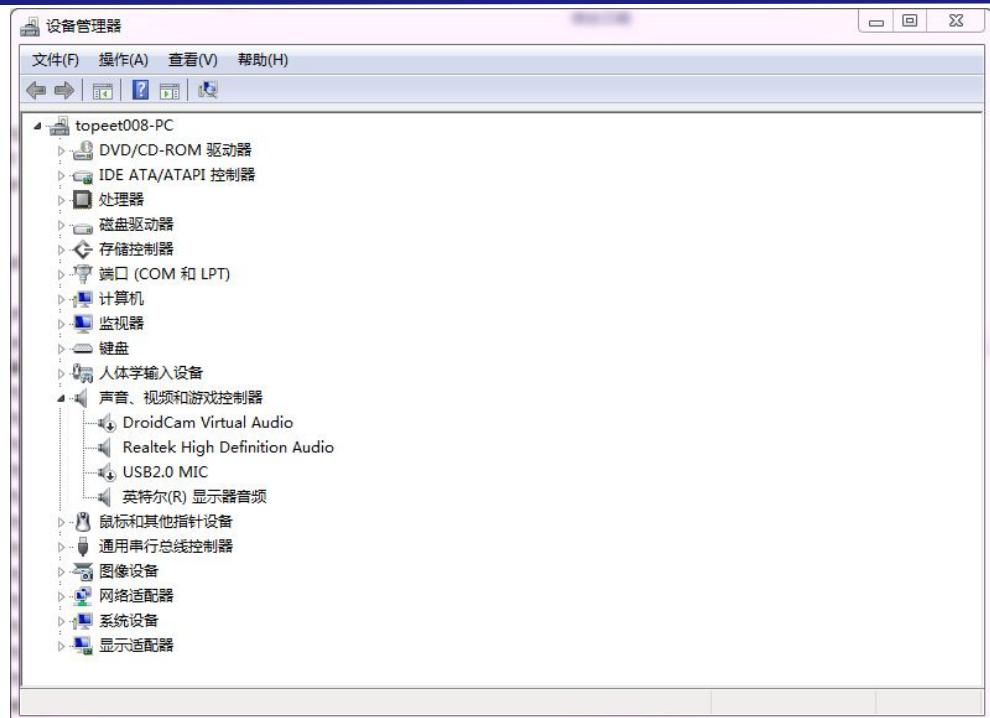
6 ) 如下图，单击按钮 “完成” 。



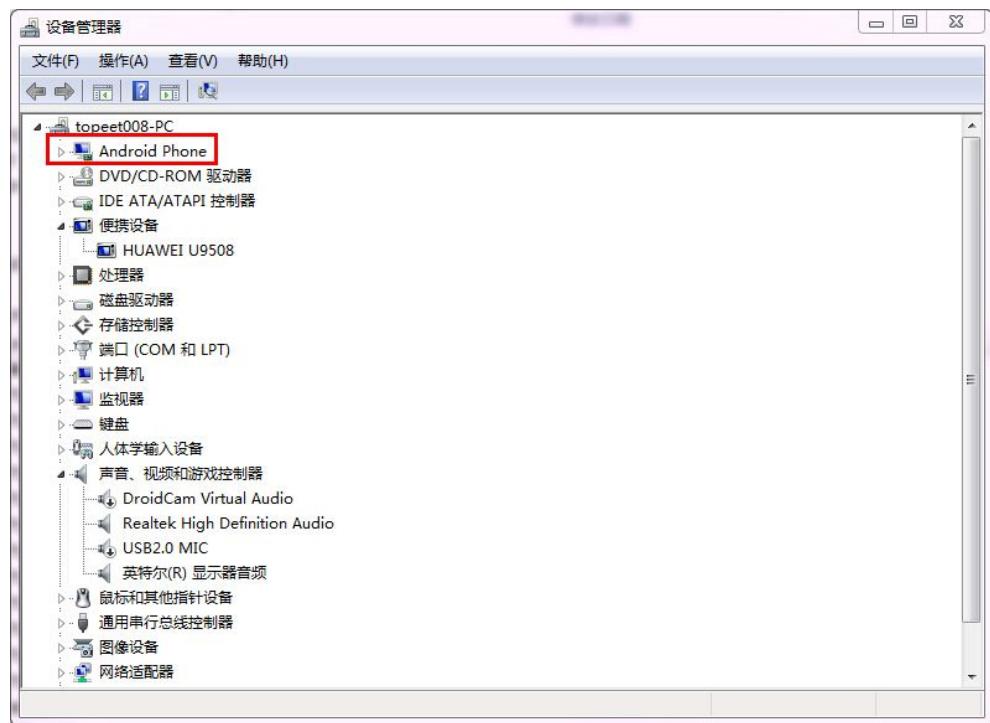
出现上图矩形框中的提示“.....可以使用了” , 就说明驱动安装成功了。驱动装完后 , 最好重启一下电脑。

#### 8 ) 下面来确认一下 , ADB 驱动是否能够使用。

首先进入操作系统的任务管理器 , 下图是 Win7-64 位操作系统的设备管理器 , “3.1.4 小节” 中有说明如何进入 Win7-64 位操作系统的设备管理器 , 用户如果使用其它操作系统 , 可以使用百度查找进入设备管理器的方法。



9 ) 接着将开发板使用 OTG 线和电脑的 USB 接口相连 , 开发板启动并进入到 Android 系统 , 如下图 , 如果出现红色矩形框中的 “Android Phone” , 就说明驱动已经装好。

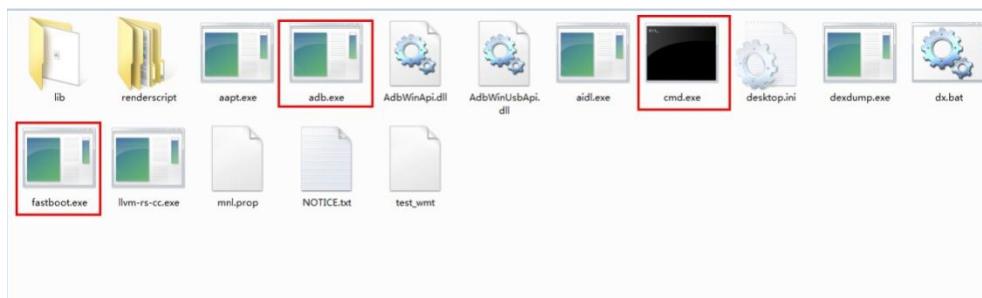


### 3.6.2 ADB 的基础知识

在第四章烧写镜像中，用户会用到 Windows 命令行。在烧写的时候，需要用到用户光盘文件夹 “02\_编译器以及烧写工具” → “tools” → “USB\_fastboot\_tool” 中的工具，拷贝 “USB\_fastboot\_tool” 文件夹到合适的目录（不要放到中文目录下）。

进入到 “USB\_fastboot\_tool” → “platform-tools” ，用户在打开 “cmd.exe” 程序后，这个就是 Windows 命令行。

在这个文件夹中，里面有以下几个小程序需要给大家介绍一下。如下图，方框中的三个小程序 “cmd.exe” “adb.exe” “fastboot.exe”



#### 3.6.2.1 cmd.exe 程序

光盘中的 “cmd.exe” 可以支持 Win7-64 操作系统，如果使用的是其他系统，参考下面的方法来处理。

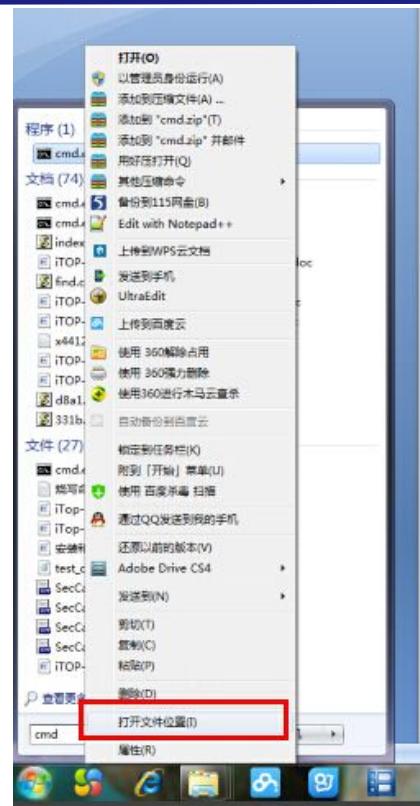
1 ) 用户打开电脑桌面系统的 “开始” 菜单，出现下图界面。在下图矩形框查找栏中输入 “cmd”



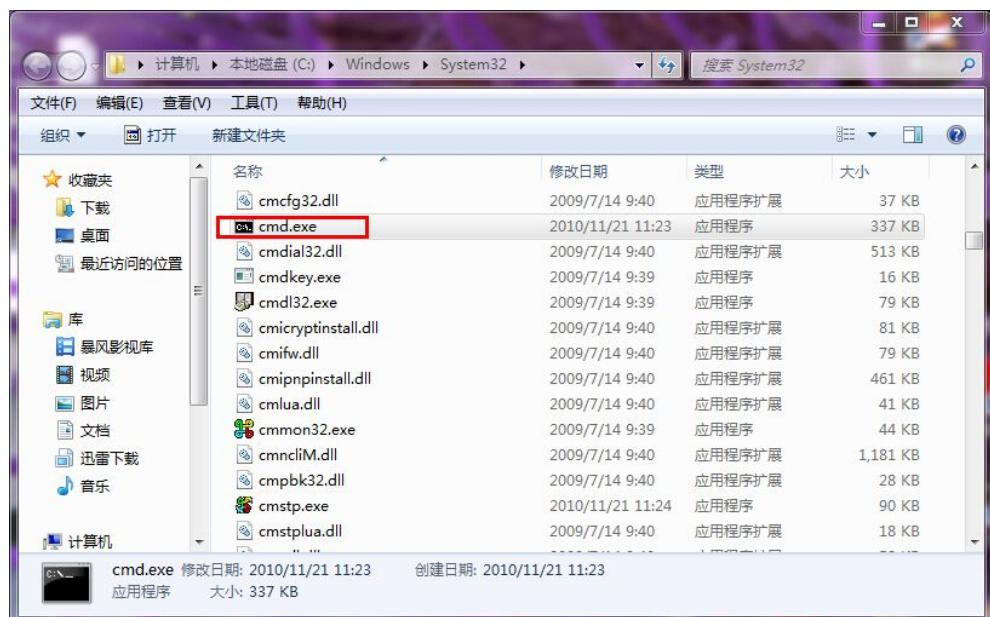
2) 如下图，出现这个操作系统自带的“cmd.exe”小程序。



3) 如下图，(不同的操作系统有点区别)右键单击该程序，找到选项“打开文件设置”



4 ) 如下图 , 可以找到用户自己操作系统的 “cmd.exe” 。



5 ) 用户将自己操作系统的 “cmd.exe” 程序拷贝到 “USB\_fastboot\_tool” 文件夹中 , 将 “cmd.exe” 覆盖 , 这样属于用户自己的 Windows 命令行就运行起来了。

在这个 Windows 命令行中，如果没有其它两个小程序 “adb.exe” 和 “fastboot.exe” ，那么在这个命令行中，就只能输入 DOS 命令。具体的 DOS 命令，用户如果感兴趣，可以百度一下，对于没有 Linux 系统经验的用户，可以了解一下。

### 3.6.2.2 fastboot.exe 程序

用户在烧写的时候，会发现所有的烧写命令前面都会添加 “fastboot” ，这些 fastboot 命令以及 “fastboot.exe” 小程序，都是三星在 Google 提供的 ADB 驱动以及 “adb.exe” 程序的基础上做的。

当然，fastboot 命令只能用来和 exynos4412 的 uboot 模式进行交互，也就是前面烧写镜像中提到的那些命令。

如果开发板完全启动之后，就可以使用 ADB 命令来交互。

### 3.6.2.3 adb.exe 程序

adb 全称是 Android Debug Bridge，是 android sdk 里的一个工具。它可以用来安装/卸载安卓的应用，上传/下载文件等。

当然，如果要用 ADB 命令和开发板交互，那么开发板一定要在文件系统模式，也就是开发板必须运行安卓操作系统，并且安卓文件系统必须完全启动。

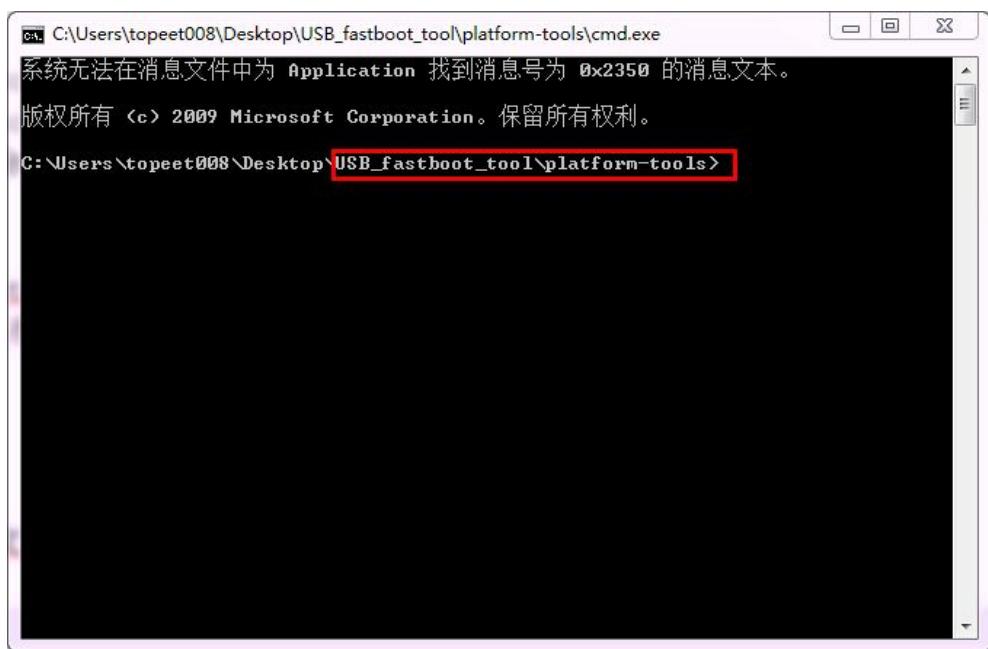
在 ADB 驱动安装完成后，电脑的 USB 接口和开发板的 OTG 接口相连，然后打开 “cmd.exe” ,就可以使用 ADB 功能。

### 3.6.3 常用的 ADB 命令

下面给大家介绍几个基本的 ADB 命令。

将光盘文件夹 “USB\_fastboot\_tool” 拷贝到电脑上。下图中文件夹是放到桌面上的，**有一点需要注意，尽量不要放到中文目录**，这一点似乎已经强调了很多次了。

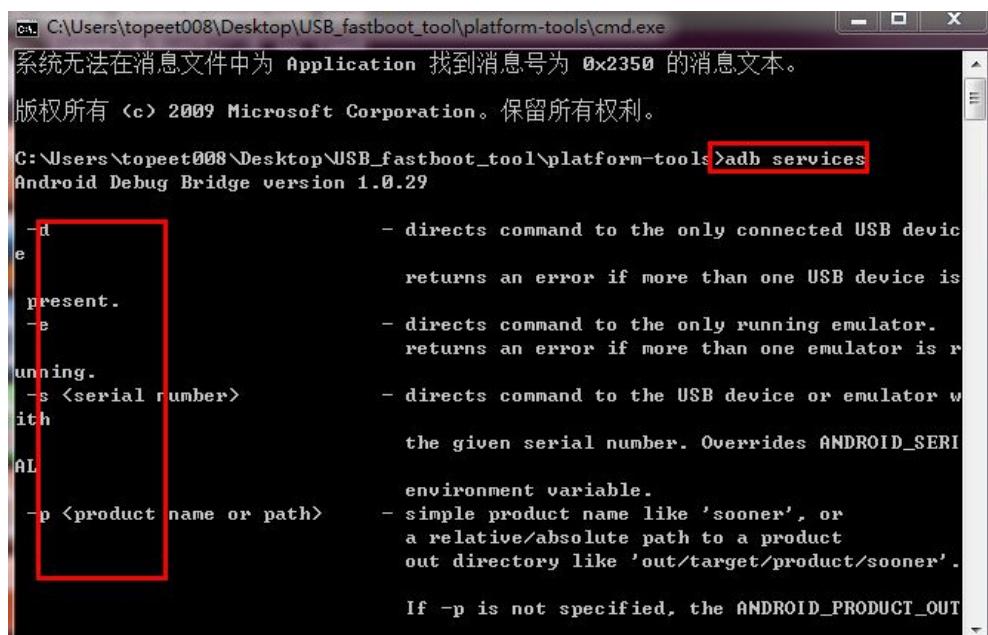
进入文件夹 “.....” --> “USB\_fastboot\_tool” --> “platform-tools” , 单击“cmd.exe” 打开 Windows 命令行,弹出如下界面。

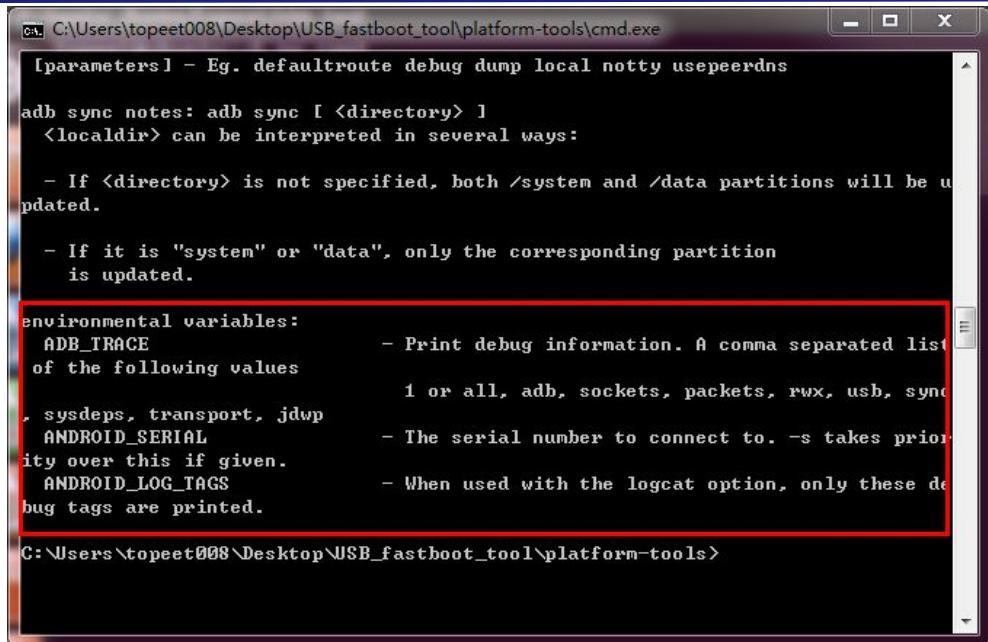


如下图，在 Windows 命令行中，输入以下命令：

adb services

会弹出 adb 命令的帮助文件。





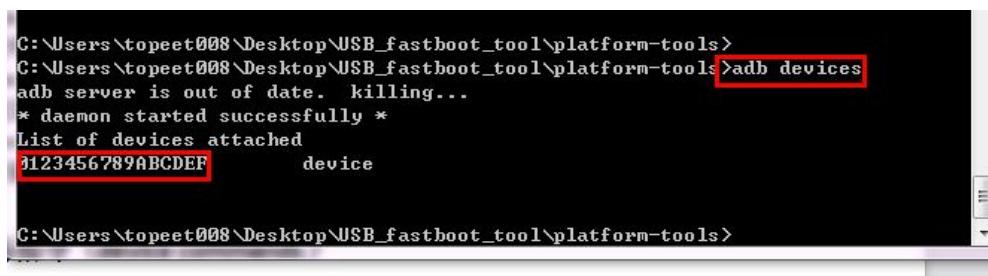
```
C:\Users\topeet008\Desktop\USB_fastboot_tool\platform-tools\cmd.exe
[parameters] - Eg. defaultroute debug dump local notty usepeerdns

adb sync notes: adb sync [ <directory> ]
    <localdir> can be interpreted in several ways:
        - If <directory> is not specified, both /system and /data partitions will be updated.
        - If it is "system" or "data", only the corresponding partition is updated.

environmental variables:
    ADB_TRACE           - Print debug information. A comma separated list of the following values
    , sysdeps, transport, jdwp
    ANDROID_SERIAL      - The serial number to connect to. -s takes priority over this if given.
    ANDROID_LOG_TAGS     - When used with the logcat option, only these debug tags are printed.

C:\Users\topeet008\Desktop\USB_fastboot_tool\platform-tools>
```

如下图，查找 adb 设备，在 Windows 命令行中，输入命令  
adb devices

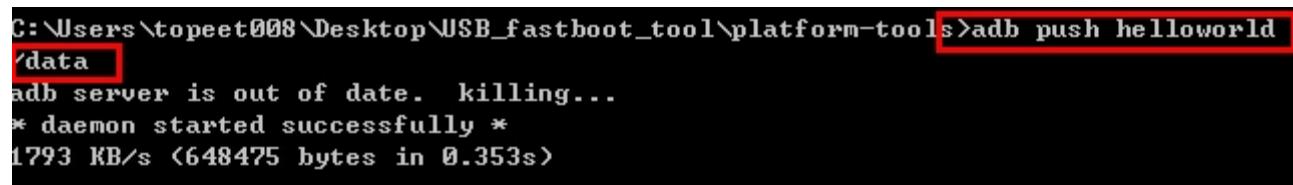


```
C:\Users\topeet008\Desktop\USB_fastboot_tool\platform-tools>
C:\Users\topeet008\Desktop\USB_fastboot_tool\platform-tools>adb devices
adb server is out of date. killing...
* daemon started successfully *
List of devices attached
3123456789ABCDEF       device

C:\Users\topeet008\Desktop\USB_fastboot_tool\platform-tools>
```

如下图，上传 C 的测试程序，到安卓的 “data” 文件夹中。  
在 Windows 命令行中，输入命令

adb push helloworld /data



```
C:\Users\topeet008\Desktop\USB_fastboot_tool\platform-tools>adb push helloworld
'/data'
adb server is out of date. killing...
* daemon started successfully *
1793 KB/s (648475 bytes in 0.353s)
```

如下图，登录开发板 Android 设备的 shell。在登录状态中，用户可以执行更多的 ADB 指令，这些指令都是直接对开发板的安卓文件系统进行操作。

登录安卓 shell 的命令，在 Windows 命令行中，输入命令  
adb shell

```
C:\Users\topeet008\Desktop\USB_fastboot_tool\platform-tools>adb shell
```

如下图，登录后，很多基本命令都是和 Linux 的命令类似，用户可以测试一下，在 Windows 命令行中，输入命令，例如：#ls，#cd 等。

```
C:\Users\topeet008\Desktop\USB_fastboot_tool\platform-tools>adb shell
root@android:/ # ls
ls
acct
cache
config
d
data
default.prop
dev
etc
```

进入上传的“helloworld”文件的安卓文件夹“/data”，在 Windows 命令行中，输入命令

```
cd /data
```

```
root@android:/ # cd /data
cd /data
```

在安卓安卓文件夹“/data”中，查看到上传的“helloworld”。

在 Windows 命令行中，输入命令

```
ls
```

```
root@android:/data # ls
data
dontpanic
drm
helloworld
local
lost+found
misc
property
resource-cache
system
```

如下图，修改文件“helloworld”的权限，

在 Windows 命令行中，输入命令

```
chmod 777 helloworld
```

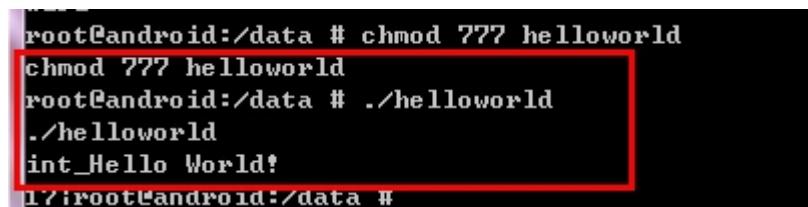


```
root@android:/data # chmod 777 helloworld
chmod 777 helloworld
```

如下图，运行“helloworld”程序。

在 Windows 命令行中，输入命令

```
./helloworld
```

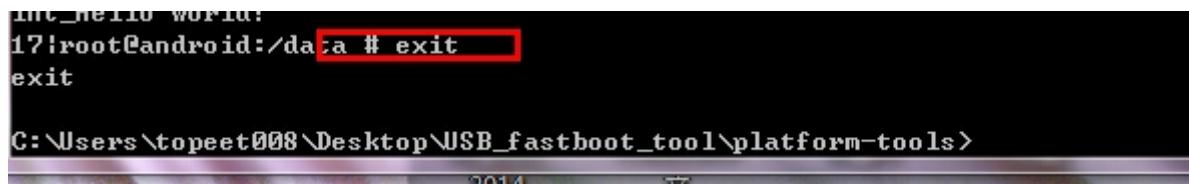


```
root@android:/data # chmod 777 helloworld
chmod 777 helloworld
root@android:/data # ./helloworld
./helloworld
int_Hello_World!
17:root@android:/data #
```

如下图，执行退出安卓 shell 的命令。

在 Windows 命令行中，输入命令

```
exit
```



```
int_Hello_World!
17:root@android:/data # exit
exit

C:\Users\topeet008\Desktop\USB_fastboot_tool\platform-tools>
```

### 3.6.4 ADB 驱动安装常见问题解决办法汇总

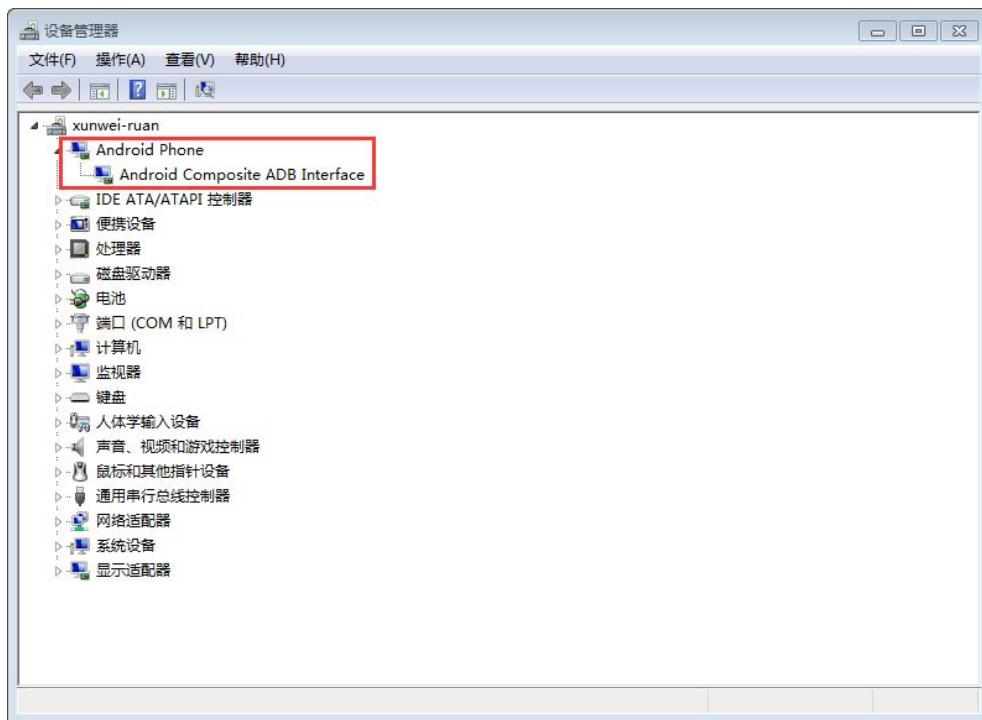
用户在使用 OTG 接口的时候，需要安装 ADB 驱动。无论是烧写系统的镜像还是安装和调试 Android 的 APK 都需要用到这个驱动。

但是经常有用户提到 PC 机无法识别到开发板，下面给大家介绍一下归纳的解决办法。

第一步安装驱动，无论是用户的 PC 机器是什么版本的操作系统，都可以用如下的软件自动安装改驱动。

我们测试了几个常见的 PC 软件，几乎可以自动安装驱动。360 系列软件、qq 以及腾讯管家、百度管家以及百度杀毒、豌豆荚（这个货好难卸载）、暴风影音、驱动精灵以及金山系列等等。

用户使用我们提供的 OTG 线，先把自己的 Android 手机和 PC 的 USB 接口相连。进入设备管理器，如下图，红色框显示设备正常，那么说明 ADB 的驱动已经装好了。



当然可以使用我们提供的驱动，我们提供了在 XP 下的驱动，win7 和 win8（同一个）的驱动，可以参考“3.6.1 安卓 ADB 驱动的安装”。

然后就是第二步，使用 OTG 线连接开发板和 PC 机。在烧写的时候或者调试安装 APK 的时候，仍然经常出问题。经过技术支持人员反复测试，发现 99% 的情况都是 PC 上的软件“打架”，抢着和开发板连接。

这个时候用户需要有点耐心，使用百度。例如下图



目前测试发现的用户可能用到的关键词如下：

禁用 360 自动连接手机

禁用 qq 自动连接手机

禁用腾讯管家自动连接手机

禁用豌豆荚自动连接手机

禁用暴风影音自动连接手机

禁用驱动精灵自动连接手机

禁用金山毒霸自动连接手机

.....

.....

用户需要好好根据实际情况，检查哪些软件需要禁止自动连接手机，一般有 2 个软件自动连接就有很大的几率无法正常使用了。

另外很多其它软件都会自动连接手机，导致无法正确连接。

第三步如果还有烧写系统镜像的问题，请参考一下使用手册 2.3，需要先搞明白 uboot 模式和文件系统模式，这种情况也经常出现。

## 3.7 win8 下基础软件的安装和学习

### 3.7.1 超级终端的安装和使用

#### 3.7.1.1 关闭 win8 的自动更新

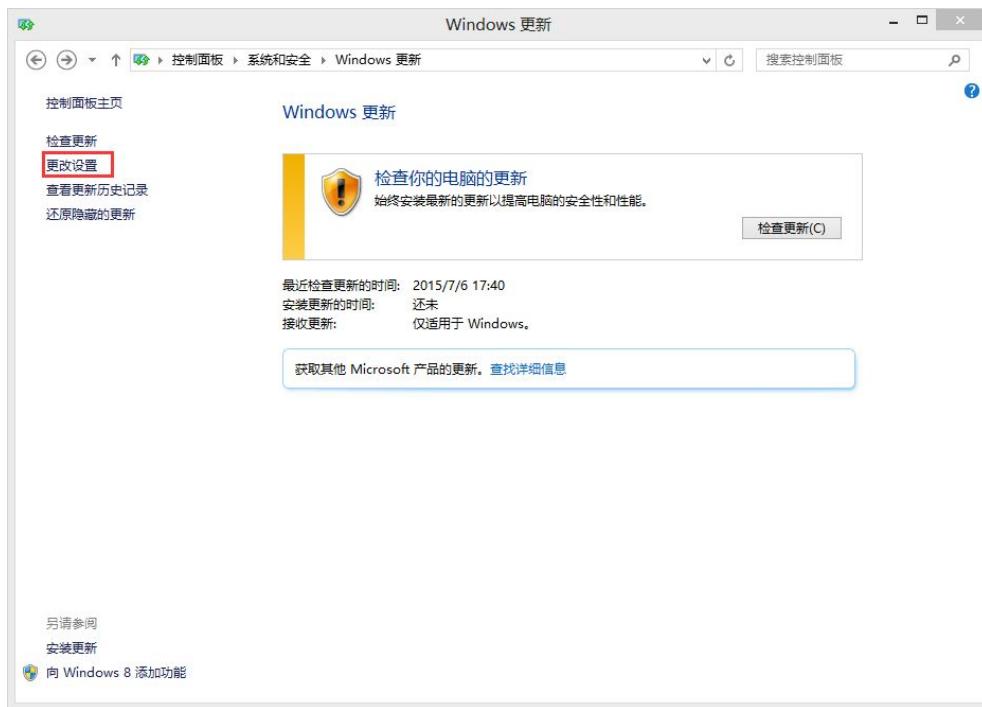
如果用户使用的 PC 机没有串口，就需要用到 USB 转串口驱动。我们给用户配套的硬件是 PL 2303，提供的驱动也是 PL2303。如果用户使用其它的设备，那么所需要的驱动就不一样了。

本来驱动安装后直接使用，但是在 win8 下有自动更新，默认自动更新的，它会更新到最新的版本，最新的版本反而不能够使用，这里建议先关闭 win8 的自动更新功能。

如下图，进入 win8 的“系统”界面，点击“window 更新”。



如下图，更改设置。



如下图所示，先把自动更新全部关掉，可以等我们的 USB 转串口驱动装好，测试完毕再打开。



如下图，单击“确定”，设置完毕。



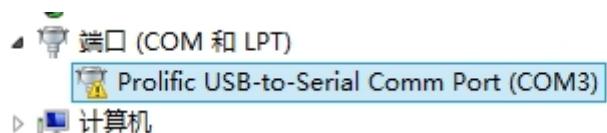
### 3.7.1.2 安装 USB 转串口驱动 PI2303

Win8 的 usb 转串口驱动和 win7 的驱动放在网盘同一位置，如下图所示。

文件名	大小	修改时间
PL2303_Prolific_DriverInstaller_v10518_win8.exe	3.03MB	2015-07-15 17:12
CH340串口线驱动.rar	141KB	2015-04-28 09:25
ReadMe.c	86B	2015-04-28 09:25
迅为usb转串口驱动.zip	2.15MB	2015-04-28 09:25

下载如上图所示对应的驱动，安装方法参考 win7。

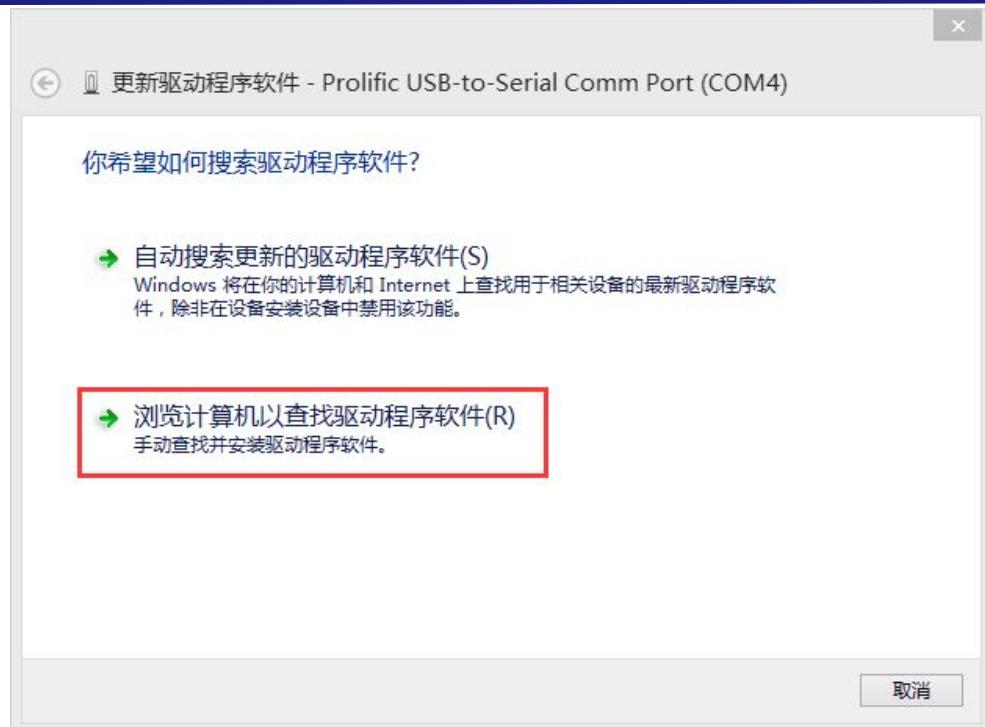
安装完之后，如下图，PC 机接上 USB 转串口模块，进入 win8 的设备管理器，会出现无法识别的“黄色感叹号”。



右键上面，如下图进入驱动设置，单击“更新驱动程序...”



如下图，单击红色框部分



如下图所示，选择红色框，单击“下一步”。



如下图，可以看到系统给安装了多个驱动，如下图所示选择红色框中的低版本驱动。



如下图所示，单击“下一步”。



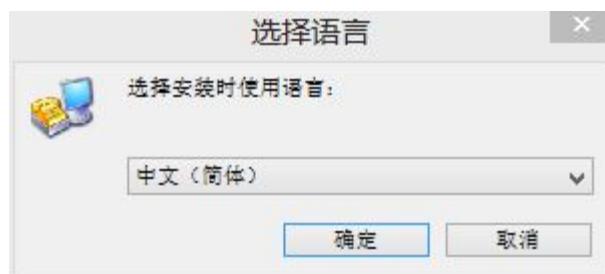
如下图，重启计算机，驱动安装完毕。



### 3.7.1.3 超级终端的安装

Win8 下的超级终端安装比较简单，和 win7 下使用的是同一个。

将软件下载之后，如下图，开始安装。



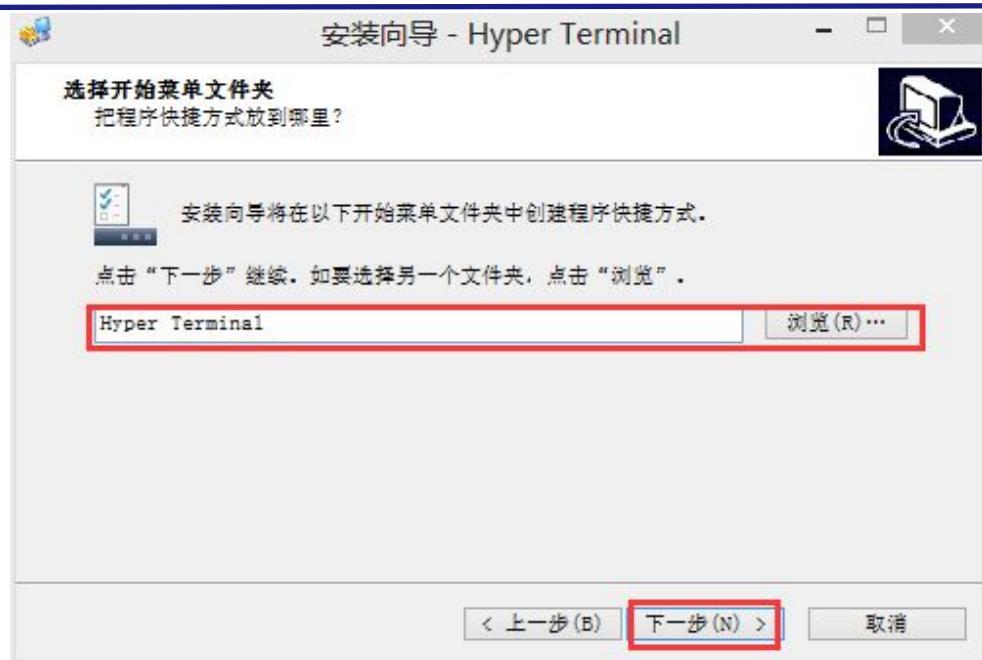
如上图，可以使用中文版本，单击“确定”。



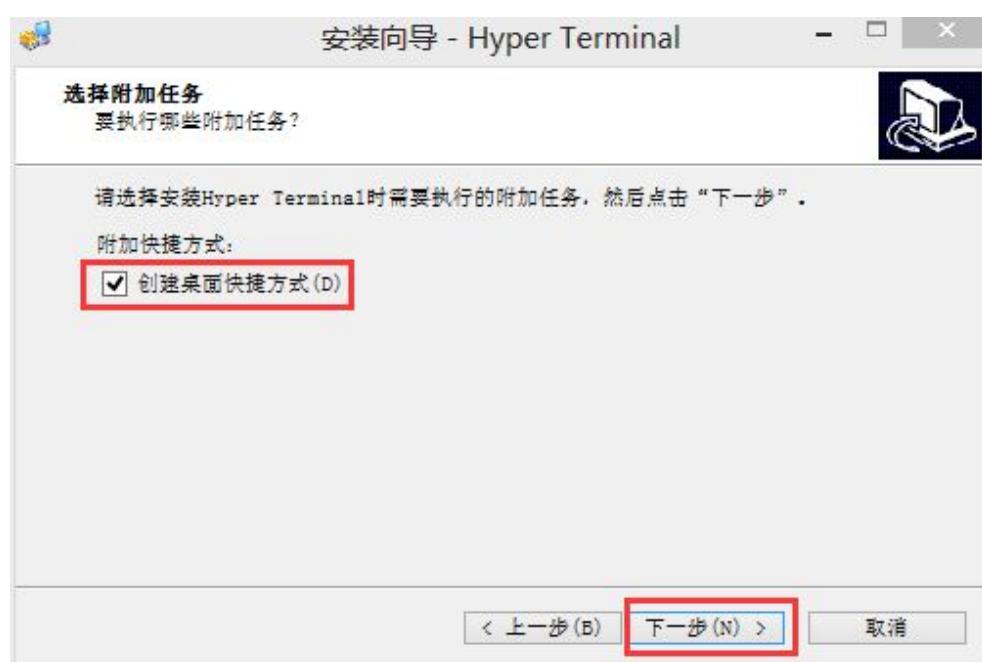
如下图，选择安装位置。单击“下一步”。



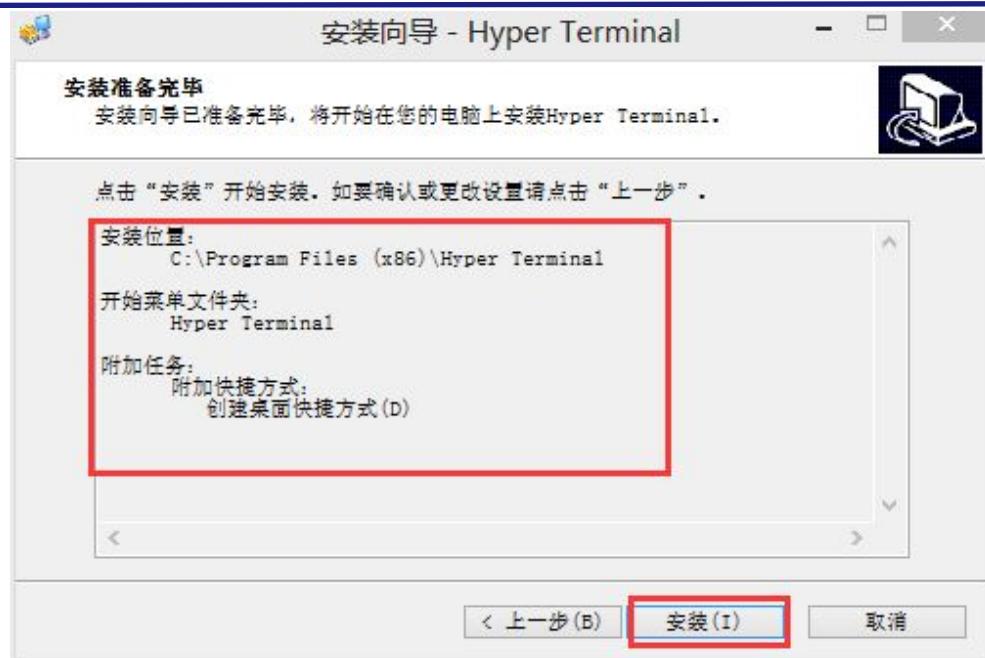
如下图，添加到 windows 的“开始”菜单中。单击“下一步”。



如下图，选择创建桌面快捷方式。单击“下一步”。



如下图所示，单击“安装”。



如下图，单击按钮“结束”。



### 3.7.1.4 其它库的安装

在 win8 下，超级终端在安装的时候可能需要安装其它的微软的各种库文件，这个没有关系，只要根据提示挨个安装即可。

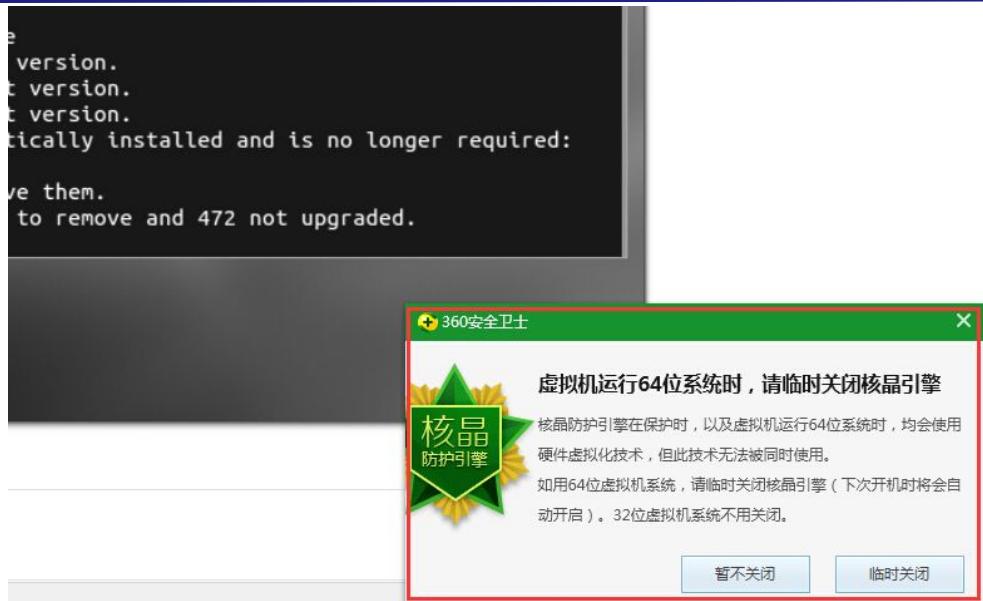
如下图所示，提示安装“.net2.0”和“.net3.0”，安装完库之后再安装超级终端。需要安装的库，在实际情况下和用户的win8系统有关系，下图仅供参考。如果需要安装其它的库，可以使用软件管家搜索下载安装，这里不再赘述。



### 3.7.2 win8 下安装虚拟机以及 Ubuntu12.04.2 等软件

在 win8 下，虚拟机和 Ubuntu 的安装方法和在 win7 下一样。

这里需要特别提醒一下的地方是，在 win8 下面的用户，如果装了 360 之类的软件，在开虚拟机 Ubuntu 的时候会弹出如下界面。这里需要选择“不关闭”，如果嫌烦可以百度一下将这一项设置为不提示。



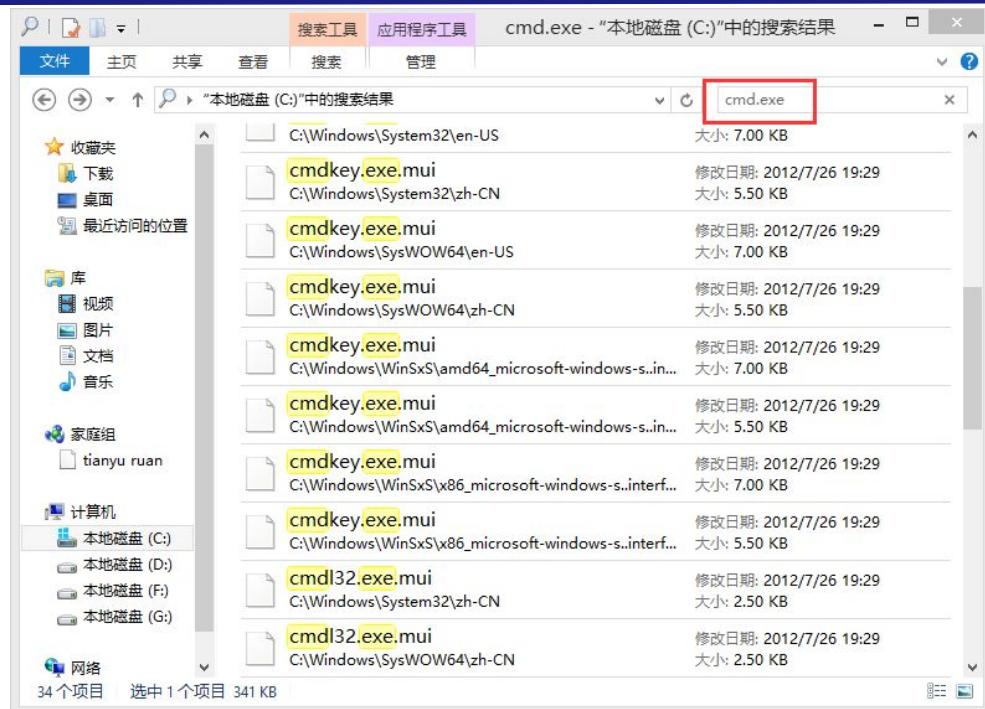
另外，其它软件弹出时候也不能关闭 64 位虚拟化！

### 3.7.3 win8 下的 cmd.exe 程序

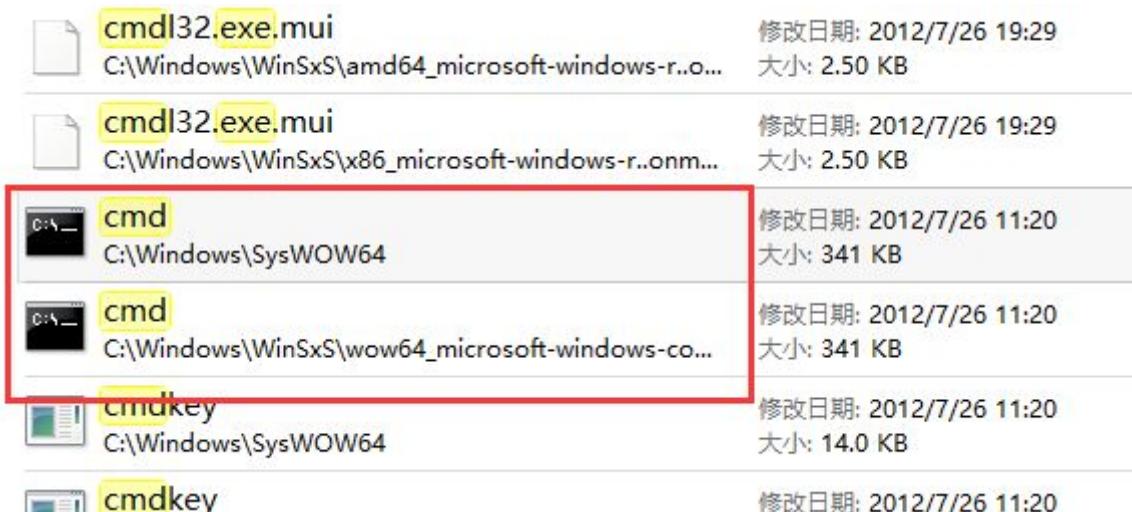
在后面烧写系统镜像的时候，需要用到 win8 自带 cmd.exe ( 参考 3.6.2.1 ) 。

我们提供的 cmd.exe 是，在 win8 下 cmd.exe 不容易那么容易找到，下面简单介绍如何在 win8 下找到 “cmd.exe” 。

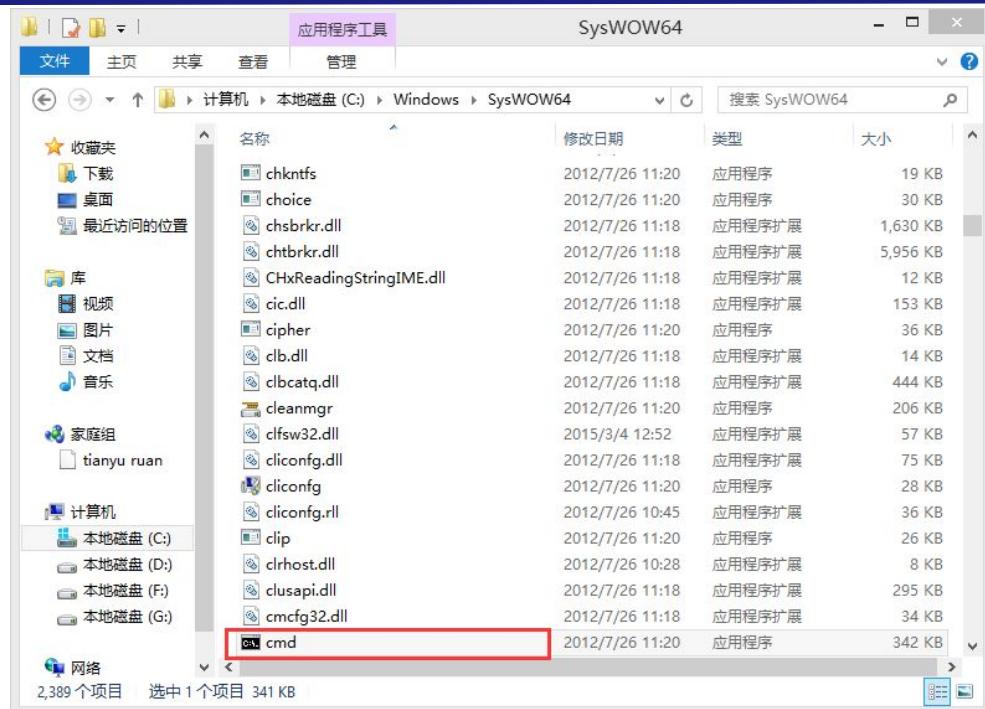
如下图进入 C 盘。搜索 “cmd.exe” ,如下图所示



如下图所示，搜索到“cmd.exe”。



进入 cmd.exe 程序的目录中。



将 cmd.exe 程序拷贝到烧写的文件夹中，该文件夹为光盘中的“USB\_fastboot\_tool”

文件夹。

名称	修改日期	类型	大小
lib	2015/7/15 10:49	文件夹	
renderscript	2015/7/15 10:49	文件夹	
aapt	2014/9/30 11:34	应用程序	805 KB
adb	2014/9/30 11:34	应用程序	159 KB
AdbWinApi.dll	2014/9/30 11:34	应用程序扩展	94 KB
AdbWinUsbApi.dll	2014/9/30 11:34	应用程序扩展	60 KB
aidl	2014/9/30 11:34	应用程序	216 KB
cmd	2014/9/30 11:34	应用程序	337 KB
dexdump	2014/9/30 11:34	应用程序	130 KB
dx	2014/9/30 11:34	Windows 批处理...	3 KB
fastboot	2014/9/30 11:34	应用程序	69 KB
llvm-rs-cc	2014/9/30 11:34	应用程序	18,866 KB
mnl.prop	2014/9/30 11:34	PROP 文件	1 KB
NOTICE	2014/9/30 11:34	文本文档	369 KB
test_wmt	2014/9/30 11:34	文件	638 KB

## 四 iTOP-4412 开发板镜像的烧写

镜像，是一种文件形式，可以把许多文件做成一个镜像文件。说到底，镜像就是源代码编译并连接以后生成的可执行文件包，把这些镜像文件烧写到开发板的存储芯片里，开机就可以运行了。

烧写方式有两种，通过 TF 卡烧写以及使用 OTG 接口烧写。

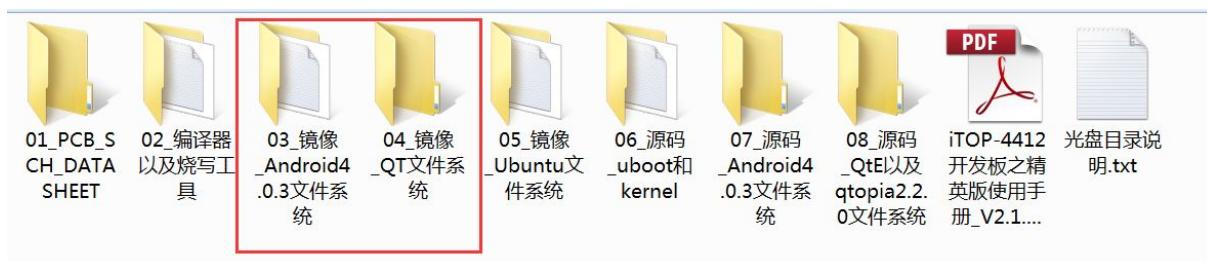
注意 1：Android 系统和 Qt 系统的烧写完全一样，是通用的。

注意 2：无论是使用 OTG 接口方式还是使用 TF 卡方式都不需要设置拨码开关，只有在出现特殊情况时才需要调整拨码开关，在本章最后一小节会提到什么情况需要重新设置拨码开关。

### 4.1 镜像文件说明

迅为 iTOP-4412 开发板平台支持的功能较多，镜像源码也很多，所以这一小节中先给大家介绍一下镜像文件在光盘中的位置。

开发板配套光盘，如下图所示，可以看到“03\_镜像\_Android4.0.3 文件系统”和“04\_镜像\_QT 文件系统”文件夹。Android4.0.3 和 Qt 操作系统和开发板配套的镜像都在这两个文件夹中。



#### 4.1.1 Android4.0.3 和三种核心板配套的镜像

Android4.0.3 系统需要的四个的镜像都在光盘的目录 “03\_镜像\_Android4.0.3 文件系统” 下，如下图所示，该目录分为 “uboot” ， “zImage” 和 “system” 三个文件夹。

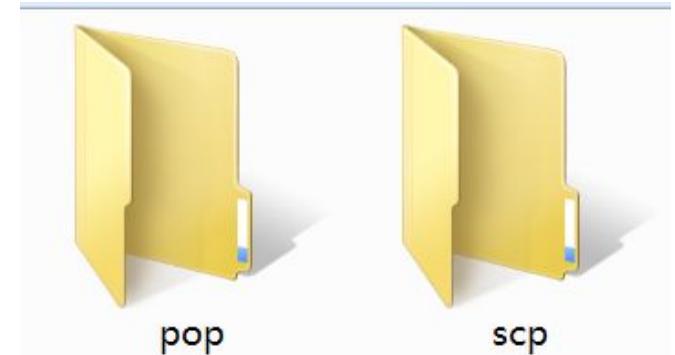


先根据手册 1.1.1 小节确定自己开发板的核心板的是 SCP 1G, SCP 2G 还是 POP。

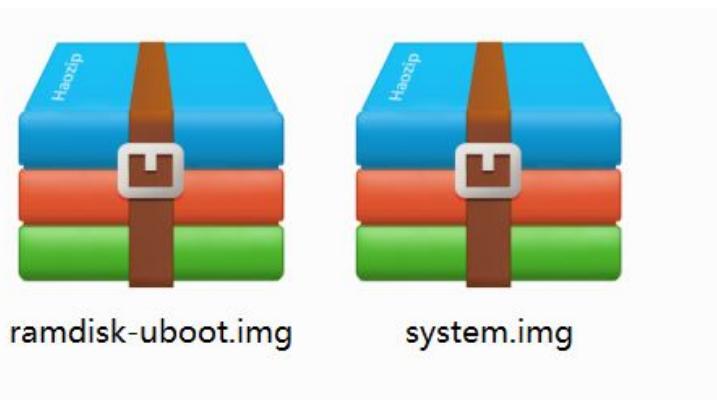
uboot 镜像文件 “u-boot-iTOP-4412.bin” , 在光盘 “03\_镜像\_Android4.0.3 文件系统” → “uboot” 目录下，如下图所示，分为 “pop” , “scp1G” , “scp2G” , 用户根据实际情况选用和核心板对应的 “u-boot-iTOP-4412.bin” 文件。



kernel 镜像文件 “zImage” , 在光盘 “03\_镜像\_Android4.0.3 文件系统” → “zImage” 目录下，如下图所示，分为 “pop” 和 “scp” 文件夹。SCP 1G 和 SCP 2G 核心板通用 SCP 目录下的 “zImage” 镜像文件，POP 核心板使用 “pop” 目录下的 “zImage” 镜像文件。



Android 镜像文件 “ramdisk-uboot.img” 和 “system.img” , 在光盘 “03\_镜像\_Android4.0.3 文件系统” → “system” 目录下 , 如下图所示 , 全部核心板通用一套镜像文件。



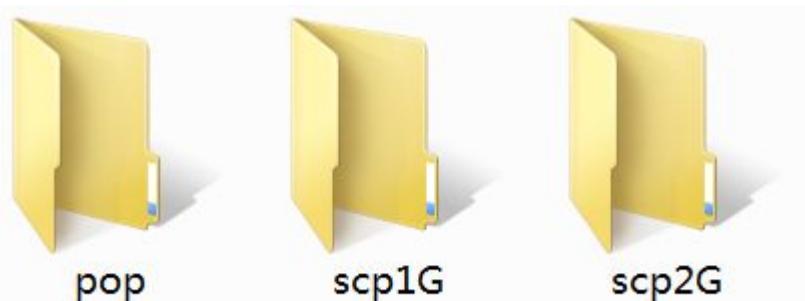
#### 4.1.2 Qt 和三种核心板配套的镜像

Qt 系统需要的四个的镜像都在光盘的目录 “04\_镜像\_QT 文件系统” 下 , 如下图所示 , 该目录分为 “uboot” , “zImage” 和 “system” 三个文件夹。

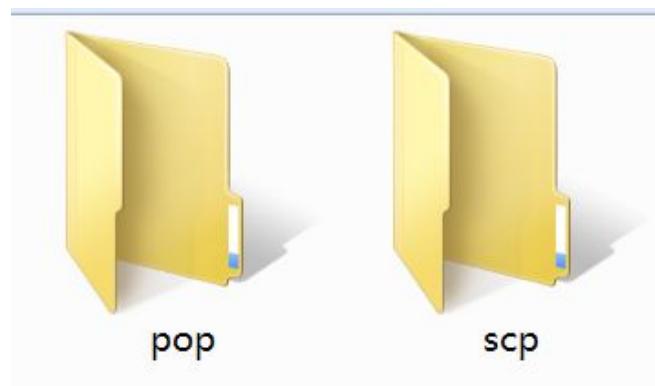


先根据手册 1.1.1 小节确定自己开发板的核心板的是 SCP 1G, SCP 2G 还是 POP。

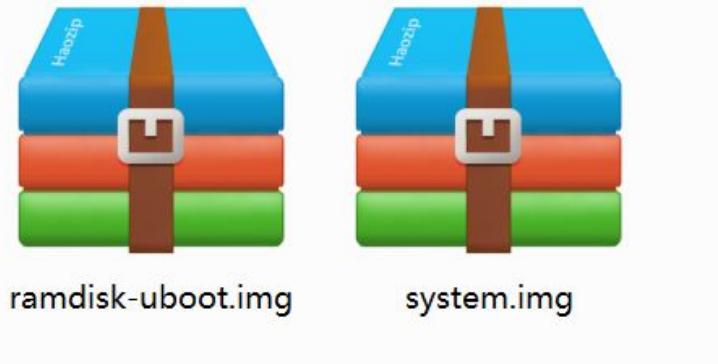
uboot 镜像文件 “u-boot-iTOP-4412.bin” , 在光盘 “04\_镜像\_QT 文件系统” → “uboot” 目录下 , 如下图所示 , 分为 “pop” , “scp1G” , “scp2G” , 用户根据实际情况选用和核心板对应的 “u-boot-iTOP-4412.bin” 文件。



kernel 镜像文件 “zImage” , 在光盘 “04\_镜像\_QT 文件系统” → “zImage” 目录下 , 如下图所示 , 分为 “pop” 和 “scp” 文件夹。SCP 1G 和 SCP 2G 核心板通用 SCP 目录下的 “zImage” 镜像文件 , POP 核心板使用 “pop” 目录下的 “zImage” 镜像文件。



Qt 镜像文件 “ramdisk-uboot.img” 和 “system.img” , 在光盘 “04\_镜像\_QT 文件系统” → “system” 目录下 , 如下图所示 , 全部核心板通用一套镜像文件。



## 4.2 OTG 接口烧写方式

通过该方式可以烧写 Android 系统和 Qt 系统。

需要准备一根 OTG 线，绝大多数智能手机和 PC 机相连接的线都是 OTG 线，都是通用的。

这种方式比 TF 卡烧写方式要快一些，建议调试的时候使用这种方法。

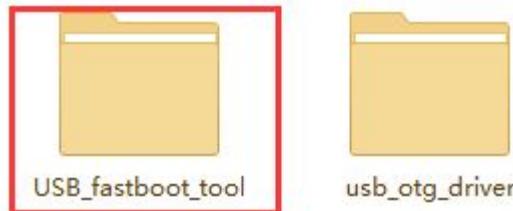
OTG 接口烧写方式也叫 fastboot 烧写方式，先介绍 OTG 烧写使用的硬件和软件平台以及烧写步骤，再介绍烧写对应的操作。

### 4.2.1 硬件平台

- 1 ) 使用串口线连接开发板串口（精英版是靠近网口的串口 CON3）到 PC 机串口
- 2 ) 使用 OTG 线，将开发板 OTG 接口和 PC 的 USB 接口相连。
- 3 ) 连接电源，屏幕等

### 4.2.2 软件平台

- 1 ) OTG 方式只能在 XP、WIN7 或者 WIN8 系统下实现。注意，如果用户不是在 Win7 环境下，需要参考 3.6.2 中的“cmd.exe”，使用用户系统自带的终端。
- 2 ) 打开串口工具（超级终端或者其它串口助手），设置默认波特率为 115200，参考“3.1 超级终端的使用”。
- 3 ) 将光盘中的文件夹“02\_编译器以及烧写工具” → “tools” → “USB\_fastboot\_tool” 文件夹拷贝到您方便使用的地方，因为这个文件夹会经常被用到，注意不要拷贝到中文目录下（桌面不算是中文目录），如下图所示。

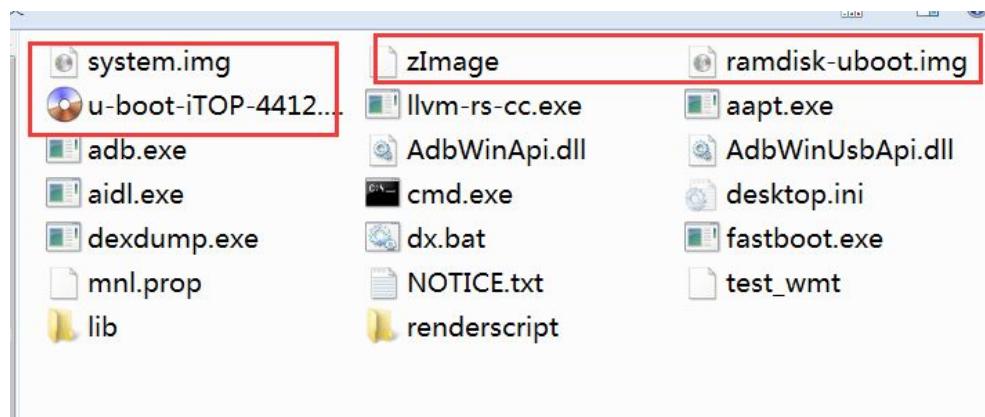


#### 4 ) 安装 PC 机的 USB 驱动。

用户光盘"usb\_driver"文件夹中有 USB 驱动，这个 USB 驱动和 Android 的 ADB 驱动通用，可以参考“3.6 安卓 ADB 功能介绍”。提供 xp 和 win7 版本的 USB 驱动。  
“android\_drv\_70000\_32.exe” 文件，这个是 xp 32 位的版本；  
“android\_drv\_90000\_64.exe” 文件，这个是 win7 64 位的版本。

### 4.2.3 烧写步骤

1 ) 将 Android4.0.3 文件系统和核心板配套的四个镜像 “ramdisk-uboot.img” 、  
“system.img” 、 “u-boot-iTOP-4412.bin” 、 “zImage” 拷贝到  
“USB\_fastboot\_tool” --> “platform-tools” 文件夹下面。 (注意拷贝前一定要先确认自己核心板是属于哪一类，然后在对应的文件夹中拷贝出来)



1 ) 打开超级终端，然后上电启动开发板，按 “回车” ，进入 Uboot 模式，不明白 uboot 模式可以参考“2.3 uboot 模式和文件系统模式”。如下图所示，进入 uboot 模式。

```
In:    serial
Out:   serial
Err:   serial
eMMC OPEN Success.!!
                           !!!Notice!!!
!You must close eMMC boot Partition after all image writing!
!eMMC boot partition has continuity at image writing time.!
!So, Do not close boot partition, Before, all images is written.

MMC read: dev # 0, block # 48, count 16 ...16 blocks read: OK
eMMC CLOSE Success.!!

Checking Boot Mode ... EMMC4.41
SYSTEM ENTER NORMAL BOOT MODE
Hit any key to stop autoboot:  0
iTOP-4412 #
```

3 ) 创建 eMMC 分区并格式化。如果原来已经做过此步骤，则可以跳过，不必每次烧写前都分区和格式化。在超级终端中，输入下面分区和格式化命令。

如下图所示，输入分区命令 “fdisk -c 0”。

```
iTOP-4412 # fdisk -c 0
.fdisk is completed

      partition #      size (MB)      block start #      block count      partition_Id
          1            1340            4862616            2744544      0x0C
          2            1026            37290            2103156      0x83
          3            1026            2140446            2103156      0x83
          4             302            4243602            619014      0x83
iTOP-4412 #
```

如下图所示，输入命令 “fatformat mmc 0:1”。

```
iTOP-4412 # fatformat mmc 0:1
Start format MMC&d partition&d ...
Partition1: Start Address(0x4a3298), Size(0x29e0e0)
.....size checking ...
Under 8G
write FAT info: 32
Fat size : 0xa78
..Erase FAT region...
.
.
.
Partition1 format complete.
iTOP-4412 #
```

如下图所示，输入命令 “ext3format mmc 0:2”。

```
iTOP-4412 # ext3format mmc 0:2
Start format MMC0 partition2 ....
** Partition2 is not ext2 file-system 0 ***
Partition2: Start Address(0x91aa), Size(0x201774)
Start ext2format...
Wirte 0/9block-group
Reserved blocks for jounaling : 8202
Start write addr : 0x91aa
.....
.....d_indirect_point:0xc2aa
..Wirte 1/9block-group
Reserved blocks for jounaling : 8202
Start write addr : 0x491aa
....Erase inode table(1) - 0x493ca.....
Wirte 2/9block-group
Reserved blocks for jounaling : 8202
Start write addr : 0x891aa
....Erase inode table(2) - 0x891ba.....
Wirte 3/9block-group
Reserved blocks for jounaling : 8202
```

如下图所示，输入命令 “ext3format mmc 0:3”。

如下图所示，输入命令 “ext3format mmc 0:4”。

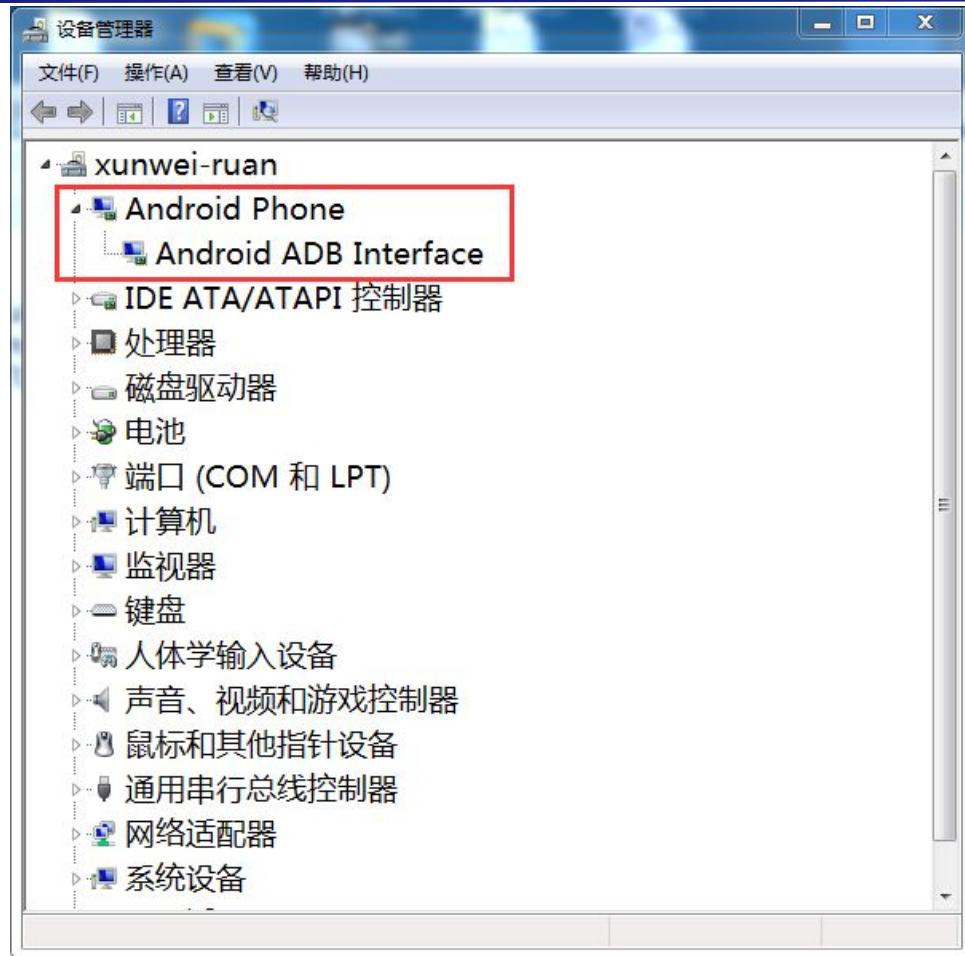
```
iTOP-4412 # ext3format mmc 0:4
Start format MMC0 partition4 .....
** Partition4 is not ext2 file-system 0 ***
Partition4: Start Address(0x40c092), Size(0x97206)
Start ext2format...
Wirte 0/3block-group
Reserved blocks for journaling : 4102
Start write addr : 0x40c092
.....Erase inode table(0) - 0x40c142.....
.....d indirect_point:0x40fb12
..Wirte 1/3block-group
Reserved blocks for journaling : 4102
Start write addr : 0x44c092
....Erase inode table(1) - 0x44c142.....
Wirte 2/3block-group
Reserved blocks for journaling : 4102
Start write addr : 0x48c092
..Erase inode table(2) - 0x48c0a2.....
iTOP-4412 #
```

如下图所示，在超级终端中，输入命令“fastboot”

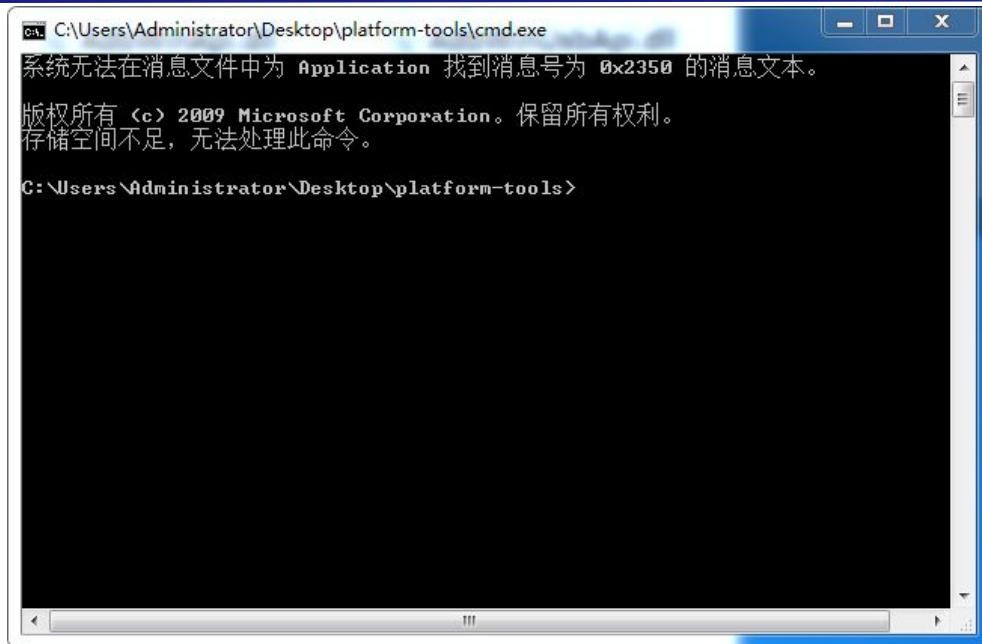
```
iTOP-4412 # fastboot
[Partition table on MoviNAND]
ptn 0 name='bootloader' start=0x0 len=N/A (use hard-coded info. ())
ptn 1 name='kernel' start=N/A len=N/A (use hard-coded info. (cmd:
ptn 2 name='ramdisk' start=N/A len=0x300000 (~3072KB) (use hard-co
ptn 3 name='Recovery' start=N/A len=0x600000 (~6144KB) (use hard-c
ptn 4 name='system' start=0x1235400 len=0x402EE800 (~1051578KB)
ptn 5 name='userdata' start=0x41523C00 len=0x402EE800 (~1051578KB)
ptn 6 name='cache' start=0x81812400 len=0x12E40C00 (~309507KB)
ptn 7 name='fat' start=0x94653000 len=0x53C1C000 (~1372272KB)
```

注意，**fastboot** 命令需要与 PC 上的 **USB\_fastboot\_tool** 工具配套使用，而且 **fastboot** 命令需要进入 **uboot** 模式中才能使用。

4 ) 如下图所示，检查一下 PC 是否识别了设备，需要用 OTG 线将开发板 OTG 接口和 PC 的 USB 接口相连。



5 ) 在 PC 机上运行 “USB\_fastboot\_tool” --> “platform-tools” 文件夹中的文件 “cmd.exe” ( cmd.exe 可执行文件是 Windows 自带的命令行工具 , 光盘里面的是 Win7 下的 , 如果提示版本不兼容 , 请使用你自己系统里面的 cmd.exe 工具 , 具体可以参考 3.6.2.1 cmd.exe 程序 ) , 如下图所示。

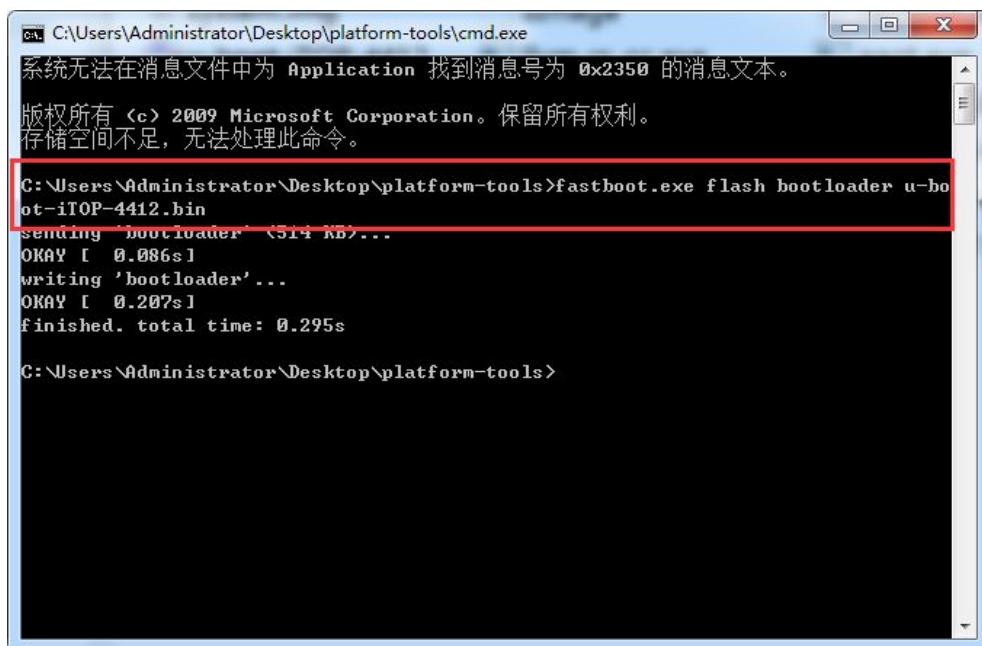


6 ) 在 Windows 命令行中 , 输入下面的命令 :

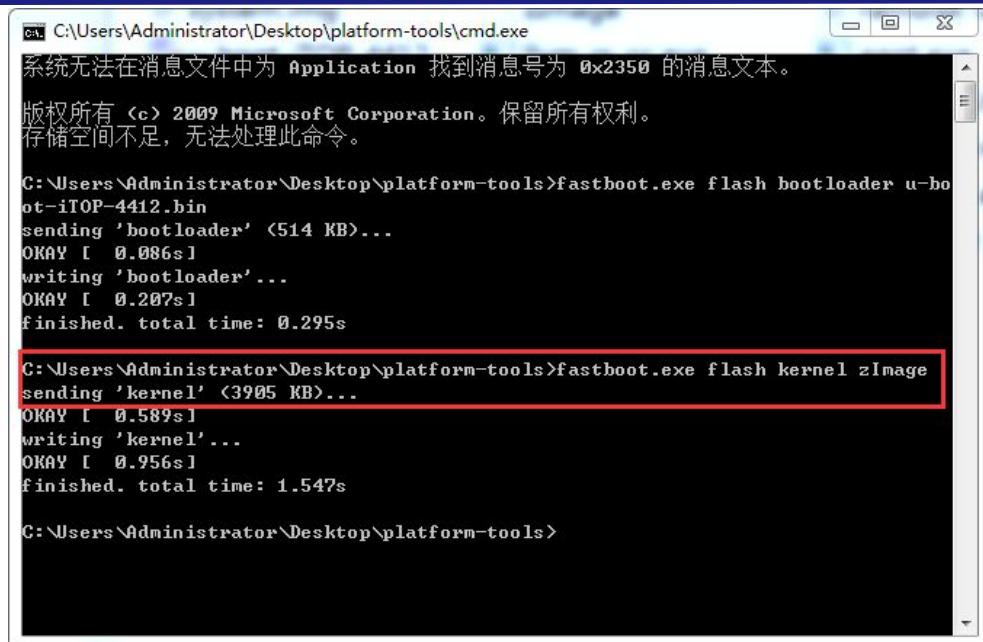
如下图所示 , 输入烧写 uboot 命令

“fastboot.exe flash bootloader u-boot-iTOP-4412.bin”

特别提醒 , 不建议用户烧写 “u-boot-iTOP-4412.bin” 这个文件 , 可跳过此步骤 , 因为出厂前已经烧写过这个镜像文件了。



如下图所示 , 输入烧写 zImage 内核命令 “fastboot.exe flash kernel zImage”



```
C:\Users\Administrator\Desktop\platform-tools\cmd.exe
系统无法在消息文件中为 Application 找到消息号为 0x2350 的消息文本。
版权所有 © 2009 Microsoft Corporation。保留所有权利。
存储空间不足，无法处理此命令。

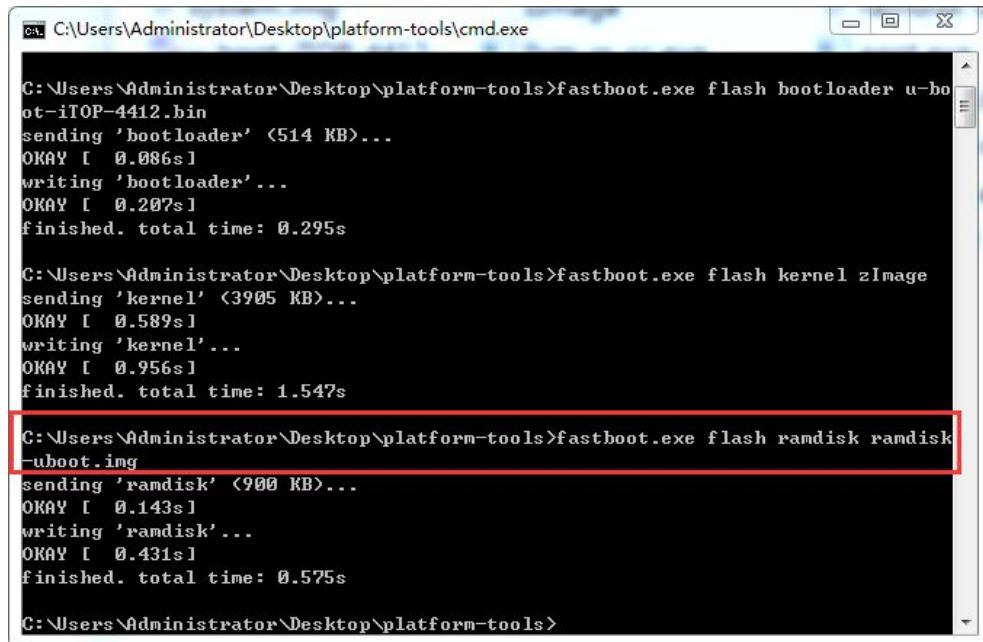
C:\Users\Administrator\Desktop\platform-tools>fastboot.exe flash bootloader u-boot-iTOP-4412.bin
sending 'bootloader' <514 KB>...
OKAY [ 0.086s]
writing 'bootloader' ...
OKAY [ 0.207s]
finished. total time: 0.295s

C:\Users\Administrator\Desktop\platform-tools>fastboot.exe flash kernel zImage
sending 'kernel' <3905 KB>...
OKAY [ 0.589s]
writing 'kernel' ...
OKAY [ 0.956s]
finished. total time: 1.547s

C:\Users\Administrator\Desktop\platform-tools>
```

如下图所示，输入烧写 ramdisk 命令

“fastboot.exe flash ramdisk ramdisk-uboot.img”



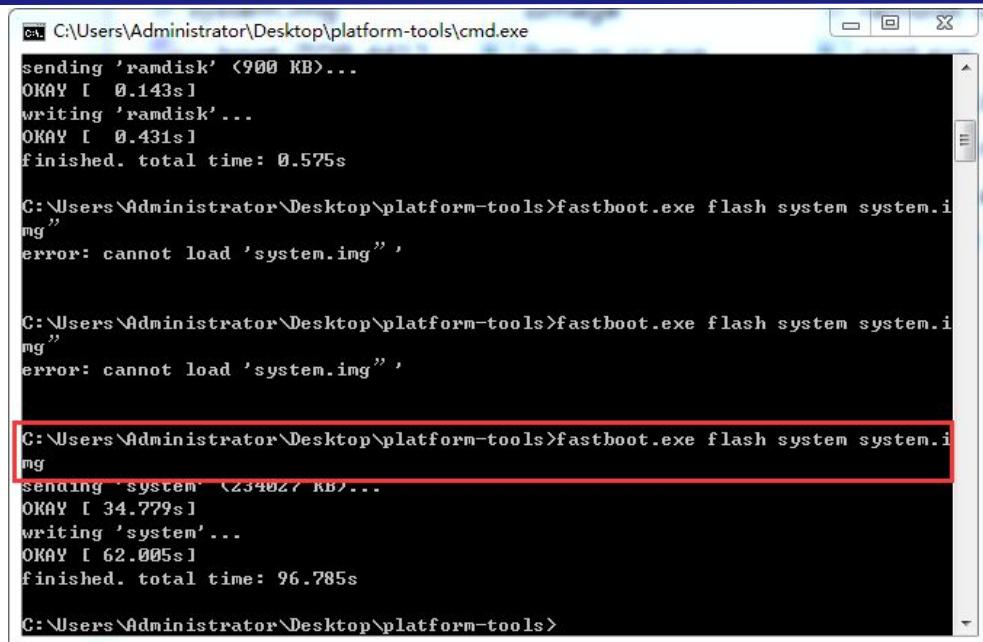
```
C:\Users\Administrator\Desktop\platform-tools\cmd.exe
C:\Users\Administrator\Desktop\platform-tools>fastboot.exe flash bootloader u-boot-iTOP-4412.bin
sending 'bootloader' <514 KB>...
OKAY [ 0.086s]
writing 'bootloader' ...
OKAY [ 0.207s]
finished. total time: 0.295s

C:\Users\Administrator\Desktop\platform-tools>fastboot.exe flash kernel zImage
sending 'kernel' <3905 KB>...
OKAY [ 0.589s]
writing 'kernel' ...
OKAY [ 0.956s]
finished. total time: 1.547s

C:\Users\Administrator\Desktop\platform-tools>fastboot.exe flash ramdisk ramdisk-uboot.img
sending 'ramdisk' <900 KB>...
OKAY [ 0.143s]
writing 'ramdisk' ...
OKAY [ 0.431s]
finished. total time: 0.575s

C:\Users\Administrator\Desktop\platform-tools>
```

如下图所示，输入烧写 system 文件系统命令 “fastboot.exe flash system system.img”



```
C:\Users\Administrator\Desktop\platform-tools\cmd.exe
sending 'ramdisk' <900 KB>...
OKAY [ 0.143s]
writing 'ramdisk'...
OKAY [ 0.431s]
finished. total time: 0.575s

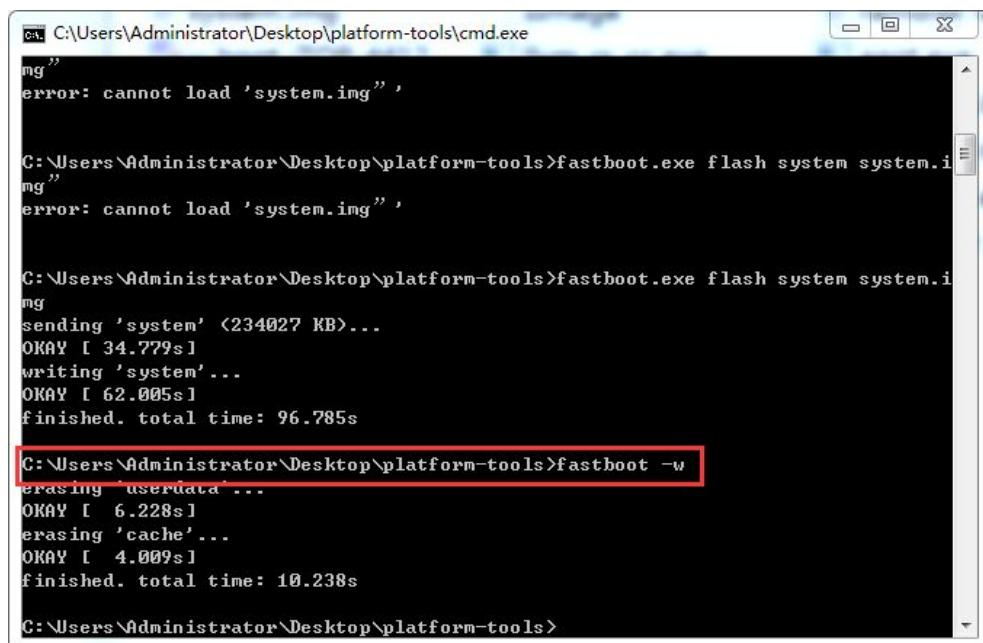
C:\Users\Administrator\Desktop\platform-tools>fastboot.exe flash system system.i
mg"
error: cannot load 'system.img'

C:\Users\Administrator\Desktop\platform-tools>fastboot.exe flash system system.i
mg"
error: cannot load 'system.img'

C:\Users\Administrator\Desktop\platform-tools>fastboot.exe flash system system.i
mg
sending 'system' <234027 KB>...
OKAY [ 34.779s]
writing 'system'...
OKAY [ 62.005s]
finished. total time: 96.785s

C:\Users\Administrator\Desktop\platform-tools>
```

如下图所示，输入擦除命令 “fastboot -w” 。



```
C:\Users\Administrator\Desktop\platform-tools\cmd.exe
mg"
error: cannot load 'system.img'

C:\Users\Administrator\Desktop\platform-tools>fastboot.exe flash system system.i
mg"
error: cannot load 'system.img'

C:\Users\Administrator\Desktop\platform-tools>fastboot.exe flash system system.i
mg
sending 'system' <234027 KB>...
OKAY [ 34.779s]
writing 'system'...
OKAY [ 62.005s]
finished. total time: 96.785s

C:\Users\Administrator\Desktop\platform-tools>fastboot -w
erasing 'userdata'...
OKAY [ 6.228s]
erasing 'cache'...
OKAY [ 4.009s]
finished. total time: 10.238s

C:\Users\Administrator\Desktop\platform-tools>
```

注意：上述 fastboot.exe flash 命令可以分开执行，只烧写单个的镜像。

7 ) 在 Windows 命令行中，输入重启开发板命令 “fastboot reboot”

```
C:\Users\Administrator\Desktop\platform-tools\cmd.exe
mg"
error: cannot load 'system.img'

C:\Users\Administrator\Desktop\platform-tools>fastboot.exe flash system system.i
mg
sending 'system' (234027 KB)...
OKAY [ 34.779s]
writing 'system'...
OKAY [ 62.005s]
finished. total time: 96.785s

C:\Users\Administrator\Desktop\platform-tools>fastboot -w
erasing 'userdata'...
OKAY [ 6.228s]
erasing 'cache'...
OKAY [ 4.009s]
finished. total time: 10.238s

C:\Users\Administrator\Desktop\platform-tools>fastboot reboot
rebooting...

finished. total time: 0.003s

C:\Users\Administrator\Desktop\platform-tools>
```

输入重启命令之后，开发板会重启，超级终端会打印启动信息，第一次Android启动需要解压和安装一些初始化文件，会花费的时间长一点。第一次启动完成之后，再次启动速度就会快一点。

## 4.3 TF 卡烧写方式

使用该方式能够烧写Android系统和QT系统。

TF卡存储容量最少要2G以上。

### 4.3.1 制作可以烧写的TF卡

注意：制作可以烧写的TF卡需要用到Ubuntu系统，用户需要学习Ubuntu的使用，Ubuntu的安装和使用可以参考3.2、3.3以及3.4小节。另外使用这种方式要求核心板的uboot可以正常启动，如果核心板的uboot无法启动，请参考使用手册“4.4 开发板出厂前首次烧写”

本节制作TF卡的方法需要核心板的eMMC能够正常启动打印信息。

使用 TF 卡之前，必须要先分区。制作 TF 卡需要在 PC 机的 Ubuntu 系统下，分 3 个步骤来完成。这里需要注意的是，TF 卡制作完成后，就可长期使用，不用每次重新制作，另外如果烧坏了 eMMC 的 uboot，那么将无法进行分区，就要参考 4.5 小节先给 TF 卡分区。

1 ) 给 TF 卡分区。需要将 TF 卡先插入开发板，然后再启动开发板并进入 Uboot 模式（如何进入可参考前面 2-5 节），然后在超级终端中，依次输入下列烧写命令：

```
— fdisk -c 1
```

注意上面的分区命令的是参数“1”，代表的是 tf 卡，如果是“0”则代表是 eMMC。

如果是 2G 的 TF 卡，请将命令“fdisk -c 1”改为“fdisk -c 1 300 300 300”。

```
— fatformat mmc 1:1
```

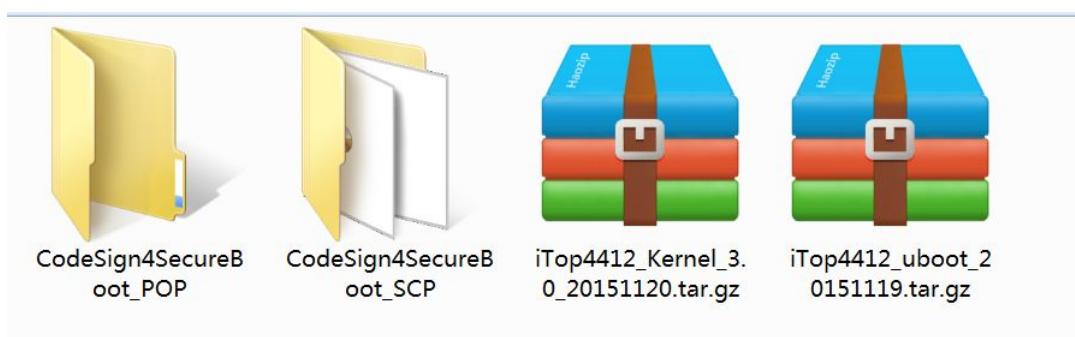
```
— ext3format mmc 1:2
```

```
— ext3format mmc 1:3
```

```
— ext3format mmc 1:4
```

执行完上面的命令之后，就要将 uboot 烧写到 TF 卡。

2 ) 如下图所示，在光盘文件夹“06\_源码\_uboot 和 kernel”中有一个压缩包“iTopy4412\_uboot\_xxx.tar.gz”，压缩包文件名中的“xxx”代表不确定，“xxx”表示日期，日期信息在系统升级后会变更。



3 ) 使用 SSH 工具（参考使用手册 3.3.5 安装和使用 SSH 软件），拷贝压缩包“iTopy4412\_uboot\_xxx.tar.gz”到 PC 机的 Ubuntu 系统中，然后解压压缩包，得到文件夹“iTopy4412\_uboot”，进入“iTopy4412\_uboot”文件夹，如下图所示。

```
root@ubuntu:/home/topeet/android4.0/iTop4412_uboot
root@ubuntu:/home/topeet/android4.0# ls
iTop4412_uboot iTop4412_uboot_20151119.tar.gz
root@ubuntu:/home/topeet/android4.0# cd iTop4412_uboot
root@ubuntu:/home/topeet/android4.0/iTop4412_uboot# ls
all00_padding.bin E4412_N.bl1.bin lib_nios onenand_ipl
api examples lib_nios2 paddingaa
board fs lib_ppc post
build_uboot.sh include lib_sh README
common lib_arm lib_sparc readme.txt
config.mk lib_avr32 MAINTAINERS rules.mk
COPYING lib_blackfin MAKEALL sdfuse
cpu lib_fdt Makefile sdifuse.d
CREDITS lib_generic mkb12 tc4_cmm.cmm
disk lib_i386 mkconfig tools
doc lib_m68k mkuboot uboot_readme.txt
drivers lib_microblaze nand_spl
E4212 lib_mips net
root@ubuntu:/home/topeet/android4.0/iTop4412_uboot#
```

4 ) 拷贝光盘中文件夹中对应核心板的镜像 “u-boot-iTOP-4412.bin” 到上一步解压出来的文件夹 “iToph4412\_uboot” 中 , 如下图所示。

```
root@ubuntu:/home/topeet/android4.0/iTop4412_uboot
common          lib_arm        lib_sparc      readme.txt
config.mk       lib_avr32      MAINTAINERS   rules.mk
COPYING         lib_blackfin  MAKEALL        sdfuse
cpu             libfdt        Makefile      sdFUSE
CREDITS         lib_generic   mklbl2        tc4_cmm.cmm
disk            lib_i386       mkconfig      tools
doc              lib_m68k      mkuboot      uboot_readme.txt
drivers          lib_microblaze nand_spl
E4212           lib_mips       net
root@ubuntu:/home/topeet/android4.0/iTop4412_uboot# ls
all00_padding.bin E4412_N.b11.bin lib_nios      onenand_ipl
api               examples      lib_nios2     paddingaa
board             fs           lib_ppc       post
build_uboot.sh   include      lib_sh        README
common            lib_arm      lib_sparc    readme.txt
config.mk         lib_avr32    MAINTAINERS  rules.mk
COPYING           lib_blackfin MAKEALL      sdfuse
cpu               libfdt      Makefile      tc4_cmm.cmm
CREDITS          lib_generic  mklbl2        tools
disk              lib_i386     mkconfig      u-boot-iTOP-4412.bin
doc               lib_m68k     mkuboot      uboot_readme.txt
drivers          lib_microblaze nand_spl
E4212           lib_mips      net
root@ubuntu:/home/topeet/android4.0/iTop4412_uboot#
```

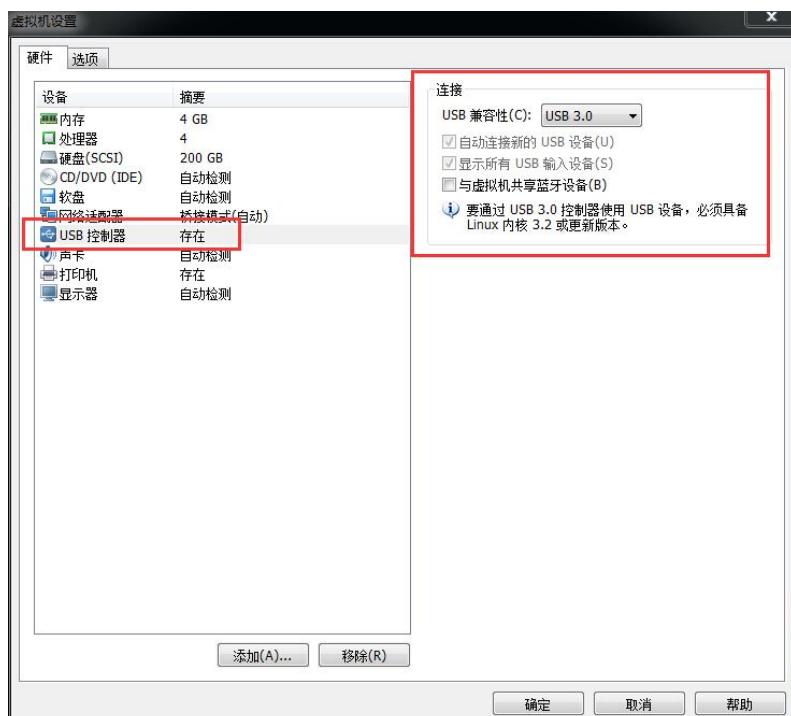
5 ) 在 Ubuntu 命令行中输入命令 “df -l” , 查看一下系统有哪些盘符。

```
root@ubuntu:/home/topeet/android4.0/iTop4412_uboot# df -l
Filesystem      1K-blocks    Used Available Use% Mounted on
/dev/sda1        60894268 10242120   47558900  18% /
udev             2013744       4   2013740   1% /dev
tmpfs            809208     808     808400   1% /run
none              5120       0      5120   0% /run/lock
none            2023012     200   2022812   1% /run/shm
root@ubuntu:/home/topeet/android4.0/iTop4412_uboot#
```

6 ) 接着使用读卡器将 TF 卡连接到 PC 机的 Ubuntu 系统下 , 如下图所示 , Ubuntu 系统识别 TF 卡后 , 可能提示下面的信息。



7 ) 在虚拟机 VMware Workstation 选项 “虚拟机 M” , 进入 “虚拟机设置” , 如下图所示 , 根据 USB 接口选择一下版本 , 如果是 USB3.0 则使用 USB3.0。



8 ) TF 卡连接之后到 Ubuntu 之后 , 再次使用 Linux 命令 “df -l” 查看盘符。将第二次查看的盘符和第一次查看的盘符对比一下 , 就会发现 Ubuntu 系统中多出了盘符 , 这个盘符就是 TF 卡的盘符 , 盘符名称在接下来的 Linux 命令中会用到 , 如下图所示。

```
root@ubuntu:/home/topeet/android4.0/iTop4412_uboot# df -l
Filesystem      1K-blocks   Used   Available  Use% Mounted on
/dev/sda1        60894268 10242120  47558900  18% /
udev             2013744     4    2013740   1% /dev
tmpfs            809208    808    808400   1% /run
none              5120      0    5120  0% /run/lock
none             2023012    200   2022812   1% /run/shm
root@ubuntu:/home/topeet/android4.0/iTop4412_uboot# df -l
Filesystem      1K-blocks   Used   Available  Use% Mounted on
/dev/sda1        60894268 10242424  47558596  18% /
udev             2013744     4    2013740   1% /dev
tmpfs            809208    832    808376   1% /run
none              5120      0    5120  0% /run/lock
none             2023012    200   2022812   1% /run/shm
/dev/sdb2        1034996   34112   948308   4% /media/37b2a37e-d68c-4259-7467-
a77513424650
/dev/sdb1        6053656     4   6053652   1% /media/0000-3333
/dev/sdb3        302612    16584   270408   6% /media/befaf008-2317-8fe3-fbaf-
677c60cccc98
/dev/sdb4        302612    16584   270408   6% /media/9ec90b96-3da4-a970-a2c0-
0e8d409bad67
root@ubuntu:/home/topeet/android4.0/iTop4412_uboot#
```

9) 在执行下面这条命令的时候，要特别特别注意！一定要分清楚，哪个盘符是 TF 卡的盘符，如果不清楚，请务必先拔掉 TF 卡，看清楚哪些盘符是属于 Ubuntu 系统的硬盘盘符，看清楚哪些盘符是 Ubuntu 系统的硬盘盘符后，再插入 TF 卡，分辨出哪个盘符是新增加的盘符，新增加的盘符才是 TF 卡的盘符。

10 ) 进入文件夹 “iTop4412\_uboot” 中。在 Ubuntu 命令行中，执行 Linux 命令：

“./mkuboot /dev/sdx” ，mkuboot 是 uboot 源码文件夹中的一个脚本，下图中脚本命令的对象是上图中识别的 “sdb” 。

(注意， sdx 用前面查看盘符，多出来的盘符名代替，不要带数字，比如 df -l 看到的 tf 卡是 /dev/sdb0，这个 0 不要带，直接写 sdb)

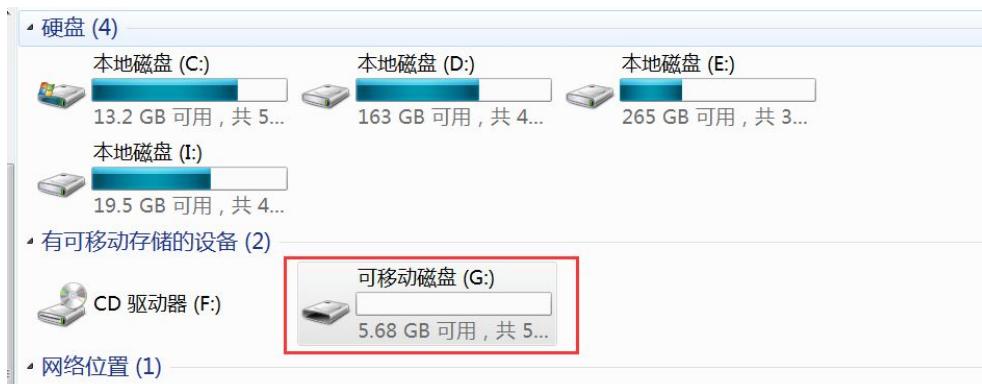
```

root@ubuntu:/home/topeet/android4.0/iTop4412_uboot# ls
all00_padding.bin  E4412_N.bl1.bin  lib_nios  onenand_ipl
api               examples      lib_nios2 paddingaa
board              fs           lib_ppc   post
build_uboot.sh    include      lib_sh    README
common             lib_arm     lib_sparc  readme.txt
config.mk          lib_avr32   MAINTAINERS rules.mk
COPYING            lib_blackfin  MAKEALL   sdfuse
cpu                lib_fdt     Makefile  sdifuse.c
CREDITS            lib_generic  mkb12    tc4_cmm.cmm
disk               lib_i386    mkconfig  tools
doc                lib_m68k    mkuboot  u-boot-iTOP-4412.bin
drivers             lib_microblaze nand_spl  uboot_readme.txt
E4212              lib_mips    net
root@ubuntu:/home/topeet/android4.0/iTop4412_uboot# ./mkuboot /dev/sdb
Fuse iTOP-4412 trustzone uboot file into SD card
/dev/sdb reader is identified.
u-boot-iTOP-4412.bin fusing...
1029+1 records in
1029+1 records out
527104 bytes (527 kB) copied, 5.47041 s, 96.4 kB/s
u-boot-iTOP-4412.bin image has been fused successfully.
Eject SD card
root@ubuntu:/home/topeet/android4.0/iTop4412_uboot#

```

11 ) 检查可以烧写的 TF 卡是否制作成功。

检查的方法很简单，使用读卡器连接 TF 卡到 Win7 系统上，如果在 Win7 系统上发现 TF 卡的存储空间减少了 2G 以上，那么这个 TF 卡就制作成功了。如下图所示，8G 的 TF 卡只剩 5.68G。



### 4.3.2 使用 TF 卡烧写

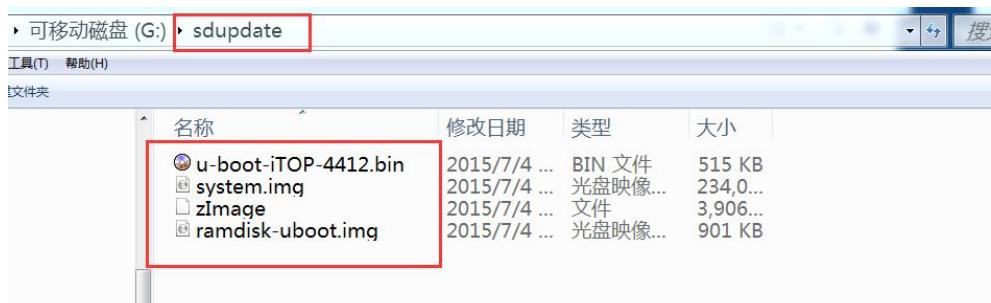
在 Win7 系统和 Ubuntu 系统下，都可以使用 TF 卡烧写。这里以 Win7 为例，Ubuntu 的也是一样，将镜像文件拷贝到 tf 卡中的“sdupdate”文件夹中。

Win7 下 TF 卡烧写步骤如下：

1 ) 将制作完成的 TF 卡接入 PC 机的 Win7 或者 Ubuntu 系统中，在 TF 卡上建立文件夹“sdupdate” 。注意，文件夹名字一定要使用“sdupdate” 。如下图所示。



2 ) 拷贝相应的镜像文件到 TF 卡的文件夹 “sdupdate” 中，如下图所示。



3 ) 将 TF 卡先插入开发板中，进入超级终端的 UBOOT 模式，如下图所示。

```
....  
eMMC OPEN Success..!!  
!!!Notice!!!  
!You must close eMMC boot Partition after all image writing!  
!eMMC boot partition has continuity at image writing time.!  
!So, Do not close boot partition, Before, all images is written.!  
  
MMC read: dev # 0, block # 48, count 16 ...16 blocks read: OK  
eMMC CLOSE Success..!!  
  
Checking Boot Mode ... EMMC4.41  
SYSTEM ENTER NORMAL BOOT MODE  
Hit any key to stop autoboot: 0  
iTOP-4412 #
```

4 ) 输入烧写命令 “sdfuse flashall” 。

这是一个全部烧写的命令，就是将 “sdupdate” 中全部的镜像烧写到开发板中，如下图所示。

```
iTOP-4412 # sdfuse flashall
SD sclk_mmc is 400K HZ
SD sclk_mmc is 50000K HZ
SD sclk_mmc is 50000K HZ
[Fusing Image from SD Card.]
.fdisk is completed

partition #    size(MB)      block start #      block count      partition_Id
    1        1340          4862616          2744544          0x0C
    2        1026          37290            2103156          0x83
    3        1026          2140446            2103156          0x83
    4         302          4243602            619014          0x83
>>>part_type : 2
Partition1: Start Address(0x4a3298), Size(0x29e0e0)
.....size checking ...
Under 8G
write FAT info: 32
Fat size : 0xa78
..Erase FAT region.
.....
```

烧写命令 “sdfuse flashall” 可以用下面替代的烧写命令：

- sdfuse flash bootloader u-boot-iTOP-4412.bin
- sdfuse flash kernel zImage
- sdfuse flash ramdisk ramdisk-uboot.img
- sdfuse flash system system.img

替代的烧写命令允许单条执行，在执行单条烧写命令的时候，只烧写相应的单个镜像文件。例如，执行烧写命令 “sdfuse flash kernel zImage” ，就只会更新 Linux 内核 “zImage” 镜像，而不会影响其他的镜像。

5 ) 等待烧写完成，最后在超级终端中，输入以下命令：

- reset ( 在超级终端中执行该命令会重启开发板 )

## 4.4 开发板出厂前首次烧写

在前面几个小节中，介绍制作 TF 卡的时候，都需要依赖核心板的 eMMC 能够启动 uboot，当核心板的 uboot 被烧写了错误的文件，导致损坏，那么就需要使用本节的方法来解决了。

全新的 TF 卡并不能直接用于烧写镜像，全新的 TF 卡需要经过分区、烧写 uboot 等步骤后才能用于烧写。

下面具体介绍如何制作可以烧写的 TF 卡，首先确定 TF 卡是 FAT32 格式的。

- 1 ) 在 Ubuntu 命令行中输入 Linux 命令 “df -l” ，查看一下 Ubuntu 系统有哪些盘符。如下图所示。

```
root@ubuntu:~# df -l
Filesystem      1K-blocks    Used Available Use% Mounted on
/dev/sda1        60894268 10242512 47558508 18% /
udev             2013744      4   2013740  1% /dev
tmpfs            809208     832   808376  1% /run
none              5120       0    5120  0% /run/lock
none            2023012     200   2022812  1% /run/shm
root@ubuntu:~#
```

- 2 ) 接着使用读卡器将 TF 卡连接到 PC 机的 Ubuntu 系统下，Ubuntu 系统识别 TF 卡后，再次使用 Linux 命令 “df -l” 查看盘符。将第二次查看的盘符和第一次查看的盘符对比一下，就会发现 Ubuntu 系统中多出一个盘符，这个盘符就是 TF 卡的盘符，TF 卡盘符名称在接下来的 Linux 命令中会用到，如下图所示。

```
root@ubuntu:~# df -l
Filesystem      1K-blocks    Used Available Use% Mounted on
/dev/sda1        60894268 10242512 47558508 18% /
udev             2013744      4   2013740  1% /dev
tmpfs            809208     832   808376  1% /run
none              5120       0    5120  0% /run/lock
none            2023012     200   2022812  1% /run/shm
root@ubuntu:~# df -l
Filesystem      1K-blocks    Used Available Use% Mounted on
/dev/sda1        60894268 10242560 47558460 18% /
udev             2013744      4   2013740  1% /dev
tmpfs            809208     820   808388  1% /run
none              5120       0    5120  0% /run/lock
none            2023012     200   2022812  1% /run/shm
/dev/sdb1        7767164      4   7767160  1% /media/disk
root@ubuntu:~#
```

- 3 ) 拷贝用户光盘 “Android 源码” 文件夹中的文件 “iT0p-4412\_uboot\_XXX” 到 Ubuntu 系统中解压，得到文件夹 “iT0p4412\_uboot” 。

拷贝对应核心板的文件 “u-boot-iTOP-4412.bin” 到前面解压出来的文件夹 “iT0p4412\_uboot” 中。进入前面解压出来的文件夹 “iT0p4412\_uboot” 中，如下图所示

```
root@ubuntu:~# cd /home/topeet/android4.0/iTop4412_uboot
root@ubuntu:/home/topeet/android4.0/iTop4412_uboot# ls
all00_padding.bin  E4412_N.bl1.bin  lib_nios  onenand_ipl
api               examples    lib_nios2  paddingaa
board              fs         lib_ppc   post
build_uboot.sh    include    lib_sh    README
common             lib_arm    lib_sparc  readme.txt
config.mk          lib_avr32  MAINTAINERS rules.mk
COPYING            lib_blackfin  MAKEALL  sdfuse
cpu                lib_fdt    Makefile  tc4_cmm.cmm
CREDITS            lib_generic  mkl2      tools
disk               lib_i386   mkconfig  u-boot-iTOP-4412.bin
doc                lib_m68k   mkuboot  uboot_readme.txt
drivers             lib_microblaze  nand_spl
E4212              lib_mips   net
root@ubuntu:/home/topeet/android4.0/iTop4412_uboot#
```

4 ) 烧写文件 “u-boot-iTOP-4412.bin” 到 TF 卡 , 具体操作如下 :

在执行下面这条命令的时候 , 要特别特别注意 ! 一定要分清楚 , 哪个盘符是 TF 卡的盘符 , 如果不清楚 , 请务必先拔掉 TF 卡 , 看清楚哪些盘符是属于 Ubuntu 系统的硬盘盘符 , 看清楚哪些盘符是 Ubuntu 系统的硬盘盘符后 , 再插入 TF 卡 , 分辨出哪个盘符是新增加的盘符 , 新增加的盘符才是 TF 卡的盘符。

在 Ubuntu 命令行中 , 执行下面命令

`./mkuboot /dev/sdx` ( “ sdx ” 就是前面查到的 TF 卡盘符名 , 不要带数字 , 比如 `df -l` 看到的 tf 卡是 `/dev/sdb0` , 这个 0 不要带 , 直接写 `sdb` )

需要注意的是 , 上面的命令需要在文件夹 " iTop4412\_uboot\_xxx " 中执行。这个文件夹是解压 uboot 源码之后得到的文件夹。

如果这里识别为 “`/dev/sdb1`” , 那么如下图所示 , 使用命令 “`./mkuboot /dev/sdb`” , 不要带后面盘符的编号。

```
root@ubuntu:/home/topeet/android4.0/iTop4412_uboot# ls
all00_padding.bin  E4412_N.bl1.bin  lib_nios  onenand_ipl
api               examples    lib_nios2  paddingaa
board              fs         lib_ppc   post
build_uboot.sh    include    lib_sh    README
common             lib_arm    lib_sparc  readme.txt
config.mk          lib_avr32  MAINTAINERS rules.mk
COPYING            lib_blackfin  MAKEALL  sdfuse
cpu                lib_fdt    Makefile  tc4_cmm.cmm
CREDITS            lib_generic  mkl2      tools
disk               lib_i386   mkconfig  u-boot-iTOP-4412.bin
doc                lib_m68k   mkuboot  uboot_readme.txt
drivers             lib_microblaze  nand_spl
E4212              lib_mips   net
root@ubuntu:/home/topeet/android4.0/iTop4412_uboot# ./mkuboot /dev/sdb
Fuse iTOP-4412 trustzone uboot file into SD card
/dev/sdb reader is identified.
u-boot-iTOP-4412.bin fusing...
1029+1 records in
1029+1 records out
527104 bytes (527 kB) copied, 5.39526 s, 97.7 kB/s
u-boot-iTOP-4412.bin image has been fused successfully.
Eject SD card
root@ubuntu:/home/topeet/android4.0/iTop4412_uboot#
```

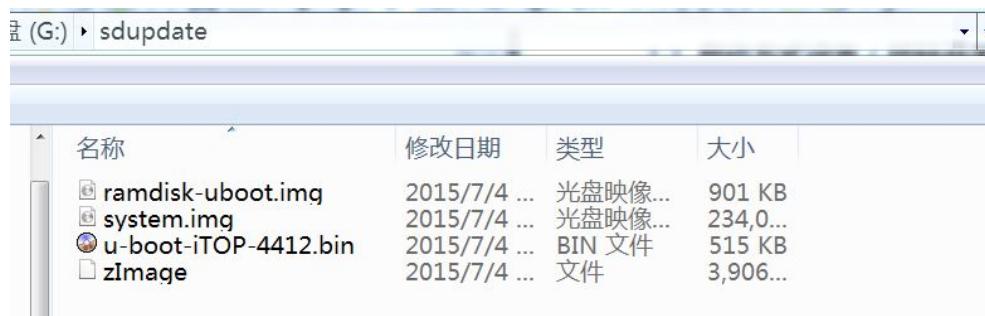
4 ) 将开发板的拨码开关置于 **TF 卡启动模式** , 先插入经过上一步处理的 TF 卡 , 启动开发板 , 进入 uboot 模式 , 对 TF 卡进行分区 , 具体操作如下 :

在超级终端中 , 执行下面的命令 :

- fdisk -c 1 300 300 300
- fatformat mmc 1:1
- ext3format mmc 1:2
- ext3format mmc 1:3
- ext3format mmc 1:4

6 ) 到上一步 , TF 卡就制作好了。下面继续介绍如何使用制作好的 TF 卡。

7 ) 将 TF 卡插入 Window 双系统 , 新建 “sdupdate” , 将 uboot 等四个文件拷贝到 “sdupdate” 文件夹中 , 也可以选择只拷贝 uboot 文件 , 如下图所示。



8 ) 将 TF 卡插入开发板 , 设置开发板为 tf 卡启动模式 , 启动开发板。对 eMMC 进行格式化分区命令以及擦除命令 , 最后使用命令 “sdfuse flashall” , 就可以将 uboot 、内核以及文件系统更新到开发板的 eMMC 中 , 这里也可以选择只更新 uboot 。

9 ) 烧写镜像完成之后 , 将开发板设置为 **eMMC 启动模式** , 开发板就可以正常启动 uboot , 然后可以使用 fastboot 烧写或者 tf 卡烧写内核或者文件系统的镜像 , 烧写方法参考 4.2 或者 4.3 小节。

## 五 Android 开发环境搭建以及编译

本章节中将为您详细介绍 uboot、Linux3.0.15 和 linux-3.5 内核、Android4.0.3 和 Android4.2.2 编译环境的搭建以及编译。

**注意:**从一开始，我们就是基于 Ubuntu12.04.2 平台做开发，所有的配置和编译脚本也是基于此平台，我们没有在其它平台上测试过。如果你对 Linux 和 Android 开发很熟悉，相信你会根据错误提示逐步找到原因并解决，错误提示一般是选用的平台缺少了某些库文件或者工具等原因造成的；否则，**我们建议初学者使用和我们一致的平台**，即 Ubuntu12.04.2，你可以在我们的网盘下载 Ubuntu12.04.2 的镜像，安装的时候请务必参考我们手册提供的步骤，这是我们经过严格测试的，以免遗漏一些开发时所需要的组件。

Linux 的发行版本众多，我们无法为此一一编写文档，敬请原谅。

Uboot、Kernel 以及 Android 的编译环境看似复杂，用户只要抓住以下几个要点就可以了：

第一：Uboot、Kernel 编译器。编译器在光盘中都有提供，在需要使用的步骤中，会说明编译器在光盘中的位置。

第二：设置环境变量。Uboot、Kernel 编译器的环境变量设置后，编译的时候，系统才能找到编译器。

第三：Android 文件系统的编译器。编译器需要使用 Ubuntu 系统自带的 gcc 编译器，但是版本不对，所以需要降低版本。迅为将这个过程编写成了几个简单的命令，用户只需要挨个执行命令就可以了。

第四：库文件。搭建过程中会给通过执行简单的脚本命令来安装库文件，复杂的步骤变的简单有效。

另外，如果用户想了解编译环境具体是怎么搭建起来的，可以利用我们提供的脚本文件来学习。

## 5.1 Android4.0.3 编译环境的两种搭建方式

迅为电子给用户提供两种搭建编译环境的方式，一种方法是用户安装虚拟机，然后安装基础的 Ubuntu12.04.2 系统，利用我们提供工具和详细的使用步骤，搭建编译环境；另外一种方法是用户安装虚拟机，然后直接加载我们“搭建好的 Ubuntu 镜像”，用户只需要修改一下编译器的环境变量，就可以直接用来编译源码。

### 5.1.1 使用已经搭建好的镜像

使用已经搭建编译环境的镜像，用户只需要安装虚拟机“Vmware\_Workstaion\_wm” ，然后用虚拟机加载搭建好环境的 Ubuntu 镜像。

“搭建好的镜像” 提供网址供大家下载，加载方法参考“3.2 安装虚拟机以及 Ubuntu12.04.2 等软件”。

### 5.1.2 自己搭建环境

另一种方法是自己安装虚拟机，安装 Ubuntu12.04.2 系统，搭建环境。大家可以参考 5.2 小节自己搭建环境。

网盘里面提供了所有需要用到的软件。

## 5.2 搭建环境

本节的主要内容是，详细讲解如何搭建编译环境。

这里需要注意的是，搭建过程中用到的各类软件，都需要和手册提到的版本保持一致，如果使用的是“搭建好的镜像”，则可以跳过这一节，但是编译的时候要针对性的设置一下环境变量。

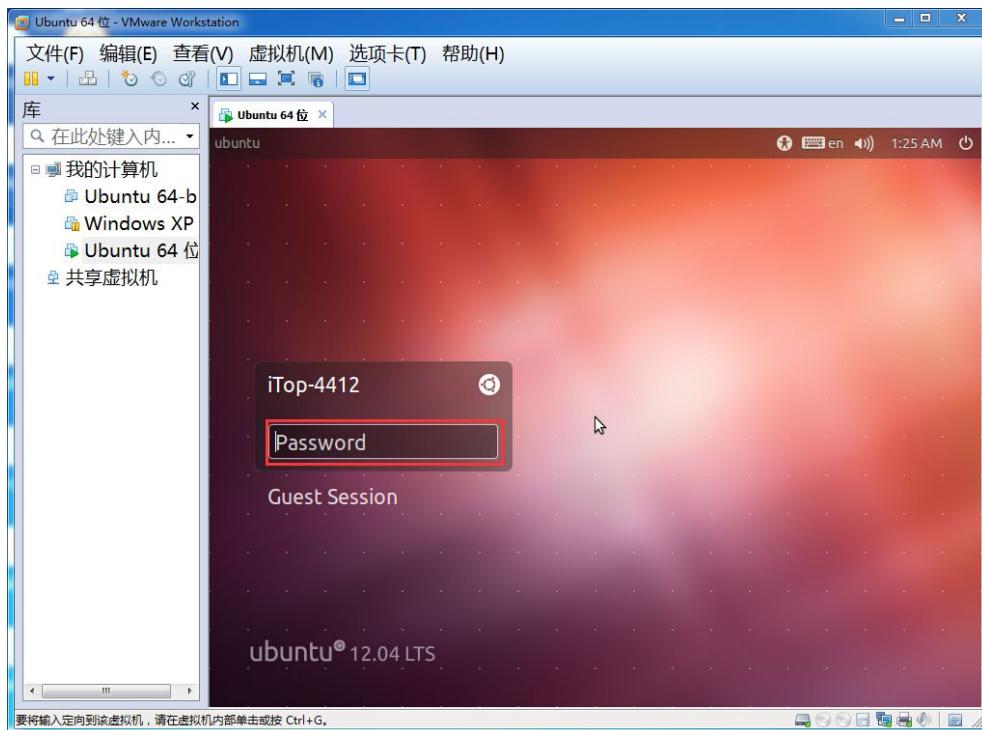
### 5.2.1 安装基本软件

首先安装虚拟机“Vmware\_Workstaion\_wm”，然后使用虚拟机安装“Ubuntu12.04.2初始系统”。

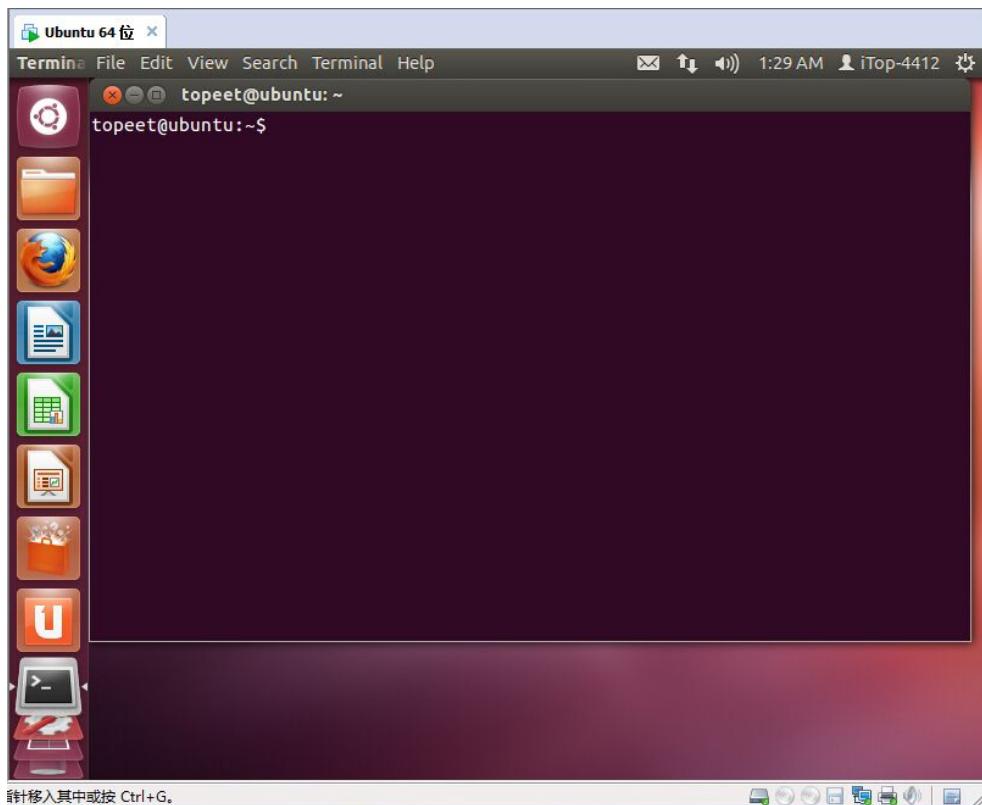
虚拟机和 Ubuntu 初始系统迅为电子在网盘提供下载。

Ubuntu 的安装方法可以参考 3.2 小节来安装 Ubuntu12.04.2 原始系统。

如下图所示，Ubuntu 初始系统安装完成。



输入密码“topeet”，登陆 Ubuntu，键盘上按“Ctrl+Alt+t”，弹出 Ubuntu 的控制台。如下图所示。



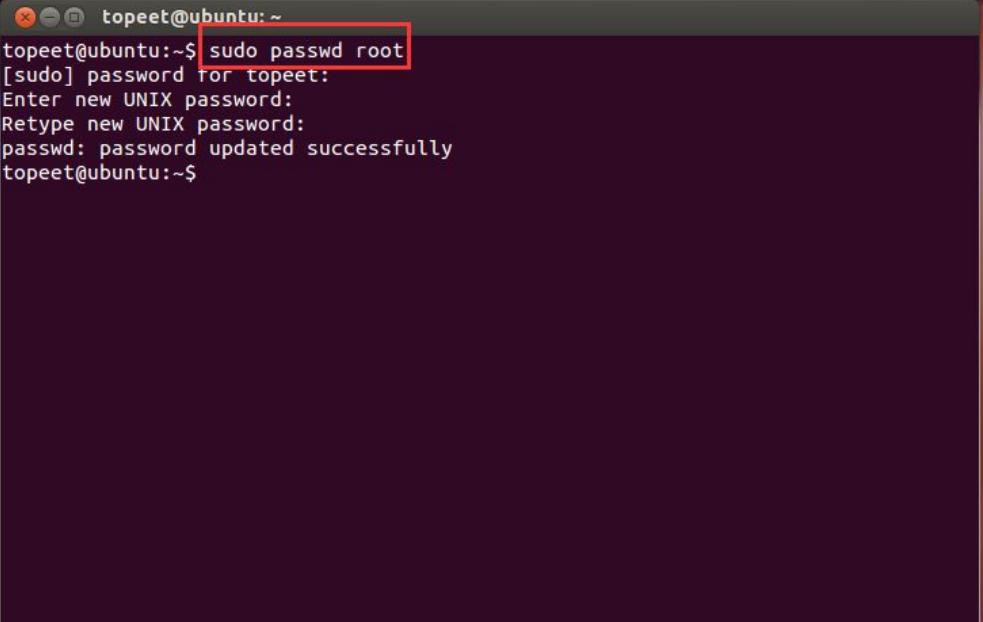
安装完成后进入 Ubuntu 的终端，激活 root 用户，具体操作如下：

在 Ubuntu 命令行中，执行下面命令：

`sudo passwd root`

接着在 Ubuntu 的终端输入安装时的密码和新密码，Ubuntu 系统中密码默认是隐藏的。

如下图所示。



```
topeet@ubuntu:~$ sudo passwd root
[sudo] password for topeet:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
topeet@ubuntu:~$
```

接着登录 root 用户，具体操作如下：

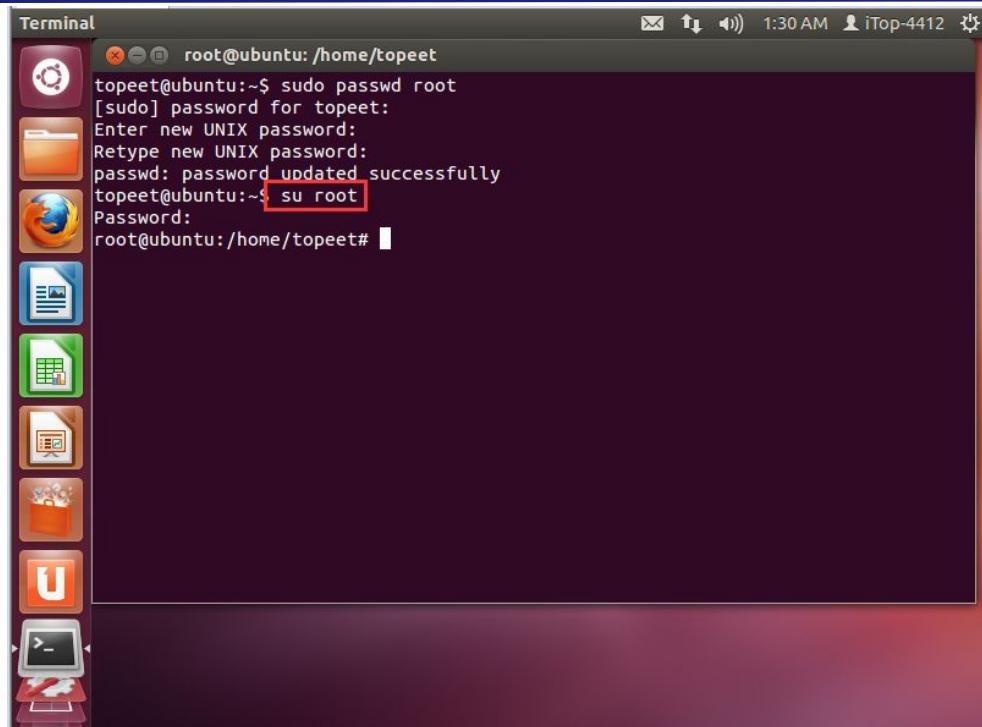
在 Ubuntu 命令行中，执行下面命令：

su root

接着输入密码，就可以登陆 root 用户。

后面所有的操作都需要在 root 用户下面进行操作。

如下图所示。



参考“3.2.4 虚拟机 VMware-workstation8.0.3 联网以及基本设置”联网。

参考“3.3.4.2 修改数据源地址”将 Ubuntu 数据源地址修改为国内地址。

参考“3.3.4.3 “apt-get update” 命令” 更新数据源。

然后在 Ubuntu 安装软件 vim 和 ssh，具体操作如下：

在 Ubuntu 命令行中，执行下面命令：

apt-get install vim

apt-get install ssh

上面安装的 ssh 软件，可以很方便在主机和虚拟机上传递文件，也可以通过远程终端控制 Ubuntu 系统，这里推荐给大家使用。ssh 软件的使用参考 3.3 小节。

## 5.2.2 安装编译组件

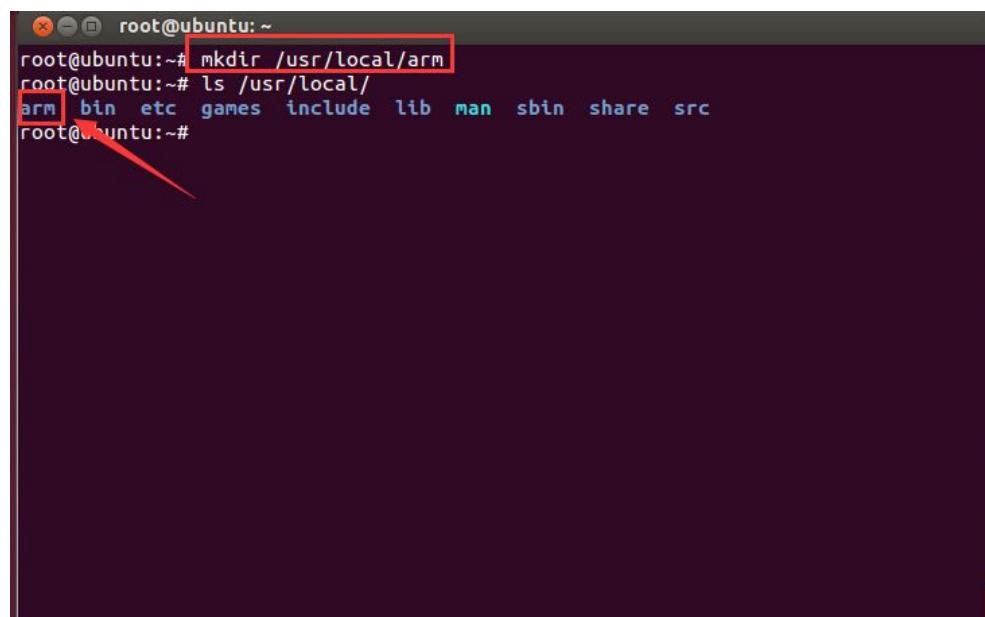
### 交叉编译工具

编译的时候需要用到交叉编译工具，我们提供的交叉编译工具是用户光盘“02\_编译器以及烧写工具”→“arm 交叉编译器”文件夹中的压缩包“arm-2009q3.tar.bz2”。

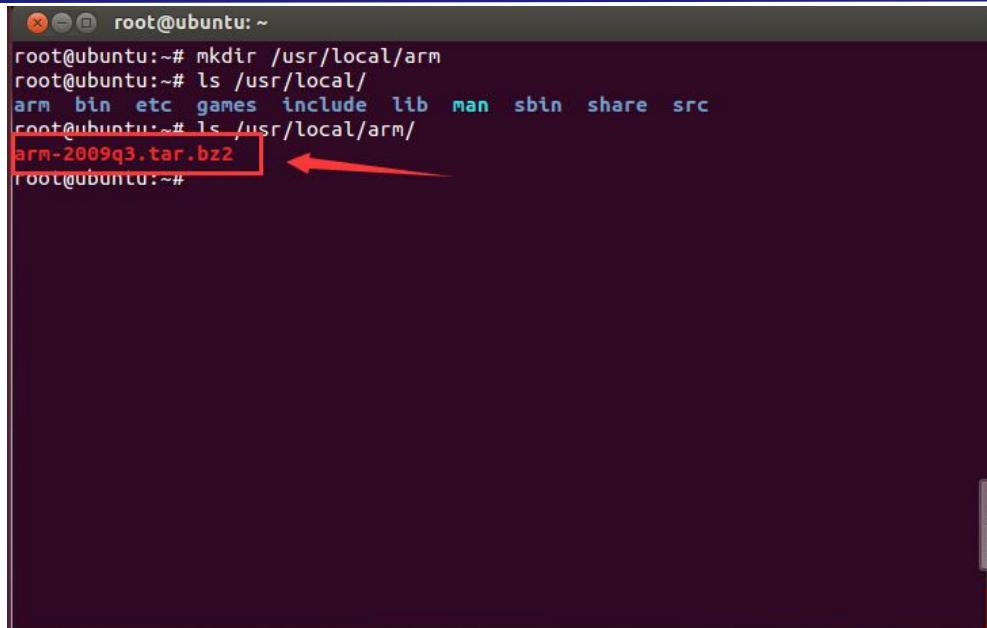


使用 SSH 工具将交叉编译工具拷贝到 Ubuntu12.04.2 系统的文件夹 “usr” --

> “local” --> “arm” 中，local 下默认没有 arm 文件夹，可以新建一个。如下图所示。



拷贝编译器之后如下图所示。

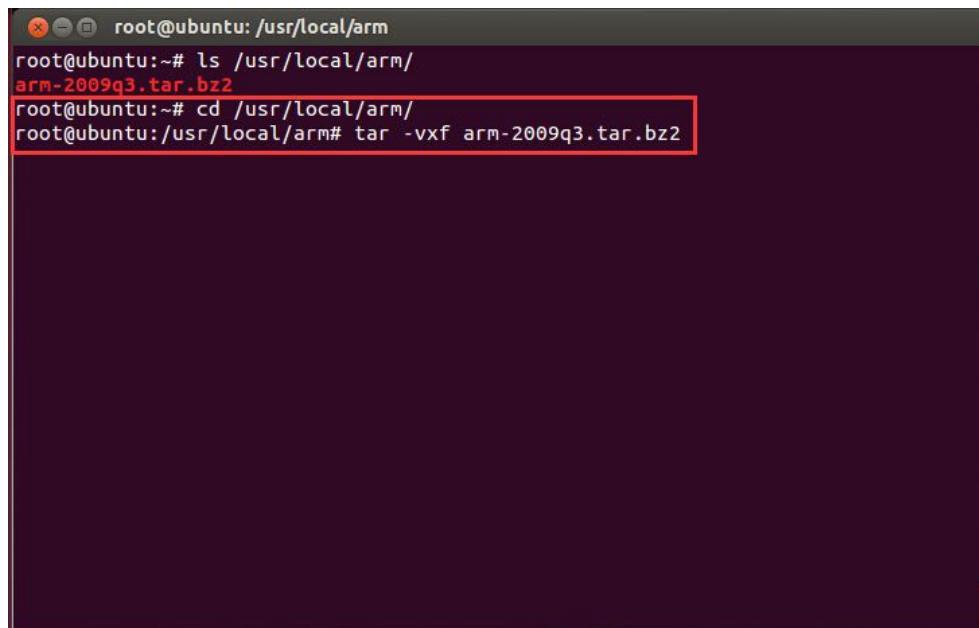


```
root@ubuntu: ~
root@ubuntu:~# mkdir /usr/local/arm
root@ubuntu:~# ls /usr/local/
arm bin etc games include lib man sbin share src
root@ubuntu:~# ls /usr/local/arm/
arm-2009q3.tar.bz2
```

A red arrow points from the bottom right towards the file "arm-2009q3.tar.bz2" in the terminal output.

然后在 Ubuntu 系统中将压缩包解压到当前目录下。

使用命令 “cd /usr/local/arm/” 进入/usr/local/arm 文件夹，然后使用解压命令 “tar -vxf arm-2009q3.tar.bz2” 解压压缩包,如下图所示。

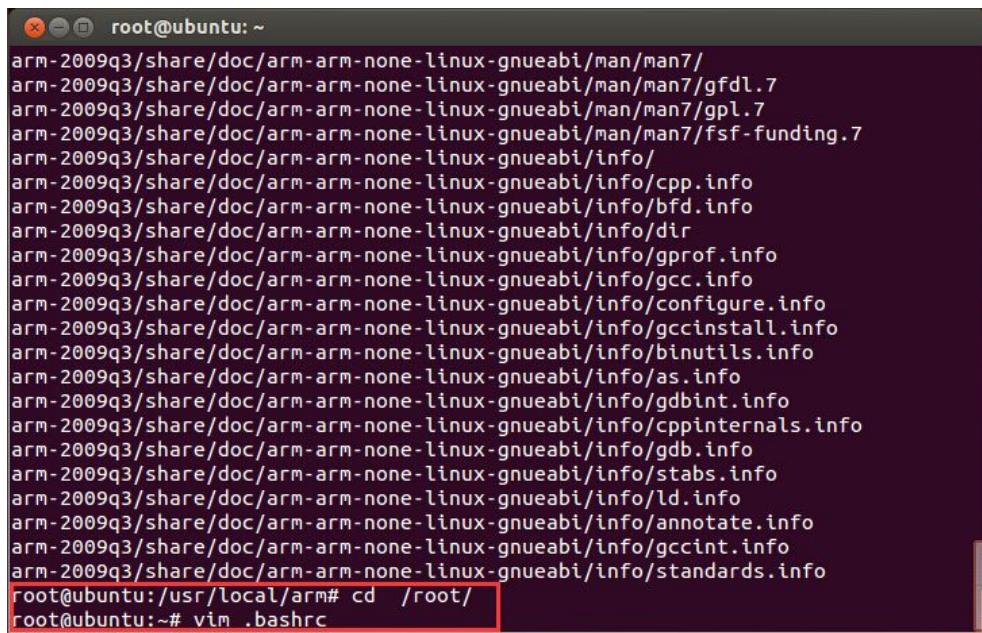


```
root@ubuntu: /usr/local/arm
root@ubuntu:~# ls /usr/local/arm/
arm-2009q3.tar.bz2
root@ubuntu:~# cd /usr/local/arm/
root@ubuntu:/usr/local/arm# tar -vxf arm-2009q3.tar.bz2
```

### 修改交叉编译工具的路径（修改环境变量）

修改交叉编译工具路径，需要修改环境变量，具体操作如下：

在 Ubuntu 命令行中，执行命令 “cd /root” 和 “vim .bashrc” ，打开环境变量文件 “.bashrc” ，如下图所示。

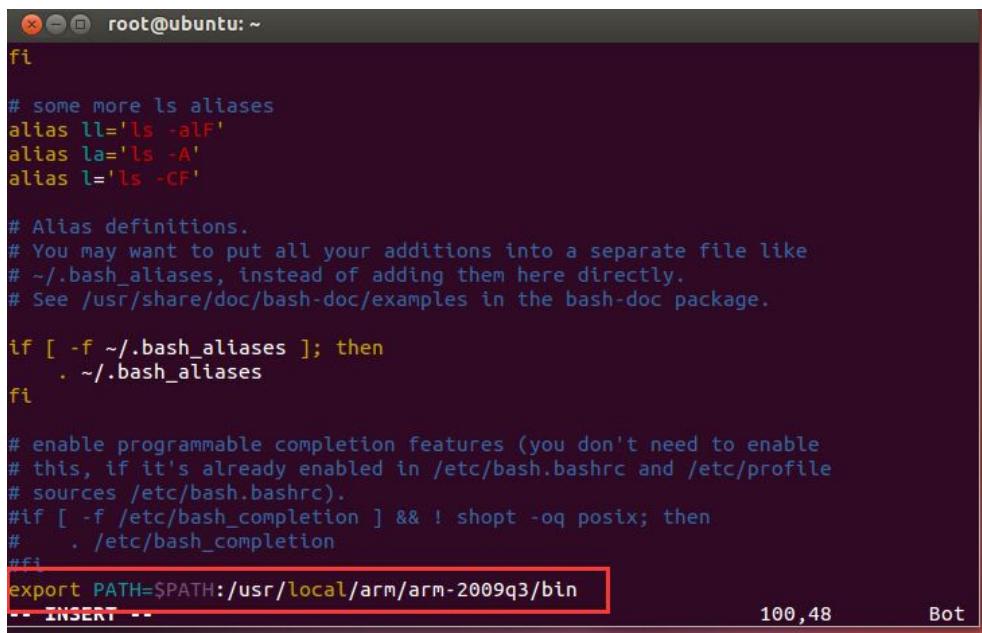


```
root@ubuntu:~ 
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/man/man7/
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/man/man7/gfdl.7
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/man/man7/gpl.7
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/man/man7/fst-funding.7
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/cpp.info
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/bfd.info
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/dir
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/gprof.info
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/gcc.info
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/configure.info
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/gccinstall.info
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/binutils.info
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/as.info
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/gdbint.info
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/cppinternals.info
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/gdb.info
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/stabs.info
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/ld.info
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/annotate.info
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/gccint.info
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/standards.info
root@ubuntu:/usr/local/arm# cd /root/
root@ubuntu:~# vim .bashrc
```

然后在 “.bashrc” 文件中的最后一行添加如下信息：

“export PATH=\$PATH:/usr/local/arm/arm-2009q3/bin”

如下图所示。



```
root@ubuntu:~ 
fi

# some more ls aliases
alias ll='ls -alF'
alias la='ls -A'
alias l='ls -CF'

# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
#if [ -f /etc/bash_completion ] && ! shopt -oq posix; then
#    . /etc/bash_completion
#fi
export PATH=$PATH:/usr/local/arm/arm-2009q3/bin
-- INSERT --
```

100,48 Bot

修改完成后保存退出。

执行下列命令，更新环境变量 “source .bashrc” 的命令，如下图所示。

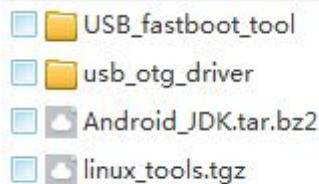
```
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/gdb.info  
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/stabs.info  
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/ld.info  
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/annotate.info  
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/gccint.info  
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/standards.info  
root@ubuntu:/usr/local/arm# cd /root/  
root@ubuntu:~# vim .bashrc  
root@ubuntu:~# source .bashrc  
root@ubuntu:~#
```

最后，在 Ubuntu 命令行中输入命令 “arm” ，然后按 TAB 键，如果在命令行中能够看到 arm 编译器的信息，就表明交叉编译工具安装成功。如下图所示。

```
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/standards.info  
root@ubuntu:/usr/local/arm# cd /root/  
root@ubuntu:~# vim .bashrc  
root@ubuntu:~# source .bashrc  
root@ubuntu:~# arm  
arm2hpdl      arm-none-linux-gnueabi-gdbtui  
arm-none-linux-gnueabi-addr2line  arm-none-linux-gnueabi-gprof  
arm-none-linux-gnueabi-ar        arm-none-linux-gnueabi-ld  
arm-none-linux-gnueabi-as        arm-none-linux-gnueabi-nm  
arm-none-linux-gnueabi-c++       arm-none-linux-gnueabi-objcopy  
arm-none-linux-gnueabi-c++filt   arm-none-linux-gnueabi-objdump  
arm-none-linux-gnueabi-cpp       arm-none-linux-gnueabi-ranlib  
arm-none-linux-gnueabi-g++       arm-none-linux-gnueabi-readelf  
arm-none-linux-gnueabi-gcc       arm-none-linux-gnueabi-size  
arm-none-linux-gnueabi-gcc-4.4.1  arm-none-linux-gnueabi-sprite  
arm-none-linux-gnueabi-gcov     arm-none-linux-gnueabi-strings  
arm-none-linux-gnueabi-gdb       arm-none-linux-gnueabi-strip  
root@ubuntu:~# arm
```

### 5.2.3 安装库文件、JDK 以及降低 GCC 版本

为了方便用户，将库文件和 JDK 的安装命令制作成了脚本文件，用户只要执行两个脚本就可以安装库文件和 JDK。这两个脚本在用户光盘 “02\_编译器以及烧写工具” → “tools” 文件夹下的压缩包 “Android\_JDK.tar.bz2” 中，如下图所示。



用户将压缩包拷贝到 Ubuntu 系统中，解压压缩包会生成文件夹 “Android\_JDK” ，如下图所示。

```
root@ubuntu:~# ls /home/topeet/
Android_JDK.tar.bz2  Documents  examples.desktop  Pictures  Templates
Desktop             Downloads  Music           Public    Videos
root@ubuntu:~# cd /home/topeet/
root@ubuntu:/home/topeet#
root@ubuntu:/home/topeet#
root@ubuntu:/home/topeet# ls
Android_JDK.tar.bz2  Documents  examples.desktop  Pictures  Templates
Desktop             Downloads  Music           Public    Videos
root@ubuntu:/home/topeet# tar -vxf Android_JDK.tar.bz2
Android_JDK/
Android_JDK/.DS_Store
Android_JDK/apt-source/
Android_JDK/apt-source/sources.list.163
Android_JDK/apt-source/sources.list.cn
Android_JDK/install-devel-packages.sh
Android_JDK/jdk6/
Android_JDK/jdk6/install-sun-java6.sh
Android_JDK/jdk6/jdk-6u43-linux-x64.bin
Android_JDK/update_gcc.txt
root@ubuntu:/home/topeet# ls
Android_JDK  Desktop  Downloads  Music  Public  Videos
Android_JDK.tar.bz2  Documents  examples.desktop  Pictures  Templates
root@ubuntu:/home/topeet#
```

## 安装库文件和 JDK

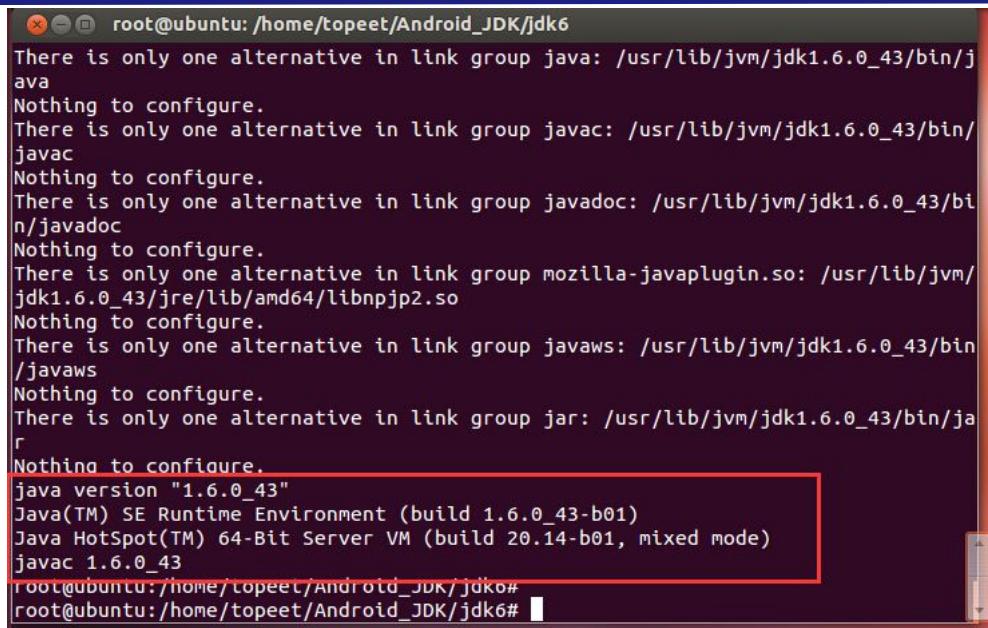
使用 cd 命令，进入解压出来的 “Android\_JDK” --> “jdk6” 文件夹，运行脚本文件 “install-sun-java6.sh” 。

需要注意的是，这条命令执行完毕可能会耗时 15 分钟以上。

如下图所示，执行 “./install-sun-java6.sh” 脚本。

```
Android_JDK/apt-source/sources.list.cn
Android_JDK/install-devel-packages.sh
Android_JDK/jdk6/
Android_JDK/jdk6/install-sun-java6.sh
Android_JDK/jdk6/jdk-6u43-linux-x64.bin
Android_JDK/update_gcc.txt
root@ubuntu:/home/topeet# ls
Android_JDK  Desktop  Downloads  Music  Public  Videos
Android_JDK.tar.bz2  Documents  examples.desktop  Pictures  Templates
root@ubuntu:/home/topeet# cd Android_JDK
root@ubuntu:/home/topeet/Android_JDK# cd jdk6/
root@ubuntu:/home/topeet/Android_JDK/jdk6# ./install-sun-java6.sh
```

执行上面的命令的时候，根据提示输入 “回车” 命令。升级完成之后如下图所示。



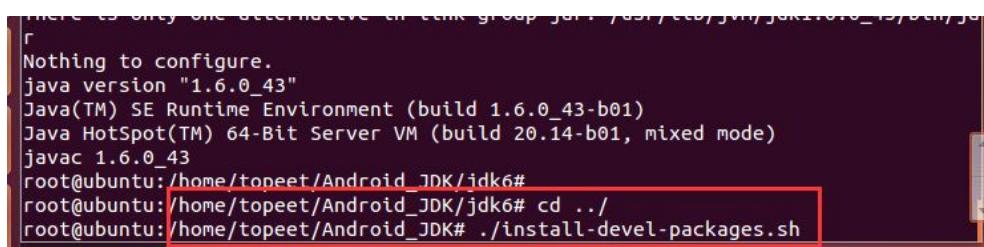
```

root@ubuntu:/home/topeet/Android_JDK/jdk6
There is only one alternative in link group java: /usr/lib/jvm/jdk1.6.0_43/bin/java
Nothing to configure.
There is only one alternative in link group javac: /usr/lib/jvm/jdk1.6.0_43/bin/javac
Nothing to configure.
There is only one alternative in link group javadoc: /usr/lib/jvm/jdk1.6.0_43/bin/javadoc
Nothing to configure.
There is only one alternative in link group mozilla-javaplugin.so: /usr/lib/jvm/jdk1.6.0_43/jre/lib/amd64/libnpjp2.so
Nothing to configure.
There is only one alternative in link group javaws: /usr/lib/jvm/jdk1.6.0_43/bin/javaws
Nothing to configure.
There is only one alternative in link group jar: /usr/lib/jvm/jdk1.6.0_43/bin/jar
Nothing to configure.
java version "1.6.0_43"
Java(TM) SE Runtime Environment (build 1.6.0_43-b01)
Java HotSpot(TM) 64-Bit Server VM (build 20.14-b01, mixed mode)
javac 1.6.0_43
root@ubuntu:/home/topeet/Android_JDK/jdk6#
root@ubuntu:/home/topeet/Android_JDK/jdk6# 
```

进入解压出来的文件夹“Android\_JDK”中运行脚本“install-devel-packages.sh”，安装库文件，具体操作如下：

在Ubuntu命令行中，执行命令“./install-devel-packages.sh”

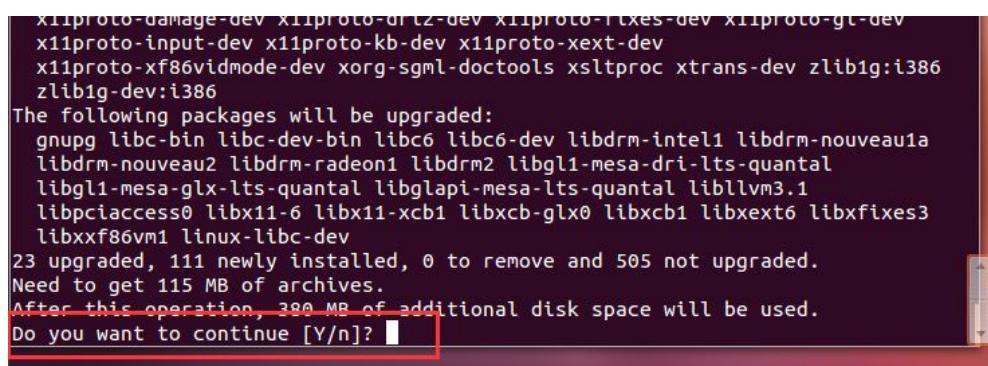
需要注意的是，这条命令可能会耗时40分钟以上。如下图所示。



```

root@ubuntu:/home/topeet/Android_JDK/jdk6#
root@ubuntu:/home/topeet/Android_JDK/jdk6# cd ..
root@ubuntu:/home/topeet/Android_JDK# ./install-devel-packages.sh 
```

上面命令执行的时候，需要根据提示输入“Y”



```

xiiproto-damage-dev xiiproto-dlz-dev xiiproto-rtxes-dev xiiproto-gt-dev
xiiproto-input-dev xiiproto-kb-dev xiiproto-xext-dev
xiiproto-xf86vidmode-dev xorg-sgml-doctools xsltproc xtrans-dev zlib1g:i386
zlib1g-dev:i386
The following packages will be upgraded:
gnupg libc-bin libc-dev-bin libc6 libgcc-dev libdrm-intel1 libdrm-nouveau1a
libdrm-nouveau2 libdrm-radeon1 libdrm2 libgl1-mesa-dri-lts-quantal
libgl1-mesa-glx-lts-quantal libglapi-mesa-lts-quantal libllvm3.1
libpciaaccess0 libxi1-6 libxi1-xcb1 libxcb-glx0 libxcb1 libxext6 libxf86input0
libxf86vm1 linux-libc-dev
23 upgraded, 111 newly installed, 0 to remove and 505 not upgraded.
Need to get 115 MB of archives.
After this operation, 380 MB of additional disk space will be used.
Do you want to continue [Y/n]? 
```

然后安装过程中，还会提示输入“y”，如下图所示。

```
==> Executing: 'apt-get install vim dos2unix minicom gawk'
Reading package lists... Done
Building dependency tree
Reading state information... Done
vim is already the newest version.
The following extra packages will be installed:
  libsigsegv2 lrzs
The following NEW packages will be installed:
  dos2unix gawk libsigsegv2 lrzs minicom
0 upgraded, 5 newly installed, 0 to remove and 505 not upgraded.
Need to get 947 kB of archives.
After this operation, 3,170 kB of additional disk space will be used.
Do you want to continue [Y/n]? ■
```

如下图所示，安装完毕。

```
Unpacking lrzs (from .../lrzs_0.12.21-5_amd64.deb) ...
Selecting previously unselected package minicom.
Unpacking minicom (from .../minicom_2.5-2_amd64.deb) ...
Selecting previously unselected package dos2unix.
Unpacking dos2unix (from .../dos2unix_5.3.1-1_amd64.deb) ...
Processing triggers for man-db ...
Setting up gawk (1:3.1.8+dfsg-0.1ubuntu1) ...
Setting up lrzs (0.12.21-5) ...
Setting up minicom (2.5-2) ...
Setting up dos2unix (5.3.1-1) ...
root@ubuntu:/home/topeet/Android_JDK# ■
```

这里需要注意的是，上面个的脚本执行完毕的时候，注意一下有些库文件是不是提示没有安装。如果发现有库文件没有安装，有可能是网络不好或者下载源丢失。这个时候用户使用一下更新下载源的命令“ apt-get update ”，然后再执行一下上面的两个脚本。

例如，如下图所示，再次运行 “./install-devel-packages.sh” 之后，提示没有无法安装的库和软件，那么表明已经安装完全了。

```
0 upgraded, 0 newly installed, 0 to remove and 505 not upgraded.
==> Executing: 'apt-get install vim dos2unix minicom gawk'
Reading package lists... Done
Building dependency tree
Reading state information... Done
gawk is already the newest version.
dos2unix is already the newest version.
minicom is already the newest version.
vim is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 505 not upgraded.
root@ubuntu:/home/topeet/Android_JDK# ■
```

## 降低 GCC 版本

使用 Ubuntu 编译 Android 的时候需要用到 Ubuntu 系统自带的 GCC4.4.7 编译器，但是安装的 Ubuntu12.04.2 版本，它的 GCC 版本过高，所以这里需将要 GCC 编译的版本降低到 4.4.7。

进入前面解压的文件夹“Android\_JDK”中，会看到一个文本“update\_gcc.txt”，打开文本“update\_gcc.txt”后会看到里面有8条命令，这8条命令需要在Ubuntu命令行中依次执行。

如下图所示，使用命令打开“update\_gcc.txt”文件。

```
root@ubuntu:/home/topeet/Android_JDK# vim update_gcc.txt
[...]
root@ubuntu:/home/topeet/Android_JDK# apt-source install-devel-packages.sh
root@ubuntu:/home/topeet/Android_JDK# vim update_gcc.txt
```

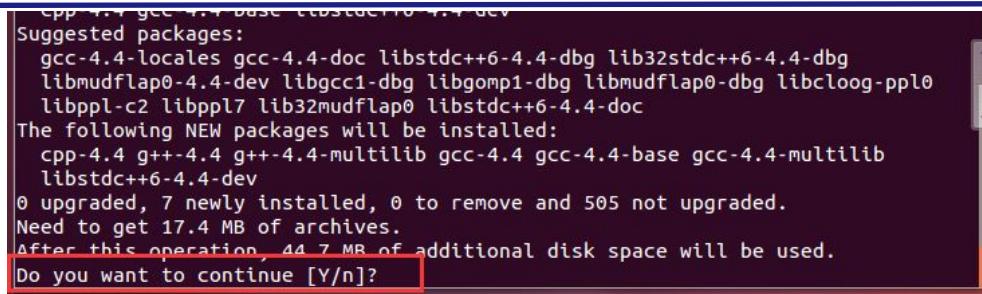
如下图所示，有8条命令。

```
root@ubuntu:/home/topeet/Android_JDK
1.apt-get install gcc-4.4 g++-4.4 g++-4.4-multilib gcc-4.4-multilib
2.update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-4.4 100
3.update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-4.6 50
4.update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.4 100
5.update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.6 50
6.update-alternatives --install /usr/bin/cpp cpp-bin /usr/bin/cpp-4.4 100
7.update-alternatives --install /usr/bin/cpp cpp-bin /usr/bin/cpp-4.6 50
8.gcc -v
~
~
"update_gcc.txt" [noeol][dos] 15L, 546C
1,1 All
```

例如执行第一条命令，

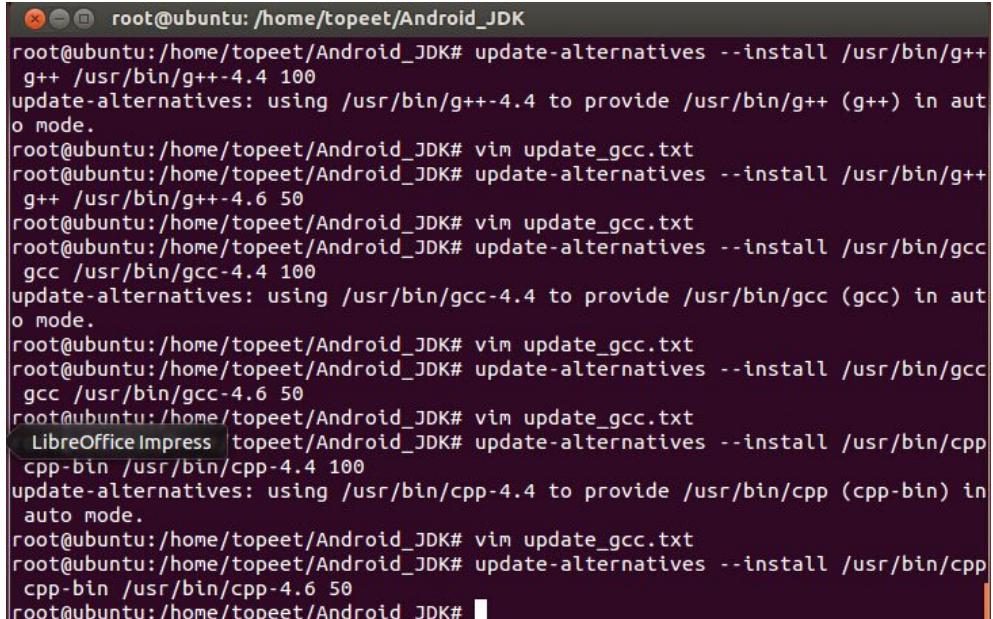
```
gawk is already the newest version.
dos2unix is already the newest version.
minicom is already the newest version.
vim is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 505 not upgraded.
root@ubuntu:/home/topeet/Android_JDK# ls
root@ubuntu:/home/topeet/Android_JDK# apt-source install-devel-packages.sh
root@ubuntu:/home/topeet/Android_JDK# vim update_gcc.txt
root@ubuntu:/home/topeet/Android_JDK# apt-get install gcc-4.4 g++-4.4 g++-4.4-mu
ltilib gcc-4.4-multilib
```

根据提示输入“y”，如下图所示。



```
CPP 4.7.1  gcc 4.7.1  basic  C/C++  compiler  4.7.1  dev
Suggested packages:
  gcc-4.4-locales  gcc-4.4-doc  libstdc++6-4.4-dbg  lib32stdc++6-4.4-dbg
  libmudflap0-4.4-dev  libgcc1-dbg  libgomp1-dbg  libmudflap0-dbg  libcloog-ppl0
  libppl-c2  libppl7  lib32mudflap0  libstdc++6-4.4-doc
The following NEW packages will be installed:
  cpp-4.4  g++-4.4  g++-4.4-multilib  gcc-4.4  gcc-4.4-base  gcc-4.4-multilib
  libstdc++6-4.4-dev
0 upgraded, 7 newly installed, 0 to remove and 505 not upgraded.
Need to get 17.4 MB of archives.
After this operation, 44.7 MB of additional disk space will be used.
Do you want to continue [Y/n]?
```

其余几条命令，如下图所示，执行起来很快。



```
root@ubuntu:/home/topeet/Android_JDK# update-alternatives --install /usr/bin/g++
g++ /usr/bin/g++-4.4 100
update-alternatives: using /usr/bin/g++-4.4 to provide /usr/bin/g++ (g++) in auto mode.
root@ubuntu:/home/topeet/Android_JDK# vim update_gcc.txt
root@ubuntu:/home/topeet/Android_JDK# update-alternatives --install /usr/bin/g++
g++ /usr/bin/g++-4.6 50
root@ubuntu:/home/topeet/Android_JDK# vim update_gcc.txt
root@ubuntu:/home/topeet/Android_JDK# update-alternatives --install /usr/bin/gcc
gcc /usr/bin/gcc-4.4 100
update-alternatives: using /usr/bin/gcc-4.4 to provide /usr/bin/gcc (gcc) in auto mode.
root@ubuntu:/home/topeet/Android_JDK# vim update_gcc.txt
root@ubuntu:/home/topeet/Android_JDK# update-alternatives --install /usr/bin/gcc
gcc /usr/bin/gcc-4.6 50
root@ubuntu:/home/topeet/Android_JDK# vim update_gcc.txt
LibreOffice Impress |topeet/Android_JDK# update-alternatives --install /usr/bin/cpp
cpp-bin /usr/bin/cpp-4.4 100
update-alternatives: using /usr/bin/cpp-4.4 to provide /usr/bin/cpp (cpp-bin) in auto mode.
root@ubuntu:/home/topeet/Android_JDK# vim update_gcc.txt
root@ubuntu:/home/topeet/Android_JDK# update-alternatives --install /usr/bin/cpp
cpp-bin /usr/bin/cpp-4.6 50
root@ubuntu:/home/topeet/Android_JDK#
```

在执行了这 8 条命令之后，Ubuntu 系统就将 gcc 的版本降低到 4.4.7。

如下图所示，使用命令 “gcc -v” ，可以看到 gcc 的版本为 4.4.7 了。

```
root@ubuntu: /home/topeet/Android_JDK
cpp-bin /usr/bin/cpp-4.4 100
update-alternatives: using /usr/bin/cpp-4.4 to provide /usr/bin/cpp (cpp-bin) in
auto mode.
root@ubuntu:/home/topeet/Android_JDK# vim update_gcc.txt
root@ubuntu:/home/topeet/Android_JDK# update-alternatives --install /usr/bin/cpp
cpp-bin /usr/bin/cpp-4.6 50
root@ubuntu:/home/topeet/Android_JDK# vim update_gcc.txt
root@ubuntu:/home/topeet/Android_JDK#
root@ubuntu:/home/topeet/Android_JDK# gcc -v
Using built-in specs.
Target: x86_64-linux-gnu
Configured with: ../src/configure -v --with-pkgversion='Ubuntu/Linaro 4.4.7-1ubu
ntu2' --with-bugurl=file:///usr/share/doc/gcc-4.4/README.Bugs --enable-languages
=c,c++,fortran,objc,obj-c++ --prefix=/usr --program-suffix=-4.4 --enable-shared
--enable-linker-build-id --with-system-zlib --libexecdir=/usr/lib --without-incl
uded-gettext --enable-threads=posix --with-gxx-include-dir=/usr/include/c++/4.4
--libdir=/usr/lib --enable-nls --with-sysroot=/ --enable-locale=gnu --enable-li
bstdcxx-debug --enable-objc-gc --disable-werror --with-arch-32=i686 --with-tune=
generic --enable-checking=release --build=x86_64-linux-gnu --host=x86_64-linux-g
nu --target=x86_64-linux-gnu
Thread model: posix
gcc version 4.4.7 (Ubuntu/Linaro 4.4.7-1ubuntu2)
root@ubuntu:/home/topeet/Android_JDK#
```

需要注意的是，在执行这 8 条命令时，只有第一条命令会耗时 10 分钟左右，其它的都会很快完成，而且命令一定要依次执行，不能有遗漏。

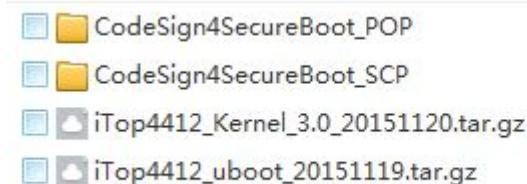
## 5.3Android4.0.3 镜像的编译

无论什么文件系统都需要和 linux 内核以及 uboot 对应，所以在本章节先介绍 Android4.0.3 文件系统对应的 uboot 以及 kernel 编译，再介绍 Android4.0.3 文件系统的编译。

### 5.3.1uboot 的编译

#### 5.3.1.1 源码目录

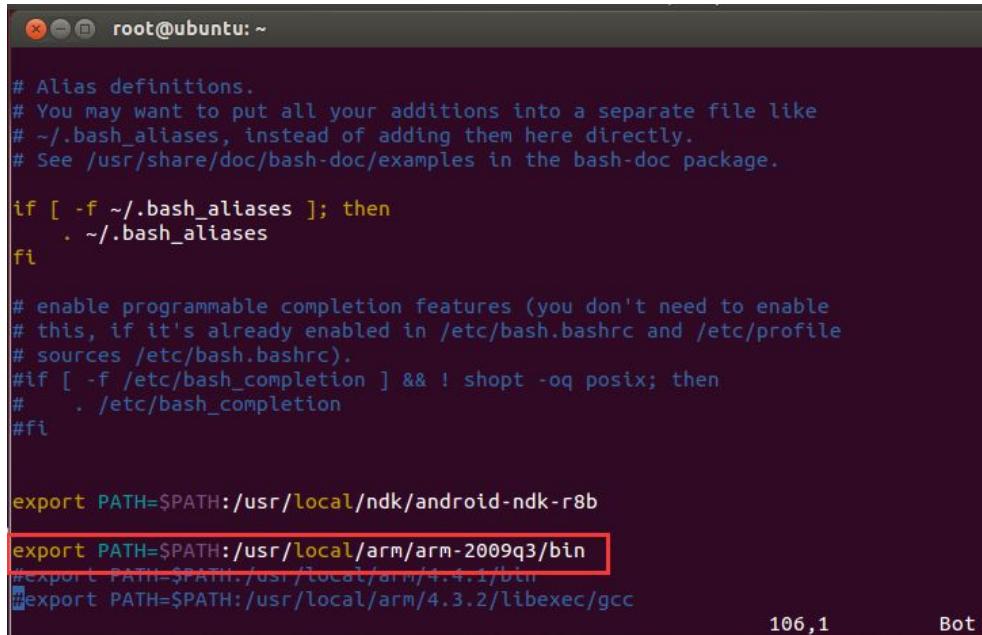
Uboot 源码在光盘 “06\_源码\_uboot 和 kernel” 目录下，如下图所示。



### 5.3.1.2 编译器

如下图所示，编译器是使用的光盘目录下，“02\_编译器以及烧写工具”→“arm 交叉编译器”下的“arm-2009q3.tar.bz2”

如果使用的是搭建好的环境，确保编译器环境变量，如下图所示。



The screenshot shows a terminal window titled "root@ubuntu: ~". The window displays the contents of the .bashrc file. The export command for the PATH variable is highlighted with a red rectangle. The command is: `export PATH=$PATH:/usr/local/arm/arm-2009q3/bin`. Other parts of the PATH export command are also visible but not highlighted.

```
# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
#if [ -f /etc/bash_completion ] && ! shopt -oq posix; then
#    . /etc/bash_completion
#fi

export PATH=$PATH:/usr/local/arm/arm-2009q3/bin
#export PATH=$PATH:/usr/local/arm/4.4.1/bin
#export PATH=$PATH:/usr/local/arm/4.3.2/libexec/gcc
```

### 5.3.1.3 参数配置

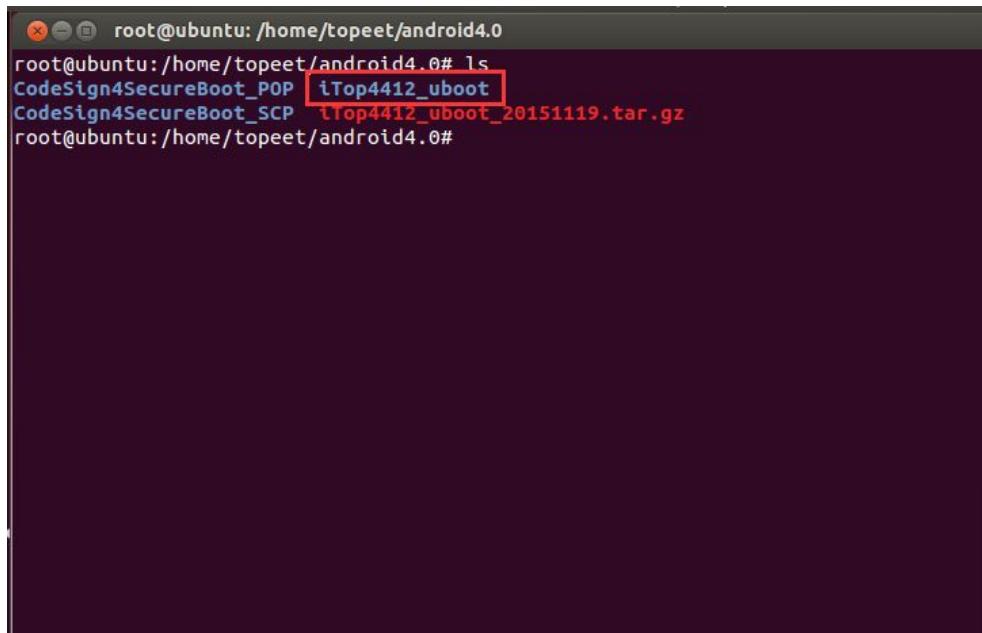
编译 uboot 的脚本是源码文件夹中的“build\_uboot.sh”，在编译的时候需要向脚本传参数，根据核心板的不同，脚本执行参数如下表所示。

硬件分类	脚本执行参数
核心板 SCP 1G 内存	SCP_1GDDR
核心板 SCP 2G 内存	SCP_2GDDR
核心板 POP 1G 内存	POP_1GDDR
核心板 POP 2G 内存	POP_2GDDR

### 5.3.1.4 编译生成 uboot 镜像举例

这里以 SCP 1G 核心板为例编译 uboot 镜像。

将光盘 “06\_源码\_uboot 和 kernel” 目录下 “CodeSign4SecureBoot\_POP” 、 “CodeSign4SecureBoot\_SCP” 以及 “iT0p4412\_uboot\_xxx.tar.gz” 拷贝到 Ubuntu 系统下，然后将 “iT0p4412\_uboot\_xxx.tar.gz” 解压，得到 “iT0p4412\_uboot” 文件夹，如下图所示。



进入 “iT0p4412\_uboot” 文件夹，使用编译脚本 “build\_uboot.sh” 编译 uboot，这里需要编译的是 “SCP 1G 核心板” 的 uboot 镜像，那么编译命令是

“./build\_uboot.sh SCP\_1GDDR”

输入编译命令，如下图所示。这里一定先确定核心板是哪种类型，然后将对应的参数传到脚本。

```

root@ubuntu:/home/topeet/android4.0/iTop4412_uboot
root@ubuntu:/home/topeet/android4.0/iTop4412_uboot# ls
all00_padding.bin  E4412_N.b11.bin  lib_nios  onenand_ipl
api               examples      lib_nios2  paddingaa
board              fs          lib_ppc   post
build_uboot.sh    include      lib_sh    README
common             lib_arm     lib_sparc  MAINTAINERS
config.mk          lib_avr32   MAKEALL   rules.mk
COPYING            lib_blackfin  lib_spice  sdfuse
cpu                lib_fdt     Makefile  sdifuse.o
CREDITS            lib_generic  mkl2      tc4_cmm.cmm
disk               lib_i386    mkconfig  tools
doc                lib_m68k    mkuboot  uboot_readme.txt
drivers             lib_microblaze  nand_spl
E4212              lib_mips    net
root@ubuntu:/home/topeet/android4.0/iTop4412_uboot# ./build_uboot.sh SCP_1GDDR

```

如下图所示，编译中。

```

enand.a drivers/mtd/ubi/libubi.a drivers/mtd/spi/libspi_flash.a drivers/net/libn
et.a drivers/net/phy/libphy.a drivers/pci/libpci.a drivers/pcmcia/libpcmcia.a dr
ivers/power/libpower.a drivers/spi/libspi.a drivers/rtc/librtc.a drivers/serial/
libserial.a drivers/twserial/libtws.a drivers/usb/gadget/libusb_gadget.a drivers
/usb/host/libusb_host.a drivers/usb/musb/libusb_musb.a drivers/usb/phy/libusb_ph
y.a drivers/video/libvideo.a drivers/watchdog/libwatchdog.a common/libcommon.a l
ibfdt/libfdt.a api/libapi.a post/libpost.a board/samsung/smdkc210/libsmdkc210.a
--end-group /home/topeet/android4.0/iTop4412_uboot/lib_arm/eabi_compatible.o -L /usr
/local/arm/arm-2009q3/bin/../lib/gcc/arm-none-linux-gnueabi/4.4.1 -lgcc -Map u-b
oot.map -o u-boot
/usr/local/arm/arm-2009q3/bin/arm-none-linux-gnueabi-objcopy -O srec u-boot u-bo
ot.srec
/usr/local/arm/arm-2009q3/bin/arm-none-linux-gnueabi-objcopy --gap-fill=0xff -O
binary u-boot u-boot.bin
make[1]: Entering directory `/home/topeet/android4.0/iTop4412_uboot/sdfuse_q'
gcc -o cksum cksum.c
gcc -o add_sign add_sign.c
gcc -o add_padding add_padding.c
make[1]: Leaving directory `/home/topeet/android4.0/iTop4412_uboot/sdfuse_q'
bl2aa file size= 14336B
before padding uboot.bin file size= 249988B
85884 B written

```

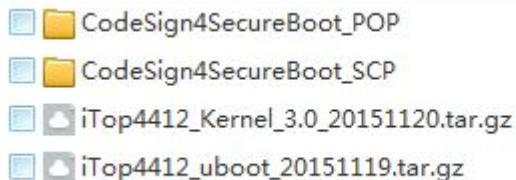
如下图所示，脚本执行完成，在“iTop4412\_uboot”文件夹下生成了“u-boot-iTOP-4412.bin”文件。生成的文件“u-boot-iTOP-4412.bin”文件就是SCP 1G 内存核心板对应的 uboot 镜像文件。

```
root@ubuntu:/home/topeet/android4.0/iTop4412_uboot# ls
all00_padding.bin    examples      lib_sh      rules.mk
api                  fs           lib_sparc   sdfuse
board                include      MAINTAINERS sdfuse.d
build_uboot.sh       lib_arm     MAKEALL     System.map
checksum_b12_14k.bin lib_avr32   Makefile    tc4_cmm.cmm
common               lib_blackfin libconfig   tools
config.mk            lib_fdt     mkbl2      u-boot
COPYING              lib_generic lib_i386    u-boot.bin
cpu                  lib_m68k    lib_microblaze onenand_ipl u-boot.lds
CREDITS              lib_mips    lib_nios    paddingaa u-boot.map
disk                 lib_nios    lib_nios2   post      u-boot.srec
doc                  lib_ppc     README     README
E4212_N.bl1.bin      lib_ppc     readme.txt
root@ubuntu:/home/topeet/android4.0/iTop4412_uboot#
```

## 5.3.2 Linux 内核的编译

### 5.3.2.1 源码目录

Linux 内核源码在光盘 “06\_源码\_uboot 和 kernel” 目录下，如下图所示。



### 5.3.2.2 编译器

内核的编译器和 uboot 的编译器一样，参考 “5.3.1.2 编译器”

### 5.3.2.3 参数配置

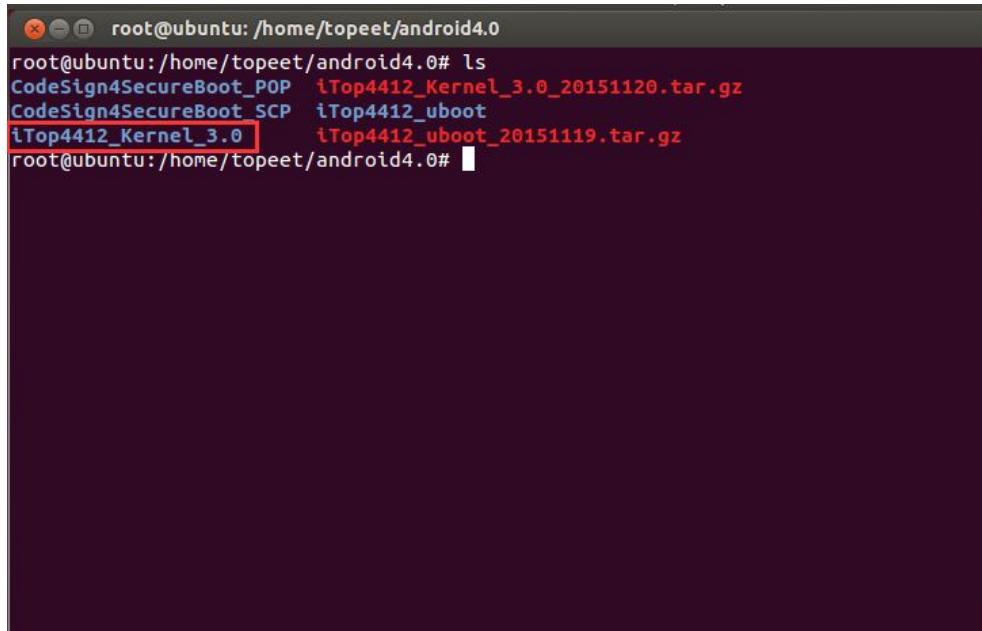
内核的编译是组合式配置文件，基本的配置文件名是 “config\_for\_android\_YY\_elite” ，  
YY 表示用下表所示的参数替代。

硬件分类	配置文件
核心板 SCP 1G 或者 2G 内存	config_for_android_scp_elite
核心板 POP 1G 内存	config_for_android_pop_elite
核心板 POP 2G 内存	config_for_android_pop2G_elite

### 5.3.2.4 编译生成内核镜像举例

这里以 SCP 1G 核心板为例编译 zImage 内核镜像,那么配置文件为 “config\_for\_android\_scp\_elite”。

将光盘 “06\_源码\_uboot 和 kernel” 目录下的压缩包 “iT0p4412\_Kernel\_3.0\_xxx.tar.gz” 拷贝到 Ubuntu , 然后解压 , 得到文件夹 “iT0p4412\_Kernel\_3.0 ” , 如下图所示。



```
root@ubuntu:/home/topeet/android4.0
root@ubuntu:/home/topeet/android4.0# ls
CodeSign4SecureBoot_POP  iT0p4412_Kernel_3.0_20151120.tar.gz
CodeSign4SecureBoot_SCP  iT0p4412_uboot
iT0p4412_Kernel_3.0     iT0p4412_uboot_20151119.tar.gz
root@ubuntu:/home/topeet/android4.0#
```

进入文件夹 “iT0p4412\_Kernel\_3.0 ” , 使用命令

“cp config\_for\_android\_scp\_elite .config” 覆盖自带的配置文件 , 如下图所示。

```
root@ubuntu:/home/topeet/android4.0# ls
CodeSign4SecureBoot_POP  iTop4412_Kernel_3.0_20151120.tar.gz
CodeSign4SecureBoot SCP  iTop4412_uboot
iTop4412_Kernel_3.0      iTop4412_uboot_20151119.tar.gz
root@ubuntu:/home/topeet/android4.0# cd iTop4412_Kernel_3.0
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0# cp config_for_android_s
cp_elite .config
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0#
```

然后使用编译命令 “make zImage” , 如下图所示。

```
root@ubuntu:/home/topeet/android4.0# ls
CodeSign4SecureBoot_POP  iTop4412_Kernel_3.0_20151120.tar.gz
CodeSign4SecureBoot SCP  iTop4412_uboot
iTop4412_Kernel_3.0      iTop4412_uboot_20151119.tar.gz
root@ubuntu:/home/topeet/android4.0# cd iTop4412_Kernel_3.0
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0# cp config_for_android_s
cp_elite .config
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0# make zImage
```

编译中 , 如下图所示。

```
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0# cp config_for_androiu_s
cp_elite .config
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0# make zImage
HOSTCC scripts/basic/fixedp
HOSTCC scripts/kconfig/conf.o
SHIPPED scripts/kconfig/zconf.tab.c
SHIPPED scripts/kconfig/lex.zconf.c
SHIPPED scripts/kconfig/zconf.hash.c
HOSTCC scripts/kconfig/zconf.tab.o
HOSTLD scripts/kconfig/conf
scripts/kconfig/conf --silentoldconfig Kconfig
CHK include/linux/version.h
UPD include/linux/version.h
CHK include/generated/utsrelease.h
UPD include/generated/utsrelease.h
Generating include/generated/mach-types.h
CC kernel/bounds.s
GEN include/generated/bounds.h
CC arch/arm/kernel/asm-offsets.s
GEN include/generated/asm-offsets.h
CALL scripts/checksyscalls.sh
CC scripts/mod/empty.o
HOSTCC scripts/mod/mk_elfconfig
```

编译完成，如下图所示。

```
CC init/version.o
LD init/built-in.o
LD .tmp_vmlinux1
KSYM .tmp_kallsyms1.S
AS .tmp_kallsyms1.o
LD .tmp_vmlinux2
KSYM .tmp_kallsyms2.S
AS .tmp_kallsyms2.o
LD vmlinux
SYSMAP System.map
SYSMAP .tmp_System.map
OBJCOPY arch/arm/boot/Image
Kernel: arch/arm/boot/Image is ready
AS arch/arm/boot/compressed/head.o
GZIP arch/arm/boot/compressed/piggy.gzip
AS arch/arm/boot/compressed/piggy.gzip.o
CC arch/arm/boot/compressed/misc.o
CC arch/arm/boot/compressed/decompress.o
SHIPPED arch/arm/boot/compressed/lib1funcs.S
AS arch/arm/boot/compressed/lib1funcs.o
LD arch/arm/boot/compressed/vmlinux
OBJCOPY arch/arm/boot/zImage
Kernel: arch/arm/boot/zImage is ready
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0#
```

文件夹“iTop4412\_Kernel\_3.0”下的“arch”-->“arm”-->“boot”会生成镜像文件“zImage”，这个zImage镜像可以给 **SCP 1G 和 SCP 2G** 的核心板使用，如下图所示。

```
LD      .tmp_vmlinux1
KSYM   .tmp_kallsyms1.S
AS     .tmp_kallsyms1.o
LD      .tmp_vmlinux2
KSYM   .tmp_kallsyms2.S
AS     .tmp_kallsyms2.o
LD      vmlinux
SYSMAP System.map
SYSMAP .tmp_System.map
OBJCOPY arch/arm/boot/Image
Kernel: arch/arm/boot/Image is ready
AS      arch/arm/boot/compressed/head.o
GZIP   arch/arm/boot/compressed/piggy.gzip
AS      arch/arm/boot/compressed/piggy.gzip.o
CC      arch/arm/boot/compressed/misc.o
CC      arch/arm/boot/compressed/decompress.o
SHIPPED arch/arm/boot/compressed/lib1funcs.S
AS      arch/arm/boot/compressed/lib1funcs.o
LD      arch/arm/boot/compressed/vmlinux
OBJCOPY arch/arm/boot/zImage
Kernel: arch/arm/boot/zImage is ready
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0# ls arch/arm/boot/
bootp compressed Image install.sh Makefile zImage
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0#
```

### 5.3.3 Android4.0.3 的编译

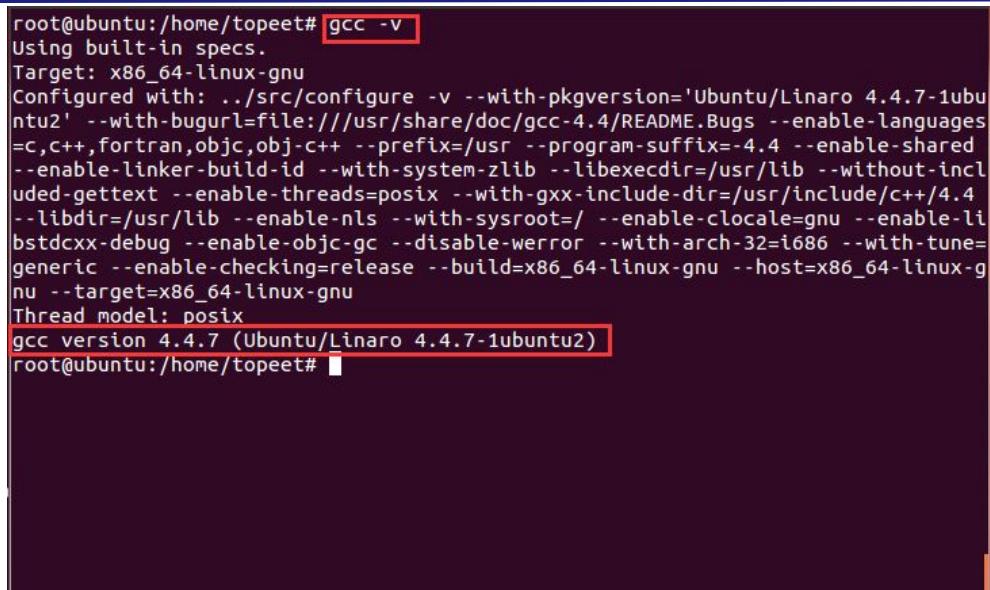
#### 5.3.3.1 源码目录

Android4.0.3 文件系统的源码在光盘 “07\_源码\_Android4.0.3 文件系统” 目录下，如下图所示。

文件名	大小
iTop4412_IKS_git_20151120.tar.gz	1.88GB

#### 5.3.3.2 编译器

Android4.0.3 的编译器是 4.4.7 版本（包括其他所有版本的 Android 编译器都是相同的），如下图所示，在控制台使用命令 “gcc -v” ，可以查看到 gcc 的版本。



```
root@ubuntu:/home/topeet# gcc -v
Using built-in specs.
Target: x86_64-linux-gnu
Configured with: ../src/configure -v --with-pkgversion='Ubuntu/Linaro 4.4.7-1ubuntu2' --with-bugurl=file:///usr/share/doc/gcc-4.4/README.Bugs --enable-languages=c,c++,fortran,objc,obj-c++ --prefix=/usr --program-suffix=-4.4 --enable-shared --enable-linker-build-id --with-system-zlib --libexecdir=/usr/lib --without-included-gettext --enable-threads=posix --with-gxx-include-dir=/usr/include/c++/4.4 --libdir=/usr/lib --enable-nls --with-sysroot=/ --enable-clocale=gnu --enable-stdcxx-debug --enable-objc-gc --disable-werror --with-arch-32=i686 --with-tune=generic --enable-checking=release --build=x86_64-linux-gnu --host=x86_64-linux-gnu --target=x86_64-linux-gnu
Thread model: posix
gcc version 4.4.7 (Ubuntu/Linaro 4.4.7-1ubuntu2)
root@ubuntu:/home/topeet#
```

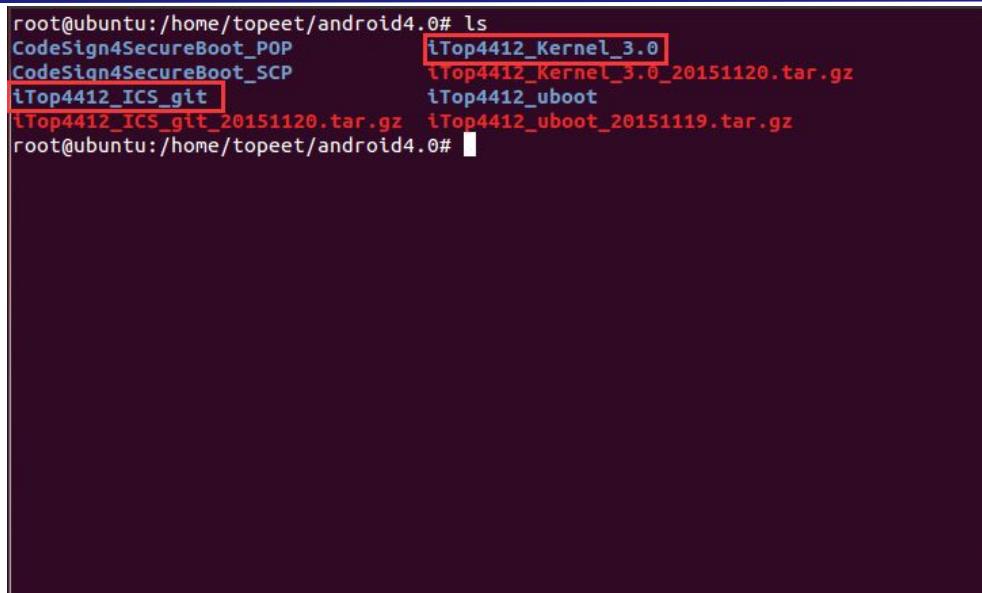
### 5.3.3.3 参数配置

无参数配置。所有种类核心板对应的 Android4.0.3 都使用同一套源码，同一种编译方法。

编译脚本是 “build\_android.sh” 。

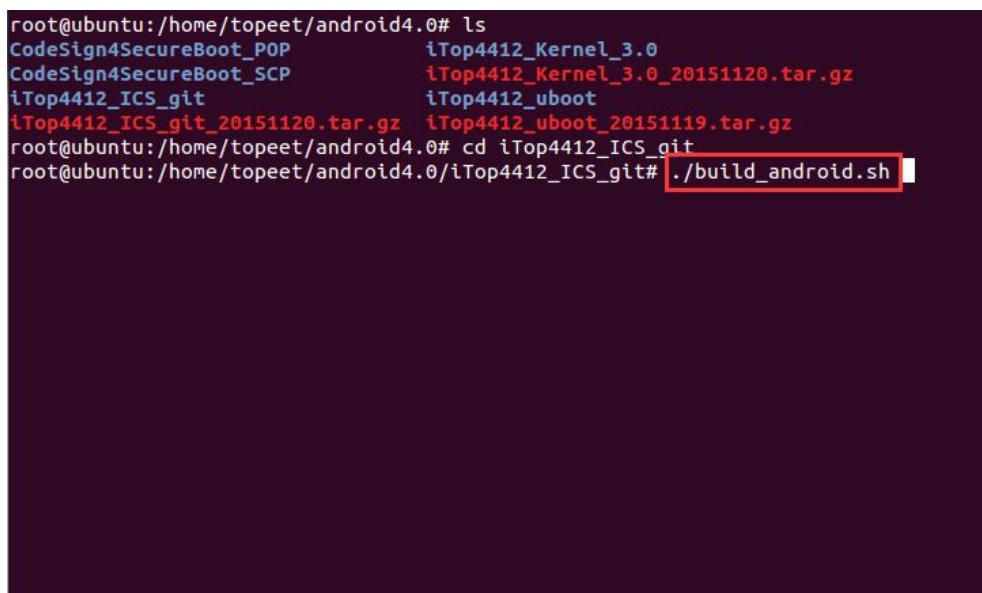
### 5.3.3.4 编译生成 Android4.0.3 镜像

将光盘 “07\_源码\_Android4.0.3 文件系统” 目录下压缩包 “iT0p4412\_ICS\_git\_xxx.tar.gz” 拷贝到 Ubuntu 系统中，解压压缩包，得到文件夹 “iT0p4412\_ICS\_git” 。这里需要注意的是，Android 源码文件夹 “iT0p4412\_ICS\_git” 和内核源码文件夹 “iT0p4412\_Kernel\_3.0” 需要放到同一目录下，如下图所示。



```
root@ubuntu:/home/topeet/android4.0# ls
CodeSign4SecureBoot_POP          iTop4412_Kernel_3.0
CodeSign4SecureBoot_SCP          iTop4412_Kernel_3.0_20151120.tar.gz
iTop4412_IKS_git               iTop4412_uboot
iTop4412_IKS_git_20151120.tar.gz iTop4412_uboot_20151119.tar.gz
root@ubuntu:/home/topeet/android4.0#
```

进入“iTop4412\_IKS\_git”目录，使用命令“./build\_android.sh”运行编译脚本,编译Android4.0.3，如下图所示。



```
root@ubuntu:/home/topeet/android4.0# ls
CodeSign4SecureBoot_POP          iTop4412_Kernel_3.0
CodeSign4SecureBoot_SCP          iTop4412_Kernel_3.0_20151120.tar.gz
iTop4412_IKS_git               iTop4412_uboot
iTop4412_IKS_git_20151120.tar.gz iTop4412_uboot_20151119.tar.gz
root@ubuntu:/home/topeet/android4.0# cd iTop4412_IKS_git
root@ubuntu:/home/topeet/android4.0/iTop4412_IKS_git# ./build_android.sh
```

开始编译，如下图所示。

```
root@ubuntu:/home/topeet/android4.0/iTop4412_ICS_git
root@ubuntu:/home/topeet/android4.0/iTop4412_ICS_git# ./build_android.sh

Build android for smdk4x12

[[[[[[ Build android platform ]]]]]]

make -j4 PRODUCT=full_smdk4x12-eng

=====
PLATFORM_VERSION_CODENAME=REL
PLATFORM_VERSION=4.0.3
TARGET_PRODUCT=full_smdk4x12
TARGET_BUILD_VARIANT=eng
TARGET_BUILD_TYPE=release
TARGET_BUILD_APPS=
TARGET_ARCH=arm
TARGET_ARCH_VARIANT=armv7-a-neon
HOST_ARCH=x86
HOST_OS=linux
HOST_BUILD_TYPE=release
BUILD_ID=IML74K
=====
```

编译比较耗费时间，第一次编译会花费 60 分钟以上。

```
out/target/product/smdk4x12/system.img+ total size is 253251420
Total compile time is 4327 seconds

[[[[[[ Make ramdisk image for u-boot ]]]]]]

Image Name: ramdisk
Created: Fri Nov 27 07:34:58 2015
Image Type: ARM Linux RAMDisk Image (uncompressed)
Data Size: 922118 Bytes = 900.51 kB = 0.88 MB
Load Address: 40800000
Entry Point: 40800000

[[[[[[ Make additional images for fastboot ]]]]]]

boot.img -> /home/topeet/android4.0/iTop4412_ICS_git/out/target/product/smdk4x12
update.zip -> /home/topeet/android4.0/iTop4412_ICS_git/out/target/product/smdk4x12
    adding: android-info.txt (stored 0%)
    adding: boot.img (deflated 1%)
    adding: system.img (deflated 35%)

ok success !!!
root@ubuntu:/home/topeet/android4.0/iTop4412_ICS_git#
```

编译完成后在文件夹 “iTop4412\_ICS” --> “out” --> “target” --> “product--> “smdk4x12” 中生成 Android4.0.3 镜像文件 “ramdisk-uboot.img” 和 “system.img” , 如下图所示。

```
Created: Fri Nov 27 07:34:58 2015
Image Type: ARM Linux RAMDisk Image (uncompressed)
Data Size: 922118 Bytes = 900.51 kB = 0.88 MB
Load Address: 40800000
Entry Point: 40800000

[[[[[[ Make additional images for fastboot ]]]]]]

boot.img -> /home/topeet/android4.0/iTop4412_ICS_git/out/target/product/smdk4x12
update.zip -> /home/topeet/android4.0/iTop4412_ICS_git/out/target/product/smdk4x12
12
    adding: android-info.txt (stored 0%)
    adding: boot.img (deflated 1%)
    adding: system.img (deflated 35%)

ok success !!!
root@ubuntu:/home/topeet/android4.0/iTop4412_ICS_git# ls out/target/product/smdk4x12/
android-info.txt  installed-files.txt      root          update.zip
boot.img          obj                  symbols       userdata.img
clean_steps.mk    previous_build_config.mk  system
data              ramdisk-uboot.img      system.img
root@ubuntu:/home/topeet/android4.0/iTop4412_ICS_git#
```

如果想要支持 wifi，在编译好内核之后，还需要在内核目录中，执行编译模块的命令“make modules”，再执行编译 Android4.0.3 文件系统的脚本。

## 5.4Android4.4.4 镜像的编译

网盘中下载 Android4.4.4 文件系统以及 uboot , kernel 的源码。

Android4.4.4 和 Android4.0.3 的编译类似。

### 5.4.1uboot 的编译

Android4.4.4 对应 uboot 的源码，编译器，参数配置，编译脚本以及编译参数和 Android4.0.3 的 uboot 全部一模一样。

### 5.4.2 Linux 内核的编译

#### 5.4.2.1 源码目录

网盘下载 Android4.4.4 对应的源码。在网盘 “iTOP-4412 开发板系统源码及镜像（其他）” → “android\_4.4.4 源码及镜像” 目录下。

### 5.4.2.2 编译器

Android4.4.4 对应内核的编译器和 Android4.0.3 的内核编译器一模一样。

### 5.4.2.3 参数配置

内核的编译是组合式配置文件，基本的配置文件名是 “config\_for\_android\_YY” , YY 表示用下表所示的参数替代。

硬件分类	配置文件
核心板 SCP 1G 或者 2G 内存	config_for_android_scp
WiFi 支持配置：	config_for_android_scp_wifi&Bluetooth
核心板 POP 1G 或者 2G 内存	config_for_android_pop
WiFi 支持配置：	config_for_android_pop_wifi&Bluetooth

如上表所示，如果需要 Android4.4.4 支持 WiFi，则需要配置对应的参数。

### 5.4.2.4 编译生成内核镜像举例

和 Android4.0.3 内核一样，如果需要编译对应核心板的内核，首先使用 cp 命令将对应的配置文件覆盖掉“ .config ”，然后在执行编译命令 “make zImage” 。

生成内核镜像的目录也是 “arch” --> “arm” --> “boot” 。

### 5.4.3 Android4.4.4 的编译

Android4.4.4 源码在网盘下载，编译器和参数配置和 Android4.0.3 一模一样。

编译 Android4.4.4，还缺少一个 “javap” 命令，使用命令：

```
"update-alternatives --install "/usr/bin/javap" "javap"
"/usr/lib/jvm/jdk1.6.0_43/bin/javap" 1"
```

更新一下，如下图所示。

```
root@ubuntu:/home/topeet/Android4.4/iTop4412_KK4.4# update-alternatives --install
l "/usr/bin/javap" "javap" "/usr/lib/jvm/jdk1.6.0_43/bin/javap" 1
```

然后进入 Android4.4.4 源码解压后得到文件夹 “iTop4412\_KK4.4” 中，使用命令 “./build\_android.sh” ，运行一键编译脚本，开始编译 Android4.4.4。

```
root@ubuntu:/home/topeet/Android4.4/iTop4412_KK4.4# ./build_android.sh
Build android for smdk4x12

including device/samsung/manta/vendorsetup.sh
including device/samsung/smdk4x12/vendorsetup.sh
including device/asus/grouper/vendorsetup.sh
including device/asus/tilapia/vendorsetup.sh
including device/asus/flounder/vendorsetup.sh
```

如下图所示，编译完成。在文件夹

“iTop4412\_KK4.4/out/target/product/smdk4x12” 中，生成镜像 “ramdisk.img” 和 “system.img” 。

```
out/target/product/smdk4x12/system.img+ maxsize=411079680 blocksize=4224 total=2
91223112 reserve=4156416
Total compile time is 5000 seconds

[[[[[[[ Make ramdisk image for u-boot ]]]]]]

Image Name: ramdisk
Image Type: ARM Linux RAMDisk Image (uncompressed)
Data Size: 325500 Bytes = 317.87 kB = 0.31 MB
Load Address: 40800000
Entry Point: 40800000
/home/topeet/Android4.4/iTop4412_KK4.4/out/target/product/smdk4x12/ramdisk-uboot
.img
OK!
ok success !!!
```

这里还需要注意的是，Android4.4.4 源码需要占用较大的空间，用户需要确认有足够的空间才能够成功编译。Android4.0.3 编译完成后总共大约占用 18G 的空间，Android4.4.4 编译完成后总共占用大约 36G 的空间，如下图。

```
root@ubuntu:/home/topeet/Android4.4/iTop4412_KK4.4# du -sh
36G ←
```

用户完全按照步骤编译错误，可以使用 “df -l” 查看一下盘符剩余空间还剩下多少，如下图所示，如果是已使用 100%，则是空间不足。

```
root@ubuntu:/home/topeet/Android4.4/iTop4412_Kernel_3.0# df -l
Filesystem      1K-blocks    Used Available Use% Mounted on
/dev/sda1        60894268  45378900   12422120  79% /
udev             2013748      4   2013744  1% /dev
tmpfs            809208     812    808396  1% /run
none              5120       0     5120  0% /run/lock
none            2023016     200   2022816  1% /run/shm
root@ubuntu:/home/topeet/Android4.4/iTop4412_Kernel_3.0#
```

# 六 Qtopia2.2.0 开发环境搭建以及编译镜像

搭建 Qtopia2.2.0 开发环境，需要先搭建 Android 的编译环境，然后在 Android 编译环境的基础上，再搭建 Qtopia2.2.0 编译环境。

Qtopia2.2.0 的编译环境看似复杂，用户只要抓住几个要点就可以了。

第一：编译器。编译器在光盘中都有提供，在需要使用的步骤中，说明其在光盘中的位置。

第二：设置环境变量。环境变量设置后，编译的时候，系统才能找到编译器。

第三：库文件。搭建过程中会给通过执行简单的脚本命令来安装库文件，复杂的步骤变的简单有效。

第四：源码。官网下载的 Qtopia2.2.0 的源文件有少量的 Bug，经过迅为工程师的修改已经可以直接使用，源码修改这一步用户可以直接跳过。

如果用户是使用“搭建好的 Ubuntu 镜像”，则只需要改一下环境变量，系统里面的工具和库文件都已经安装完毕了。

## 6.1 uboot 的编译

Qtopia2.2.0 系统中 Uboot 和 Android4.0.3 的 Uboot 源码,编译器，参数配置，编译都是通用的，参考 5.3.1 小节。

## 6.2 Linux 内核的编译

Qtopia2.2.0 系统中 Linux 内核和 Android4.0.3 中的 Linux 内核源码是一样的，编译环境和编译方法也一样，参考 5.2 小节。

## 6.2.1 参数配置

Qtopia2.2.0 文件系统对应的内核，源码以及编译环境都和 Android4.0.3 的内核一样。主要是配置文件不一样。

内核的编译是组合式配置文件，基本的配置文件名是 “config\_for\_linux\_YY\_elite” ，YY 表示用下表所示的参数替代。

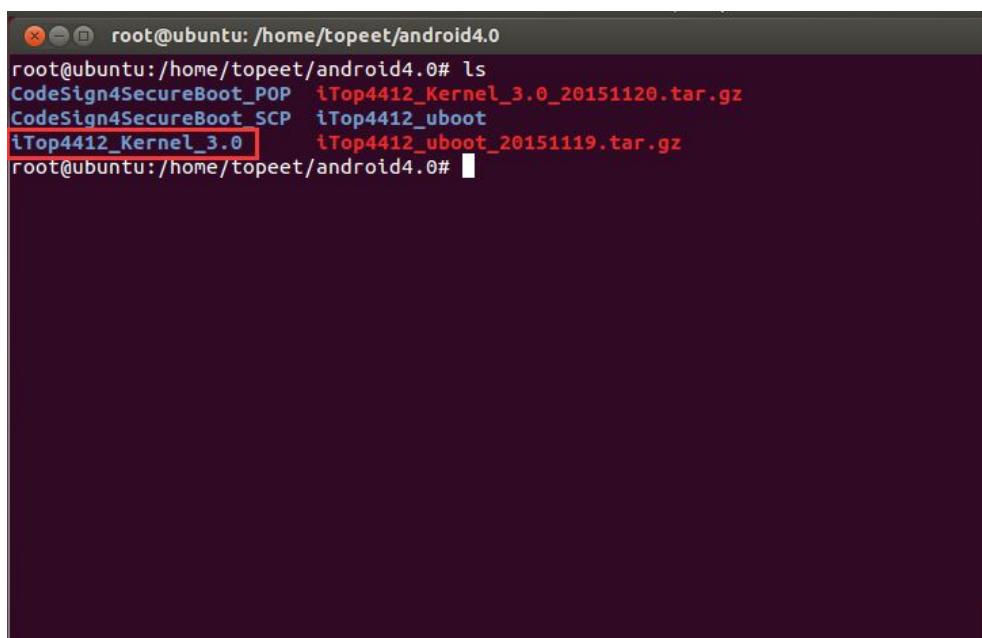
硬件分类	配置文件
核心板 SCP 1G 或者 2G 内存	config_for_linux_scp_elite
核心板 POP 1G 内存	config_for_linux_pop_elite
核心板 POP 2G 内存	config_for_linux_pop2G_elite

## 6.2.2 编译生成内核镜像举例

这里以 SCP 1G 核心板为例编译 zImage 内核镜像,那么配置文件为 “config\_for\_linux\_scp\_elite” 。

将光盘 “06\_源码\_uboot 和 kernel” 目录下的压缩包

“iTTop4412\_Kernel\_3.0\_xxx.tar.gz” 拷贝到 Ubuntu , 然后解压 , 得到文件夹 “iTTop4412\_Kernel\_3.0 ” , 如下图所示。



```
root@ubuntu:/home/topeet/android4.0
root@ubuntu:/home/topeet/android4.0# ls
CodeSign4SecureBoot_POP  iTTop4412_Kernel_3.0_20151120.tar.gz
CodeSign4SecureBoot_SCP  iTTop4412_uboot
iTTop4412_Kernel_3.0    iTTop4412_uboot_20151119.tar.gz
root@ubuntu:/home/topeet/android4.0#
```

进入文件夹 “iT0p4412\_Kernel\_3.0” , 使用命令

“cp config\_for\_linux\_scp\_elite .config” 覆盖自带的配置文件 , 如下图所示。

```
root@ubuntu:/home/topeet/android4.0# cd iT0p4412_Kernel_3.0
root@ubuntu:/home/topeet/android4.0/iT0p4412_Kernel_3.0# cp config_for_linux_scp
_elite .config
root@ubuntu:/home/topeet/android4.0/iT0p4412_Kernel_3.0#
```

然后使用编译命令 “make zImage” , 如下图所示。

```
root@ubuntu:/home/topeet/android4.0# cd iT0p4412_Kernel_3.0
root@ubuntu:/home/topeet/android4.0/iT0p4412_Kernel_3.0# cp config_for_linux_scp
_elite .config
root@ubuntu:/home/topeet/android4.0/iT0p4412_Kernel_3.0# make zImage
```

编译中 , 如下图所示。

```
root@ubuntu:/home/topeet/android4.0# cd iTop4412_Kernel_3.0
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0# cp config_for_linux_scp_elite .config
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0# make zImage
scripts/kconfig/conf --silentoldconfig Kconfig
  CHK  include/linux/version.h
  CHK  include/generated/utsrelease.h
make[1]: `include/generated/mach-types.h' is up to date.
  CALL  scripts/checksyscalls.sh
  CHK  include/generated/compile.h
  CC   arch/arm/kernel/setup.o
  LD   arch/arm/kernel/built-in.o
```

编译完成，如下图所示。

```
CC      init/version.o
LD      init/built-in.o
LD      .tmp_vmlinux1
KSYM   .tmp_kallsyms1.S
AS      .tmp_kallsyms1.o
LD      .tmp_vmlinux2
KSYM   .tmp_kallsyms2.S
AS      .tmp_kallsyms2.o
LD      vmlinux
SYSMAP System.map
SYSMAP .tmp_System.map
OBJCOPY arch/arm/boot/Image
Kernel: arch/arm/boot/Image is ready
AS      arch/arm/boot/compressed/head.o
GZIP   arch/arm/boot/compressed/piggy.gzip
AS      arch/arm/boot/compressed/piggy.gzip.o
CC      arch/arm/boot/compressed/misc.o
CC      arch/arm/boot/compressed/decompress.o
SHIPPED arch/arm/boot/compressed/libifuncs.S
AS      arch/arm/boot/compressed/libifuncs.o
LD      arch/arm/boot/compressed/vmlinux
OBJCOPY arch/arm/boot/zImage
Kernel: arch/arm/boot/zImage is ready
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0#
```

文件夹“iTop4412\_Kernel\_3.0”下的“arch”-->“arm”-->“boot”会生成镜像文件“zImage”，这个zImage镜像可以给 **SCP 1G 和 SCP 2G** 的核心板使用，如下图所示。

```
LD      .tmp_vmlinux1
KSYM   .tmp_kallsyms1.S
AS     .tmp_kallsyms1.o
LD      .tmp_vmlinux2
KSYM   .tmp_kallsyms2.S
AS     .tmp_kallsyms2.o
LD      vmlinux
SYSMAP System.map
SYSMAP .tmp_System.map
OBJCOPY arch/arm/boot/Image
Kernel: arch/arm/boot/Image is ready
AS      arch/arm/boot/compressed/head.o
GZIP   arch/arm/boot/compressed/piggy.gzip
AS      arch/arm/boot/compressed/piggy.gzip.o
CC      arch/arm/boot/compressed/misc.o
CC      arch/arm/boot/compressed/decompress.o
SHIPPED arch/arm/boot/compressed/lib1funcs.S
AS      arch/arm/boot/compressed/lib1funcs.o
LD      arch/arm/boot/compressed/vmlinux
OBJCOPY arch/arm/boot/zImage
Kernel: arch/arm/boot/zImage is ready
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0# ls arch/arm/boot/
bootp compressed Image install.sh Makefile zImage
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0#
```

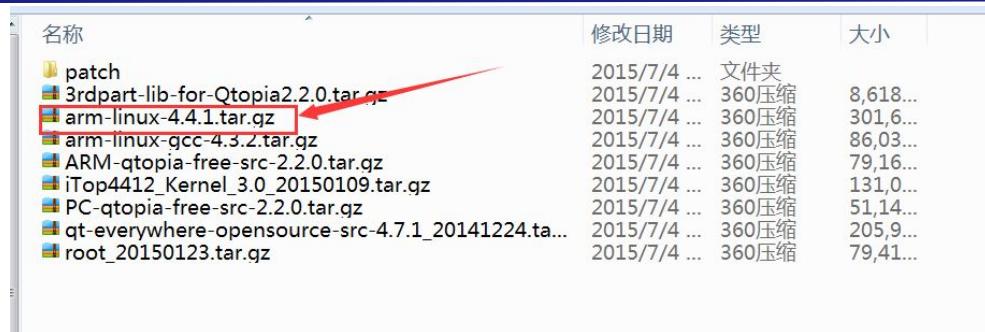
## 6.3 Qtopia2.2.0 编译的环境以及编译

针对 Qt 文件系统，迅为电子在 iTOP-4412 开发板上移植的是 Qtopia2.2.0 版本和 Qte4.7.1 版本，用户在参照本章节后编译后的文件图形界面是 Qtopia2.2.0 版本。Qte4.7.1 的编译方法则在第七章。

如果用户使用的是“搭建好的 Ubuntu 镜像”，则只需要修改一下环境变量。

### 6.3.1 编译器和基本库文件的安装

Qtopia2.2.0 文件系统的编译器和 Android4.0.3 的编译器不一样，Qtopia2.2.0 的编译器包含在用户光盘“08\_源码\_QtE 以及 qtopia2.2.0 文件系统”文件夹的压缩包“arm-linux-4.4.1.tar.g”中，如下图所示。



名称	修改日期	类型	大小
patch	2015/7/4 ...	文件夹	
3rdpart-lib-for-Qtopia2.2.0.tar.gz	2015/7/4 ...	360压缩	8,618...
<b>arm-linux-4.4.1.tar.gz</b>	2015/7/4 ...	360压缩	301,6...
arm-linux-gcc-4.3.2.tar.gz	2015/7/4 ...	360压缩	86,03...
ARM-qtopia-free-src-2.2.0.tar.gz	2015/7/4 ...	360压缩	79,16...
iTop4412_Kernel_3.0_20150109.tar.gz	2015/7/4 ...	360压缩	131,0...
PC-qtopia-free-src-2.2.0.tar.gz	2015/7/4 ...	360压缩	51,14...
qt-everywhere-opensource-src-4.7.1_20141224.t...	2015/7/4 ...	360压缩	205,9...
root_20150123.tar.gz	2015/7/4 ...	360压缩	79,41...

将压缩包拷贝到 Ubuntu 系统下，如下图所示。

```
root@ubuntu:~# ls /usr/local/arm/
arm-2009q3 arm-2009q3.tar.bz2 arm-linux-4.4.1.tar.gz
root@ubuntu:~#
```

接着将压缩包解压到 Ubuntu 系统的文件夹 “usr” --> “local” --> “arm” 下，进入 “/usr/local/arm” 目录使用解压命令 “tar -vxf arm-linux-4.4.1.tar.gz” 解压 “arm-linux-4.4.1.tar.gz” ，如下图所示。

```
root@ubuntu:~# ls /usr/local/arm/
arm-2009q3 arm-2009q3.tar.bz2 arm-linux-4.4.1.tar.gz
root@ubuntu:~# cd /usr/local/arm/
root@ubuntu:/usr/local/arm# tar -vxf arm-linux-4.4.1.tar.gz
```

如下图所示，解压完成，生成了文件夹 “4.4.1” 文件夹。

```
4.4.1/bin/arm-linux-gcc-4.4.1
4.4.1/bin/arm-none-linux-gnueabi-ar
4.4.1/bin/arm-none-linux-gnueabi-addr2line
4.4.1/bin/arm-none-linux-gnueabi-ld
4.4.1/bin/arm-linux-gcc
4.4.1/bin/arm-linux-sprite
root@ubuntu:/usr/local/arm# ls
4.4.1 arm-2009q3 arm-2009q3.tar.bz2 arm-linux-4.4.1.tar.gz
root@ubuntu:/usr/local/arm#
```

然后安装 X11 的 SDK 库，具体操作如下：

执行命令 “apt-get install libx11-dev libxext-dev libxtst-dev” ，如下图所示。

```
root@ubuntu:/usr/local/arm# ls
4.4.1 arm-2009q3 arm-2009q3.tar.bz2 arm-linux-4.4.1.tar.gz
root@ubuntu:/usr/local/arm# cd
root@ubuntu:~#
root@ubuntu:~# apt-get install libx11-dev libxext-dev libxtst-dev
```

安装库过程提示是否要继续，如下图所示，选择 “y” ，继续。

```
The following NEW packages will be installed:
  libxi-dev libxtst-dev x11proto-record-dev
The following packages will be upgraded:
  libxi6 libxtst6
2 upgraded, 3 newly installed, 0 to remove and 503 not upgraded.
Need to get 348 kB of archives.
After this operation, 997 kB of additional disk space will be used.
Do you want to continue [Y/n]? y
```

如下图所示，更新完成。

```
Selecting previously unselected package libxtst-dev.
Unpacking libxtst-dev (from .../libxtst-dev_2%3a1.2.0-4ubuntu0.1_amd64.deb) ...
Processing triggers for man-db ...
Setting up libxi6 (2:1.7.1.901-1ubuntu1~precise3) ...
Setting up libxtst6 (2:1.2.0-4ubuntu0.1) ...
Setting up libxi-dev (2:1.7.1.901-1ubuntu1~precise3) ...
Setting up x11proto-record-dev (1.14.1-2) ...
Setting up libxtst-dev (2:1.2.0-4ubuntu0.1) ...
Processing triggers for libc-bin ...
ldconfig deferred processing now taking place
root@ubuntu:~#
```

接着修改环境变量，如下图所示，在 root 目录下（使用 cd 命令之后就会回到 root 目录）使用命令 “vim .bashrc”

```
root@ubuntu:~#
root@ubuntu:~#
root@ubuntu:~# cd
root@ubuntu:~# vim .bashrc
```

使用 vim 编辑器打开环境变量文件 “.bashrc” 后，修改 Qtopia2.2.0 编译器的路径，添加

“export PATH=\$PATH:/usr/local/arm/4.4.1/bin”

到文件 “.bashrc” 的最后一行。然后注释掉其它编译器，例如下图所示的 arm-2009q3 编译器。

```
. ~/.bashrc
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
#if [ -f /etc/bash_completion ] && ! shopt -oq posix; then
#  . /etc/bash_completion
##fi
#export PATH=$PATH:/usr/local/arm/arm-2009q3/bin
export PATH=$PATH:/usr/local/arm/4.4.1/bin
-- INSERT --
```

101,43

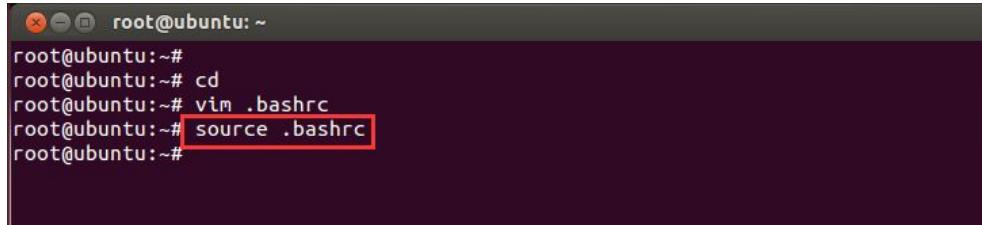
Bot

修改完成后保存退出 “.bashrc” 文件。



```
root@ubuntu:~#
root@ubuntu:~# cd
root@ubuntu:~# vim .bashrc
root@ubuntu:~#
```

更新一下环境变量，如下图所示，使用命令“source .bashrc”更新环境变量。

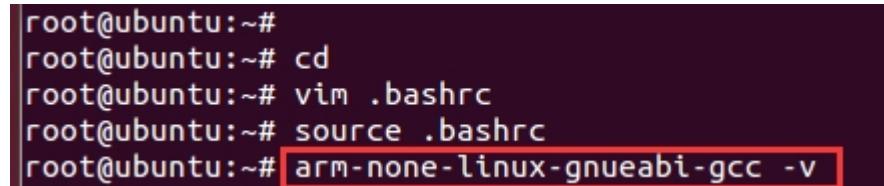


```
root@ubuntu:~#
root@ubuntu:~# cd
root@ubuntu:~# vim .bashrc
root@ubuntu:~# source .bashrc
root@ubuntu:~#
```

这里测试一下编译器是否正确安装，执行下命令

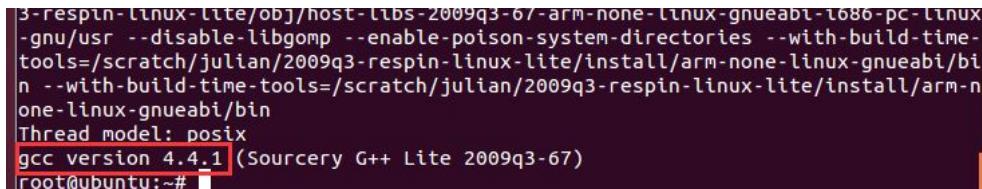
“arm-none-linux-gnueabi-gcc -v”

如下图所示。



```
root@ubuntu:~#
root@ubuntu:~# cd
root@ubuntu:~# vim .bashrc
root@ubuntu:~# source .bashrc
root@ubuntu:~# arm-none-linux-gnueabi-gcc -v
```

如下图所示，可以看到系统显示 arm-gcc 编译器的版本为“gcc version 4.4.1”。



```
3-respin-linux-lite/obj/host-libs-2009q3-67-arm-none-linux-gnueabi-i686-pc-linux
-gnu/usr --disable-libgomp --enable-poison-system-directories --with-build-time-
tools=/scratch/julian/2009q3-respin-linux-lite/install/arm-none-linux-gnueabi/bi
n --with-build-time-tools=/scratch/julian/2009q3-respin-linux-lite/install/arm-n
one-linux-gnueabi/bin
Thread model: posix
gcc version 4.4.1 (Sourcery G++ Lite 2009q3-67)
root@ubuntu:~#
```

大家注意一下，在前面搭建 Android4.0.3 编译环境的时候，其中提到了一步操作“降低 gcc 版本”，但是前面那个“gcc”是 x86 的编译器（通过命令#gcc -v 可以查看其版本）。这里用到的“gcc”编译器是 arm 编译器，它们是两个完全不同的编译器，大家不要弄混了。

### 6.3.2 Qtopia2.2.0 源文件和补丁文件

在 Ubuntu 环境中，“root”目录下新建文件夹“yizhi”，具体操作如下，在 Ubuntu 命令行中，执行下面命令：

```
cd /root
```

```
mkdir yizhi
```

这里需要注意的是，新建的文件夹一定要在这个“root”文件夹下建立，而且一定要使用“yizhi”这个名字。

如下图所示。

```
Thread model: posix
gcc version 4.4.1 (Sourcery G++ Lite 2009q3-67)
root@ubuntu:~# cd /root/
root@ubuntu:~# mkdir yizhi
root@ubuntu:~# ls
yizhi
root@ubuntu:~#
```

接着找到用户光盘“08\_源码\_QtE 以及 qtopia2.2.0 文件系统”文件夹下的压缩包“ARM-qtopia-free-src-2.2.0.tar.gz”如下图所示。

名称	修改日期	类型	大
patch	2015/7/4 ...	文件夹	
3rdpart-lib-for-Qtopia2.2.0.tar.gz	2015/7/4 ...	360压缩	8,6
arm-linux-4.4.1.tar.gz	2015/7/4 ...	360压缩	30
arm-linux-gcc-4.3.2.tar.gz	2015/7/4 ...	360压缩	86
ARM-qtopia-free-src-2.2.0.tar.gz	2015/7/4 ...	360压缩	79
iTop4412_Kernel_3.0_20150109.tar.gz	2015/7/4 ...	360压缩	13
PC-qtopia-free-src-2.2.0.tar.gz	2015/7/4 ...	360压缩	51
qt-everywhere-opensource-src-4.7.1_20141224.t...	2015/7/4 ...	360压缩	20
root_20150123.tar.gz	2015/7/4 ...	360压缩	79

将压缩包“ARM-qtopia-free-src-2.2.0.tar.gz”拷贝到前面新建的“yizhi”文件夹中，如下图所示。

```
root@ubuntu:~# cd /root/
root@ubuntu:~# mkdir yizhi
root@ubuntu:~# ls
yizhi
root@ubuntu:~# ls yizhi/
ARM-qtopia-free-src-2.2.0.tar.gz
root@ubuntu:~#
```

进入“yizhi”目录，使用命令“tar -vxf ARM-qtopia-free-src-2.2.0.tar.gz”解压压缩包，如下图所示。

```
root@ubuntu:~# ls
yizhi
root@ubuntu:~# ls yizhi/
ARM-qtopia-free-src-2.2.0.tar.gz
root@ubuntu:~# cd yizhi/
root@ubuntu:~/yizhi# tar -vxf ARM-qtopia-free-src-2.2.0.tar.gz
```

解压压缩包后得到源码文件“qtopia-free-src-2.2.0.tar.gz”和脚本文件“build”，如下图所示。

```
root@ubuntu:~# ls yizhi/
ARM-qtopia-free-src-2.2.0.tar.gz
root@ubuntu:~# cd yizhi/
root@ubuntu:~/yizhi# tar -vxf ARM-qtopia-free-src-2.2.0.tar.gz
build
qtopia-free-src-2.2.0.tar.gz
root@ubuntu:~/yizhi# ls
ARM-qtopia-free-src-2.2.0.tar.gz build qtopia-free-src-2.2.0.tar.gz
root@ubuntu:~/yizhi#
```

这里需要注意的是，用户光盘里面提供的 qtopia2.2.0 源文件是经过修改的，没有 bug 的代码，如果大家对 qtopia2.2.0 官网的源码感兴趣，可以在网盘下载源码（网盘中的源码是从 qt 官网下载的源代码）。但是假如用户在这里使用 qtopia2.2.0 官网的源码，在最后编译的时候会报很多错，这些错误需要额外的方法去排除，具体排除的方法可以参考后面的附录二。

用户光盘 “08\_源码\_QtE 以及 qtopia2.2.0 文件系统” --> “patch” 文件夹下的压缩包 “tslib.tar.gz” 是触摸的库文件，如下图所示。

名称	修改日期	类型	大小
libICE.so.6.3.0	2015/7/4 ...	0 文件	83 KB
libSM.so.6.0.1	2015/7/4 ...	1 文件	26 KB
libuuid.so.1.3.0	2015/7/4 ...	0 文件	14 KB
libXext.so.6.4.0	2015/7/4 ...	0 文件	55 KB
libXmu.so.6.2.0	2015/7/4 ...	0 文件	87 KB
libXt.so.6.0.0	2015/7/4 ...	0 文件	324 KB
tslib.tar.gz	2015/7/4 ...	360压缩	59 KB

将触摸的库文件拷贝到 Ubuntu，然后到 Ubuntu 系统的 “usr” --> “local” 文件夹下，使用命令 “tar -vxf tslib.tar.gz” 解压，如下图所示。

```
root@ubuntu:~#
root@ubuntu:~# cd /usr/local/arm/
root@ubuntu:/usr/local/arm# ls
4.4.1 arm-2009q3 arm-2009q3.tar.bz2 arm-linux-4.4.1.tar.gz tslib.tar.gz
root@ubuntu:/usr/local/arm# tar -vxf tslib.tar.gz
tslib/bin/
tslib/bin/ts_print_raw
tslib/bin/ts_calibrate
tslib/bin/ts_test
tslib/bin/ts_harvest
tslib/bin/ts_print
```

生成的文件夹 “tslib”，如下图所示。

```
tslib/lib/pkgconfig/tslib-0.0.pc
tslib/lib/libts.la
tslib/lib/libts-0.0.so.0
tslib/lib/libts-0.0.so.0.1.1
root@ubuntu:/usr/local/arm# ls
4.4.1 arm-2009q3.tar.bz2 tslib
arm-2009q3 arm-linux-4.4.1.tar.gz tslib.tar.gz
root@ubuntu:/usr/local/arm#
```

### 6.3.3 库文件和编译 Qtopia2.2.0

编译 Qtopia2.2.0 文件还需要一些额外的 6 个库文件，这 6 个库文件全部在用户光盘“08\_源码\_QtE 以及 qtopia2.2.0 文件系统” --> “patch” 文件夹下。这六个库文件分别是：libXext.so.6.4.0, libXmu.so.6.2.0, libSM.so.6.0.1, libICE.so.6.3.0, libXt.so.6.0.0, libuuid.so.1.3.0

如下图所示。

名称	修改日期	类型	大小
libICE.so.6.3.0	2015/7/4 ...	0 文件	83 KB
libSM.so.6.0.1	2015/7/4 ...	1 文件	26 KB
libuuid.so.1.3.0	2015/7/4 ...	0 文件	14 KB
libXext.so.6.4.0	2015/7/4 ...	0 文件	55 KB
libXmu.so.6.2.0	2015/7/4 ...	0 文件	87 KB
libXt.so.6.0.0	2015/7/4 ...	0 文件	324 KB
tslib.tar.gz	2015/7/4 ...	360压缩	59 KB

其中的 5 个库文件，包括 “libXext.so.6.4.0 ”、“ libXmu.so.6.2.0 ”、“ libSM.so.6.0.1 ”、“ libICE.so.6.3.0 ”、“ libXt.so.6.0.0” ，全部拷贝到 Ubuntu 系统的文件夹 “usr” --> “lib32” 下。

然后创建链接文件，具体操作如下，进入 Ubuntu 系统的文件夹 “usr” --> “lib32” 下，然后在 Ubuntu 命令行中，执行下面的命令：

ln -s libXext.so.6.4.0 libXext.so.6

创建链接文件 libXext.so.6

ln -s libXext.so.6 libXext.so

创建链接文件 libXext.so

ln -s libXmu.so.6.2.0 libXmu.so.6

创建链接文件 libXmu.so.6

ln -s libXmu.so.6 libXmu.so

创建链接文件 libXmu.so

ln -s libSM.so.6.0.1 libSM.so.6

创建链接文件 libSM.so.6

ln -s libSM.so.6 libSM.so

创建链接文件 libSM.so

ln -s libICE.so.6.3.0 libICE.so.6

创建链接文件 libICE.so.6

ln -s libICE.so.6 libICE.so

创建链接文件 libICE.so

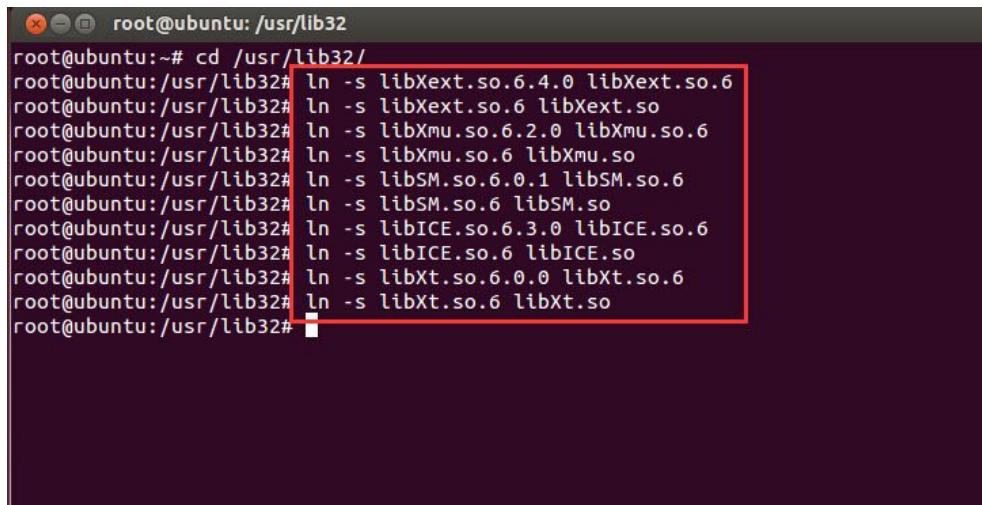
ln -s libXt.so.6.0.0 libXt.so.6

创建链接文件 libXt.so.6

ln -s libXt.so.6 libXt.so

创建链接文件 libXt.so

如下图所示，文件拷贝完成之后，执行创建链接的命令。



```
root@ubuntu:~# cd /usr/lib32/
root@ubuntu:/usr/lib32# ln -s libXext.so.6.4.0 libXext.so.6
root@ubuntu:/usr/lib32# ln -s libXext.so.6 libXext.so
root@ubuntu:/usr/lib32# ln -s libXmu.so.6.2.0 libXmu.so.6
root@ubuntu:/usr/lib32# ln -s libXmu.so.6 libXmu.so
root@ubuntu:/usr/lib32# ln -s libSM.so.6.0.1 libSM.so.6
root@ubuntu:/usr/lib32# ln -s libSM.so.6 libSM.so
root@ubuntu:/usr/lib32# ln -s libICE.so.6.3.0 libICE.so.6
root@ubuntu:/usr/lib32# ln -s libICE.so.6 libICE.so
root@ubuntu:/usr/lib32# ln -s libXt.so.6.0.0 libXt.so.6
root@ubuntu:/usr/lib32# ln -s libXt.so.6 libXt.so
```

拷贝剩下的文件 “libuuid.so.1.3.0” 到 Ubuntu 系统的 “lib32” 文件夹下，然后在

Ubuntu 命令行中，执行下面的命令：

ln -s libuuid.so.1.3.0 libuuid.so.1

创建链接文件 libuuid.so.1

ln -s libuuid.so.1 libuuid.so

创建链接文件 libuuid.so

如下图所示，文件拷贝完成之后，执行创建链接的命令。

```
root@ubuntu:/usr/lib32# cd /lib32
root@ubuntu:/lib32# ln -s libuuid.so.1.3.0 libuuid.so.1
root@ubuntu:/lib32# ln -s libuuid.so.1 libuuid.so
root@ubuntu:/lib32#
```

库文件全部处理完成后，接着就可以编译 Qtopia2.2.0 源码了，使用命令 “cd /root/yizhi” 进入 Qtopia2.2.0 源码文件夹，执行编译脚本命令 “./build” ，如下图所示。

```
root@ubuntu:/lib32# ln -s libuuid.so.1.3.0 libuuid.so.1
root@ubuntu:/lib32# ln -s libuuid.so.1 libuuid.so
root@ubuntu:/lib32# cd /root/yizhi/
root@ubuntu:~/yizhi# ./build
```

编译 qtopia2.2.0 源文件是一个比较漫长的过程。

编译完成后会在 Ubuntu 系统文件夹 “root” --> “yizhi” 下生成文件夹 “qtopia-free-2.2.0” ，这个文件夹就是编译好的 Qtopia2.2.0 文件，如下图所示。

```
qtopia/techno
make[7]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia/etc/themes/medi
alplayer/techno'
make[6]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia/etc/themes'
make[5]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia/etc/themes'
make[4]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src'
make[3]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src'
make[2]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src'
make[1]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia'
root@ubuntu:~/yizhi# ls
ARM-qtopia-free-src-2.2.0.tar.gz  qtopia-free-2.2.0
build                            qtopia-free-src-2.2.0.tar.gz
qtopia2.2.0Makelog
root@ubuntu:~/yizhi#
```

在编译好的 Qtopia2.2.0 文件夹下，我们需要接着处理一下字库文件。具体操作如下，在 Ubuntu 命令行中，执行下面的命令：

```
cp -r /root/yizhi/qtopia-free-2.2.0/qt2/lib/fonts/helvetica* /root/yizhi/qtopia-
free-2.2.0/qtopia/image/opt/Qtopia/lib/fonts/
```

如下图所示。

```
root@ubuntu:~/yizhi# ls
ARM-qtopia-free-src-2.2.0.tar.gz  qtopia-free-2.2.0
build                               qtopia-free-src-2.2.0.tar.gz
qtopia2.2.0Makelog
root@ubuntu:~/yizhi# cp -r /root/yizhi/qtopia-free-2.2.0/qt2/lib/fonts/helvetica
* ./root/yizhi/qtopia-free-2.2.0/qtopia/image/opt/Qtopia/lib/fonts/
```

然后我们把文件夹"Qtopia"拷贝到 Ubuntu 系统的 opt 文件夹下，具体操作如下，在 Ubuntu 命令行中，执行下面的命令：

```
cp -r /root/yizhi/qtopia-free-2.2.0/qtopia/image/opt/Qtopia /opt
```

如下图所示。

```
root@ubuntu:~/yizhi# ls
ARM-qtopia-free-src-2.2.0.tar.gz  qtopia-free-2.2.0
build                               qtopia-free-src-2.2.0.tar.gz
qtopia2.2.0Makelog
root@ubuntu:~/yizhi# cp -r /root/yizhi/qtopia-free-2.2.0/qt2/lib/fonts/helvetica
* ./root/yizhi/qtopia-free-2.2.0/qtopia/image/opt/Qtopia/lib/fonts/
root@ubuntu:~/yizhi# cp -r /root/yizhi/qtopia-free-2.2.0/qtopia/image/opt/Qtopia
./opt
root@ubuntu:~/yizhi#
```

### 6.3.4 第三方库文件

在前一小节中，编译的时候用到了第三方库文件“3rdpart-lib-for-Qtopia2.2.0.tar.gz”。由于这个库文件直接包含在提供的编译器压缩包“arm-linux-4.4.1.tar.gz”中，用户在前面 6.3.1 小节中，解压编译器压缩包的时候，库文件就已经直接解压到 Ubuntu 系统中了，所以在编译 QT 的时候，用户不用进行额外的处理就可以直接编译生成 QT 文件系统。

如果用户使用自己下载的编译器，那么就需要自己编译第三方库文件，第三方库文件具体的编译方法可以参考附录一。

### 6.3.5 生成 system.img

生成可以下载的 system.img 文件需要工具“mkimage”，这个工具在用户光盘“02\_编译器以及烧写工具”→“tools”文件夹下的压缩包“linux\_tools.tgz”中，如下图所示。



拷贝压缩包到 Ubuntu 系统的 “/” 目录下，注意目录是 “/” 。

```
root@ubuntu:~/yizhi# cd /
root@ubuntu:/# ls
bin  dev  initrd.img  lib64  media  proc  sbin  sys  var
boot  etc  lib   linux_tools.tgz  mnt  root  selinux  tmp  vmlinuz
cdrom  home  lib32  lost+found  opt  run  srv  usr
root@ubuntu:/#
```

进入 “/” 目录，然后使用命令 “tar -vxf linux\_tools.tgz” ，将压缩包解压

```
root@ubuntu:/# cd /
root@ubuntu:/# tar -vxf linux_tools.tgz
```

解压后如下图所示，在 “/usr/local/bin/” 目录下生成了两个文件。

```
root@ubuntu:/# ls /usr/local/bin/
make_ext4fs  mkimage
root@ubuntu:/#
```

使用命令 “cd /home/topeet/” 进入 topeet 目录，然后使用命令 “mkdir Linux+QT” 新建一个 “Linux+QT” 文件夹，如下图所示。

```
root@ubuntu:~# cd /home/topeet/
root@ubuntu:/home/topeet# mkdir Linux+QT
root@ubuntu:/home/topeet# ls
android4.0      Desktop    examples.desktop  Pictures   Videos
Android_JDK     Documents   Linux+QT        Public
Android_JDK.tar.bz2 Downloads  Music         Templates
root@ubuntu:/home/topeet#
```

找到用户光盘 “08\_源码\_QtE 以及 qtopia2.2.0 文件系统” 目录下的压缩包 “root.tar.gz” ，如下图所示。

名称	修改日期
patch	2015/7/4 ...
3rdpart-lib-for-Qtopia2.2.0.tar.gz	2015/7/4 ...
arm-linux-4.4.1.tar.gz	2015/7/4 ...
arm-linux-gcc-4.3.2.tar.gz	2015/7/4 ...
ARM-qtopia-free-src-2.2.0.tar.gz	2015/7/4 ...
iTop4412_Kernel_3.0_20150109.tar.gz	2015/7/4 ...
PC-qtopia-free-src-2.2.0.tar.gz	2015/7/4 ...
qt-everywhere-opensource-src-4.7.1_20141224.tar.gz	2015/7/4 ...
root_20150123.tar.gz	2015/7/4 ...

拷贝用户光盘 “linux” 目录下的压缩包 “root.tar.gz” 到新建的 “Linux+QT” 文件夹下，如下图所示。

```
root@ubuntu:/home/topeet#
root@ubuntu:/home/topeet# cd Linux+QT/
root@ubuntu:/home/topeet/Linux+QT# ls
root_20150123.tar.gz
root@ubuntu:/home/topeet/Linux+QT#
```

使用命令 “tar -vxf root\_20150123.tar.gz” 解压压缩包，如下图所示。

```
root@ubuntu:/home/topeet# cd Linux+QT/
root@ubuntu:/home/topeet/Linux+QT# ls
root_20150123.tar.gz
root@ubuntu:/home/topeet/Linux+QT# tar -vxf root_20150123.tar.gz
```

解压后会生成文件夹 “root” ，如下图所示。

```
root@root/Settings/Contacts.com
root@root/Settings/Beam.conf
root@root/Settings/MineSweep.conf
root@root/Settings/Categories.xml
root@root/Settings/handwriting.conf
root@root/dev/
root@ubuntu:/home/topeet/Linux+QT# ls
root@root_20150123.tar.gz
root@ubuntu:/home/topeet/Linux+QT#
```

然后把前面编译生成的文件夹 “Qtopia” 拷贝到解压出来的 “opt” 文件夹中，具体操作如下，在 Ubuntu 命令行中，执行下面的命令：

```
cp -r /root/yizhi/qtopia-free-2.2.0/qtopia/image/opt/Qtopia
/home/topeet/Linux+QT/root/opt
```

```
root@root/Settings/handwriting.conf
root@root/dev/
root@ubuntu:/home/topeet/Linux+QT# ls
root@root_20150123.tar.gz
root@ubuntu:/home/topeet/Linux+QT# cp -r /root/yizhi/qtopia-free-2.2.0/qtopia/image/Qtopia /home/topeet/Linux+QT/root/opt
```

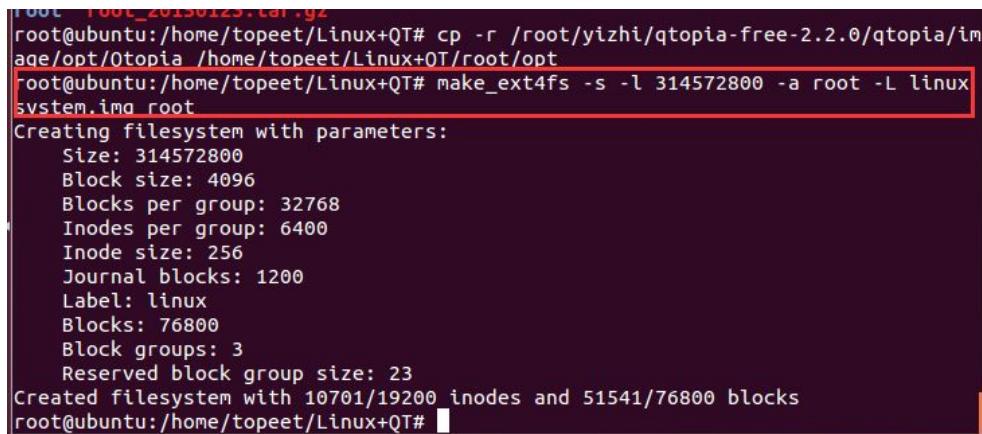
注意红色的 topeet 是用户文件夹，如果用户自己搭建环境，则需要替换成自己设置的用户名。

在执行上面的操作后，最后执行生成二进制文件的命令，在目录

“/home/topeet/Linux+QT” 中，使用命令

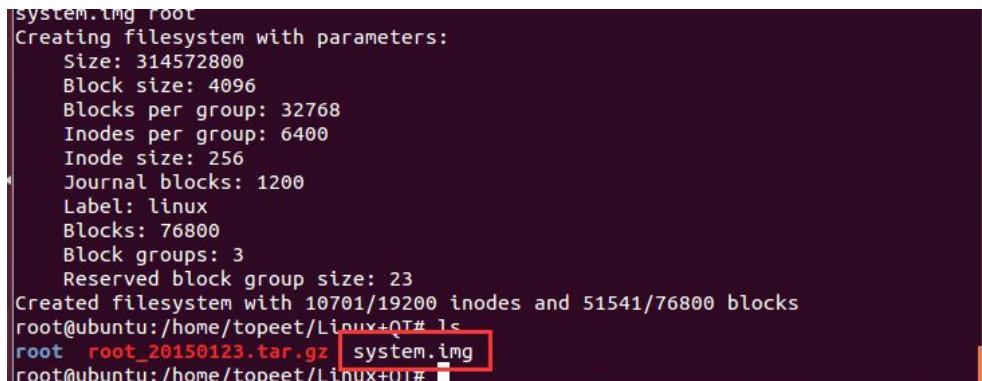
“make\_ext4fs -s -l 314572800 -a root -L linux system.img root”

如下图所示。



```
root@ubuntu:/home/topeet/Linux+QT# cp -r /root/yizhi/qtopia-free-2.2.0/qtopia/im  
age/opt/Otopia /home/topeet/Linux+QT/root/opt  
root@ubuntu:/home/topeet/Linux+QT# make_ext4fs -s -l 314572800 -a root -L linux  
system.img root  
Creating filesystem with parameters:  
Size: 314572800  
Block size: 4096  
Blocks per group: 32768  
Inodes per group: 6400  
Inode size: 256  
Journal blocks: 1200  
Label: linux  
Blocks: 76800  
Block groups: 3  
Reserved block group size: 23  
Created filesystem with 10701/19200 inodes and 51541/76800 blocks  
root@ubuntu:/home/topeet/Linux+QT#
```

执行这一步后，在 “Linux+QT” 文件夹中就生成了 “system.img” 文件，如下图所示。



```
system.img root  
Creating filesystem with parameters:  
Size: 314572800  
Block size: 4096  
Blocks per group: 32768  
Inodes per group: 6400  
Inode size: 256  
Journal blocks: 1200  
Label: linux  
Blocks: 76800  
Block groups: 3  
Reserved block group size: 23  
Created filesystem with 10701/19200 inodes and 51541/76800 blocks  
root@ubuntu:/home/topeet/Linux+QT# ls  
root root_20150123.tar.gz system.img  
root@ubuntu:/home/topeet/Linux+QT#
```

最后 Qtopia2.2.0 系统还需要一个镜像文件 “ramdisk-uboot.img” ，这个镜像文件是通用的，可以直接用编译好的镜像。这个镜像在用户光盘 “04\_镜像\_QT 文件系统” → “system” 中，如下图所示。



那么到这一步，Qtopia2.2.0 需要的全部镜像就都已经制作完成。

## 七 Qt/E4.7 的编译和使用说明

Qt4.7 的发布使 Qt 又有了更为长足的进步，通过官方提供的“changes”，可以看到如下改变：

1、首先是 QtQuick UI Tools 的正式发布，这可以说是 Qt4.7 里最重大的改变，再次强调了 Qt 发展方向，把 UI 设计交给专业的设计人员，功能模块和界面设计独立，减短软件开发的周期。

2、文档的巨大变化，可以说最近 Qt 的三个版本已经发生了翻天覆地的变化，在 Qt4.5 中，文档保持了 Qt 一贯的风格，而 Q4.6，则在内容组织和布局上进行了改变，Qt4.7 文档则是一次变革，不仅在布局和内容上更加人性化，新的 UI 风格也给人眼前一亮的感觉。

3、QtCreator 正式迈入 2.0 时代，在保持原来桌面开发内容的基础下，更考虑到未来移动开发的趋势，增加了 symbian 开发环境，而是，对于中国开发者来说，终于拥有了中文的界面，这是令许多开发者高兴的事情。

4、网络方面得到了加强，有新的类加入，弥补原来的众多不足。

5、质量和性能上得到提升，Qt4.7 中采用了最新的 webkit2.0 模块，同时，大量第三方相关库也得到了更新，并且补充了新的功能类和函数，保证 Qt 功能上的强大。

注意：Qt4.7.1 是 Qt4.7.0 一个 bug 修复版本，它保持对 Qt4.7.0 向前和向后的兼容（源代码和二进行制），iTOP-4412 使用 Qt4.7.1 版本。

Qt/E4.7.1 的 u-boot-iTOP-4412.bin、zImage 以及 ramdisk-uboot.img 和 Qtopia 通用，编译方法也一样。它们的区别是“Qtopia”带有一个桌面系统，“Qt/E4.7.1”只是一个强大的库。

Qt/E4.7.1 使用的编译器是交叉编译器编译工具“arm-linux-gcc-4.3.2.tar.gz”，编译器在用户光盘文件夹“linux”中。

Qt/E4.7.1 的源码压缩包 “qt-everywhere-opensource-src-4.7.1.tar.gz” 在用户光盘文件夹 “08\_源码\_QtE 以及 qtopia2.2.0 文件系统” 中。

## 7.1 Qt/E4.7.1 编译器的安装

Qt/E4.7.1 使用的编译器是交叉编译器编译工具 “arm-linux-gcc-4.3.2.tar.gz” , 编译器在用户光盘文件夹 “08\_源码\_QtE 以及 qtopia2.2.0 文件系统” 中。

将 “arm-linux-gcc-4.3.2.tar.gz” 解压到 Ubuntu 系统的文件夹 “/usr/local/arm” 中 , 解压后 , 如下图所示 :

```
root@ubuntu:/usr/local/arm# ls
4.3.2 ← arm-2009q3      arm-linux-4.4.1.tar.gz
4.4.1  arm-2009q3.tar.bz2 arm-linux-gcc-4.3.2.tar.gz
```

然后修改环境变量 , 修改环境变量前 , 确定是在 root 用户下 , 接着输入命令 “cd” , 确定修改的是 root 用户的环境变量 , 如下图。

```
root@ubuntu:/usr/local/arm# cd
root@ubuntu:~#
```

然后输入命令 “vim .bashrc” , 打开设置环境变量的文件 “.bashrc” 。

```
root@ubuntu:/usr/local/arm# ls
4.3.2  arm-2009q3          arm-linux-4.4.1.tar.gz
4.4.1  arm-2009q3.tar.bz2  arm-linux-gcc-4.3.2.tar.gz
root@ubuntu:/usr/local/arm# cd
root@ubuntu:~# vim .bashrc ←
```

输入如上图所示的命令 “vim .bashrc” 后 , 输入回车 , 进入 “.bashrc” 文件 , 然后进入最后行 , 如下图 , 将环境变量修改为 “export PATH=\$PATH:/usr/local/arm/4.3.2/bin”

```
#export PATH=$PATH:/usr/local/ndk/android-ndk-r8b
#export PATH=$PATH:/usr/local/arm/arm-2009q3/bin
#export PATH=$PATH:/usr/local/arm/4.4.1/bin
export PATH=$PATH:/usr/local/arm/4.3.2/bin
```

106,1

保存退出 , 然后更新一下环境变量 , 输入命令 “source .bashrc” , 如下图。

```
root@ubuntu:~# vim .bashrc
root@ubuntu:~# source .bashrc
root@ubuntu:~#
```

接着测试一下，编译器路径设置的对不对。如下图，在 Ubuntu 命令行中输入命令“arm”，然后按键盘“Tab”，出现编译器 “arm-none-linux-gnueabi-gcc-4.3.2” ，这就说明编译器路径设置正确。

```
root@ubuntu:~# arm
arm2hpdl
arm-linux-addr2line
arm-linux-ar
arm-linux-as
arm-linux-c++
arm-linux-c++filt
arm-linux-cpp
arm-linux-g++
arm-linux-gcc
arm-linux-gcc-4.3.2
arm-linux-gcov
arm-linux-gdb
arm-linux-gdbtui
arm-linux-gprof
arm-linux-ld
arm-linux-nm
arm-linux-objcopy
arm-linux-objdump
arm-none-linux-gnueabi-addr2line
arm-none-linux-gnueabi-ar
arm-none-linux-gnueabi-as
arm-none-linux-gnueabi-c++
arm-none-linux-gnueabi-c++filt
arm-none-linux-gnueabi-cpp
arm-none-linux-gnueabi-g++
arm-none-linux-gnueabi-qcc
arm-none-linux-gnueabi-gcc-4.3.2
arm-none-linux-gnueabi-gcov
arm-none-linux-gnueabi-gdb
arm-none-linux-gnueabi-gdbtui
arm-none-linux-gnueabi-gprof
arm-none-linux-gnueabi-ld
arm-none-linux-gnueabi-nm
arm-none-linux-gnueabi-objcopy
arm-none-linux-gnueabi-objdump
arm-none-linux-gnueabi-ranlib
```

## 7.2 Qt/E4.7.1 的编译

如果用户直接编译 QtE，没有编译第六章的 Linux-QT，那么这一步还需要参考使用手册“6.3.5 生成 system.img”小节，添加工具“mkimage”和解压出 root 文件夹。

将光盘文件夹“08\_源码\_QtE 以及 qtopia2.2.0 文件系统”中的 Qt/E4.7.1 源码压缩包“qt-everywhere-opensource-src-4.7.1.tar.gz”拷贝到 Ubuntu 的文件夹“root/yizhi”中，没有这个文件夹则可以新建一个。

然后，在 Ubuntu 命令行中输入解压命令“tar -vxf qt-everywhere-opensource-src-4.7.1.tar.gz”，解压后得到文件夹“qt-everywhere-opensource-src-4.7.1” ，如下图所示

```
root@ubuntu:~/yizhi# ls
3rdpart-lib-for-Qtopia2.2.0
3rdpart-lib-for-Qtopia2.2.0.tar.gz
ARM-qtopia-free-src-2.2.0.tar.gz
build
qte4.7.1Makelog
root@ubuntu:~/yizhi#                               qt-everywhere-opensource-src-4.7.1
                                         qt-everywhere-opensource-src-4.7.1.tar.gz
                                         qtopia2.2.0Makelog
                                         qtopia-free-2.2.0
                                         qtopia-free-src-2.2.0.tar.gz
```

进入 “qt-everywhere-opensource-src-4.7.1” 文件夹中，执行编译脚本 “./build-all” ，

**注意**这个命令有个点 “.”，如下图所示：

```
root@ubuntu:~/yizhi# cd qt-everywhere-opensource-src-4.7.1
root@ubuntu:~/yizhi/qt-everywhere-opensource-src-4.7.1# ls
build-all      qt-everywhere-opensource-src-4.7.1
qte4.7.1Makelog  qt-everywhere-opensource-src-4.7.1.tar.gz
root@ubuntu:~/yizhi/qt-everywhere-opensource-src-4.7.1# ./build-all
```

输入回车，如下图所示，开始编译，编译比较耗费时间，在一个小时左右。

```
qt-everywhere-opensource-src-4.7.1/doc/html/opengl-textures-textures-pro.html
qt-everywhere-opensource-src-4.7.1/doc/html/declarative-cppextensions-reference
xamples-default-main-cpp.html
qt-everywhere-opensource-src-4.7.1/doc/html/linguist-manager.html
qt-everywhere-opensource-src-4.7.1/doc/html/demos-declarative-webbrowser.html
qt-everywhere-opensource-src-4.7.1/doc/html/itemviews-addressbook-adddialog-cpp
html
qt-everywhere-opensource-src-4.7.1/doc/html/qpicture-qt3.html
```

编译完成后，如下图，进入 “/opt” 目录，可以看到编译生成的 “qt-4.7.1” 文件夹。

```
root@ubuntu:~/yizhi/qt-everywhere-opensource-src-4.7.1#
root@ubuntu:~/yizhi/qt-everywhere-opensource-src-4.7.1#
root@ubuntu:~/yizhi/qt-everywhere-opensource-src-4.7.1#
root@ubuntu:~/yizhi/qt-everywhere-opensource-src-4.7.1# cd /opt/
root@ubuntu:/opt# ls
qt-4.7.1 ←
root@ubuntu:/opt#
```

进入文件夹 “/home/topeet/Linux+QT/root/opt”（红色的目录 topeet 根据用户实际建立的文件夹调整），然后将 “qt-4.7.1” 文件夹拷贝到该目录下，如下图。红色框中的 “qt-4.7.1” 文件夹是 Qt/E4.7，蓝色框中的 “Qtopia” 就是第六章编译得到的 “Qtopia” 文件系统。

```
root@ubuntu:~# cd /home/topeet/Linux+QT/root/opt/
root@ubuntu:/home/topeet/Linux+QT/root/opt# cp -r /opt/qt-4.7.1 ./
root@ubuntu:/home/topeet/Linux+QT/root/opt# ls
qt-4.7.1 Qtopia
root@ubuntu:/home/topeet/Linux+QT/root/opt#
```

如下图，进入文件夹 “/home/topeet/Linux+QT” 中，输入命令 “make\_ext4fs -s -l 314572800 -a root -L linux system.img root” ，编译生成二进制文件 “system.img” 。

```
root@ubuntu:/home/topeet/Linux+QT/root/opt# cd ../../
root@ubuntu:/home/topeet/Linux+QT# make_ext4fs -s -l 314572800 -a root -L linux
system.img root
Creating filesystem with parameters:
  Size: 314572800
  Block size: 4096
  Blocks per group: 32768
  Inodes per group: 6400
  Inode size: 256
  Journal blocks: 1200
  Label: linux
  Blocks: 76800
  Block groups: 3
  Reserved block group size: 23
Created filesystem with 10062/19200 inodes and 45588/76800 blocks
```

如下图，文件“system.img”就是Qt/E4.7的镜像。

```
root@ubuntu:/home/topeet/Linux+QT# ls
root  root_20140912.tar.gz  system.img
root@ubuntu:/home/topeet/Linux+QT#
```

其它三个文件和Qtopia文件系统对应的镜像一模一样。

那么到这一步，需要的全部镜像就都已经制作完成。

如果想要和光盘“04\_镜像\_QT文件系统”→“system”中的镜像“system.img”一模一样，只需要将编译Qtopia和编译Qt/E4.7得到的文件夹同时放到

“/home/topeet/Linux+QT/root/opt”目录中，然后再使用“make\_ext4fs -s -l 314572800 -a root -L linux system.img root”，编译可以得到相同的“system.img”。

## 7.3 Qt/E4.7 和 Qtopia 的切换

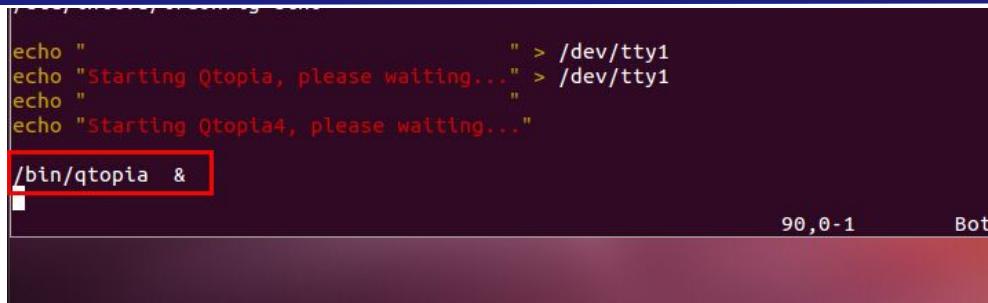
### 7.3.1 设置开发板优先运行的文件系统

源代码编译后，默认是运行Qtopia，下面讲一下如何直接运行Qt/E4.7。

这里需要修改“root/etc/init.d/rcS”文件。如下图所示，打开“root/etc/init.d/rcS”文件。

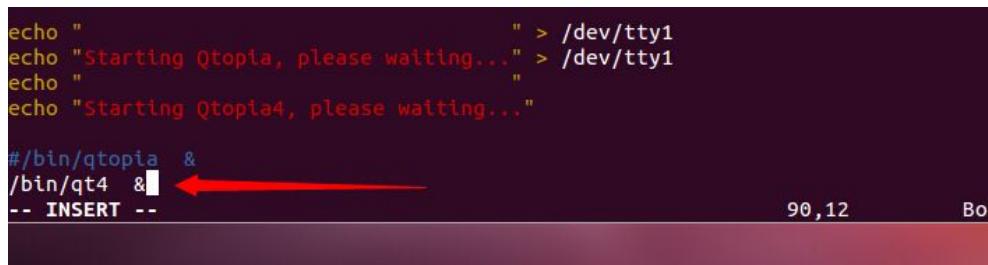
```
root@ubuntu:/home/topeet/Linux+QT# ls
root  root_20140912.tar.gz  system.img
root@ubuntu:/home/topeet/Linux+QT# vim root/etc/init.d/rcS
```

打开文件“rcS”后，进入文件中的最后一行，如下图所示，这是源码的状态，系统启动后，会默认运行“qtopia”。



```
echo "                                " > /dev/tty1
echo "Starting Qtopia, please waiting..." > /dev/tty1
echo "
echo "Starting Qtopia4, please waiting..."
```

如果要默认启动“qt-4.7.1”，则将修改为上图中的“qtopia”修改为“qt4”，如下图所示。注意这里的修改语法和格式一定要和源代码的一样。

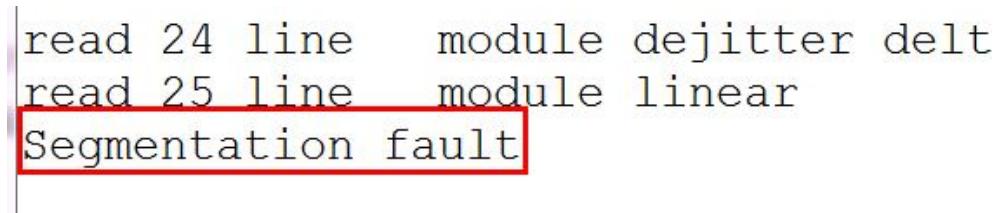


```
echo "                                " > /dev/tty1
echo "Starting Qtopia, please waiting..." > /dev/tty1
echo "
echo "Starting Qtopia4, please waiting..."
```

修改后，重新编译生成二进制文件，系统就会默认运行 Qt/E4.7。

### 7.3.2 Qt/E4.7 和 Qtopia2.2.0 的触摸校准

如果用户烧写镜像后，第一次运行正常，断电重启后，文件系统出现如下图的错误。这是由于开机后“校准文件为空”。



```
read 24 line      module dejitter delt
read 25 line      module linear
Segmentation fault
```

出现上图中的错误，则需要在超级终端中，输入命令“rm -rf /etc/point\*”，然后输入命令“reboot”重启开发板，如下图所示。

```
read 22 line    module pthres pmin=1
read 23 line    module variance delta=30
read 24 line    module dejitter delta=100
read 25 line    module linear
Segmentation fault
```

```
[root@iTOP-4412]# rm -rf /etc/point*
[root@iTOP-4412]# reboot ←
```

如上图，重启后就可以重新校准。

为了避免这个错误，用户需要在文件系统的校准阶段，按照屏幕界面“十字”标识，依次点击标识。这个过程就是触摸屏的校准阶段，确保校准的每一次都是按在系统指示的位置。

### 7.3.3 系统运行后 Qt/E4.7 和 Qtopia2.2.0 的切换

输入切换命令的时候如果已经打开过一个文件系统，则需要先关闭已启动文件系统的进程。

下面举例说明，如何关闭文件系统的进程。

如下图，已经运行了 Qtopia2.2.0 文件系统。

```
Took 1 samples...
Center : X = 400 Y = 523
-12.464966 -0.009929 1.003578
5.352295 1.018584 -0.024886
Calibration constants: -816904 -650 65770 350768 66753 -1630 65536

[root@iTOP-4412]# [ 19.671302] CPU3: Booted secondary processor
[ 19.675033] Switched to NOHZ mode on CPU #3

[root@iTOP-4412]#
[root@iTOP-4412]#
```

如下图，输入命令“ps”，查看系统进程。

```
[root@iTOP-4412]#
[root@iTOP-4412]# ps
 PID  USER      TIME      COMMAND
 1  root      0:02  {linuxrc} init
 2  root      0:00  [kthreadd]
 3  root      0:00  [ksoftirqd/0]
 4  root      0:00  [kworker/0:0]
 5  root      0:00  [kworker/u:0]
```

如下图，在超级终端中找出和 Qtopia2.2.0 文件系统相关的 ID 号，这里的进程 ID 号是 935、1027 以及 1028。

```
ps aux | grep /opt/Qtopia/bin/qpe
935 root      0:05 /opt/Qtopia/bin/qpe
936 root      0:00 -/bin/sh
965 root      0:00 [flush-179:0]
1027 root     0:00 /opt/Qtopia/bin/qss
1028 root     0:00 /opt/Qtopia/bin/quicklauncher
1067 root     0:00 [sh]
```

如下图，使用 kill 命令将 Qtopia2.2.0 的进程关掉，

```
[root@iTOP-4412]# kill 935
[root@iTOP-4412]# kill 1027
[root@iTOP-4412]# kill 1028
```

然后输入切换命令 “qt4” ，就可以切换到 Qt/E4.7。

Qt/E4.7 文件系统启动后，再切换到 Qtopia2.2.0，也是使用和上面类似的方法，这里就不再重复讲解了。

## 7.4 QtE 库的编译配置选项简介

在 QtE 库编译的时候，可以根据需求来配置，迅为提供的是已经配置好的。用户如果有其他功能的需求，可以自行配置。

下面给大家简单介绍一下 QtE 库编译时的配置。

如下图，进入 QtE4.7 的库文件源码目录。

```
root@ubuntu:~/yizhi/qt-everywhere-opensource-src-4.7.1# ls
build-all          qt-everywhere-opensource-src-4.7.1
qte4.7.1Makelog   qt-everywhere-opensource-src-4.7.1.tar.gz
root@ubuntu:~/yizhi/qt-everywhere-opensource-src-4.7.1#
```

打开 “build-all” 文件。如下图，可以看到 “./configure” 后面的配置，通过这些配置可以确定编译出来库的属性。

```
#/bin/bash

export PATH=/usr/local/arm/4.3.2/bin:$PATH
export PKG_CONFIG_PREFIX=$TOOLCHAIN/arm-none-linux-gnueabi
export TB_CC_PREFIX=arm-linux-
export TOOLCHAIN=/usr/local/arm/4.3.2

rm -fr qt-everywhere-opensource-src-4.7.1
rm -fr /opt/qt-4.7.1

tar xfvz qt-everywhere-opensource-src-4.7.1.tar.gz

cd qt-everywhere-opensource-src-4.7.1

echo yes | ./configure -opensource -embedded arm -xplatform qws/linux-arm-g++ -no-webkit -qt-libtiff -qt-libmng -qt-mouse-tslib -qt-mouse-pc -no-mouse-linuxtp -prefix /opt/qt-4.7.1 -I /usr/local/tslib/include -L /usr/local/tslib/lib

make 2>&1 | tee ../qte4.7.1Makelog && make install

~ "build-all" 20L, 631C 18,0-1 All
```

通过 “./configure --help” 可以查看每一个配置的具体含义。

```
everywhere-opensource-src-4.7.1
root@ubuntu:~/yizhi/qt-everywhere-opensource-src-4.7.1/qt-everywhere-opensource-src-4.7.1# ls
bin          doc          lib          qmake
changes-4.7.1 examples      LICENSE.FDL  README
config.tests imports      LICENSE.GPL3  src
configure     include      LICENSE.LGPL  templates
configure.exe INSTALL     mkspecs    tools
demos        LGPL_EXCEPTION.txt projects.pro translations
root@ubuntu:~/yizhi/qt-everywhere-opensource-src-4.7.1/qt-everywhere-opensource-src-4.7.1# ./configure --help
```

如下图，有大量的可配置选项。

```
-no-fontconfig ..... Do not compile FontConfig (anti-alias
ed font) support.
* -fontconfig ..... Compile FontConfig support.
          Requires fontconfig/fontconfig.h, lib
fontconfig,
          freetype.h and libfreetype.

      -no-xinput ..... Do not compile Xinput support.
* -xinput ..... Compile Xinput support. This also ena
bled tablet support
          which requires IRIX with wacom.h and
libXi or
          XFree86 with X11/extensions/XInput.h
and libXi.

      -no-xkb ..... Do not compile XKB (X KeyBoard extens
ion) support.
* -xkb ..... Compile XKB support.

      -no-glib ..... Do not compile Glib support.
+ -glib ..... Compile Glib support.

root@ubuntu:~/yizhi/qt-everywhere-opensource-src-4.7.1/qt-eve
rywhere-opensource-src-4.7.1# █
```

下面给大家简要介绍使用到的配置参数。

-opensource 自由版本（免费的）

-embedded arm 嵌入式 arm 架构

-xplatform qws/linux-arm-g++ 交叉编译的目标平台

-no-webkit 去掉 webkit 插件

-qt-libtiff 支持 tiff 插件

-qt-libmng 支持 mng 插件

-qt-mouse-tslib , -qt-mouse-pc , -no-mouse-linuxtp 触摸相关

-I ... Add an explicit include path , 添加路径，这里包含了

“/usr/local/tslib/include”

-L Add an explicit library path 添加库路径，这里包含了 “/usr/local/tslib/lib”

其它常用参数，例如 “-verbose” 可以打印每一步的信息，编译查找编译过程中的问题； “-static” 可以将库文件编译成静态，通过静态库移植的应用，不用将 QtE 库下载到开发板上。

还有上面的 “/usr/local/tslib” 则是迅为电子移植的触摸库文件。

Qt/E 的另外一个特征是拥有大量的库文件，跨平台，还可以很容易移植已有的第三方库文件。

## 八 基于 Linux-C 的测试程序

iTOP-4412 开发板可以运行的文件系统很多，在具体的文件系统上实现特定功能前，可以使用 Linux-C 程序来测试硬件以及驱动。而且这些程序很容易移植到 Android、Qt/E 以及最小文件系统上。

特别提醒：以前只接触过单片机而没有操作系统经验的用户，Linux-C 程序是跨平台的，只要按照下面介绍的方法去编译，就可以将 Linux-C 的程序和 Android 系统一起运行，使用 Linux-C 的程序测试我们关注的内容。本质上，我们可以这样理解，Android 只是一个大的文件而已，以下面第一个 helloworld 为例，Linux 内核上运行着两个程序“helloworld”+“Android”。

### 8.1 测试程序的编译和运行

#### 8.1.1 编译环境的设置

C 程序的应用程序在 Android 上运行，使用的编译器是 gcc4.4.1。编译器的安装方法参考第五章。

如下图所示，修改环境变量。

```
#export PATH=$PATH:/usr/local/ndk/android-ndk-r8b  
export PATH=$PATH:/usr/local/arm/arm-2009q3/bin  
#export PATH=$PATH:/usr/local/arm/4.1.1/bin  
#export PATH=$PATH:/usr/local/arm/4.3.2/bin
```

修改完之后，更新一下环境变量，如下图。

```
root@ubuntu:~# vim .bashrc  
root@ubuntu:~# source .bashrc
```

如下图所示，输入“arm”，然后按“TAB”键，会显示后面需要用到的编译器“arm-none-linux-gnueabi-gcc-4.4.1”。

```
root@ubuntu:~# arm
arm2hdpl
arm-linux-addr2line
arm-linux-ar
arm-linux-as
arm-linux-c+
arm-linux-c++filt
arm-linux-cpp
arm-linux-g++
arm-linux-gcc
arm-linux-gcc-4.3.2
arm-linux-gcc-4.4.1
arm-none-linux-gnueabi-addr2line
arm-none-linux-gnueabi-ar
arm-none-linux-gnueabi-as
arm-none-linux-gnueabi-c+
arm-none-linux-gnueabi-c++filt
arm-none-linux-gnueabi-cpp
arm-none-linux-gnueabi-g++
arm-none-linux-gnueabi-gcc
arm-none-linux-gnueabi-gcc-4.3.2
arm-none-linux-gnueabi-gcc-4.4.1
arm-none-linux-gnueabi-ncov
```

### 8.1.2 编译 helloworld

程序 helloworld.c 的源码如下：

```
#include <stdio.h>

int main(void)
{
    printf("Hello World!\n");
}
```

编译 helloworld 程序，输入命令“arm-none-linux-gnueabi-gcc-4.4.1 -o helloworld helloworld.c -static”，如下图所示。

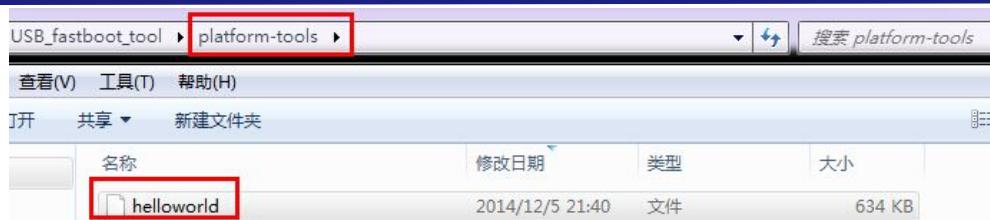
```
root@ubuntu:/home/topeet/Android-c# arm-none-linux-gnueabi-gcc-4.4.1 -o helloworld helloworld.c -static
```

如下图，生成可执行文件 helloworld。

```
root@ubuntu:/home/topeet/Android-c# ls
helloworld  helloworld.c
```

### 8.1.3 上传 helloworld 到开发板

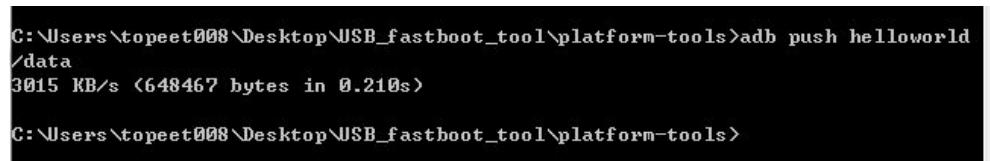
将可执行文件 helloworld 拷贝到 fastboot 烧写目录中，如下图所示，这个目录是烧写 fastboot 工具所在的目录。fastboot 工具的使用方法参考 3.6 小节。



开发板的 Android 系统运行稳定后，将 OTG 接口和电脑的 USB 连接，打开“USB\_fastboot\_tool\platform-tools”目录中的“cmd.exe”，如下图所示。



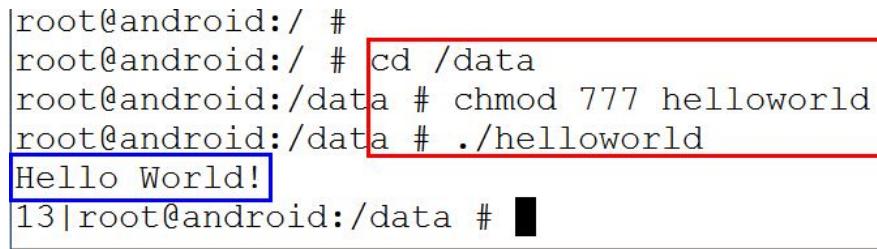
如下图所示，输入命令“adb push helloworld /data”，将程序上传到开发板的“/data”目录中。



当然也可以通过手机助手、TF 卡或者 U 盘来上传可执行文件 helloworld。

### 8.1.4 修改程序权限和运行 helloworld

如下图所示，在超级终端中，输入“cd /data”进入"/data"目录，使用“chmod 777 helloworld”修改权限，最后输入命令“./helloworld”运行程序。超级终端中会打印出“Hello world！” ，表明程序运行成功。



## 8.2 Led 灯的测试

如下图，编译 leds 测试程序，在 Ubuntu 系统中，输入编译命令 “arm-none-linux-gnueabi-gcc-4.4.1 -o leds leds.c -static” ，生成 leds 可执行程序”

```
root@ubuntu:/home/topeet/Android-c# arm-none-linux-gnueabi-gcc-4.4.1 -o leds leds.c -static
root@ubuntu:/home/topeet/Android-c# ls
helloworld  helloworld.c  leds  leds.c
root@ubuntu:/home/topeet/Android-c#
```

如下图，上传文件到开发板的 “/data” 。在 cmd 命令行中，输入 adb 传文件的命令 “adb push leds /data” 。

```
C:\Users\topeet008\Desktop\USB_fastboot_tool\platform-tools>adb push leds /data
adb server is out of date. killing...
* daemon started successfully *
2433 KB/s (650317 bytes in 0.261s)

C:\Users\topeet008\Desktop\USB_fastboot_tool\platform-tools>
```

如下图，修改测试程序的权限。在超级终端中，输入命令 “cd /data” ,输入修改权限命令 “chmod 777 leds” 。

```
root@android:/ #
root@android:/ # cd /data
root@android:/data # chmod 777 leds
root@android:/data #
```

如下图，在超级终端中，输入运行命令 “./leds” 。

```
root@android:/data # ./leds
leds light on and[ 480.105792] LEDS_CTL DEBUG:Device Opened Success!
[ 480.110704] debug: leds_ioctl cmd is 0
[ 480.114107] debug: leds_ioctl cmd is 0
    off 5 times
open /dev/leds success
[ 481.118091] debug: leds_ioctl cmd is 1
[ 481.120458] debug: leds_ioctl cmd is 1
```

测试结果：两个 led 灯同时闪烁 10 次。

## 8.3 Buzzer 蜂鸣器的测试

如下图，编译 buzzer 测试程序，在 Ubuntu 系统中，输入编译命令 “arm-none-linux-gnueabi-gcc-4.4.1 -o buzzer buzzer.c -static” ，生成 buzzer 可执行程序。

```
root@ubuntu:/home/topeet/Android-c# arm-none-linux-gnueabi-gcc-4.4.1 -o buzzer buzzer.c -static
root@ubuntu:/home/topeet/Android-c# ls
buzzer  buzzer.c  helloworld  helloworld.c  leds  leds.c
root@ubuntu:/home/topeet/Android-c#
```

如下图，上传文件到开发板的 “/data” ，在 cmd 命令行中，输入 adb 传文件的命令 “adb push buzzer /data” 。

```
C:\Users\topeet008\Desktop\USB_fastboot_tool\platform-tools>adb push buzzer /data
2805 KB/s <649201 bytes in 0.226s>
C:\Users\topeet008\Desktop\USB_fastboot_tool\platform-tools>
```

如下图，修改测试程序的权限。在超级终端中，输入命令 “cd /data” ,输入修改权限命令 “chmod 777 buzzer” 。

```
root@android:/root # cd /data
root@android:/data # chmod 777 buzzer
root@android:/data #
```

如下图，在超级终端中，输入运行命令 “./buzzer” 。

```
root@android:/data # ./buzzer
buzzer light on a[ 2836.130686] itop4412_buzzer_ioctl: cmd = 1
nd off 5 times
open /dev/buzzer_ctl success
ioctl buzzer 2 times
[ 2837.134586] itop4412_buzzer_ioctl: cmd = 0
ioctl buzzer 1 ti[ 2838.137640] itop4412_buzzer_ioctl: cmd = 1
mes
[ 2839.141903] itop4412_buzzer_ioctl: cmd = 0
ioctl buzzer 0 ti[ 2840.144875] itop4412_buzzer_ioctl: cmd = 1
mes
[ 2841.149137] itop4412_buzzer_ioctl: cmd = 0
root@android:/data #
```

测试结果：蜂鸣器响 3 次。

## 8.4 ADC 数模转换的测试

如下图，编译 ADC 测试程序，在 Ubuntu 系统中，输入编译命令 “arm-none-linux-gnueabi-gcc-4.4.1 -o ADC ADC.c -static” ，生成 ADC 可执行程序。

```
root@ubuntu:/home/topeet/Android-c# arm-none-linux-gnueabi-gcc-4.4.1 -o ADC ADC.c -static
root@ubuntu:/home/topeet/Android-c# ls
ADC  ADC.c  buzzer  buzzer.c  helloworld  helloworld.c  leds  leds.c
root@ubuntu:/home/topeet/Android-c#
```

如下图，上传文件到开发板的 “/data” ，在 cmd 命令行中，输入 adb 传文件的命令 “adb push leds /data” 。

```
C:\Users\topeet008\Desktop\USB_fastboot_tool\platform-tools>adb push ADC /data
3819 KB/s (649228 bytes in 0.166s)

C:\Users\topeet008\Desktop\USB_fastboot_tool\platform-tools>
```

如下图，修改测试程序的权限。在超级终端中，输入命令 “cd /data” ,输入修改权限命令 “chmod 777 ADC” 。

```
root@android:/dev # cd /data
root@android:/data # chmod 777 ADC
root@android:/data #
```

如下图，在超级终端中，输入运行命令 “./ADC” 。

```
root@android:/data # ./ADC
adc ready! [ 3850.050748] adc opened
[ 3850.052434] sampling ...
[ 3850.054981] read 1: 0x86e
[ 3850.057668] sampling ...
[ 3850.060244] read 1: 0x864
[ 3850.062795] sampling ...
[ 3850.065446] read 1: 0x865
[ 3850.068002] sampling ...
[ 3850.070648] read 1: 0x86d
[ 3850.073211] sampling ...
[ 3850.075852] read 1: 0x86a
[ 3850.078418] sampling ...
[ 3850.081061] read 1: 0x869
[ 3850.083627] sampling ...
[ 3850.086268] read 1: 0x866
[ 3850.088835] sampling ...
[ 3850.091476] read 1: 0x86c
[ 3850.094042] sampling ...
[ 3850.096684] read 1: 0x86b
[ 3850.099251] sampling ...
[ 3850.102377] read 1: 0x869
[ 3850.104462] value = 0x869
[ 3850.107867] adc closed

open adc success!
res value is 1343
18|root@android:/data #
```

测试结果：如上图，读取到滑动变阻器的数值为 1343 欧姆。旋转滑动变阻器后，再次执行，电阻值会改变。

## 8.5 串口的测试

如下图，编译串口测试程序，在 Ubuntu 系统中，输入编译命令“arm-none-linux-gnueabi-gcc-4.4.1 -o Uart\_ttySAC3 Uart\_ttySAC3.c -static”，生成 Uart\_ttySAC3 可执行程序”

```
on screen
root@ubuntu:/home/topeet/Android-c# arm-none-linux-gnueabi-gcc-4.4.1 -o Uart_tty
SAC3 Uart_ttySAC3.c -static
root@ubuntu:/home/topeet/Android-c# ls
ADC  buzzer  helloworld  leds  Uart_ttySAC3
ADC.c  buzzer.c  helloworld.c  leds.c  Uart_ttySAC3.c
```

如下图，上传文件到开发板的 “/data” ，在 cmd 命令行中，输入 adb 传文件的命令 “adb push Uart\_ttySAC3/data” 。

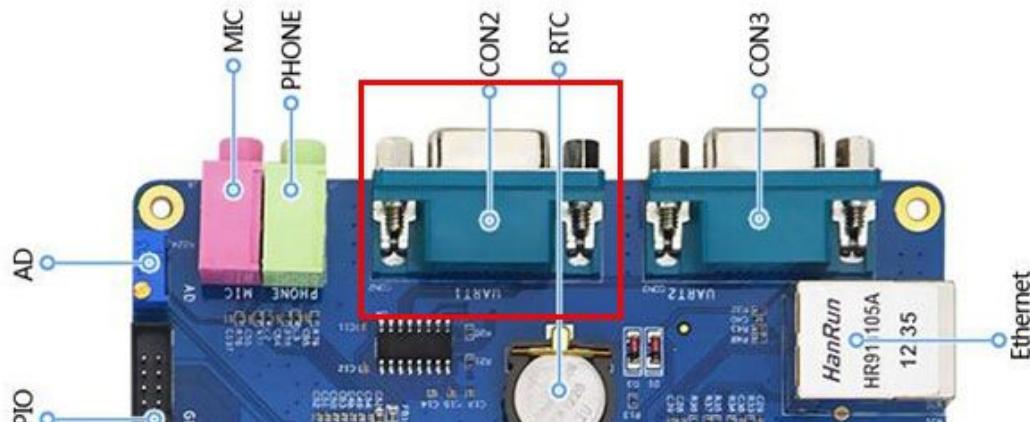
```
C:\Users\topeet008\Desktop\USB_fastboot_tool\platform-tools>adb push Uart_ttySAC
3 /data
2624 KB/s (650311 bytes in 0.242s)

C:\Users\topeet008\Desktop\USB_fastboot_tool\platform-tools>
```

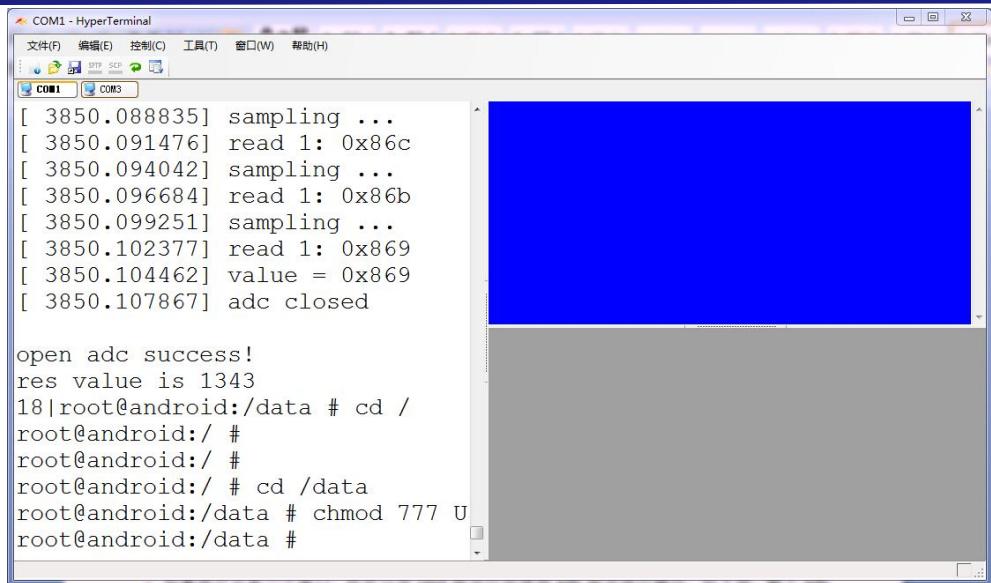
如下图，修改测试程序的权限。在超级终端中，输入命令 “cd /data” ,输入修改权限命令 “chmod 777 Uart\_ttySAC3” 。

```
root@android:/ # cd /data
root@android:/data # chmod 777 Uart_ttySAC3
root@android:/data #
```

如下图，将精英版红色框中靠近耳机接口的串口和上位机相连。该串口的参数和调试串口的参数相同。



如下图，在 PC 中连接精英版的 COM2，蓝色部分就是后面运行程序后要接收数据的串口；左边是调试输入命令串口控制端。

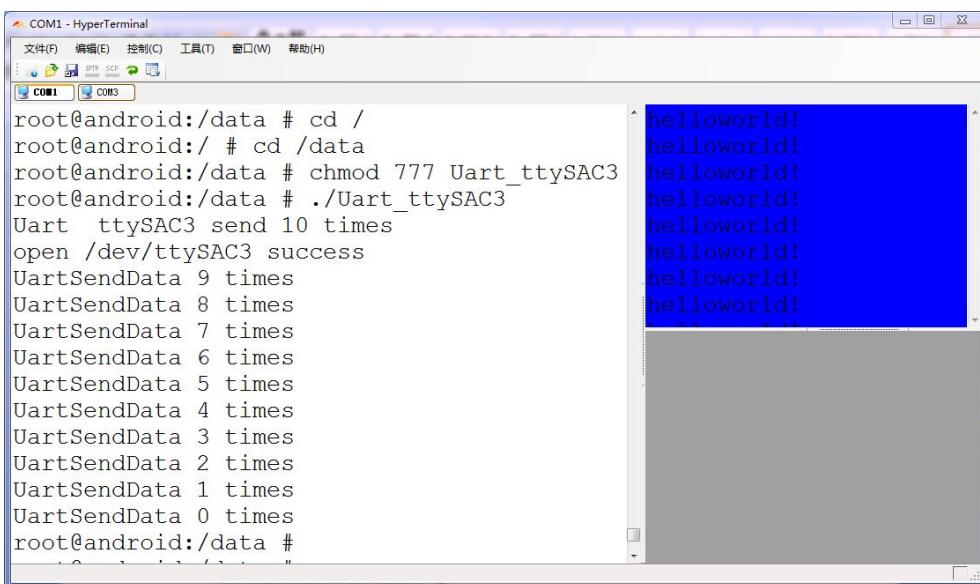


在超级终端中，修改权限命令 “chmod 777 Uart\_ttySAC3” , 输入运行命令

“./Uart\_ttySAC3” 。

```
root@android:/ # cd /data
root@android:/data # chmod 777 Uart_ttySAC3
root@android:/data #
```

如下图，修改 Uart\_ttySAC3 的权限 “chmod 777 Uart\_ttySAC3” ，执行运行命令 “./Uart\_ttySAC3” ,可以接收到数据 “helloworld ! ”



如果用户只有一个串口，也可以在 cmd 命令行中使用 adb 命令进行操作，如下图。输入 adb 命令 “adb shell” ，然后输入 “cd /data” 进入 “/data” 目录，输入 “chmod 777 Uart\_ttySAC3” 修改权限，然后执行测试程序 “./Uart\_ttySAC3” 。

```
C:\Users\topeet008\Desktop\USB_fastboot_tool\platform-tools>adb push Uart_ttySAC3 /data
2624 KB/s (650311 bytes in 0.242s)

C:\Users\topeet008\Desktop\USB_fastboot_tool\platform-tools>adb shell
root@android:/ # cd /data
cd /data
root@android:/data # chmod 777 Uart_ttySAC3
chmod 777 Uart_ttySAC3
root@android:/data # ./Uart_ttySAC3
```

## 8.6 全能版 485 的测试

本节测试 RS485 使用了两块全能版，用户可以根据实际情况修改测试的例子。

如下图，编译 485 测试程序，在 Ubuntu 系统中，输入编译命令 “arm-none-linux-gnueabi-gcc-4.4.1 -o test\_485 test\_485.c -static” ，生成 test\_485 可执行程序

```
root@ubuntu:/home/topeet/Android-c# arm-none-linux-gnueabi-gcc-4.4.1 -o test_485 test_485.c -static
root@ubuntu:/home/topeet/Android-c# ls
ADC  buzzer  helloworld  leds  test_485  Uart_ttySAC3
ADC.c  buzzer.c  helloworld.c  leds.c  test_485.c  Uart_ttySAC3.c
root@ubuntu:/home/topeet/Android-c#
```

如下图，上传文件到开发板的 “/data” ，在 cmd 命令行中，输入 adb 传文件的命令 “adb push test\_485 /data” 。

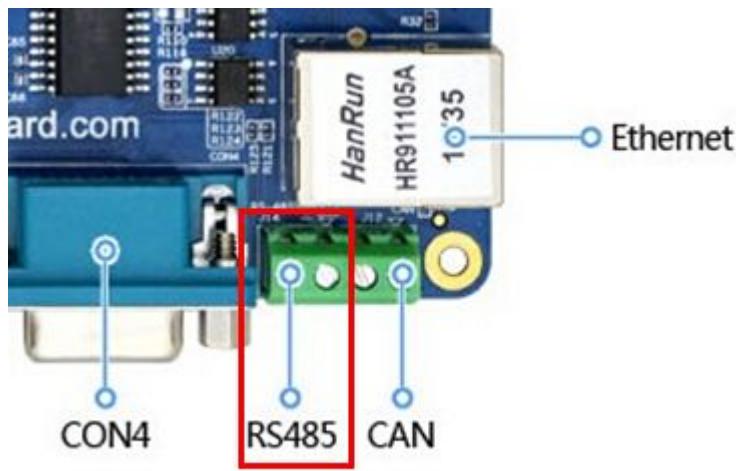
```
C:\Users\topeet008\Desktop\USB_fastboot_tool\platform-tools>adb push test_485 /data
adb server is out of date. killing...
* daemon started successfully *
2705 KB/s (656528 bytes in 0.237s)

C:\Users\topeet008\Desktop\USB_fastboot_tool\platform-tools>
```

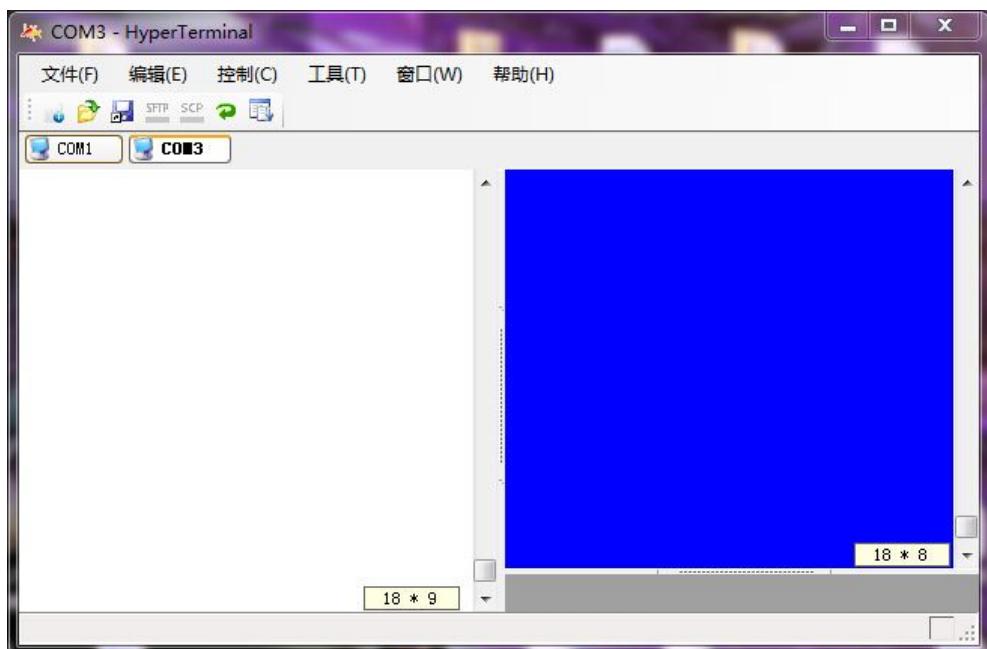
如下图，修改测试程序的权限。在超级终端中，输入命令 “cd /data” ,输入修改权限命令 “chmod 777 test\_485” 。

```
root@android:/ #
root@android:/ # cd /data
root@android:/data # chmod 777 test_485
root@android:/data #
```

这里的测试方法使用两块全能版，两块开发板的测试程序都是一样的。如下图，在网口旁的座子就是 RS485。485 有两根线，连接方式是 A 接 A，B 接 B，也就是不用交叉。



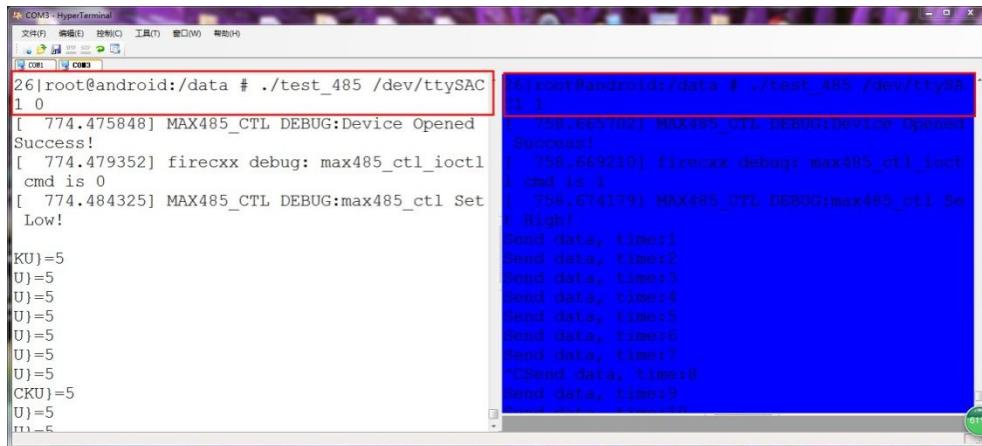
PC 机器上，如下图，两个串口分别连接开发板的调试串口 CON2。



如下图，开发板启动后，在超级终端中，输入运行命令 “./test\_485” 。

```
26|root@android:/data # ./test_485  
Usage: test_485 [uart port] [type]  
type: 0--recv, 1--send
```

如上图，可以看到，test\_485 程序需要输入参数，第一个参数是选择测试的串口，第二个是设置该口是发送数据还是接收数据。全能版连接控制 485 的串口是 ttySAC1，所以第一个参数为 ttySAC1。运行时命令分别为 “./test\_485 /dev/ttySAC1 0” 和 “./test\_485 dev/ttySAC1 0” 。运行后，超级终端打印信息，如下图所示，表明发送和接收都成功了。



# 九 定制 Linux 内核

Linux 内核看起来非常庞大，但是对于初学者以及开发应用程序的用户，根本不需要一开始就埋头于内核中，但是对于配置内核中的一些常用选项，并且编译出来下载到开发板上测试，则是必须掌握的，这是学习和掌握 Linux 的必行之路。

本章节的内容不会谈到具体的代码，但是这里需要提醒的是，初学者特别是以前只接触过单片机的用户，Linux 学习过程中不需要也不可能从代码开始学习，Linux 驱动的学习步骤是“01-基础知识”“02-搭建环境”“03-编译烧写”，然后就是本章节的“定制内核”。通过定制内核来加深对 Linux 的理解。

另外本章节针对的内核版本是“linux3.0.15”。

特别提醒：迅为的精英版和全能版内核中都带有以下提到的代码，但是精英版和全能版的硬件却有区别。例如：can 目前就只有全能版支持，在精英版提供的源码中驱动已经有了。

## 9.1 使用缺省文件配置和编译内核

为了方便用户能够方便的使用，我们针对不同的功能分别作了相应的内核配置文件。

首先必须了解的是迅为电子的内核有 SCP 和 POP 的分别，针对不同型号的核心板，内核有一些区别，不能混用了。SCP 和 POP 是不能混用的。

对应的 4412 开发板，解压内核源码包之后，如下图所示，可以看到红色方框中的内核缺省文件。

```
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0# ls
arch                                lib
binary                             MAINTAINERS
block                               Makefile
config_for_android_2M_pop          mm
config_for_android_2M_scp          modem.patch
config_for_android_pop2G_elite    Module.symvers
config_for_android_pop2G_super
config_for_android_pop_elite
config_for_android_pop_super
config_for_android_scp_elite
config_for_android_scp_super
config_for_linux_pop2G_elite
config_for_linux_pop2G_super
config_for_linux_pop_elite
config_for_linux_pop_super
config_for_linux_scp_elite
config_for_linux_scp_super
config_for_ubuntu_hdmi_pop
config_for_ubuntu_hdmi_pop_sd
config_for_ubuntu_hdmi_scp
config_for_ubuntu_hdmi_scp_sd

COPYING                            net
CREDITS                            pull_log.bat
crypto                             README
Documentation                      REPORTING-BUGS
drivers                           samples
firmware                          scripts
fs                                 security
include                           sound
init                               System.map
ipc                                tools
iTop4412_Kernel_3.0               usr
Kbuild                            virt
Kconfig                           vmlinuz
kernel                            vmlinuz.o
kernel_readme.txt
```

config\_for\_android\_xxx 支持 Android 的缺省文件

config\_for\_linux\_xxx 支持 Qt 的缺省文件

config\_for\_ubuntu\_xxx 支持 Ubuntu 的内核缺省文件

字符 xxx 含义：

scp : 表示支持 SCP 核心板

pop : 表示支持 POP 1G 核心板

pop2G : 表示支持 POP 2G 核心板

super : 表示支持全能版

elite : 表示支持精英版

hdmi : 表示在 Ubuntu 系统下，支持 HDMI 显示

sd : 表示在 Ubuntu 系统下，支持 sd 卡启动

2M : 表示支持 200 万摄像头（默认支持 500 万摄像头）

## 9.2 驱动程序源代码的位置

解压内核后，可以找到如下驱动。注意，部分驱动在缺省状态下不会编译。

设备	源码位置	设备名
蜂鸣器	drivers/char/itop4412_buzzer.c	/dev/buzzer_ctl

LED 驱动	drivers/char/itop4412_leds.c	/dev/leds
AD 数模转换	drivers/char/itop4412_adc.c	/dev/adc
485 驱动	drivers/char/max485_ctl.c	/dev/max485_ctl_pin
GPS 驱动	drivers/char/gps.c	/dev/gps
RTC 实时时钟	drivers/rtc/rtc-s3c.c	/dev/rtc*
串口 0-3	drivers/tty/serial/samsung.c	/dev/ttySAC*
i2c 总线 0-8	drivers/i2c/busses/i2c-s3c2410.c	/dev/i2c*
SPI 总线	drivers/spi/spi_s3c64xx.c	
can 驱动	net/can/af_can.c	
触摸屏驱动 ( TP )	drivers/input/touchscreen/ft5x06_ts.c	
开机画面 ( Log )	drivers/video/samsung/iTop-4412.h	
显卡驱动	drivers/video/samsung/s3cfb_wa101s.c	
LCD 背光	drivers/video/backlight/backlight.c	
HDMI_HPD	drivers/media/video/samsung/tvout/s5p_tvout_hpd.c	/dev/HPD
HDMI_Audio	drivers/media/video/samsung/tvout/hw_if/mixer.c	
SD/eMMC	drivers/mmc/host/sdhci-s3c.c	
U 盘驱动	drivers/usb/storage/usb-storage.c	/mnt/udisk*
网卡	drivers/net/usb/dm9620.c	
USB 转串口	drivers/usb/serial/pl2303.c	
USB 摄像头	drivers/media/video/uvc/*	
MIPI-CSI 摄像头	drivers/video/s5p_mipi_dsi_lowlevel.c	
MIPI-DSI 显卡	drivers/media/video/exynos/mipi-csis/mipi-csis.c	
声卡驱动	/sound/*	/dev/snd/

矩阵键盘	drivers/input/keyboard/gpio_keys.c	
MFC	drivers/media/video/samsung	
JPEG	drivers/media/video/samsung	
USB_蓝牙	drivers/bluetooth	
SDIO_WIFI	drivers/mtk_wcn_combo/*	
USB 鼠标	drivers/hid	/dev/input/mice
USB 键盘	drivers/hid	/dev/input/evnt*

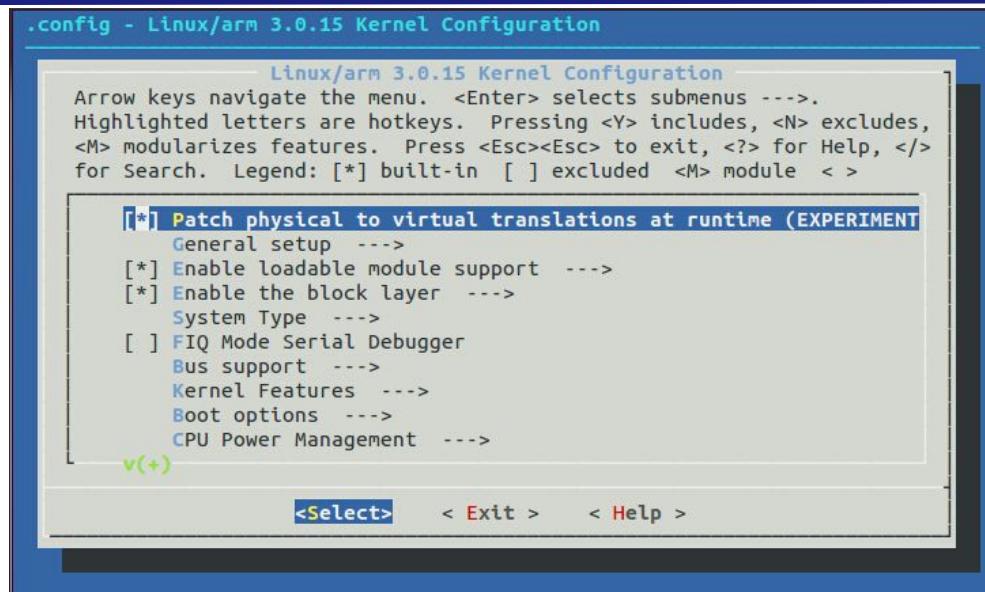
## 9.3 Menuconfig 的用法

在手动定制内核前，需要先了解一下如何使用配置工具“menuconfig”。

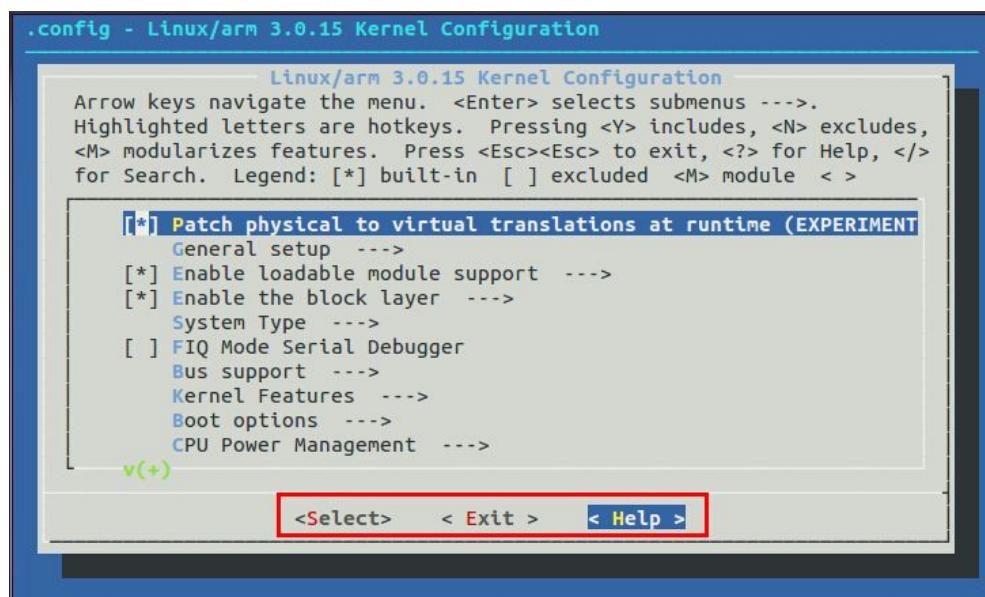
进入解压的内核文件夹，如下图所示，输入命令“#make menuconfig”，然后按“回车”。

```
root@ubuntu:/home/topeet/zImage/iTop4412_Kernel_3.0# make menuconfig
```

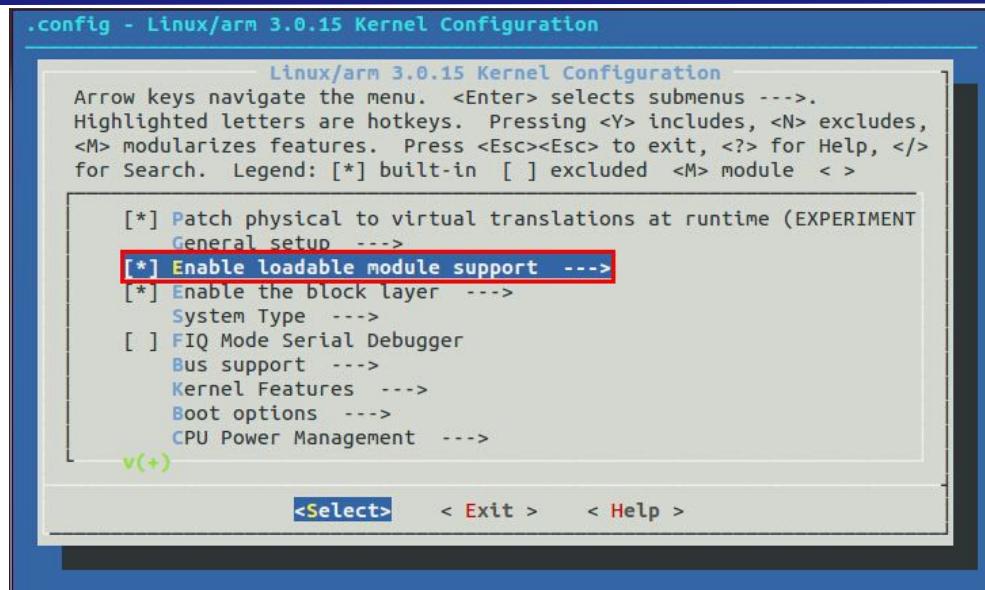
如下图所示，就是 Linux 内核的配置界面



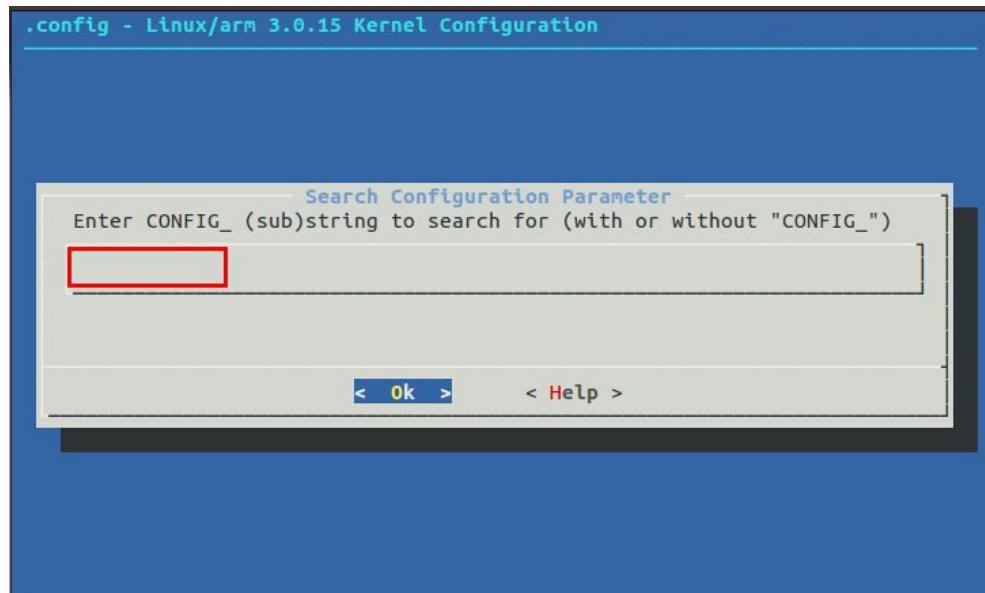
如下图所示，“方向按键”中的“左右”可以选择你需要的操作。“<Select>”表示进入选择的配置界面，“< Exit >”表示返回，“< Help >”可以阅读帮助文档。



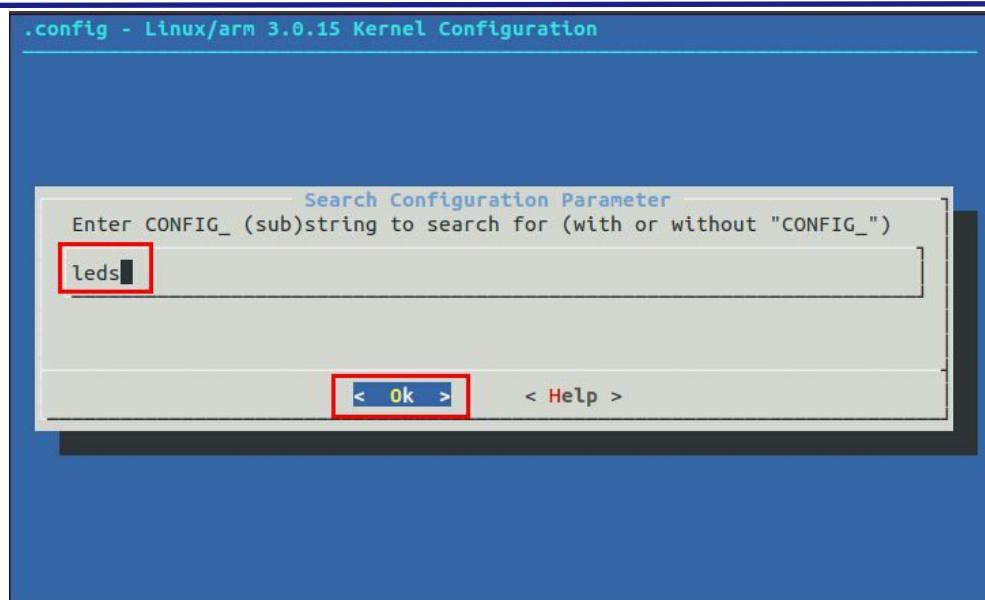
如下图所示，“方向按键”中的“上下”可以选择配置的选项。



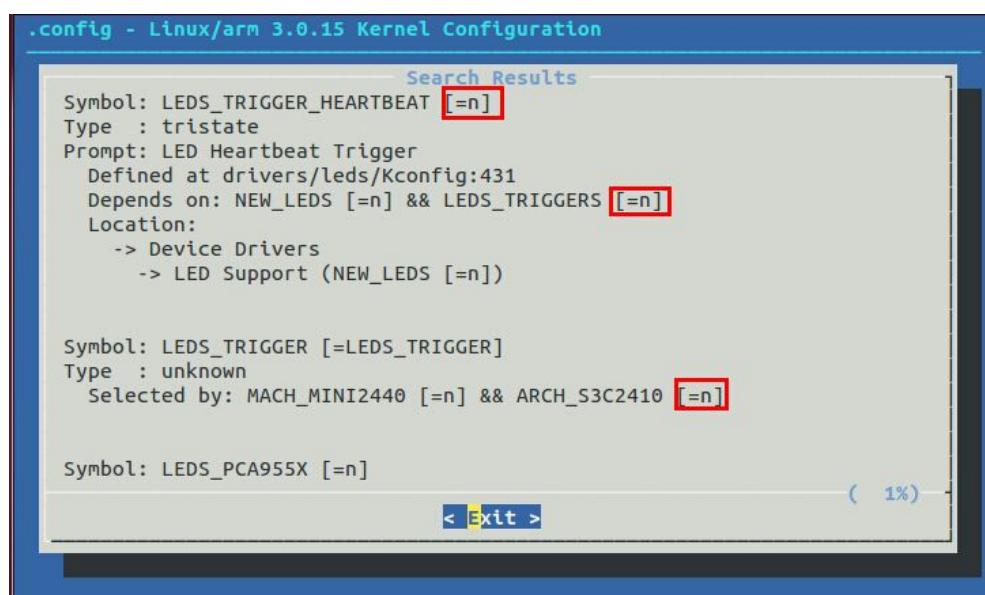
如下图所示，输入 “/” ，可以进入搜索界面。



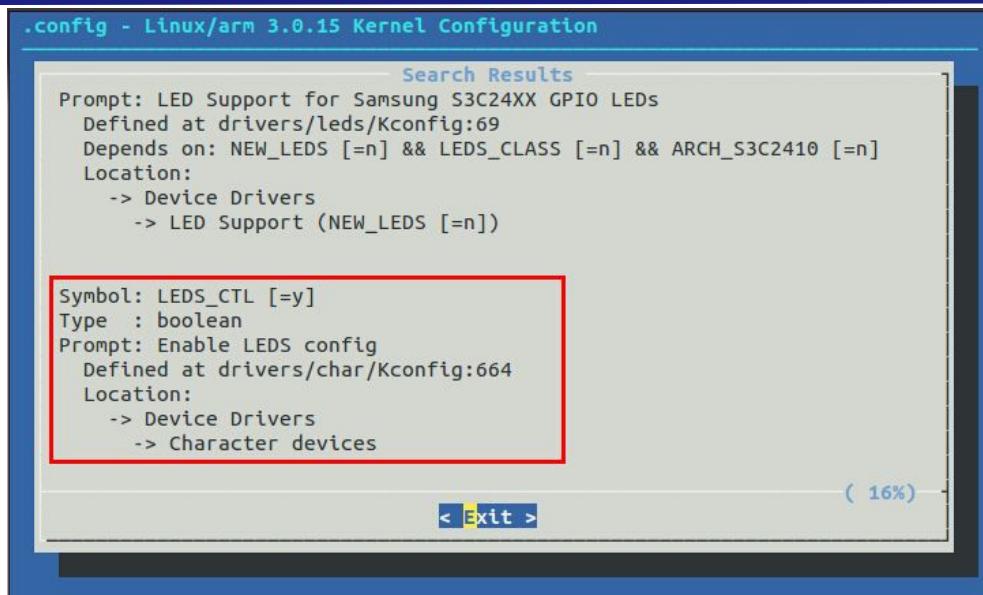
如下图所示，这里来查找一下 “leds” 的驱动，输入 “leds” ，然后按 “回车” 。



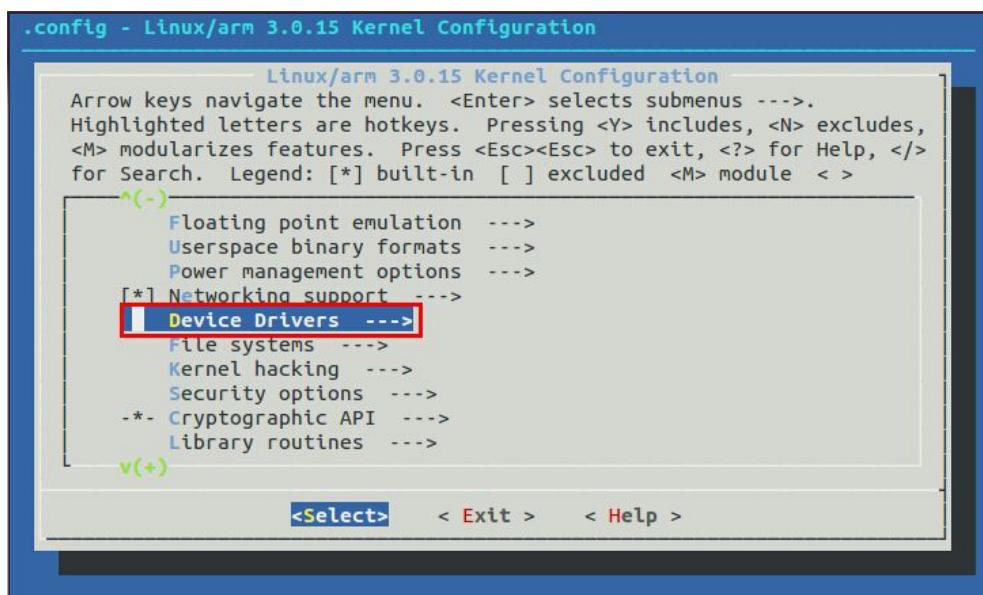
如下图所示，发现很多配置都是“=n”，通过方向按键，控制向下翻页，然后观察那个选项配置成了“=y”。



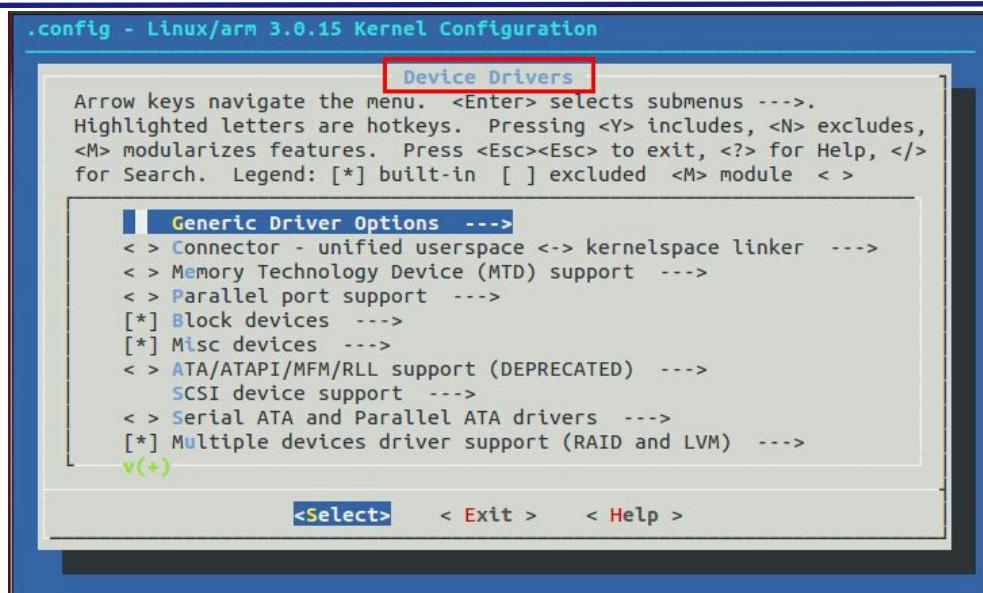
如下图所示，这里可以看到这个 leds 驱动的目录“Device Drivers” “Character devices”



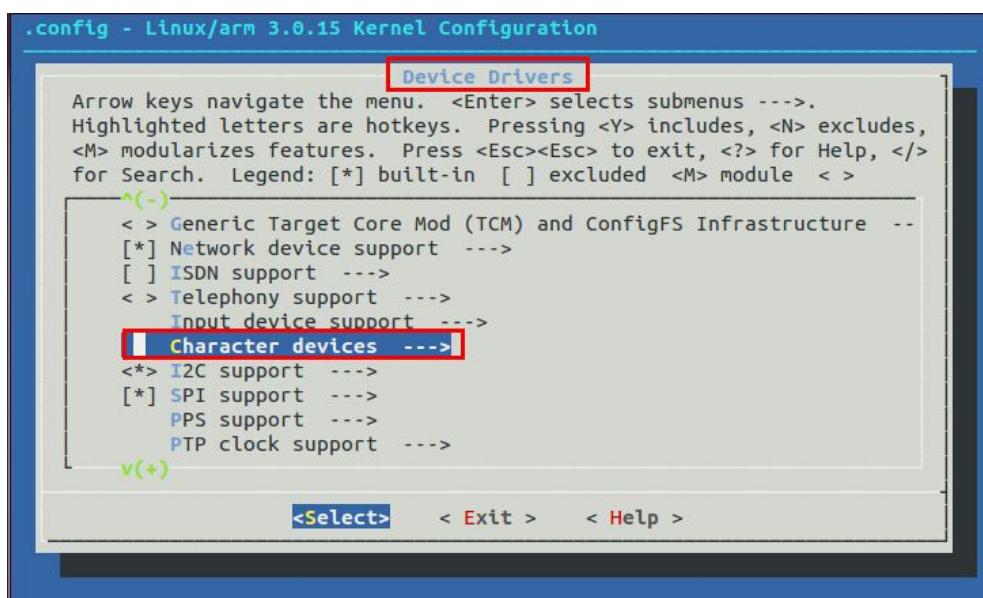
然后，根据查找出来的信息，找到对应的 leds 驱动。如下图，返回配置界面。找到“Device Drivers”目录。输入“回车”。



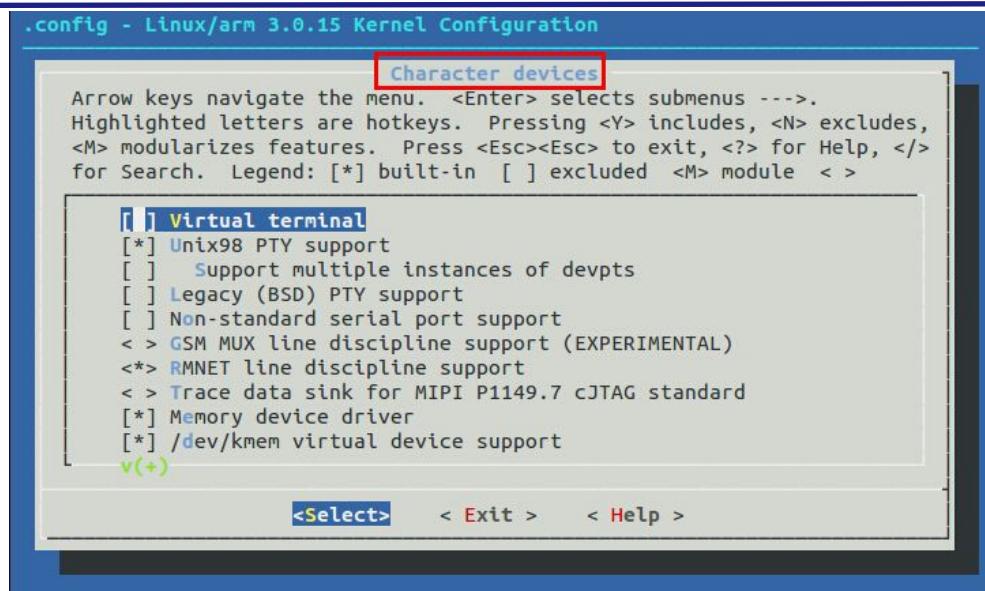
如下图所示，进入“Device Drivers” 对应的配置界面。



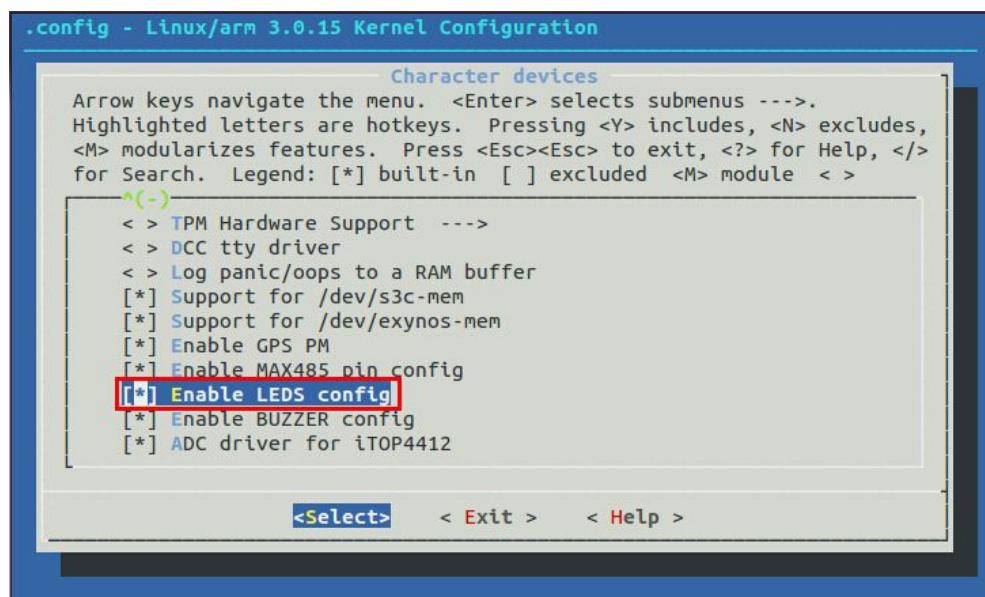
然后，如下图所示，找到“Character devices”，输入“回车”。



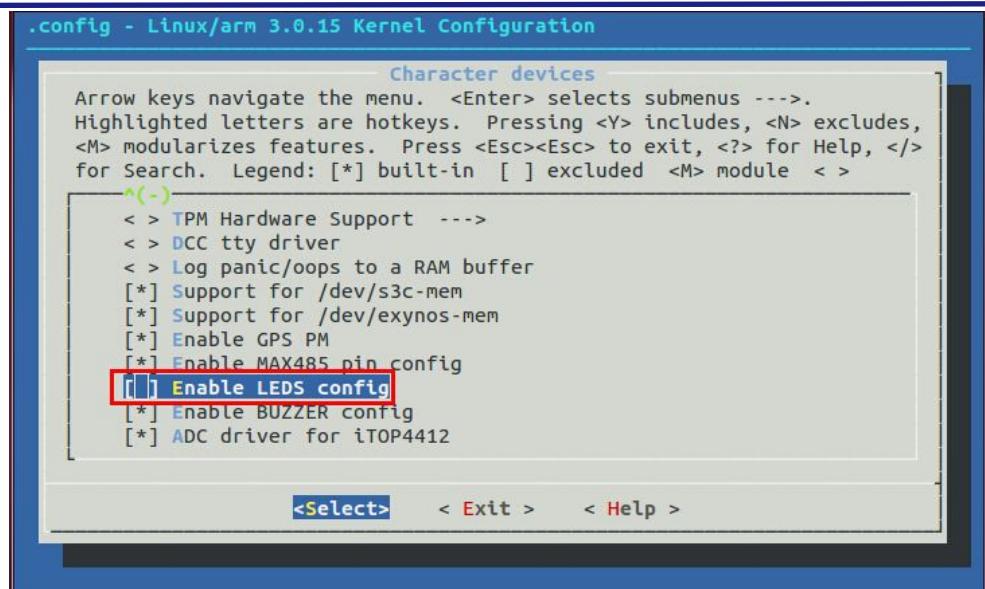
如下图所示，进入“Character devices”配置界面。



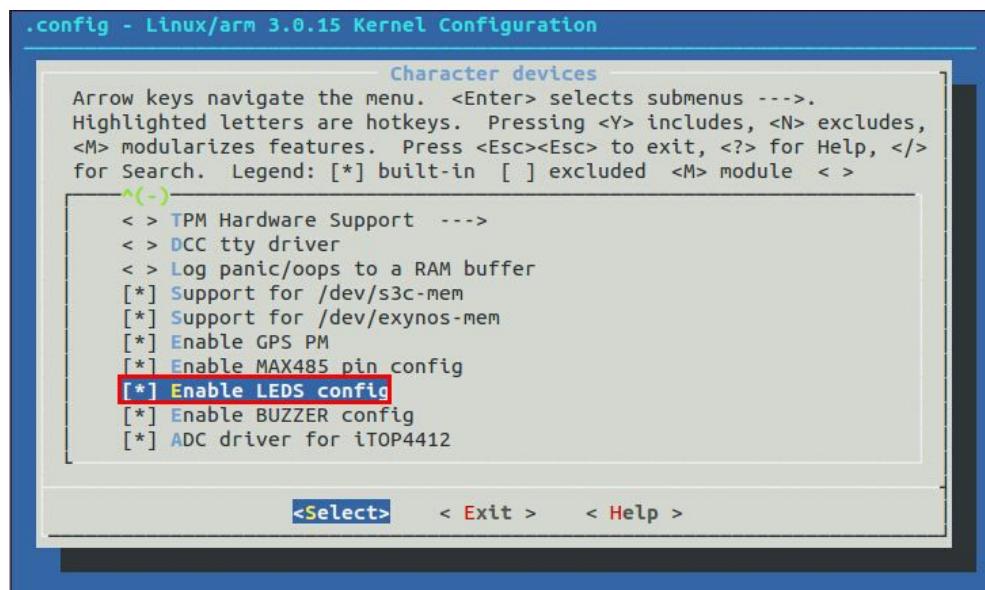
如下图所示，“Enable LEDS config” 找到对应的 leds 驱动配置选项。缺省配置文件里，这个已经选上了。



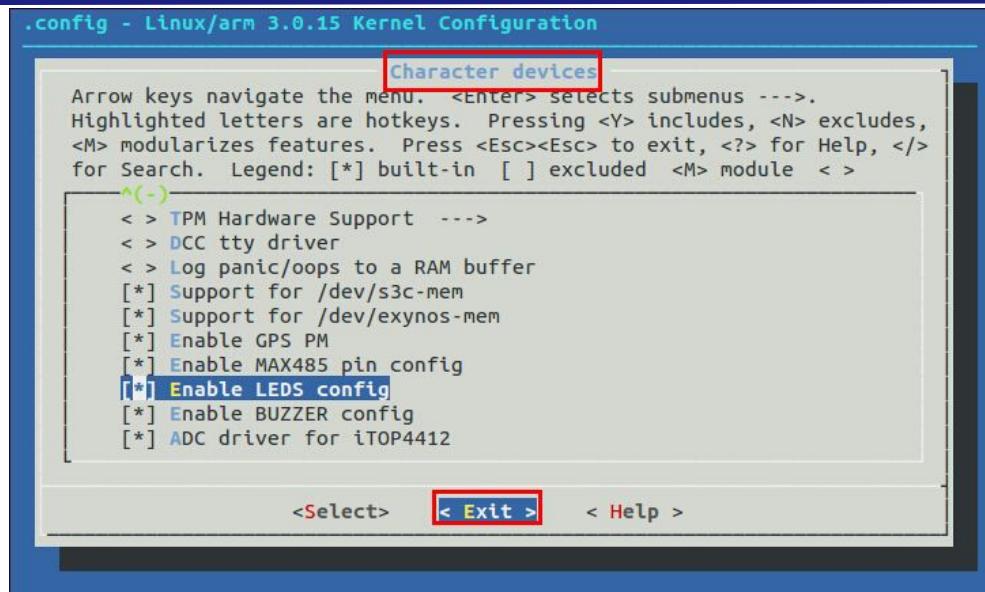
单击“空格”键后，去掉 leds 驱动选项。



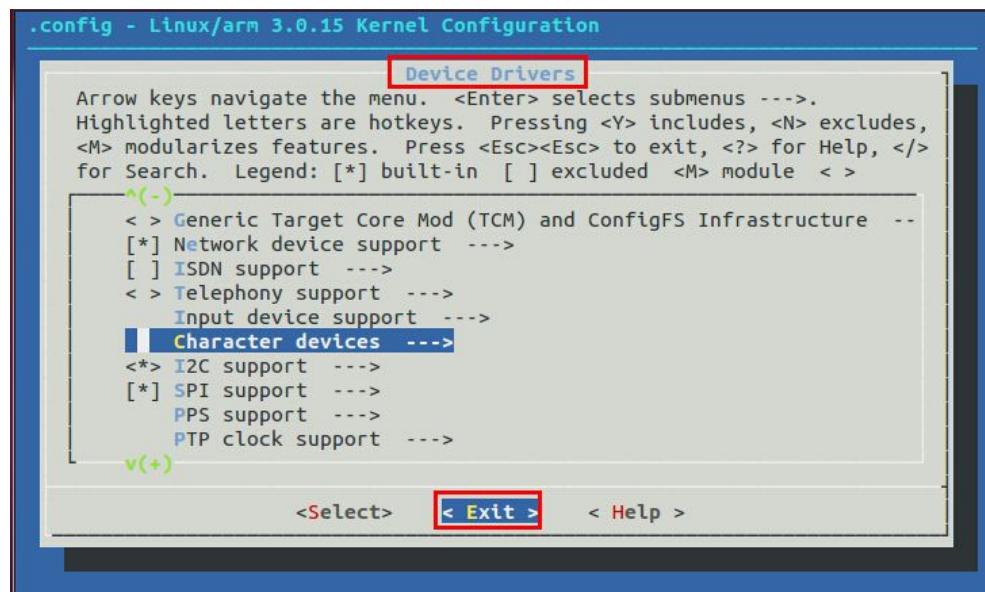
再次敲击“空格”，选上 leds 驱动的选项。



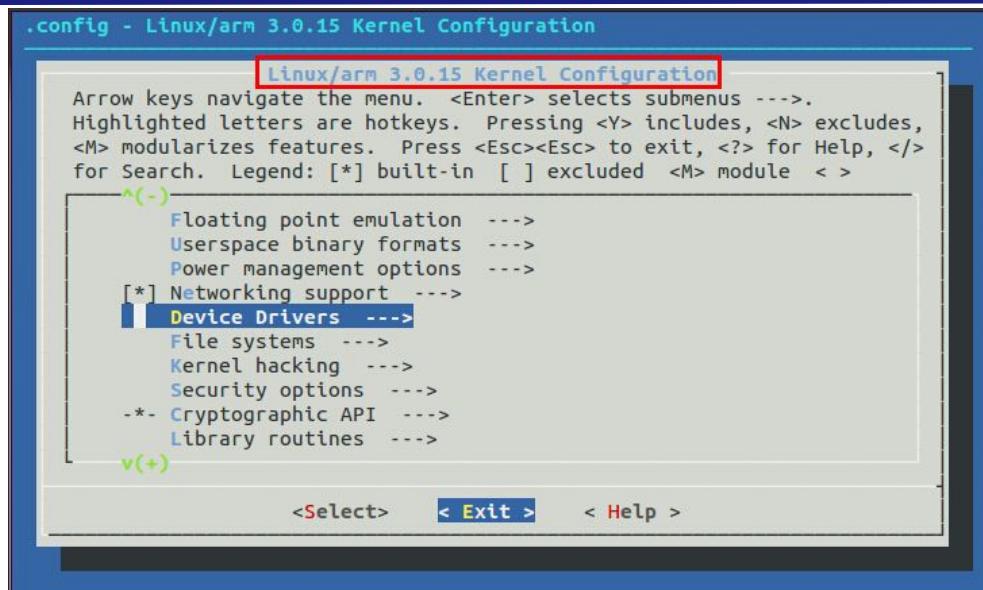
然后，选上“Exit”，如下图所示，输入“回车”。



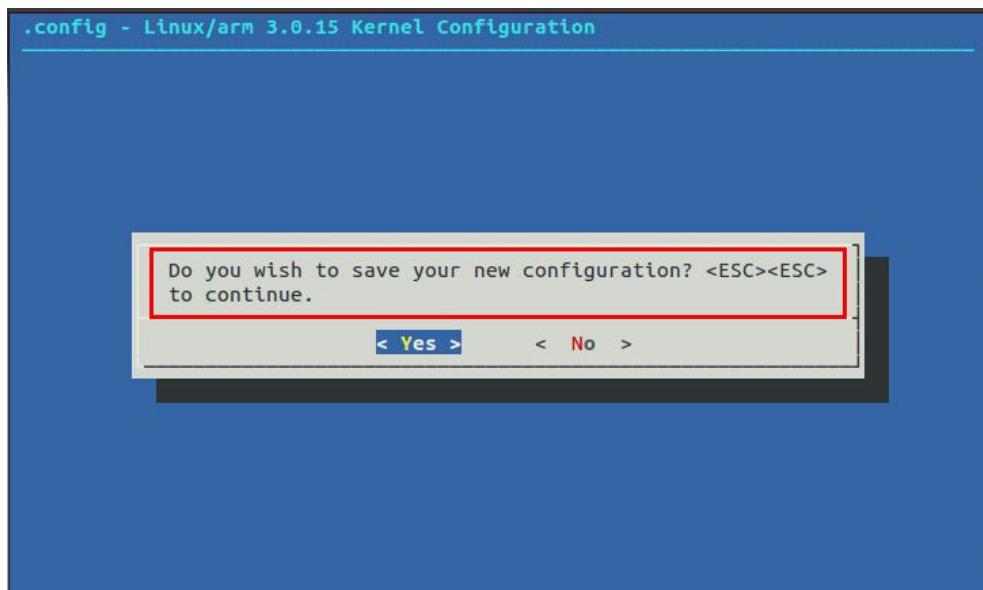
如下图所示，继续退出。



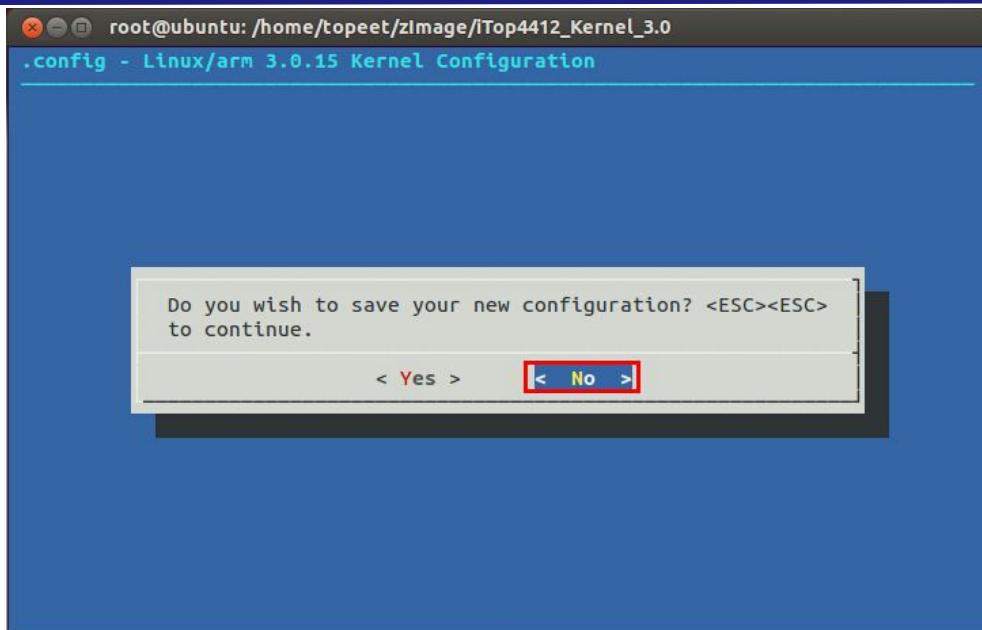
如下图所示，继续退出。



如下图所示，因为修改过配置选项，所以退出的时候会提醒“是否保存新的配置”。



如下图所示，因为第一次操作，担心用户在无意间动了某个配置选项，编译后无法启动，建议选择“No”，不保存退出。



到这里，整个 Menuconfig 配置的操作以及流程就完全介绍完了。如果修改了配置文件，如下图所示的 “.config” 文件就会被修改。再次编译内核的时候，系统会根据新的 config 文件来编译整个内核。

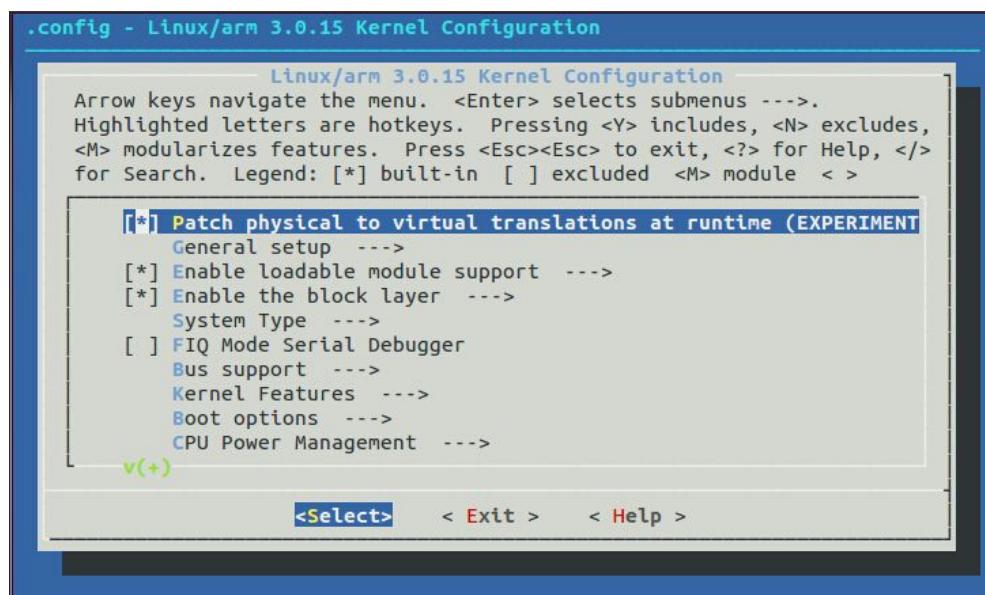
```
root@ubuntu:/home/topeet/zImage/iTop4412_Kernel_3.0# ls -a
.
..
arch
binary
block
.config
COPYING
CREDITS
crypto
Documentation
drivers
firmware
Kconfig
kernel
kernel_readme.txt
lib
MAINTAINERS
Makefile
README
REPORTING-BUGS
samples
scripts
security
sound
```

## 9.4 手动定制 Linux 内核

前面介绍了如何使用缺省文件配置内核，但是对于 Linux 内核具体的配置选项有很多，下面介绍常见的一些。

在进行配置前，请务必注意先按照前面的方法先加载缺省的配置文件，否则下面有些配置选项可能不会出现。

进入解压的内核文件夹，输入配置命令“make menuconfig”，如下图所示的Menuconfig 配置界面。

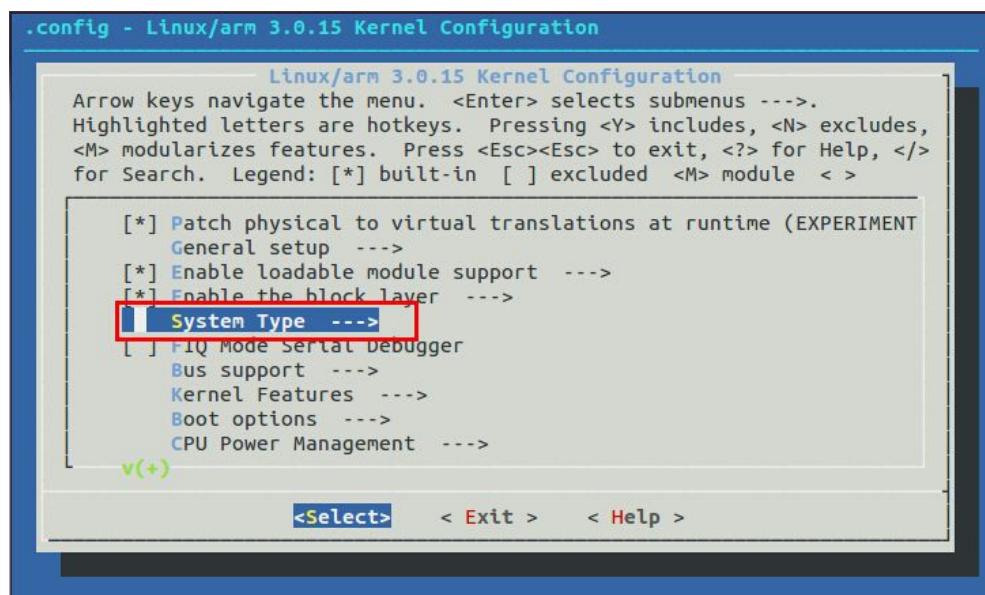


#### 9.4.1 配置 CPU 平台文件

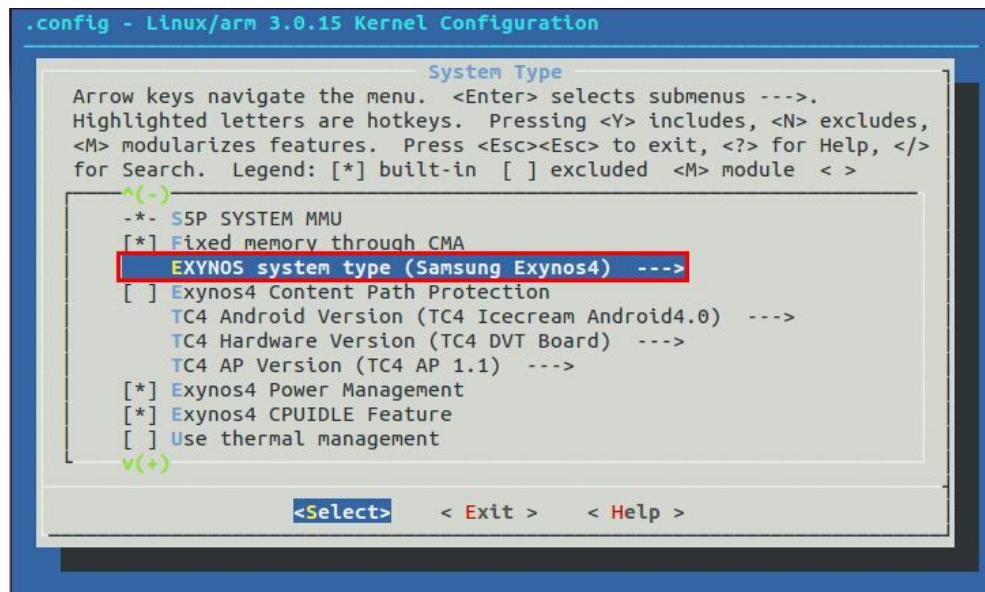
每一个内核都需要对应特定的平台，下面介绍如何配置成为 ITOP-4412 的平台内核。

配置完后对应平台文件“arch/arm/mach-exynos/mach-itop4412.c”

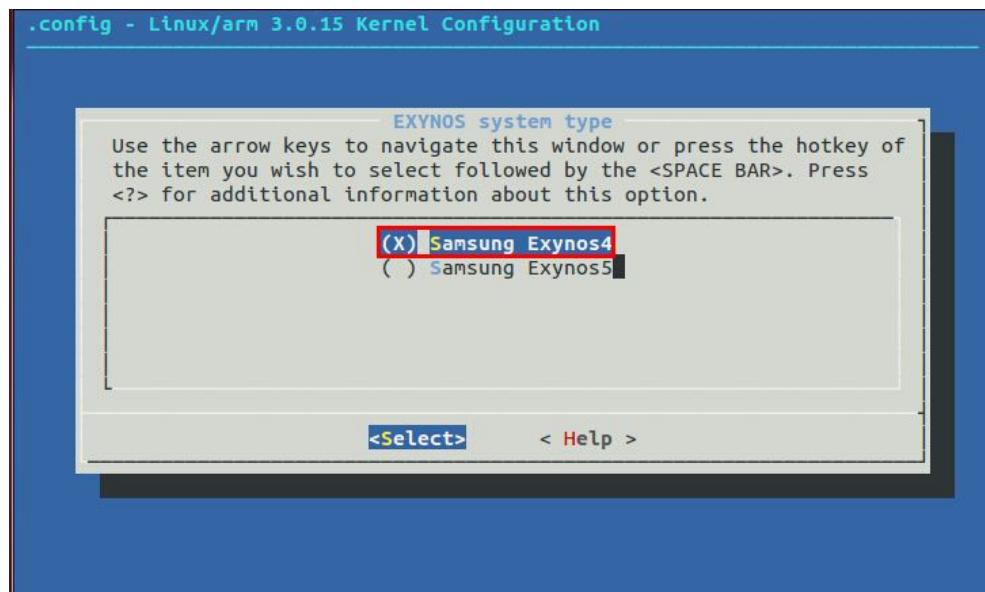
如下图所示。



如下图所示，选择“EXYNOS system type (Samsung Exynos4) --->”，按“回车”进入。



如下图所示，选择“(X) Samsung Exynos4” ，按“回车”退出该窗口。

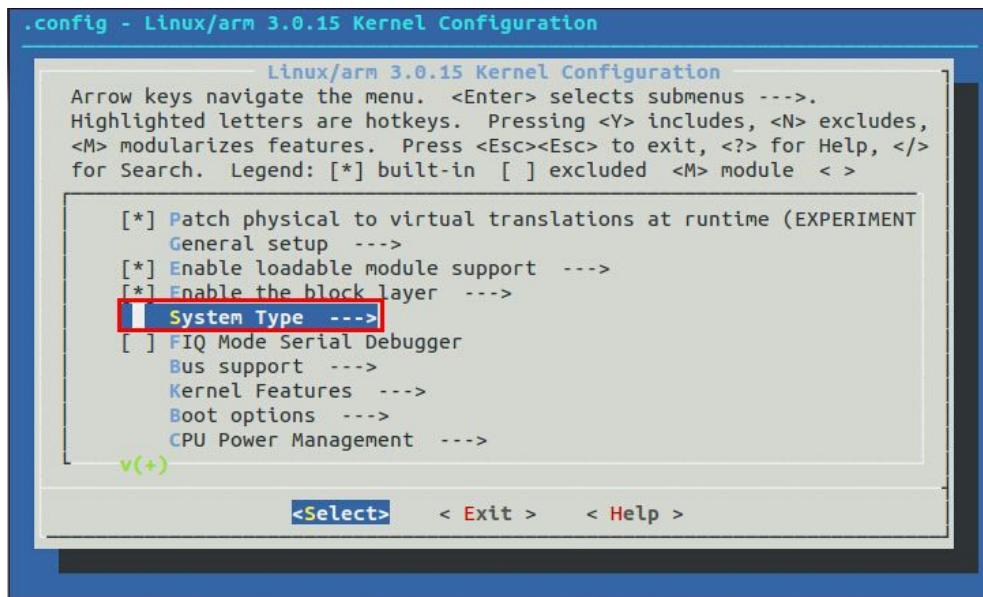


## 9.4.2 Vibrator 振动器

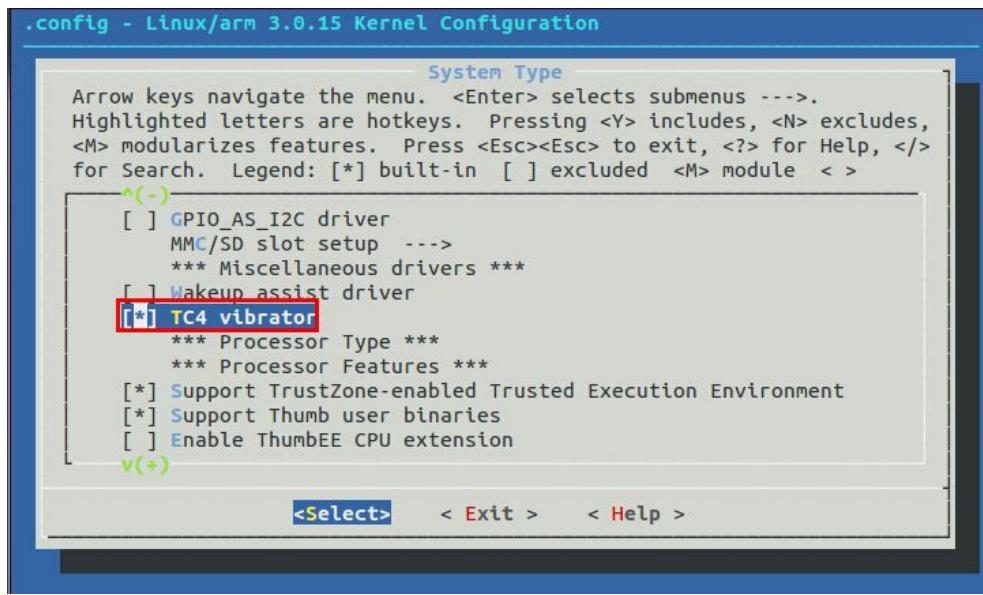
Vibrator 驱动对应 Android 的振动器，配置好之后 Android 触摸可以发出声响。

Vibrator 驱动对应源码 “arch/arm/mach-exynos/ asv-4x12\*” , 该驱动和 9.4.3 节蜂鸣器复用一个 IO。

如下图所示，配置 Vibrator 驱动。



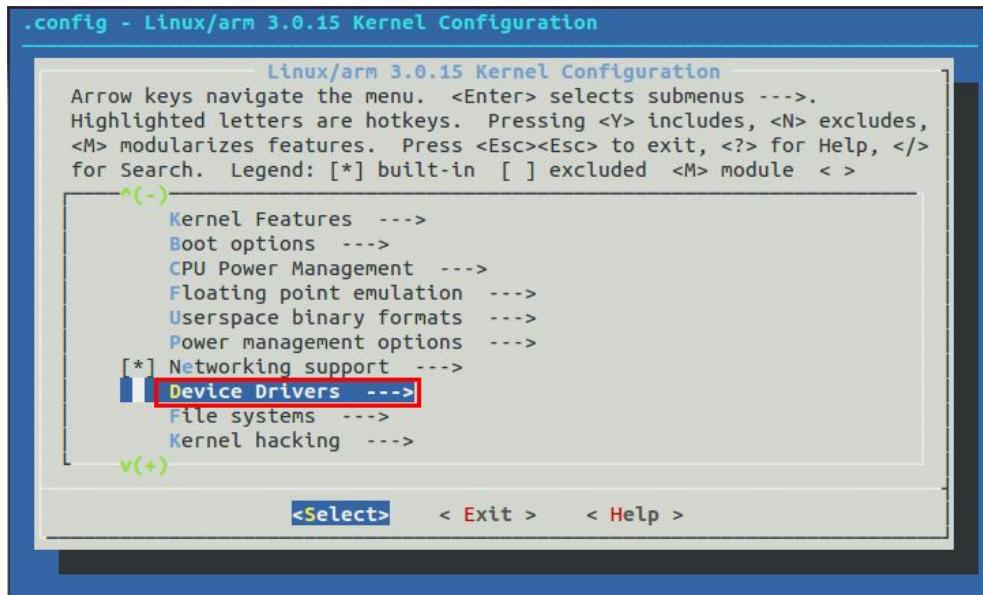
如下图，配置“TC4 vibrator”。



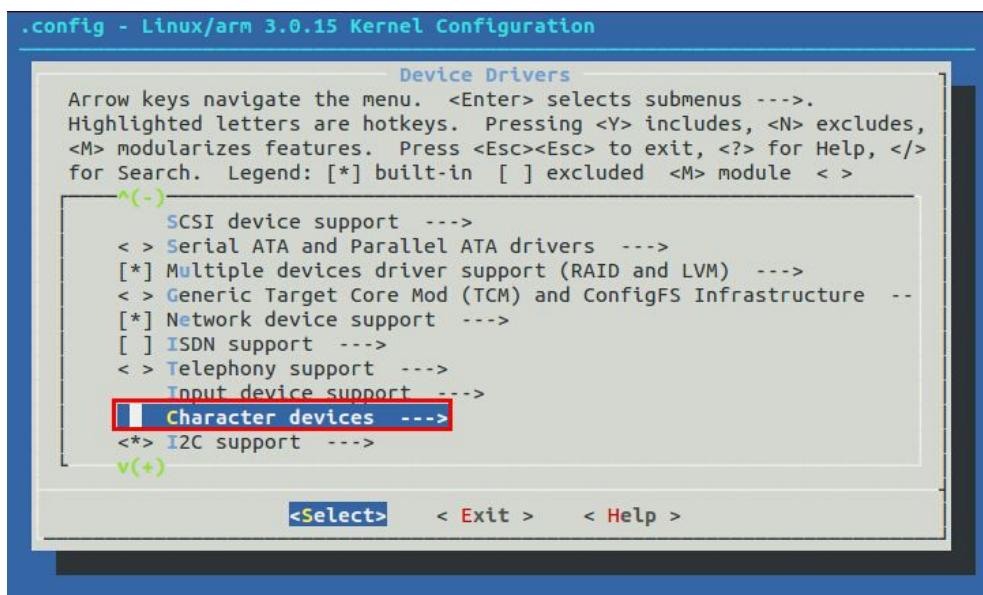
### 9.4.3 蜂鸣器 Buzzer

设置蜂鸣器驱动之后，对应设备节点 “/dev/buzzer\_ctl” ，该驱动对应源码文件 “drivers/char/itop4412\_buzzer.c” 。

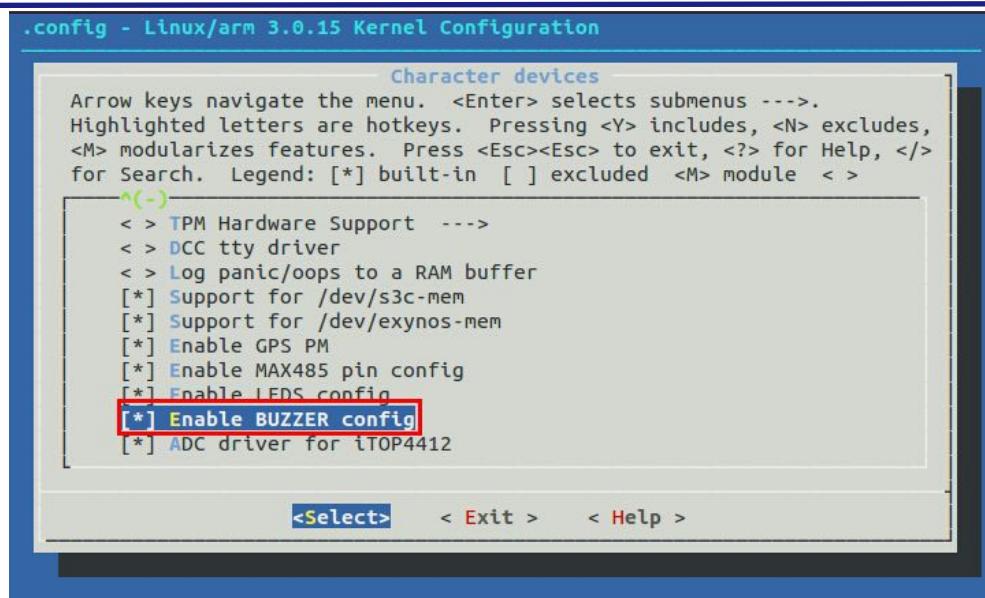
如下图，进入“Device Drivers --->”。



如下图，进入“Character devices --->”。



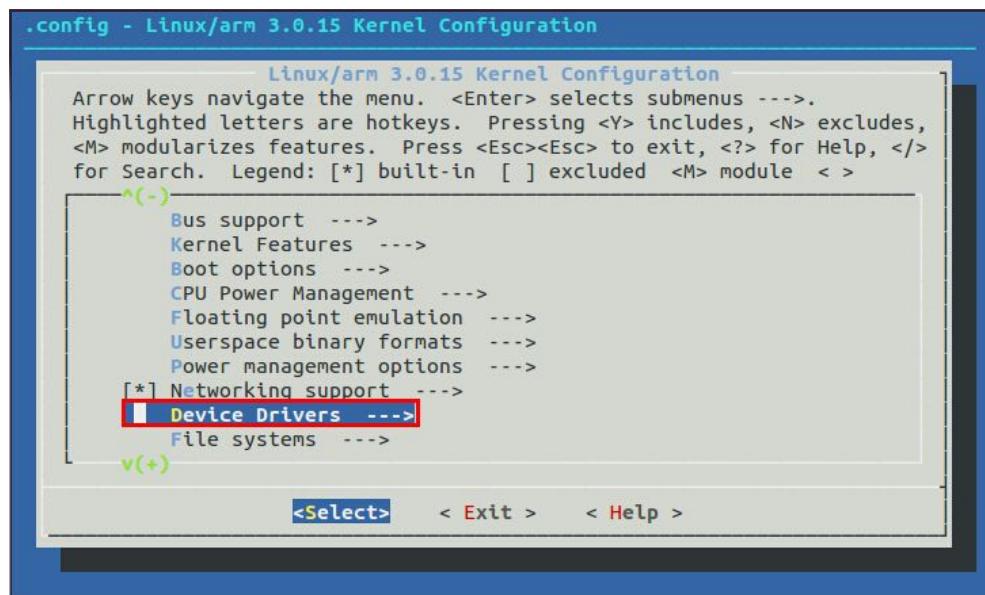
如下图，配置蜂鸣器驱动“Enable BUZZER config”。



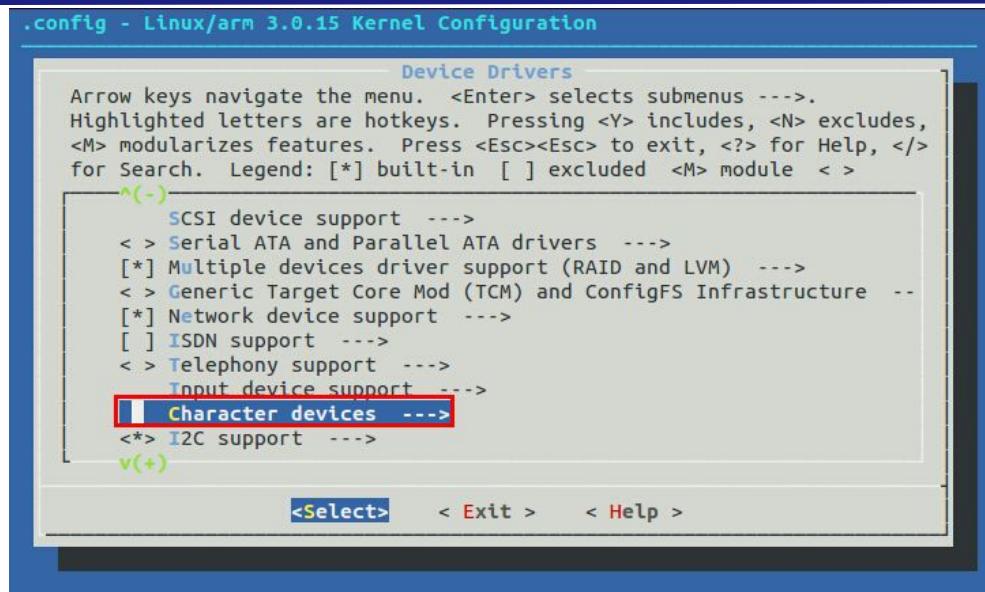
## 9.4.4 Leds

该驱动对应源码 “drivers/char/iTOP4412\_leds.c” , 对应设备节点 “/dev/leds” 。

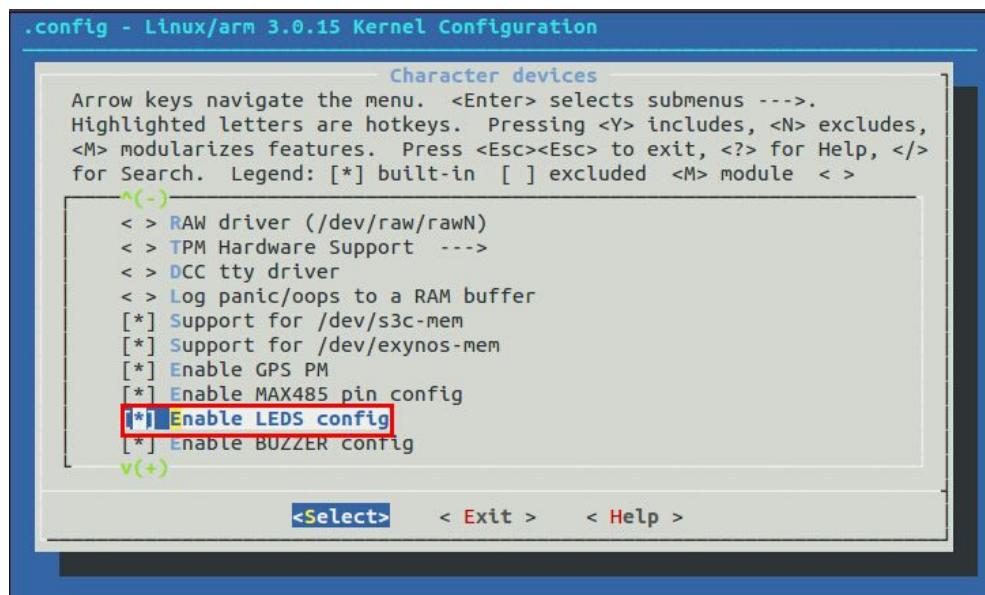
如下图，进入 “Device Drivers --->” 。



如下图，进入 “Character devices --->” 。



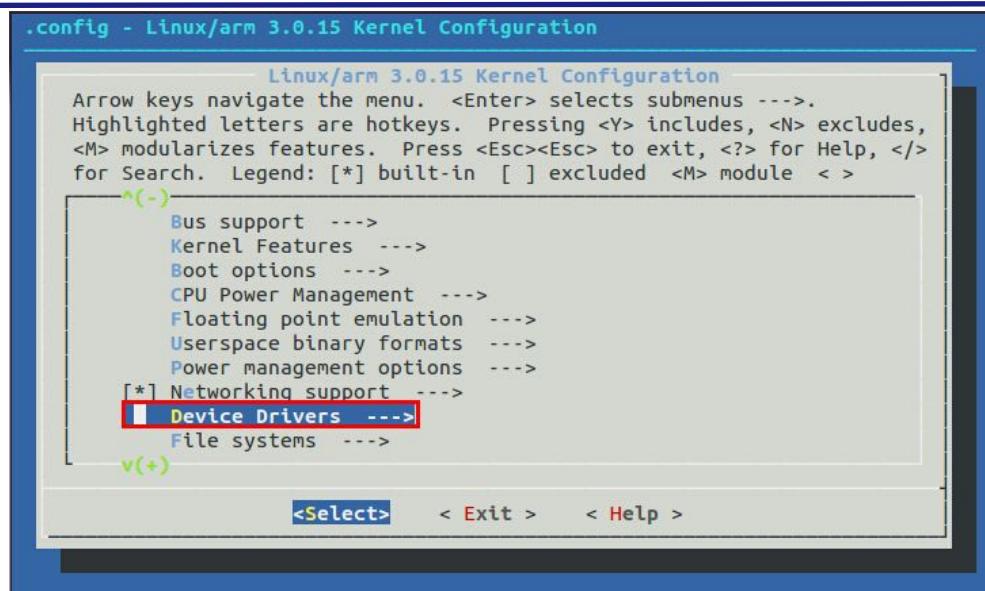
如下图所示，配置 LEDS 驱动 “Enable LEDS config”。



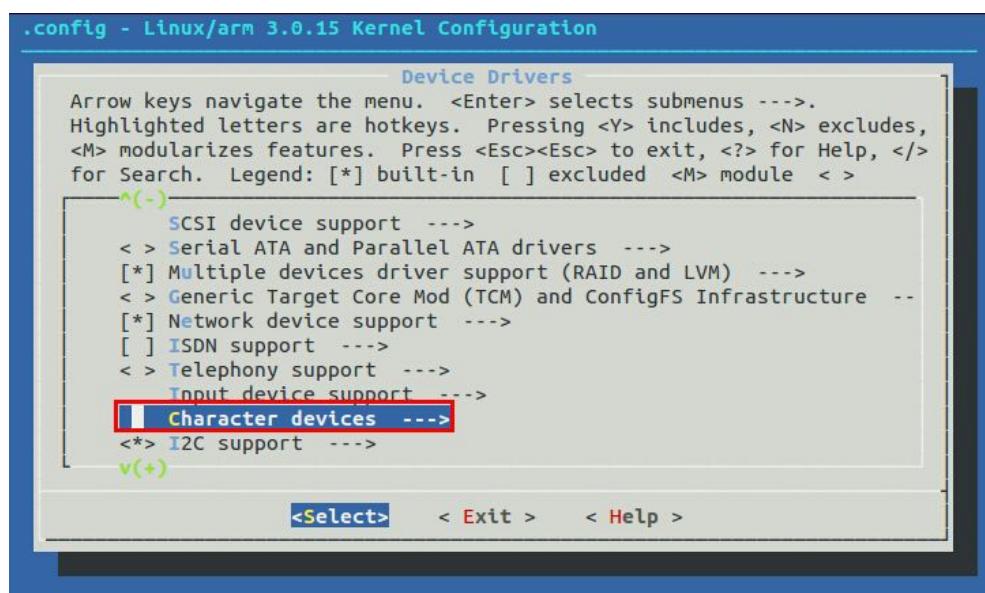
## 9.4.5 ADC 数模转换

该驱动对应源码 “drivers/char/itop4412\_adc.c”，对应设备节点 “/dev/adc”。

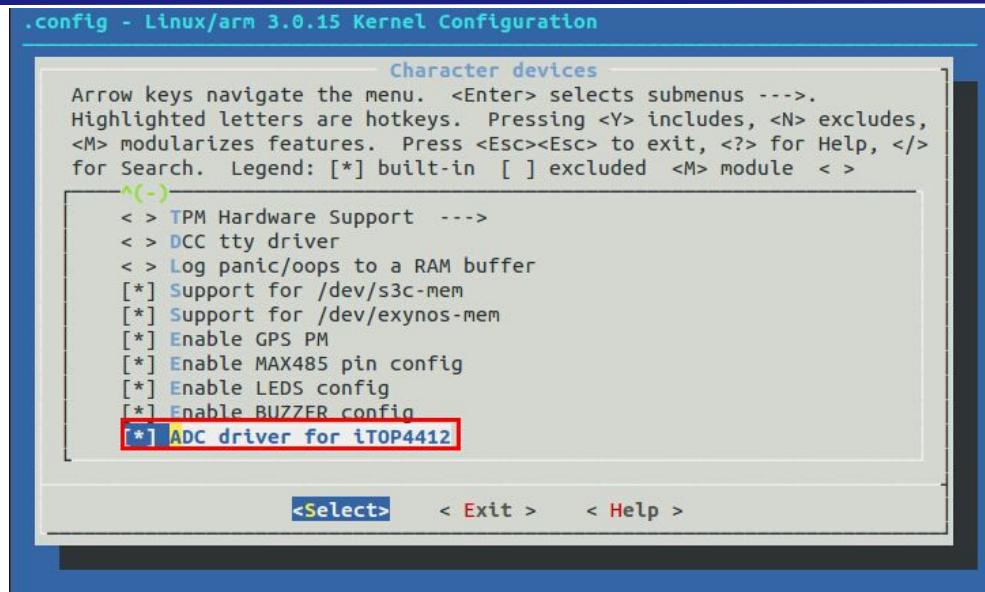
如下图，进入 “Device Drivers --->”。



如下图，进入 “Character devices --->” 。



如下图所示，配置 ADC 驱动 “ADC driver for iTOP4412” 。

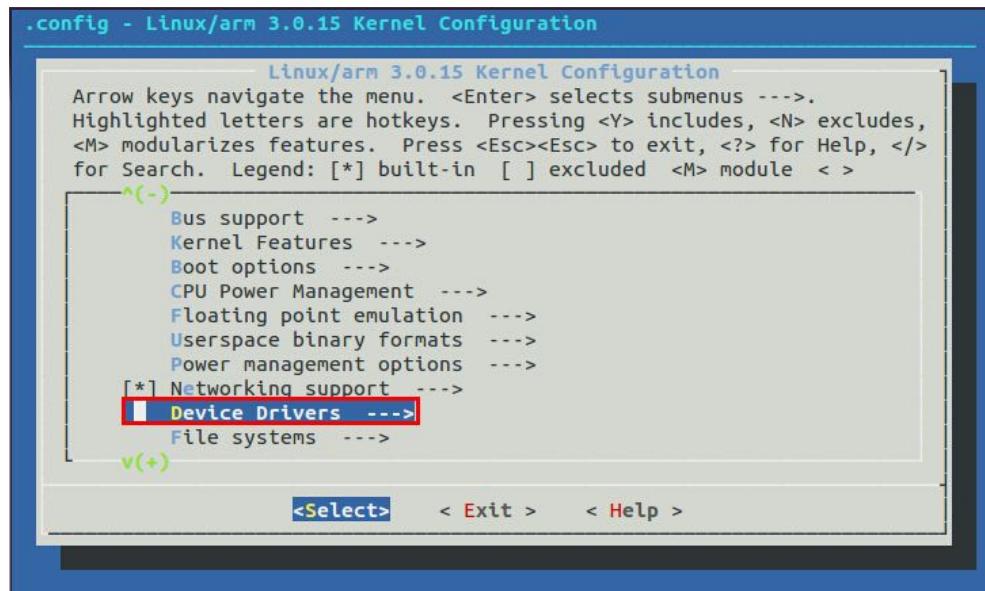


## 9.4.6 RS-485

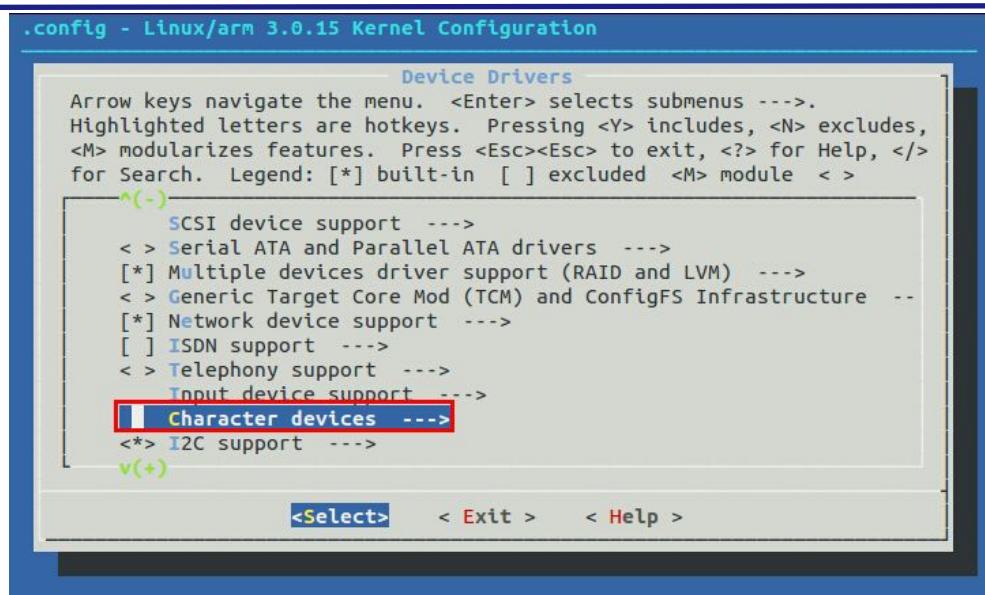
该驱动对应源码 “drivers/char/max485\_ctl.c” , 对应设备节点

“/dev/max485\_ctl\_pin” 。

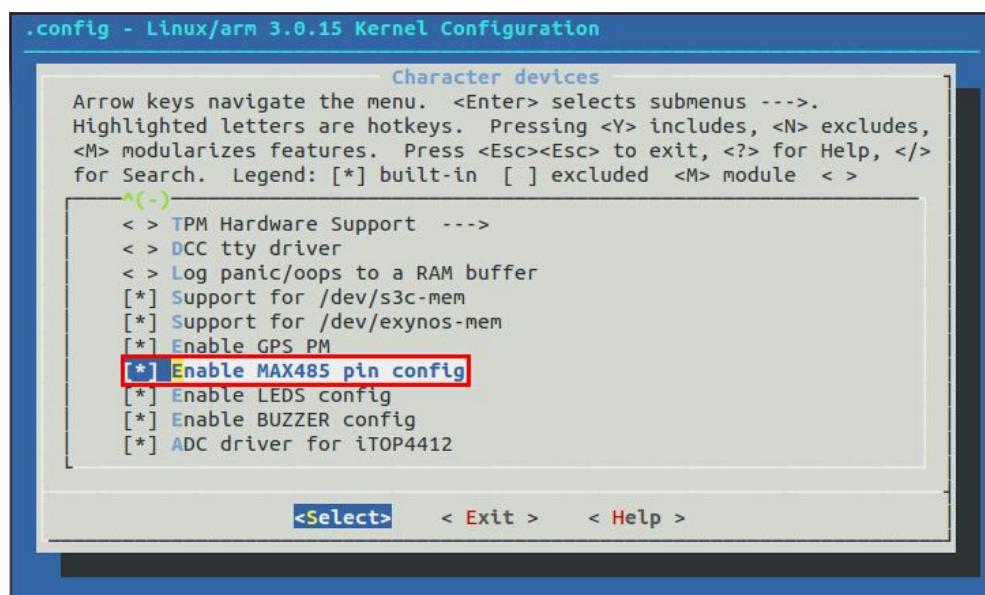
如下图 , 进入 “Device Drivers --->” 。



如下图 , 进入 “Character devices --->” 。



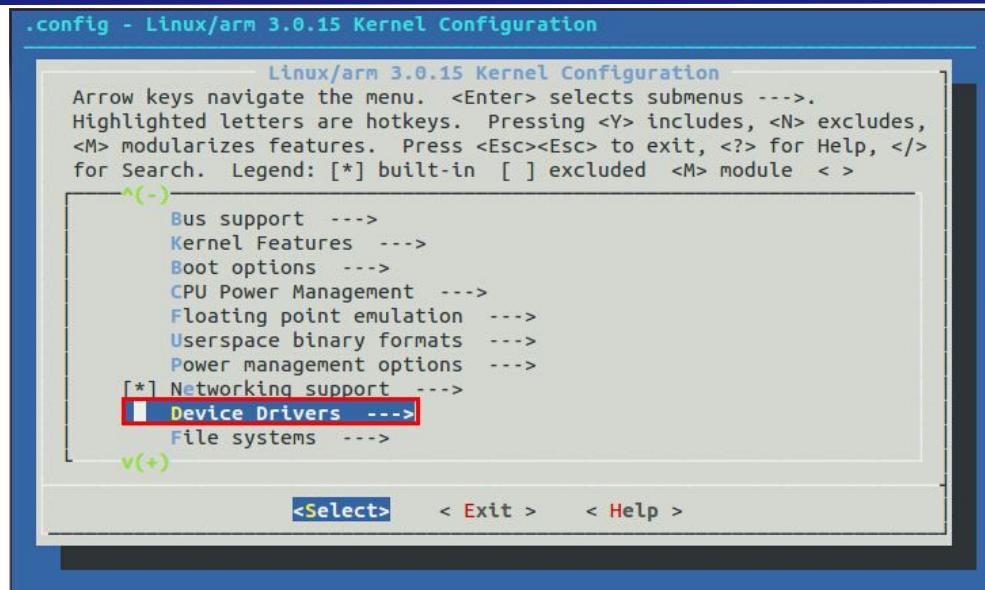
如下图所示，配置 485 驱动 “Enable MAX485 pin config”。



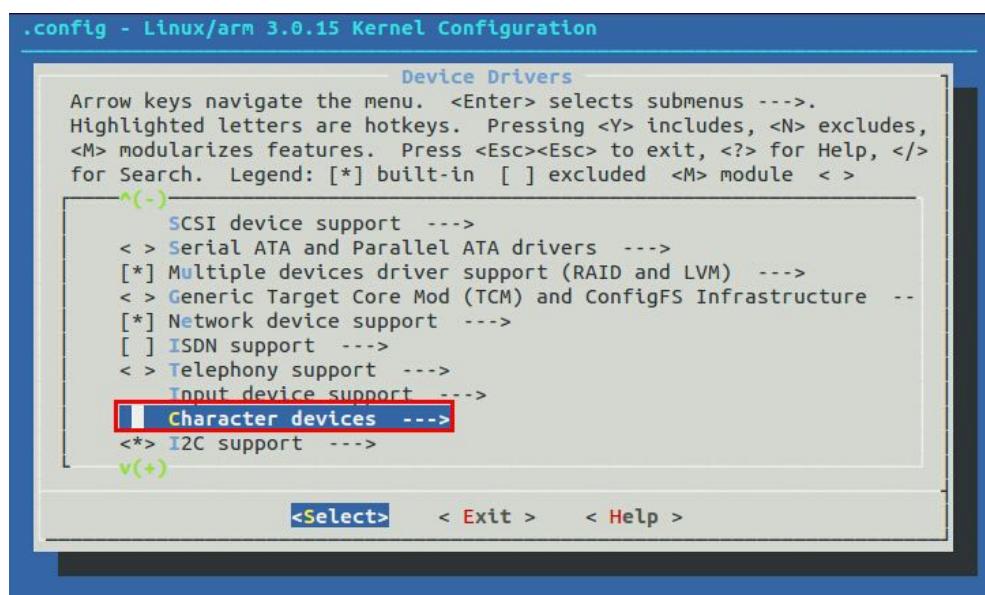
## 9.4.7 GPS 导航

该驱动对应源码 “drivers/char/gps.c” ，对应设备节点 “/dev/gps”。

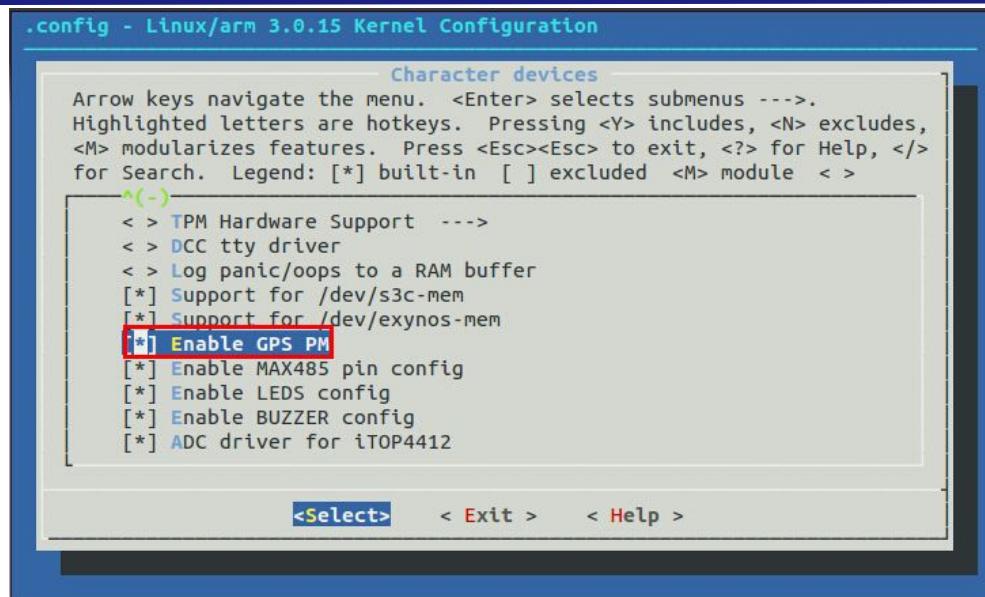
如下图，进入 “Device Drivers --->”。



如下图，进入“Character devices --->”。



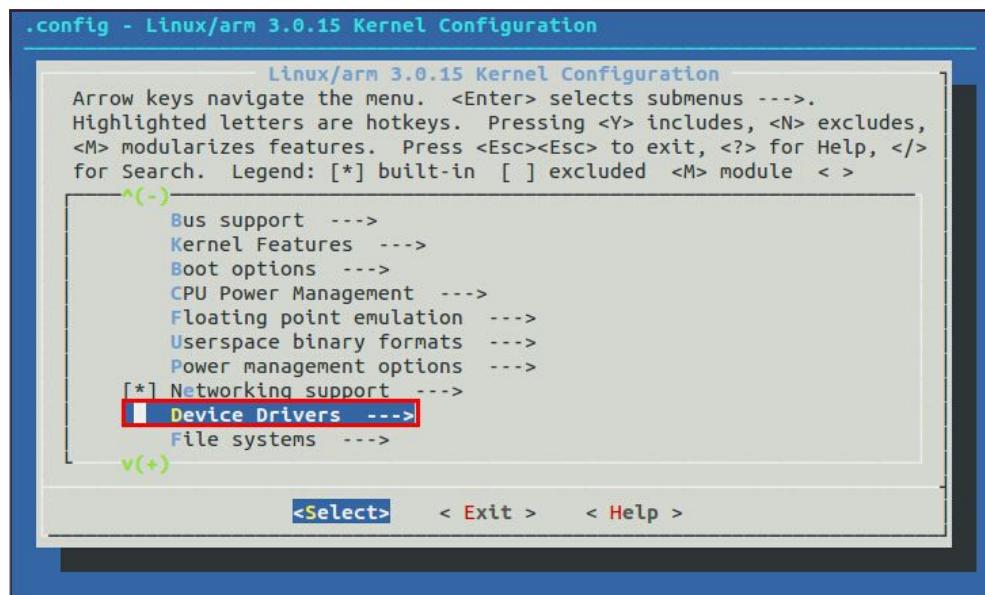
如下图所示，配置 GPS 驱动 “Enable GPS PM”。



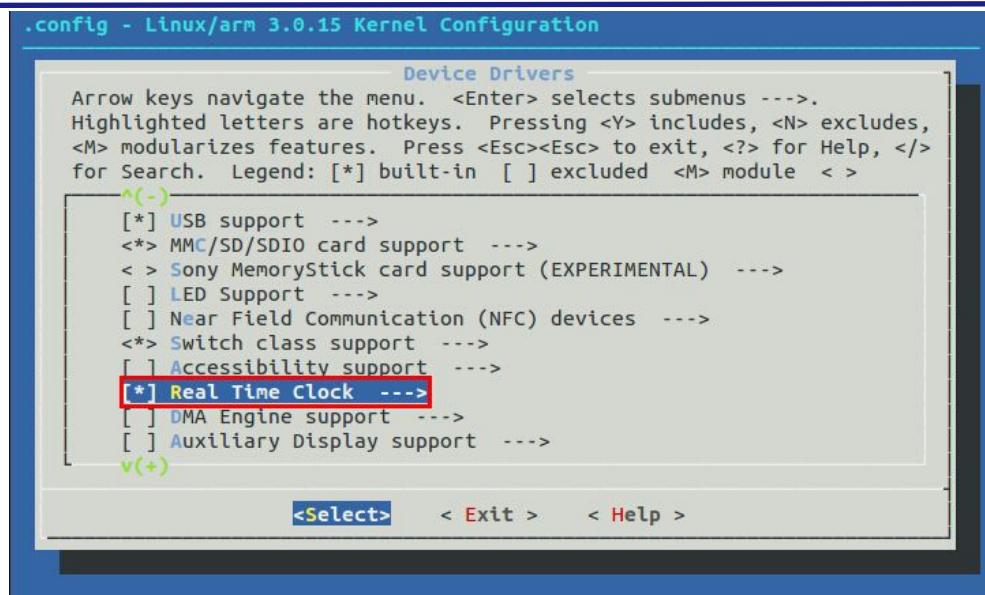
## 9.4.8 RTC 实时时钟

RTC 实时时钟对应源码为 “drivers/rtc/rtc-s3c.c” 。

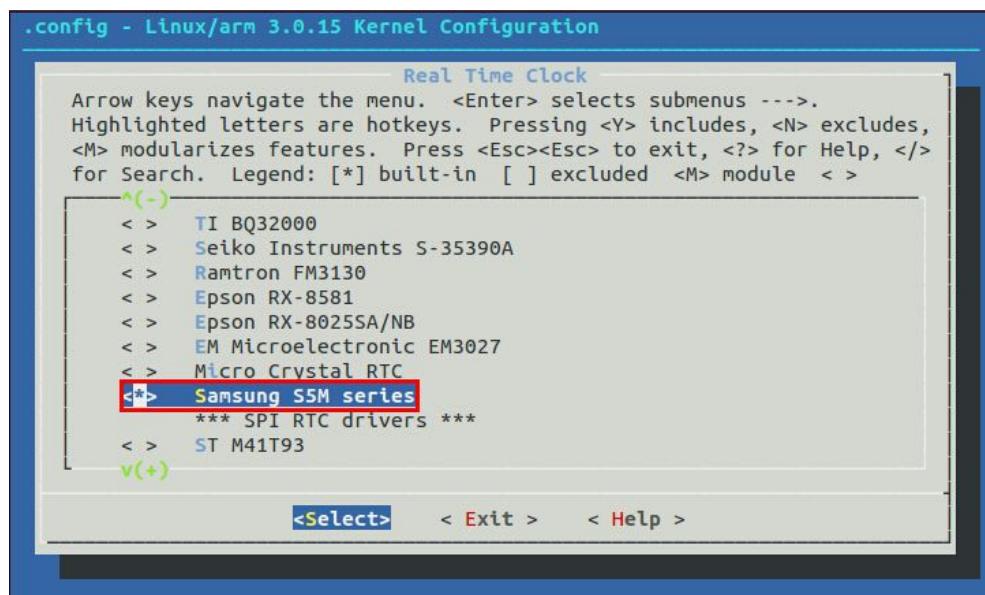
如下图，进入 “Device Drivers --->” 。



如下图，进入 “Real Time Clock --->” 。



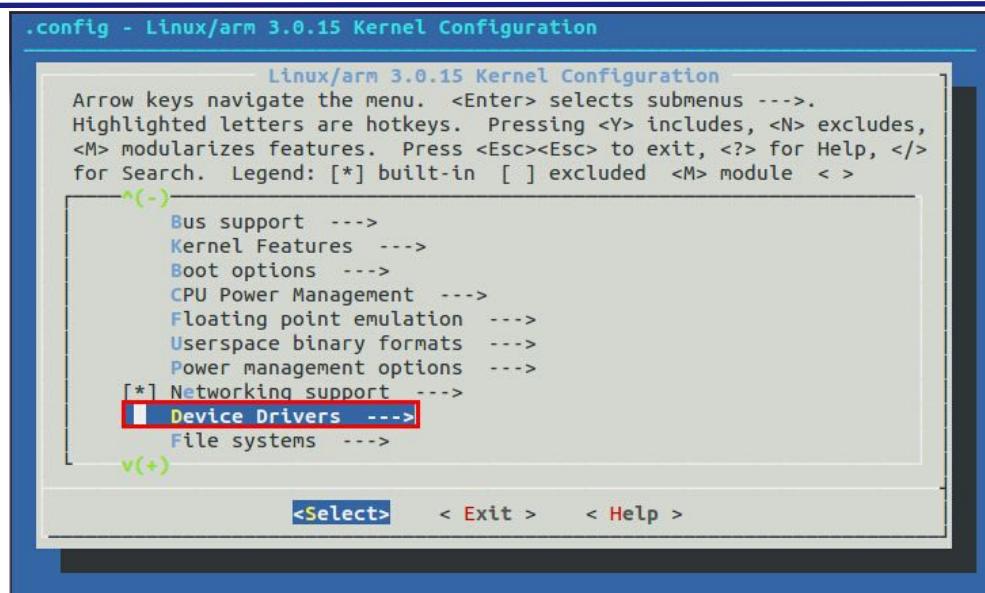
如下图，配置“Samsung S5M series”。



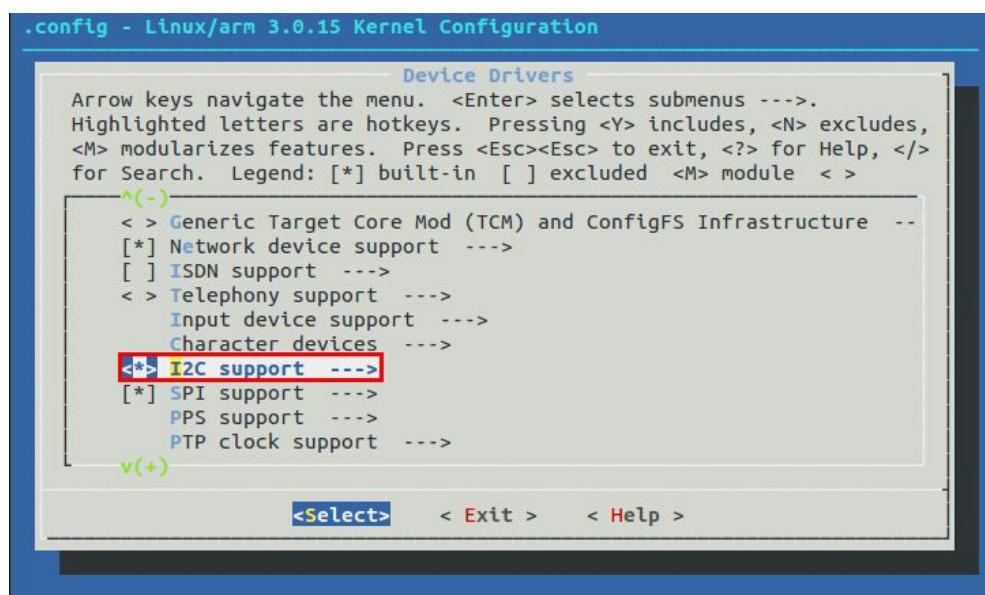
#### 9.4.9 I2C 总线

I2C 是总线，其它驱动如果要使用 I2C 总线，则需要先配置 I2C。

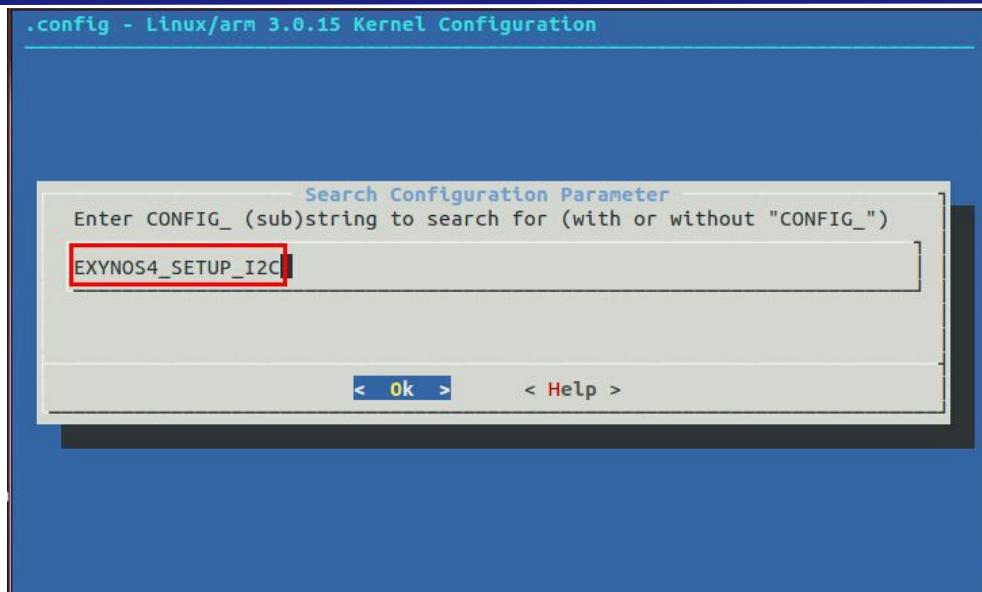
如下图，进入“Device Drivers --->”。



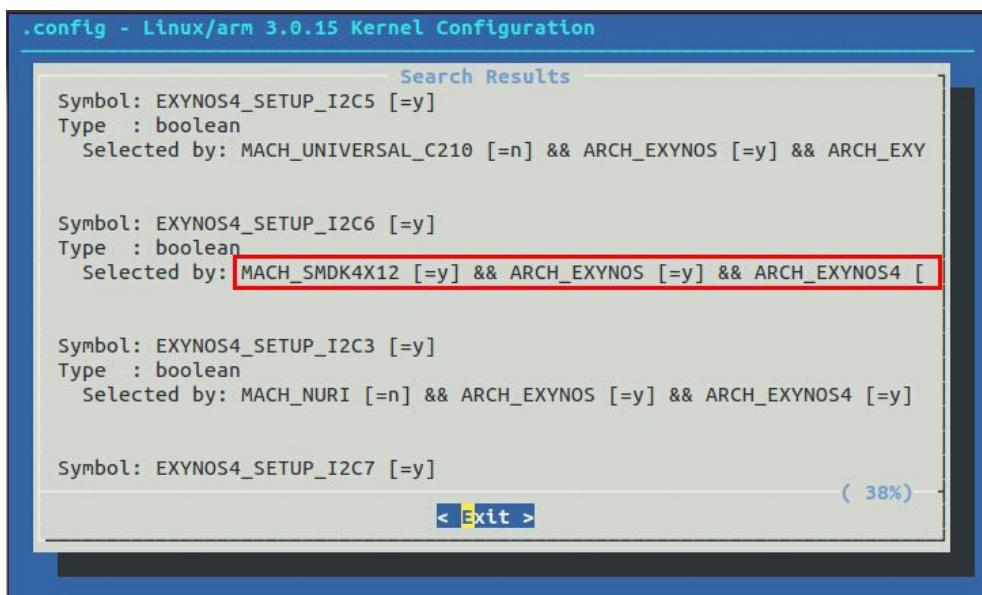
如下图，进入 “I2C support --->” 。



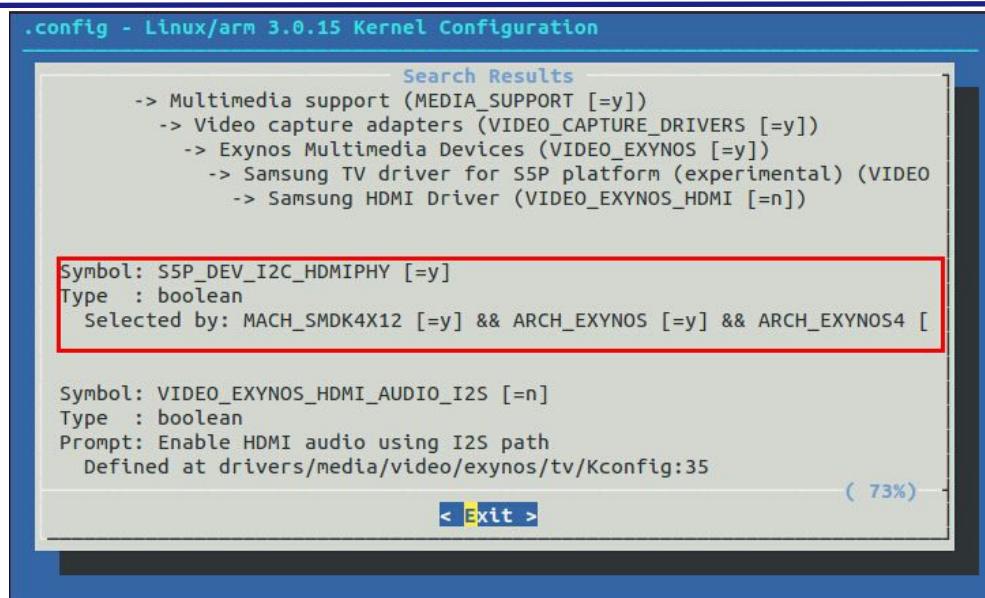
I2C 总线有一些依赖关系，如下图，输入 “/” ,进入查找模式,搜索关键字“EXYNOS4\_SETUP\_I2C” 。



如下图，例如，I2C6 依赖 “MACH\_SMDK4X12” “ARCH\_EXYNOS” 等。把依赖的文件配置上就可以实现对应的 I2C 总线功能。



另外，通过上图，细心的用户可能会发现 I2C0 没有，这是因为已经被 HDMI 中的 I2C 占用了。如下图所示，搜索关键字 “HMDI” ，可以看到 “S5P\_DEV\_I2C\_HDMIPHY” 。



```
.config - Linux/arm 3.0.15 Kernel Configuration

Search Results
-> Multimedia support (MEDIA_SUPPORT [=y])
-> Video capture adapters (VIDEO_CAPTURE_DRIVERS [=y])
-> Exynos Multimedia Devices (VIDEO_EXYNOS [=y])
-> Samsung TV driver for S5P platform (experimental) (VIDEO
-> Samsung HDMI Driver (VIDEO_EXYNOS_HDMI [=n])

Symbol: S5P_DEV_I2C_HDMIPHY [=y]
Type : boolean
Selected by: MACH_SMDK4X12 [=y] && ARCH_EXYNOS [=y] && ARCH_EXYNOS4 [ ]
```

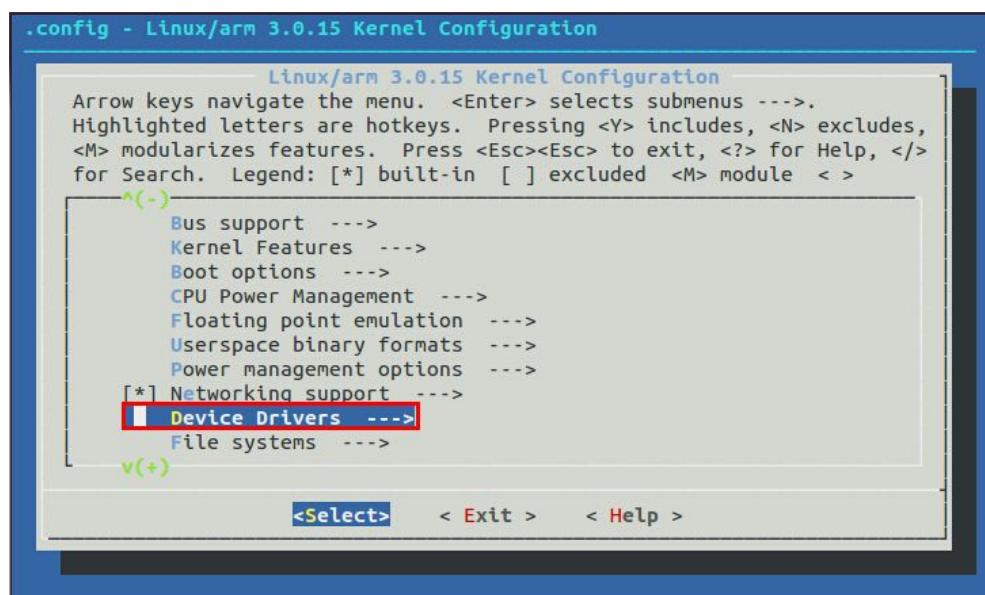
Symbol: VIDEO\_EXYNOS\_HDMI\_AUDIO\_I2S [=n]  
Type : boolean  
Prompt: Enable HDMI audio using I2S path  
Defined at drivers/media/video/exynos/tv/Kconfig:35

( 73%)

#### 9.4.10 SPI 总线

SPI 是总线，其它驱动如果要使用 SPI 总线，则需要先配置 SPI。

如下图，进入 “Device Drivers --->”。



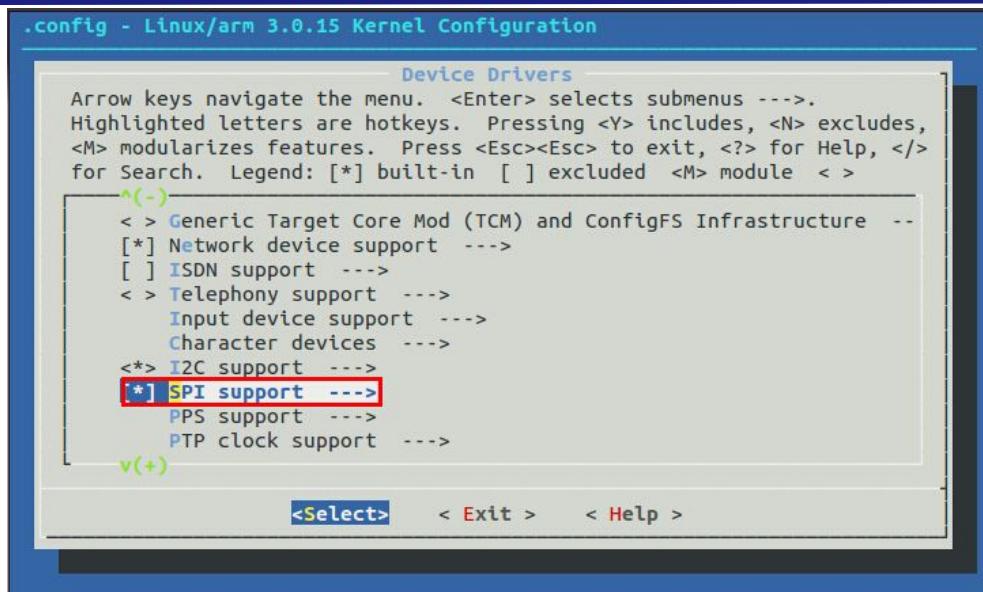
```
.config - Linux/arm 3.0.15 Kernel Configuration

Linux/arm 3.0.15 Kernel Configuration
Arrow keys navigate the menu. <Enter> selects submenus --->.
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes,
<M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </>
for Search. Legend: [*] built-in [ ] excluded <M> module < >

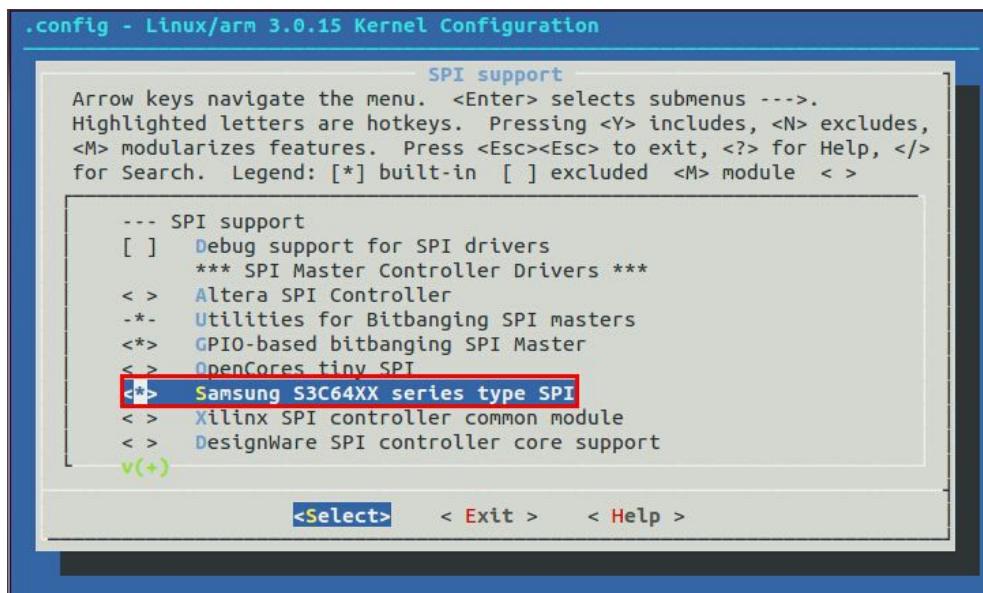
^(-)
  Bus support --->
  Kernel Features --->
  Boot options --->
  CPU Power Management --->
  Floating point emulation --->
  Userspace binary formats --->
  Power management options --->
[*] Networking support --->
  Device Drivers ---> [ ]
  File systems --->
v(+)

<Select>  < Exit >  < Help >
```

如下图，进入 “SPI support --->”。



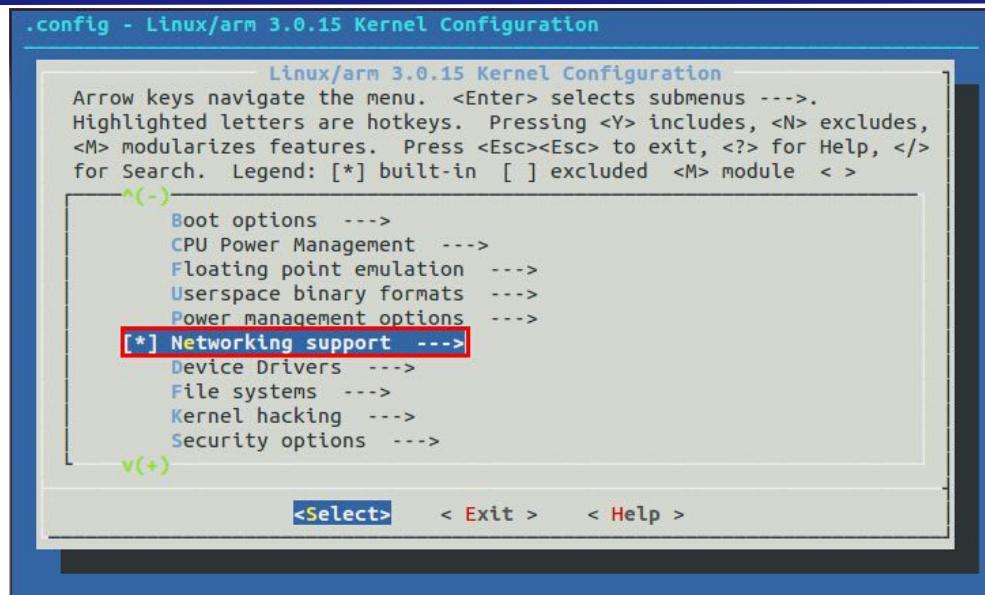
如下图所示，配置 “Samsung S3C64XX series type SPI”。



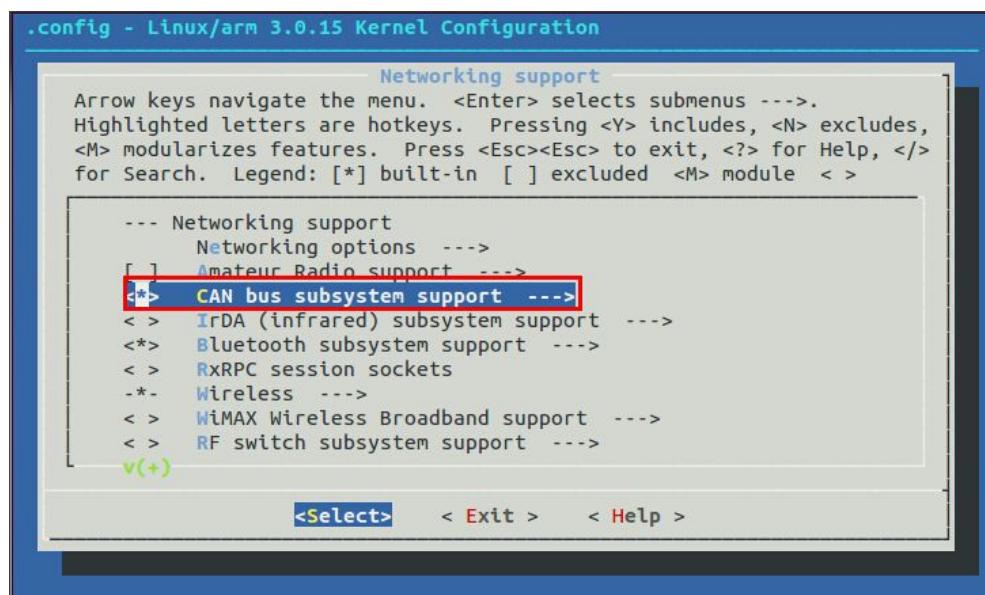
## 9.4.11 CAN 总线

CAN 驱动需要 SPI 总线支持。CAN 驱动的源码是 “net/can/af\_can.c”。

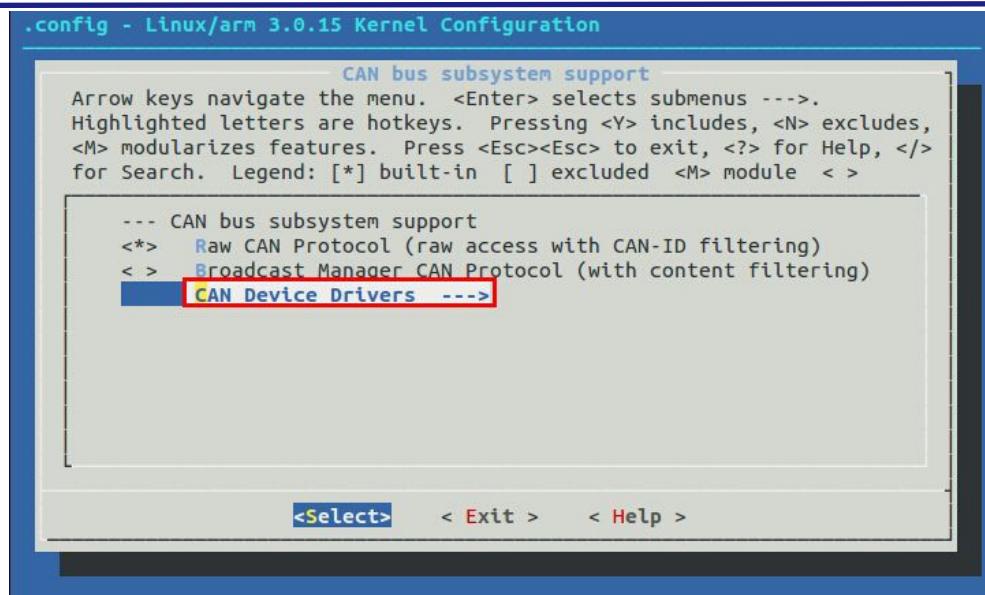
如下图，进入 “Networking support --->”。



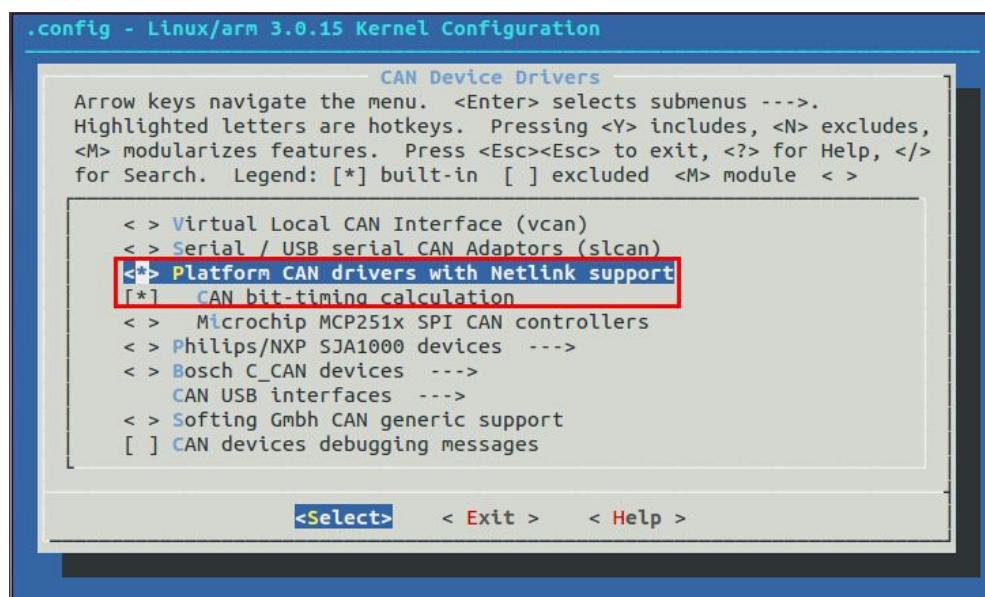
如下图，进入“CAN bus subsystem support --->”。



如下图，进入“CAN Device Drivers --->”。



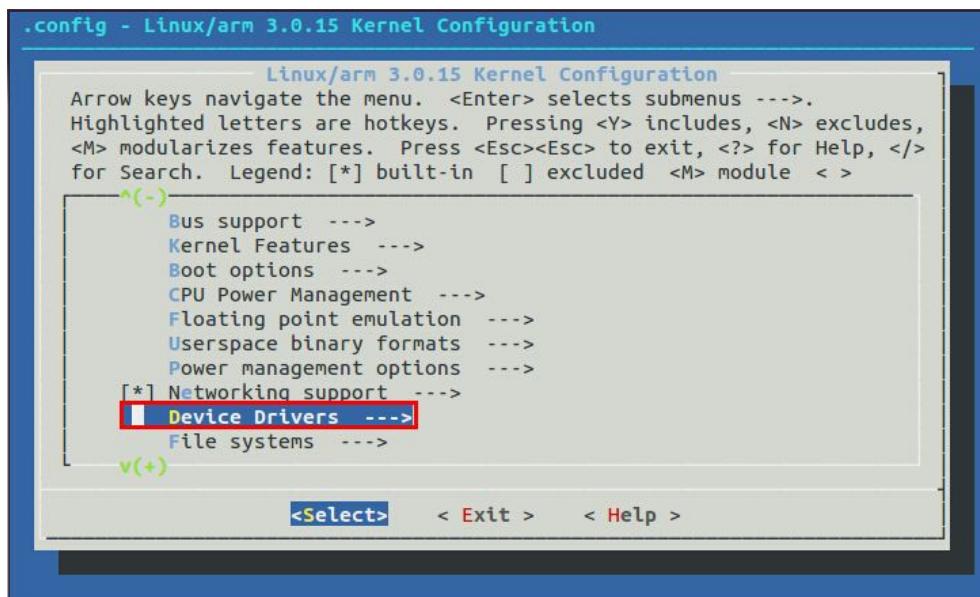
如下图，配置“Platform CAN drivers with Netlink support”和“CAN bit-timing calculation”。



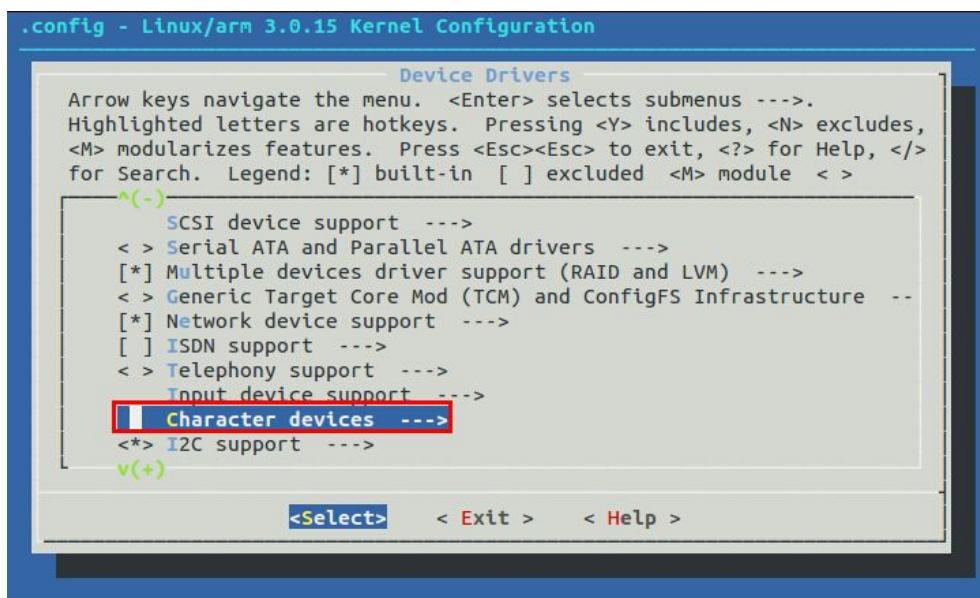
#### 9.4.12 串口 UART

该驱动对应源码“drivers/tty/serial/samsung.c”，对应设备节点“/dev/ttySAC\*”（\*表示数字0、1、2、3）。

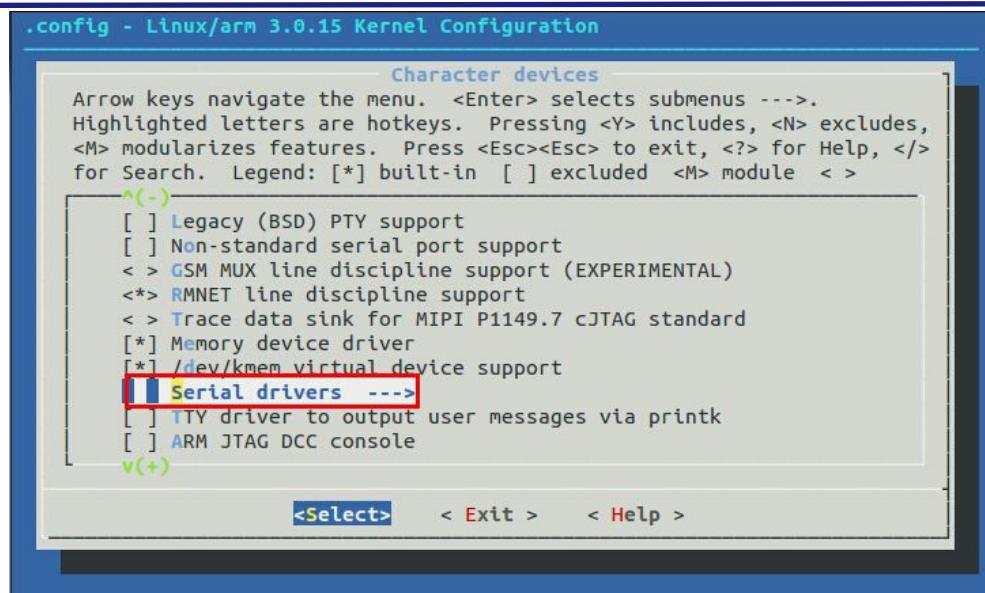
如下图，进入“Device Drivers --->”。



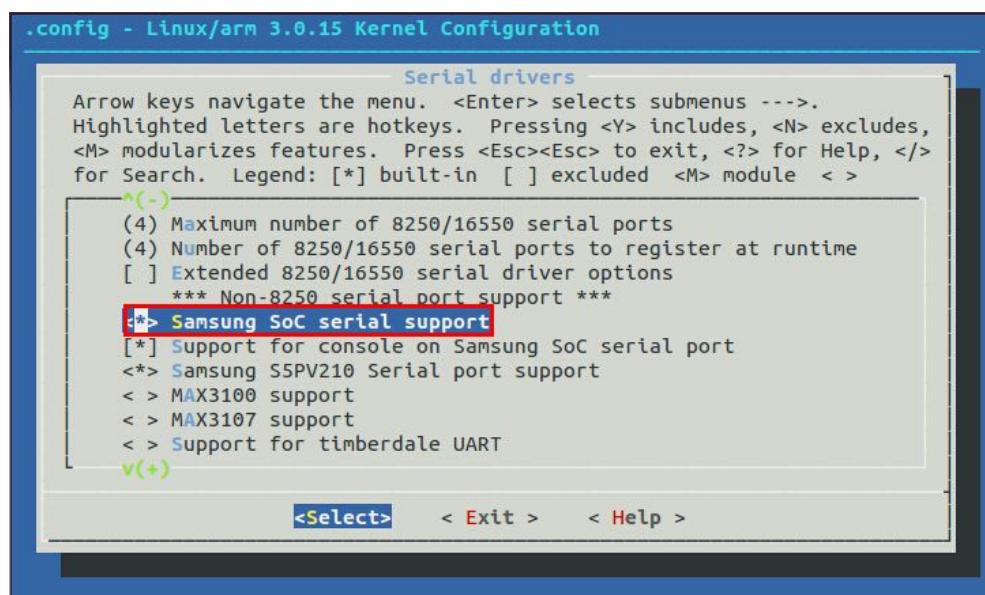
如下图，进入“Character devices --->”。



如下图，进入“Serial drivers --->”。



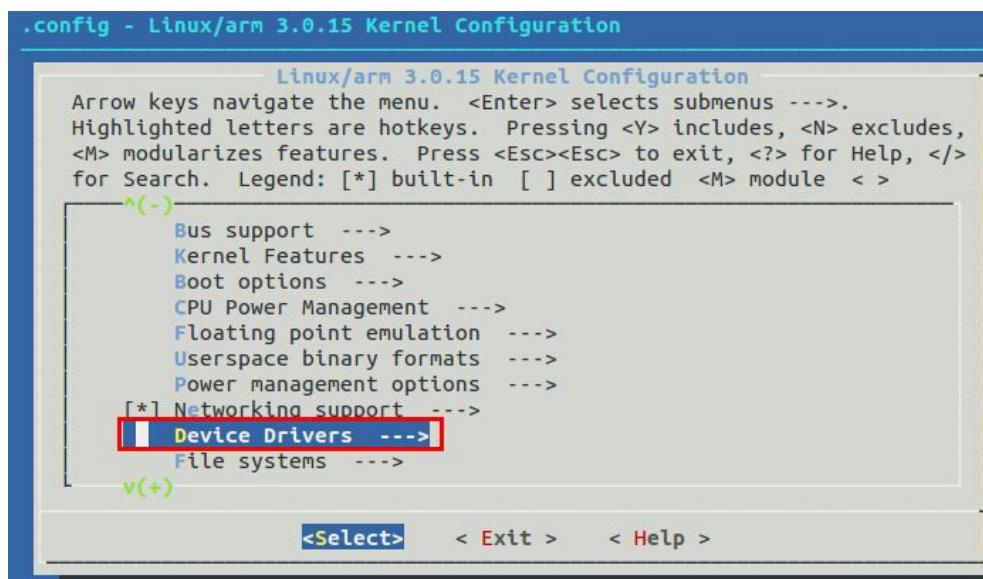
配置“Samsung SoC serial support”。



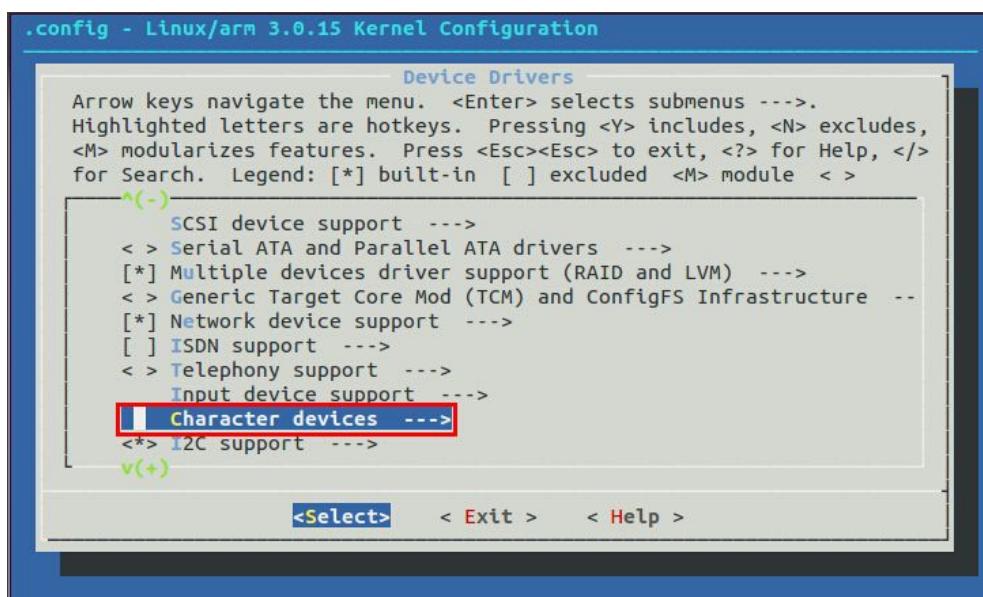
#### 9.4.13 串口虚拟控制台 console

在 iTOP-4412 开发平台中，会经常用到串口虚拟控制台。例如，启动过程中的内核打印信息，就是使用串口虚拟控制台。但是当用户最终程序调试完成后，可能需要关闭虚拟控制台，下面介绍一下这个串口虚拟控制台是如何配置。

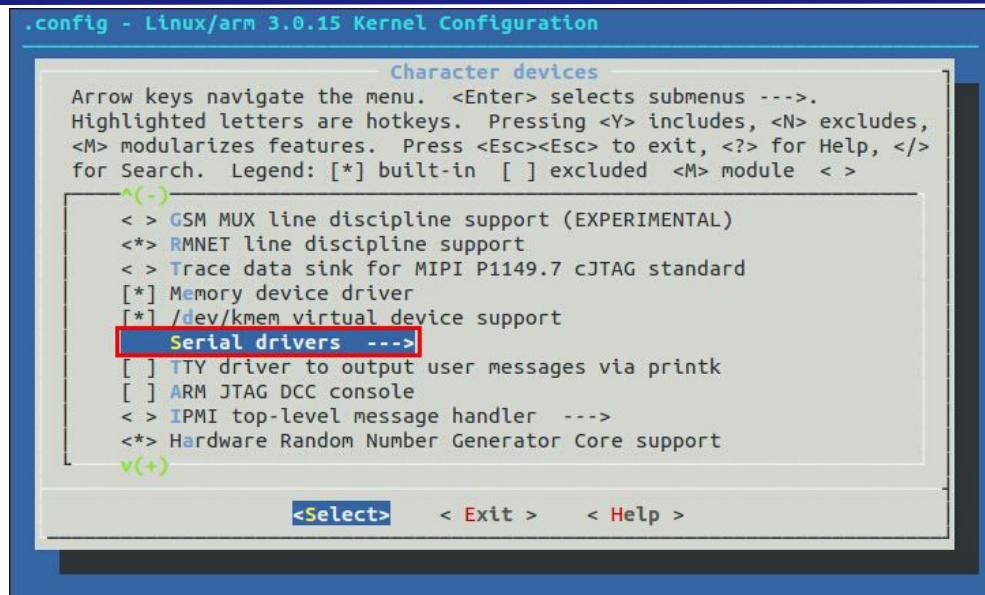
如下图，进入“Device Drivers --->”。



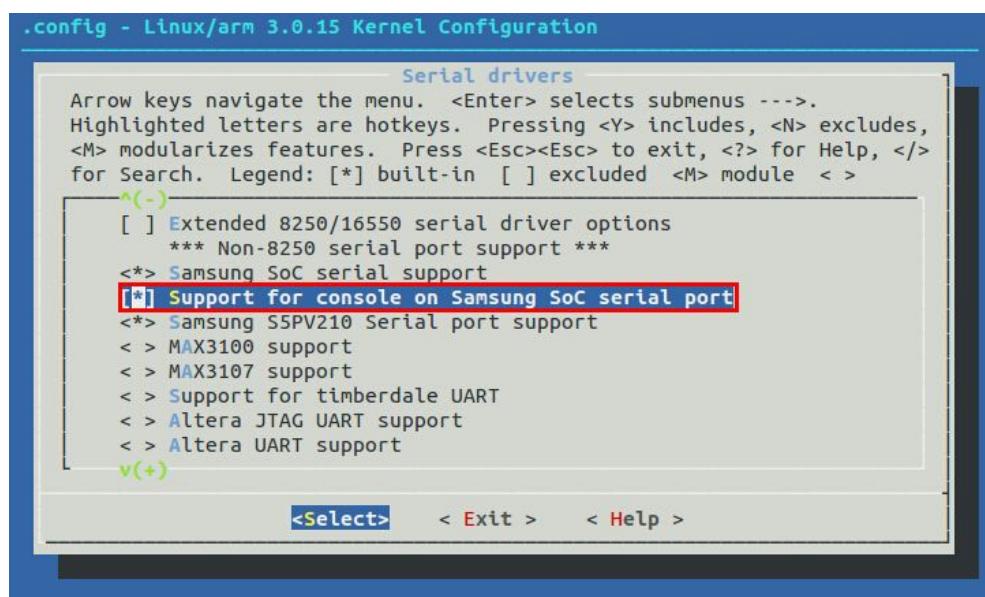
如下图，进入“Character devices --->”。



如下图，进入“Serial drivers --->”。

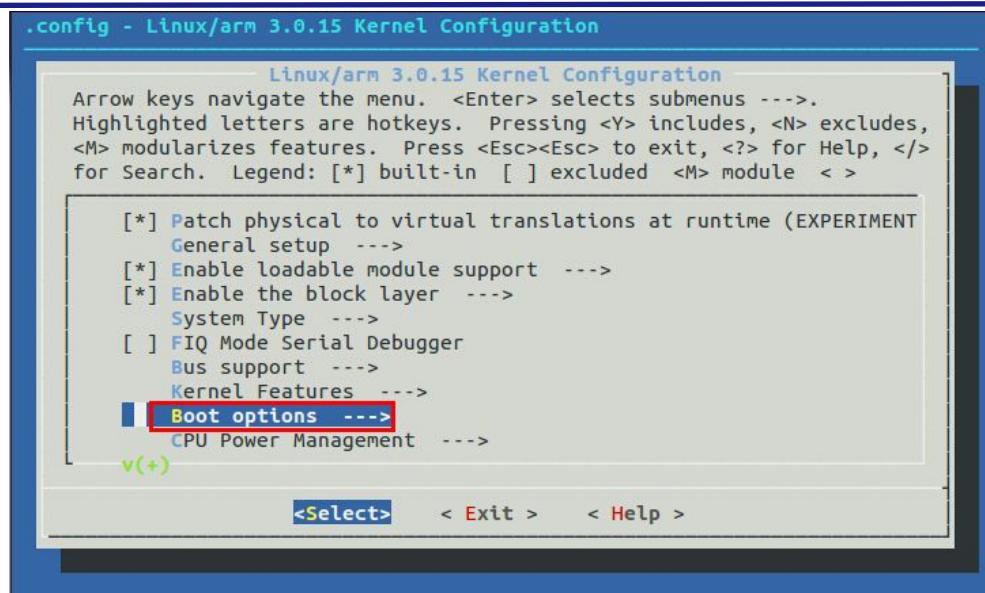


如下图，设置“Support for console on Samsung SoC serial port”。取消配置，则内核在启动的时候，将不会打印信息。

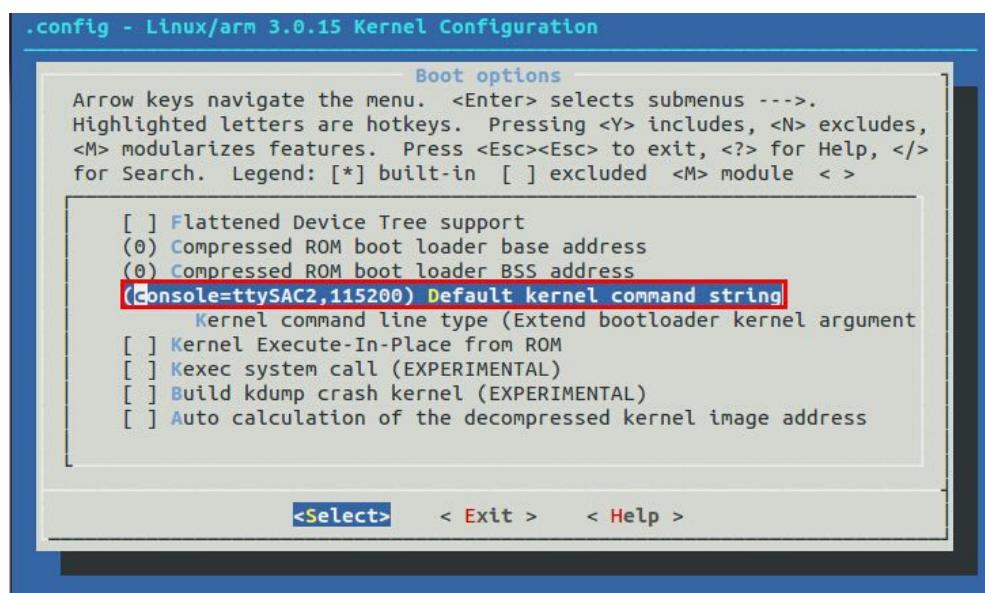


虚拟控制台在内核中可以修改串口号和波特率，也可以关闭。

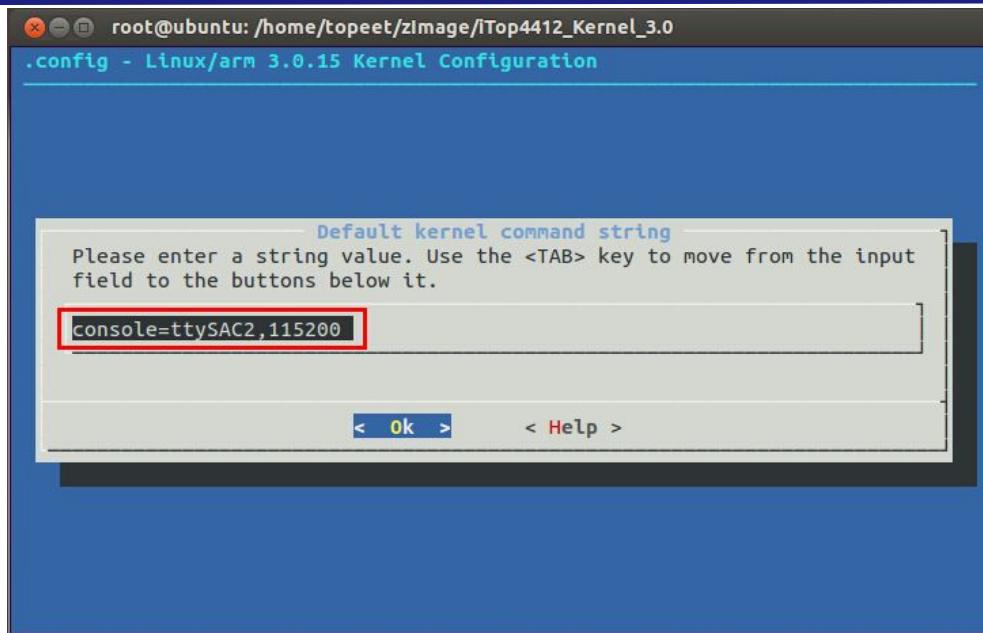
如下图，进入“Boot options --->”。



如下图，进入“Default kernel command string”。



如下图，默认是“console=ttySAC2,115200”，按照这个格式可以修改调试端口和波特率，如果“console=null”，则不打印。

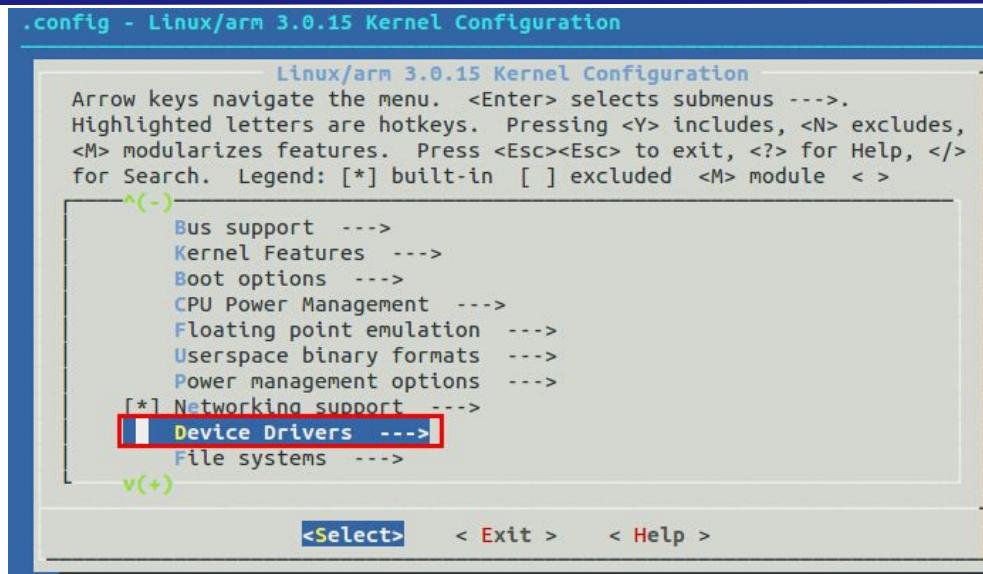


#### 9.4.14 USB 转串口 PL2303

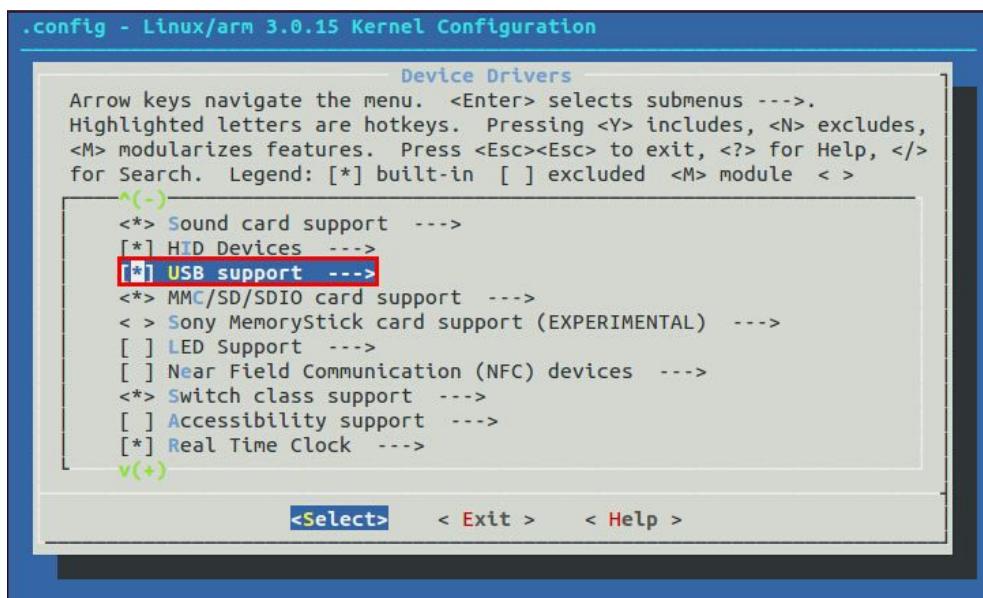
USB 驱动，在缺省状态中就已经配置，但是 USB 转串口的驱动在默认状态下并没有编译进内核，如果需要使用开发板上的 USB 转串口驱动，则设置 menuconfig，重新编译内核。

下面个介绍一下如何支持 PL2303 的 USB 转串口设备。

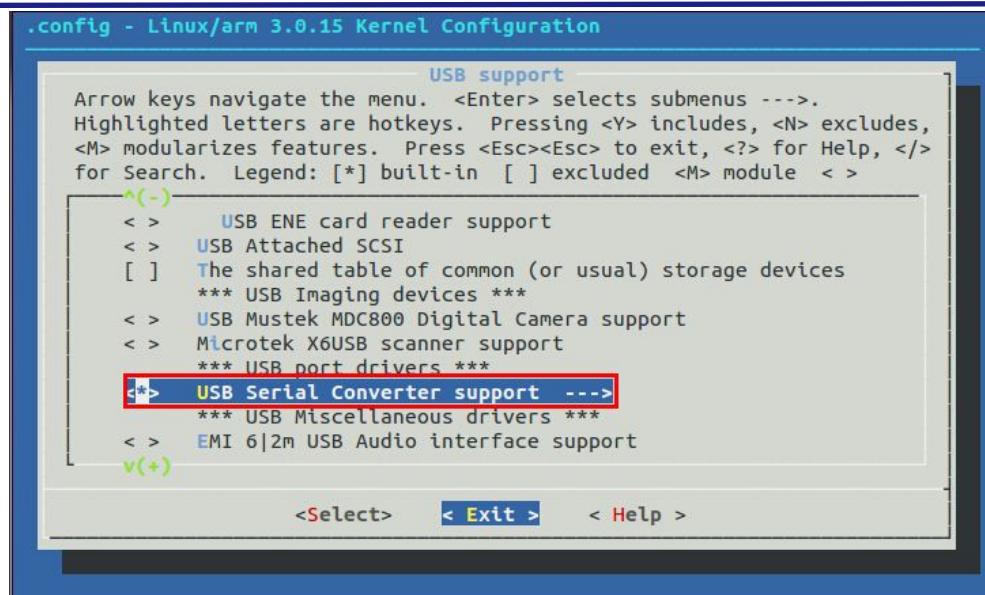
如下图，进入 “Device Drivers --->” 。



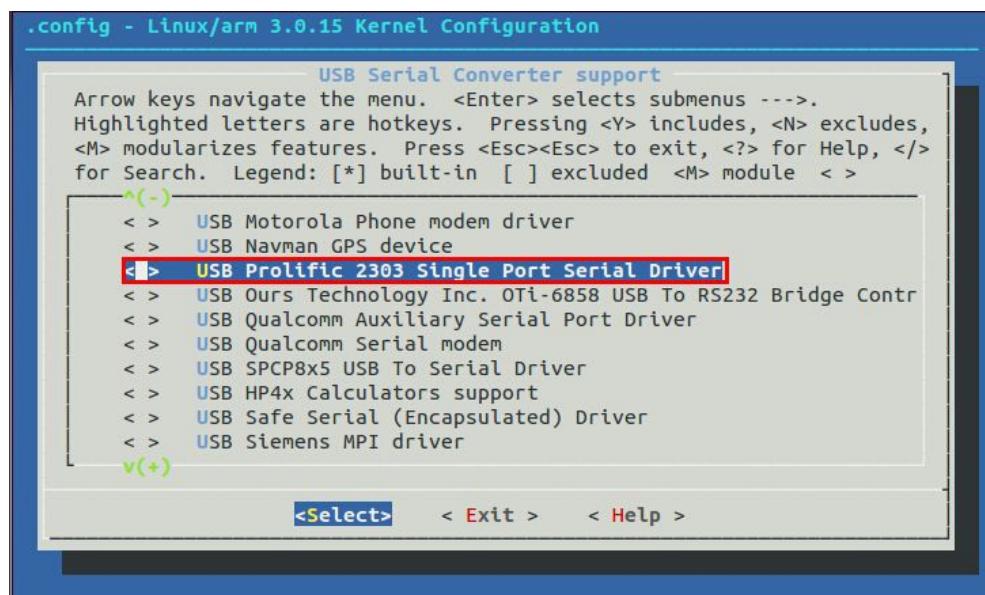
如下图，进入“USB support --->”。



如下图，进入“USB Serial Converter support --->”。



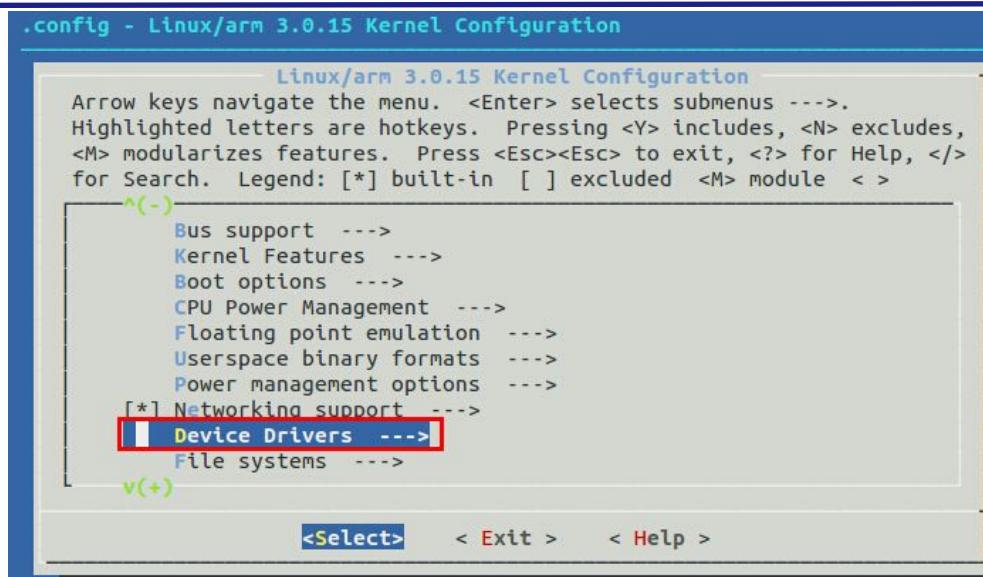
如下图，配置“USB Prolific 2303 Single Port Serial Driver”。这样就可以支持PL2303的USB转串口驱动。



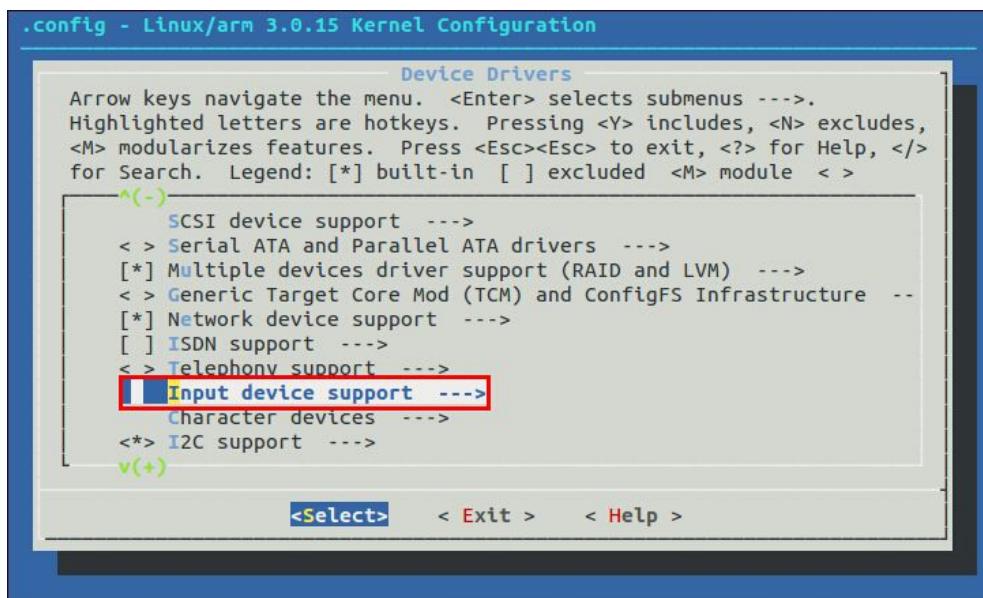
#### 9.4.15 触摸屏 TP

触摸屏驱动的源码是“drivers/input/touchscreen/ft5x06\_ts.c”。

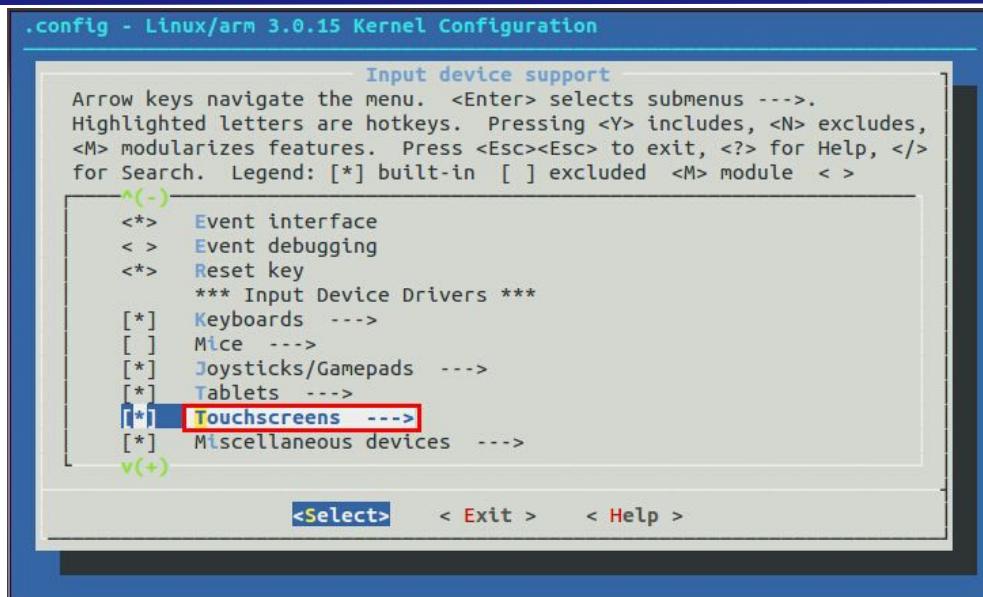
如下图，进入“Device Drivers --->”。



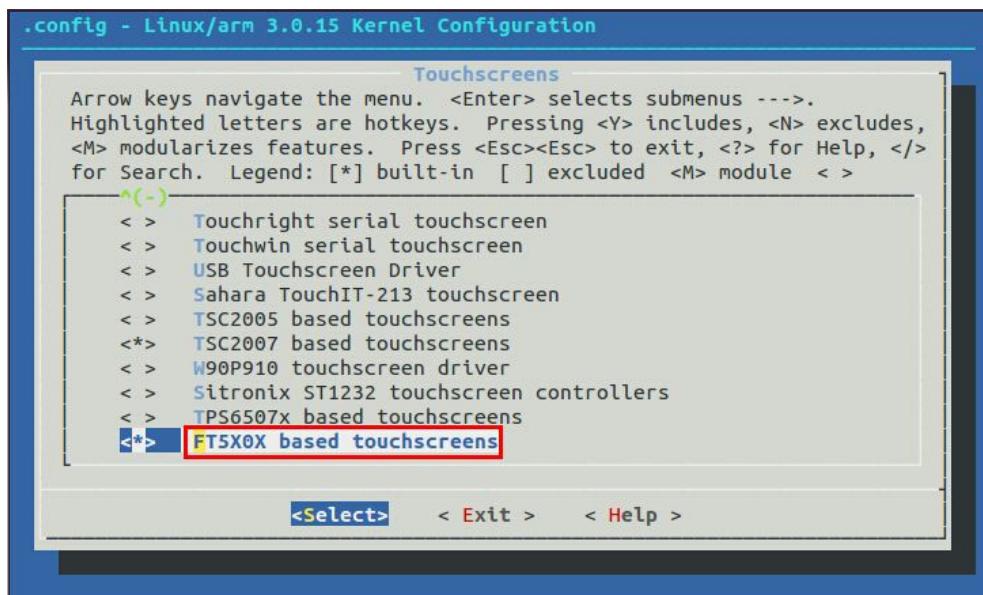
如下图，进入 “Input device support --->”。



如下图，进入 “Touchscreens --->”。



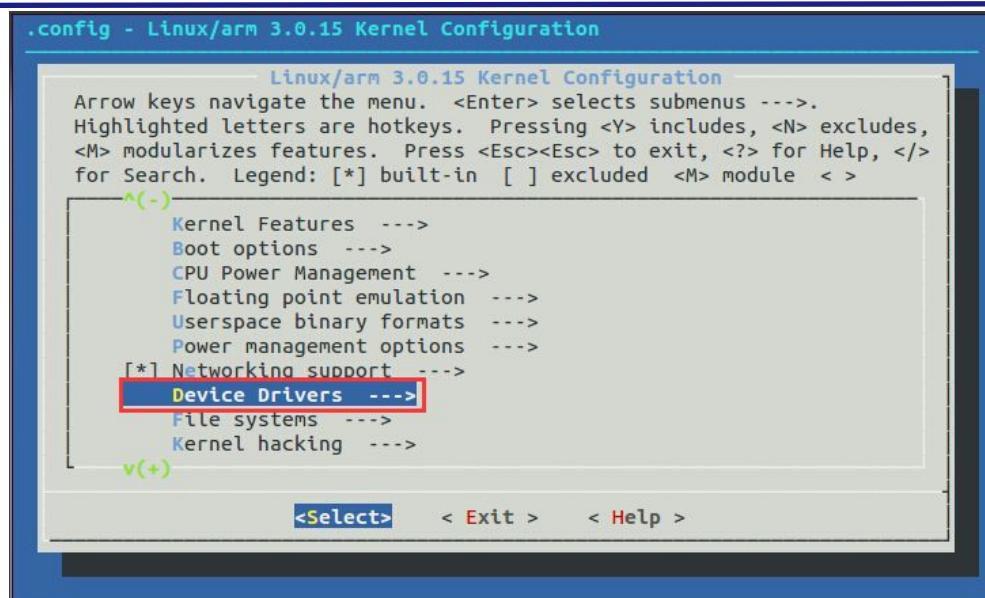
如下图，配置“FT5X0X based touchscreens”。



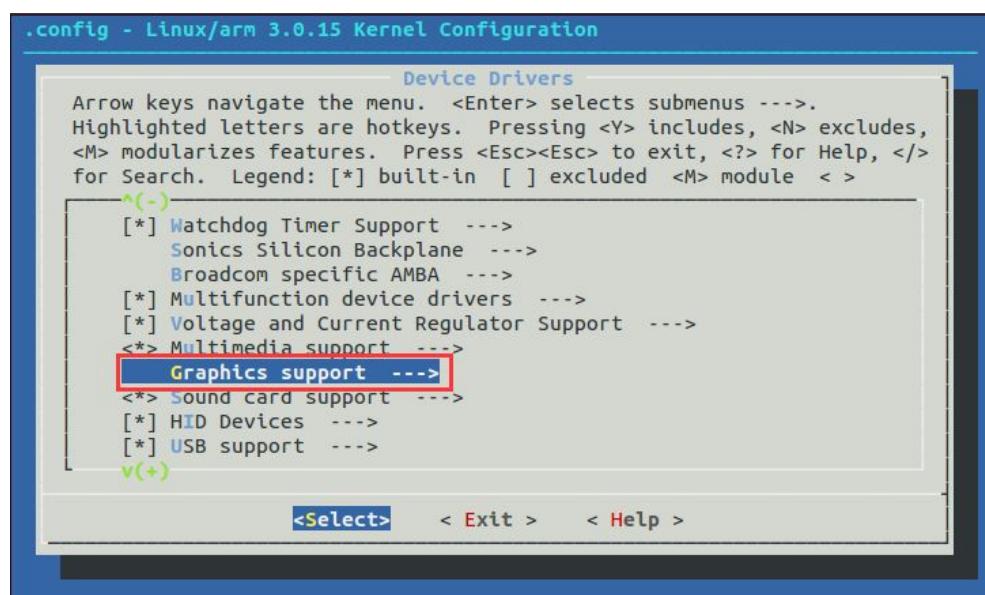
## 9.4.16 显卡 Graphics Card

显卡驱动对应源码是“drivers/video/samsung/s3cfb\_wa101s.c”。

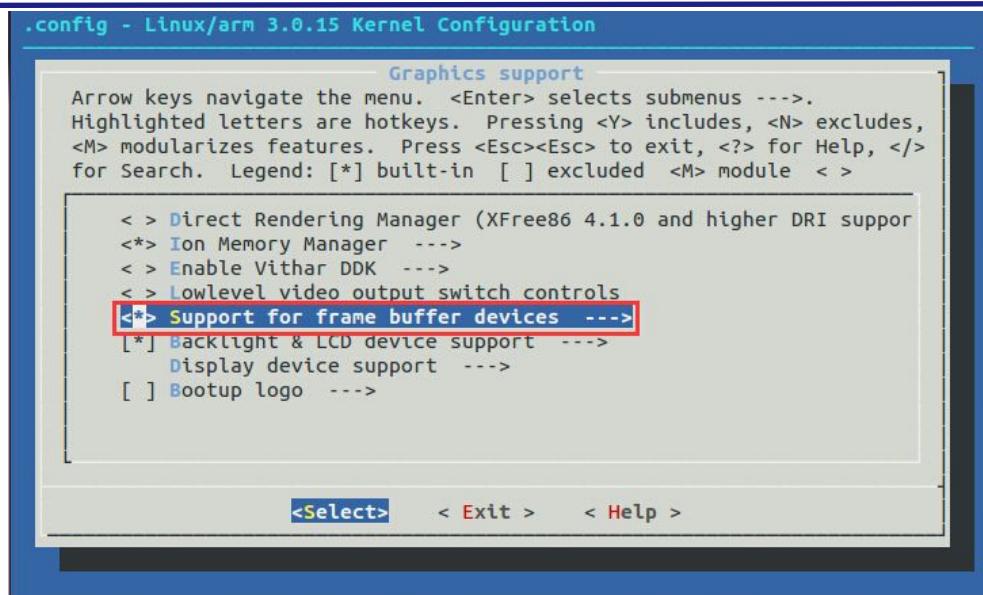
如下图，进入“Device Drivers --->”。



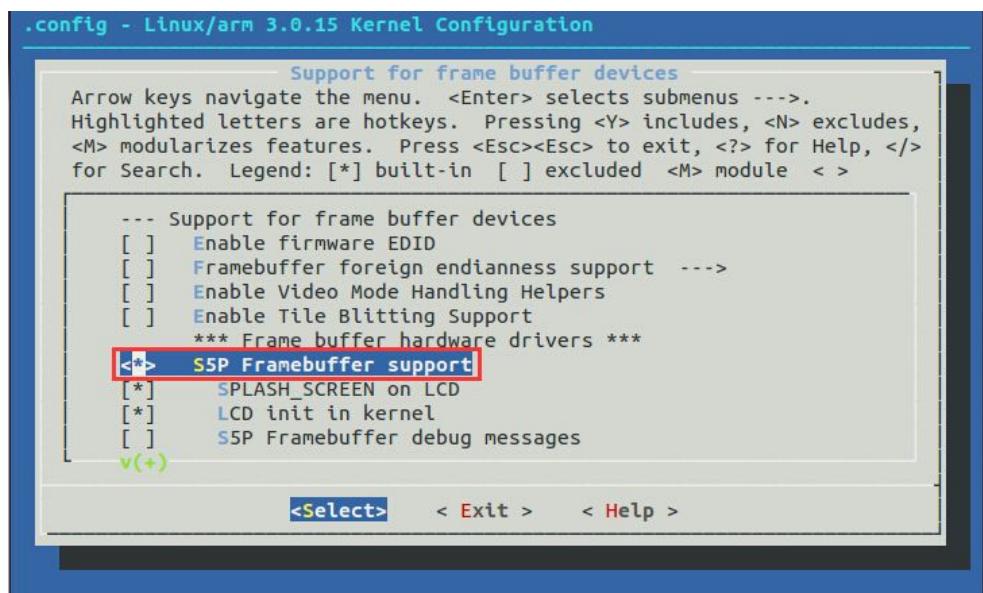
如下图，进入“Graphics support --->”。



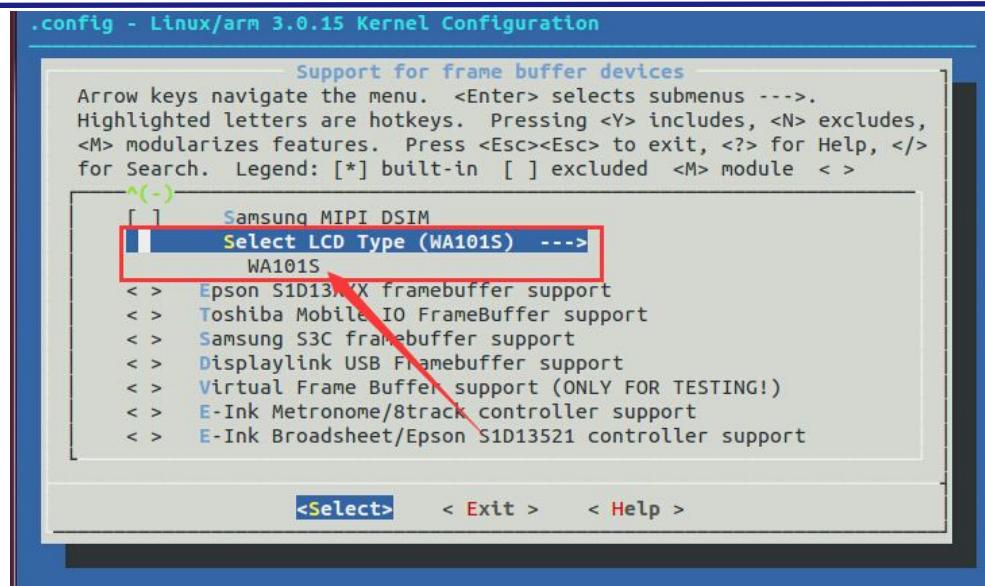
如下图，配置“Support for frame buffer devices --->”。



如下图，配置 “S5P Framebuffer support --->”。



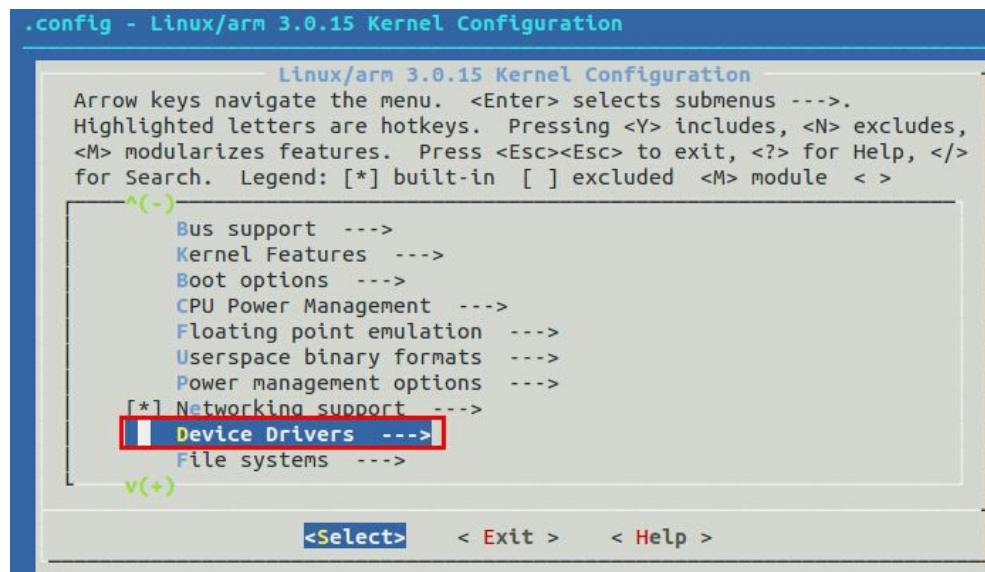
如下图，配置 “Select LCD Type --->”。



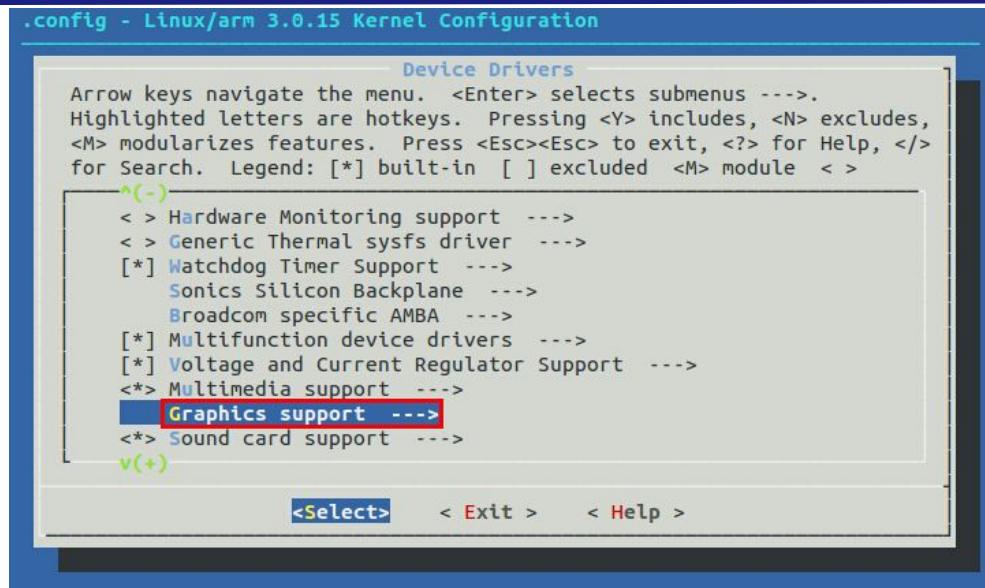
#### 9.4.17 背光 Backlight

背光驱动对应源码是 “drivers/video/backlight/backlight.c”

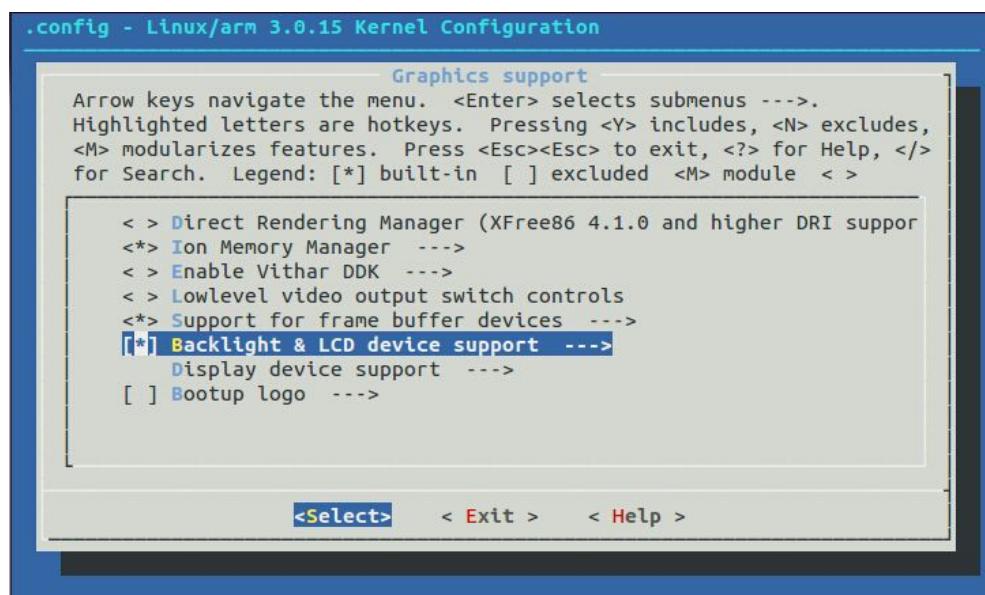
如下图，进入 “Device Drivers --->”。



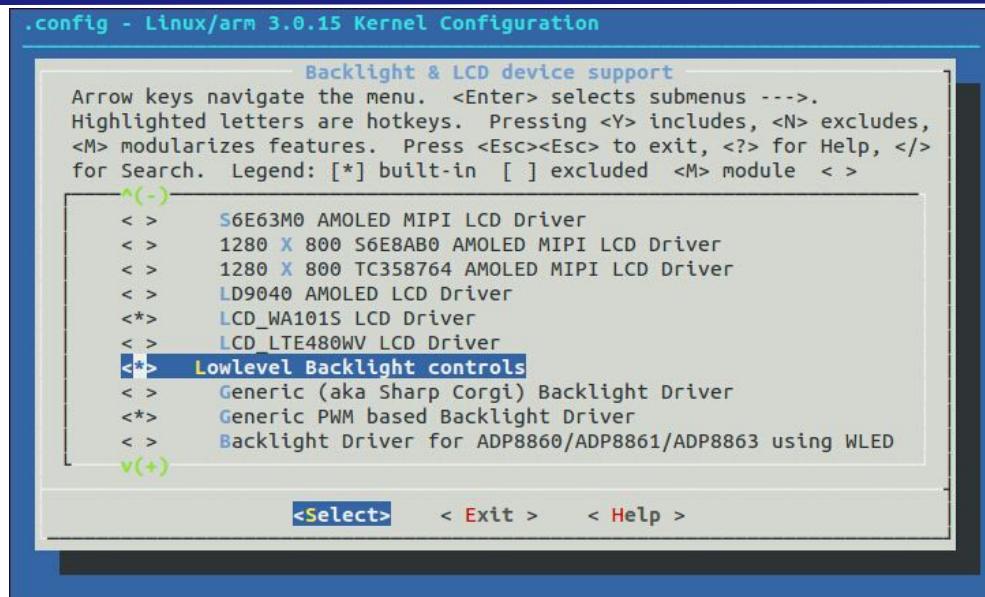
如下图，进入 “Graphics support --->”。



如下图，进入“Backlight & LCD device support --->”。



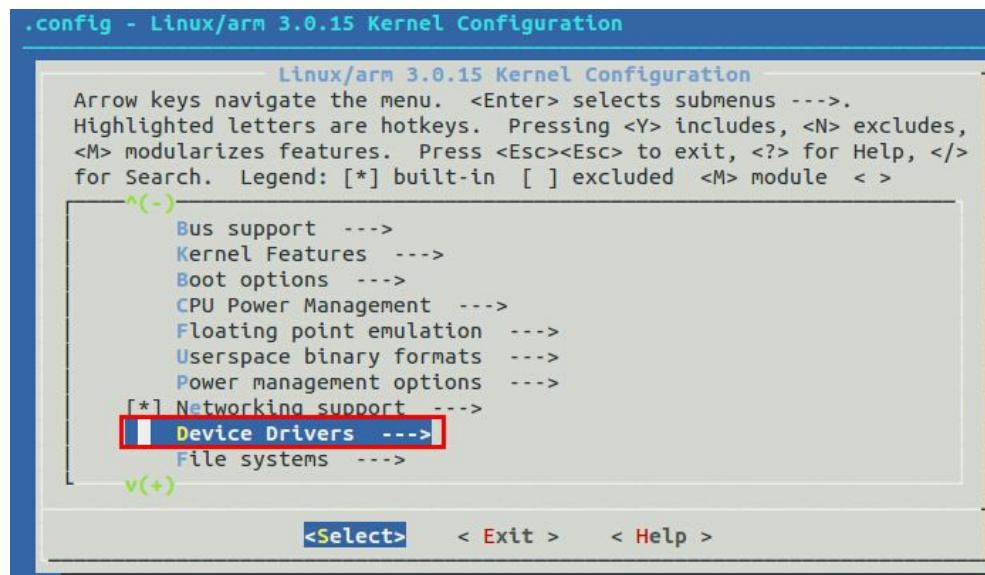
如下图，配置“Lowlevel Backlight controls”。



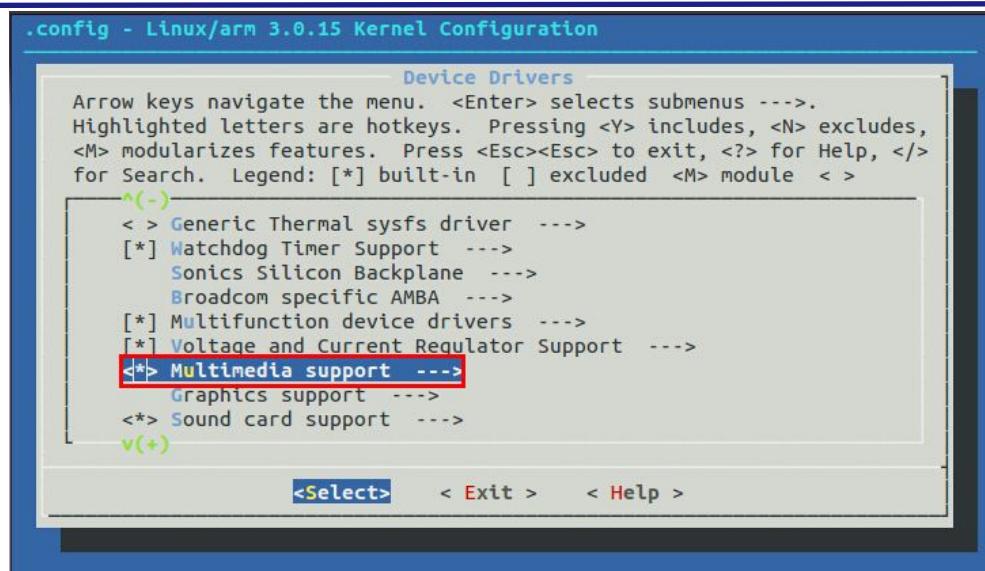
## 9.4.18 高清晰度多媒体接口 HDMI\_HPD

该驱动对应源码"drivers/media/video/samsung/tvout/s5p\_tvout\_hpd.c"。对应设备节点 "/dev/ HPD" 。

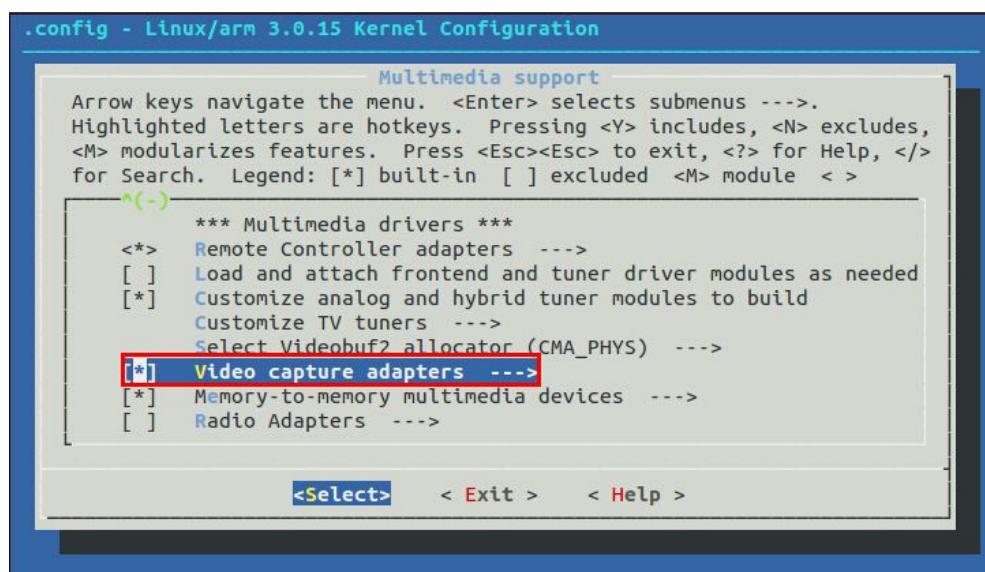
如下图，进入 "Device Drivers --->" 。



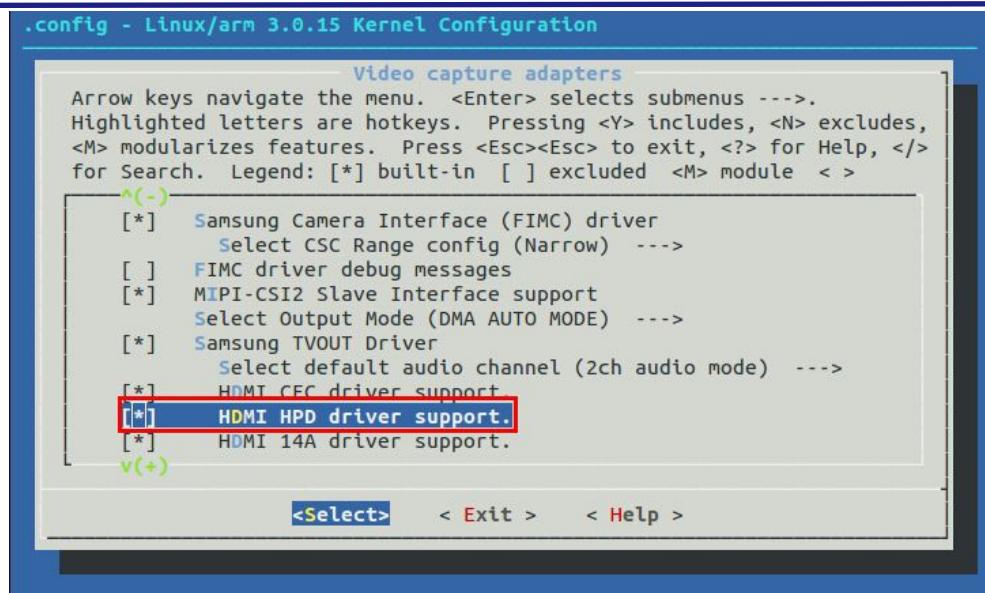
如下图，进入 "Multimedia support --->" 。



如下图，进入“Video capture adapters --->”。



如下图，配置“HDMI HPD driver support”。



#### 9.4.19 高清晰度多媒体接口 HDMI\_Audio

该驱动对应源码 “drivers/media/video/samsung/tvout/hw\_if/mixer.c” 。

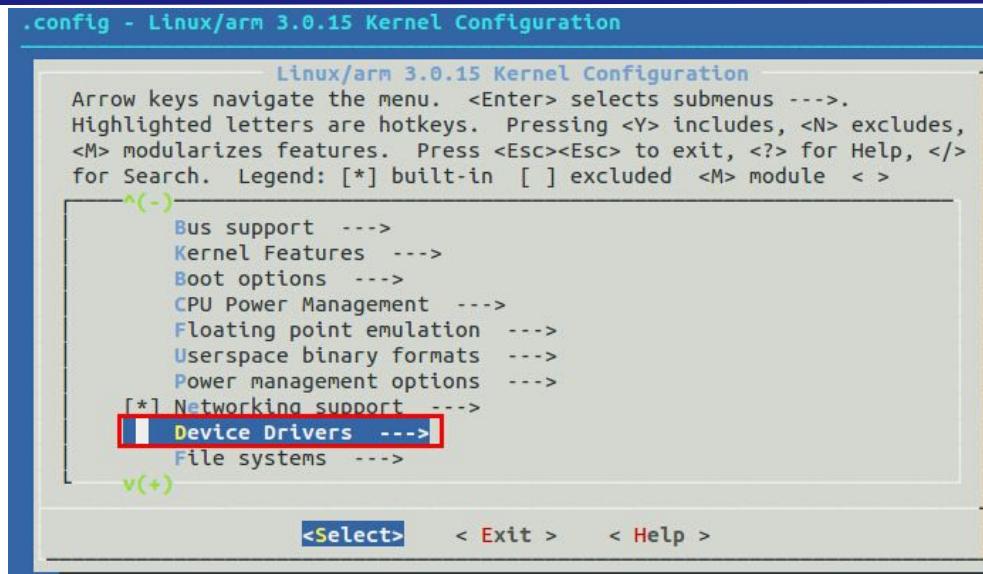
配置 “显卡 Graphics Card” ，内核即可支持该功能。

参考 9.4.16 小节。

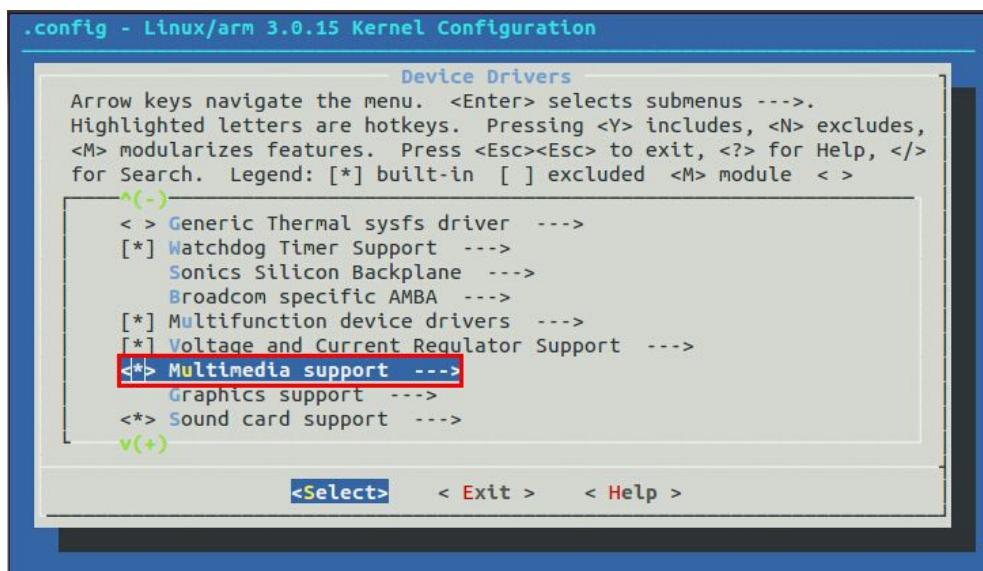
#### 9.4.20 USB 摄像头 Camera

该驱动对应源码为 “drivers/media/video/uvc/\*” 。

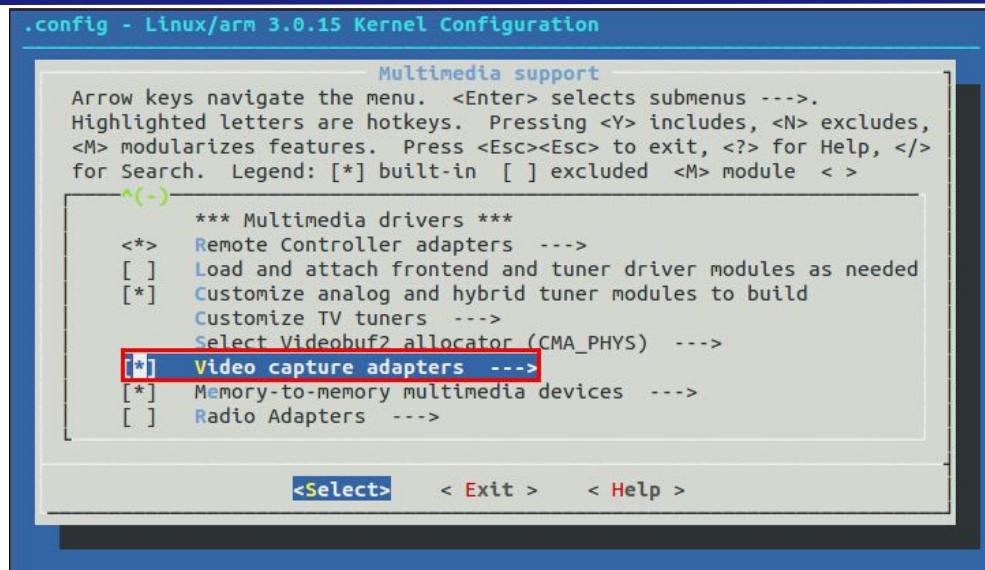
如下图，进入 “Device Drivers --->” 。



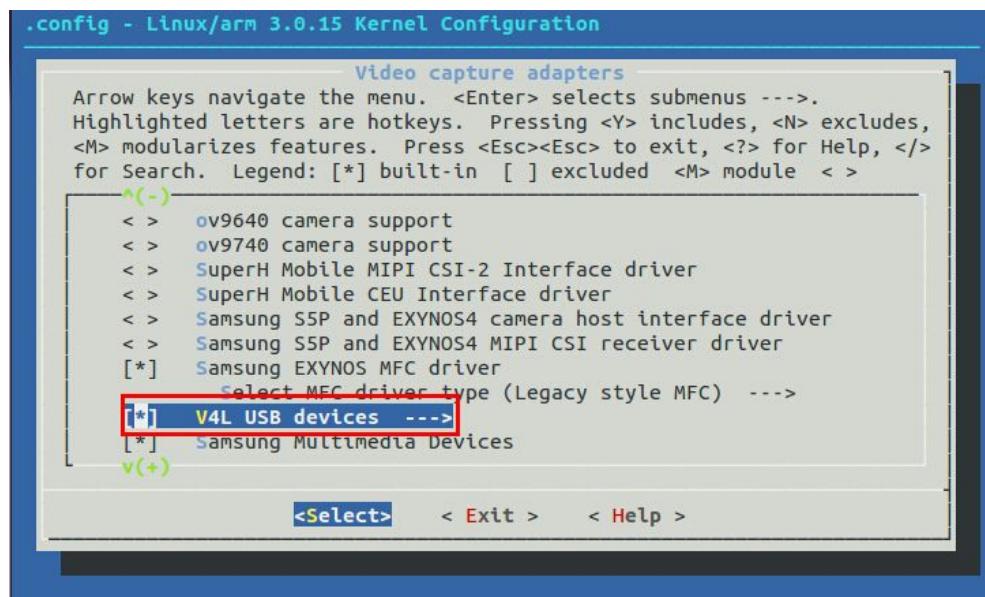
如下图，进入“Multimedia support --->”。



如下图，进入“Video capture adapters --->”。



如下图，配置“V4L USB devices --->”。

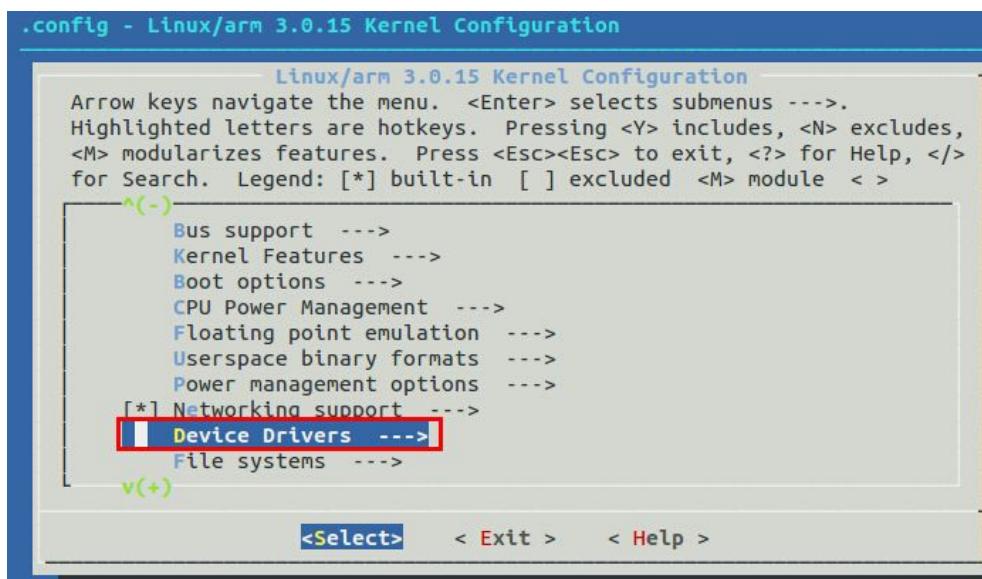


#### 9.4.21 USB 键盘和键盘

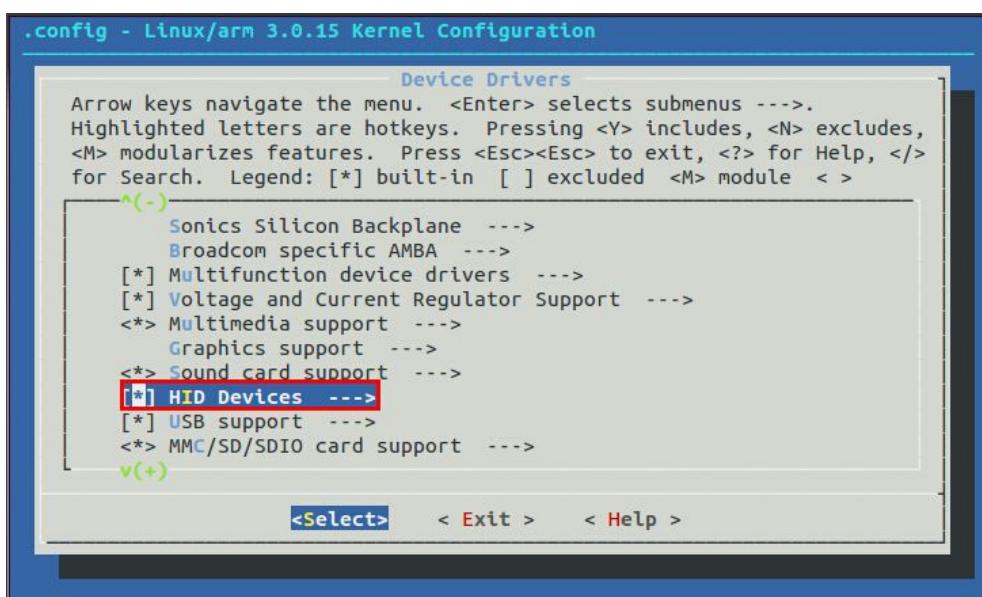
USB 键盘驱动对应源码为“drivers/hid”，对应的设备节点是“/dev/input/mice”。

USB 键盘该驱动对应源码为“drivers/hid”，对应的设备节点是“/dev/input/event\*”。

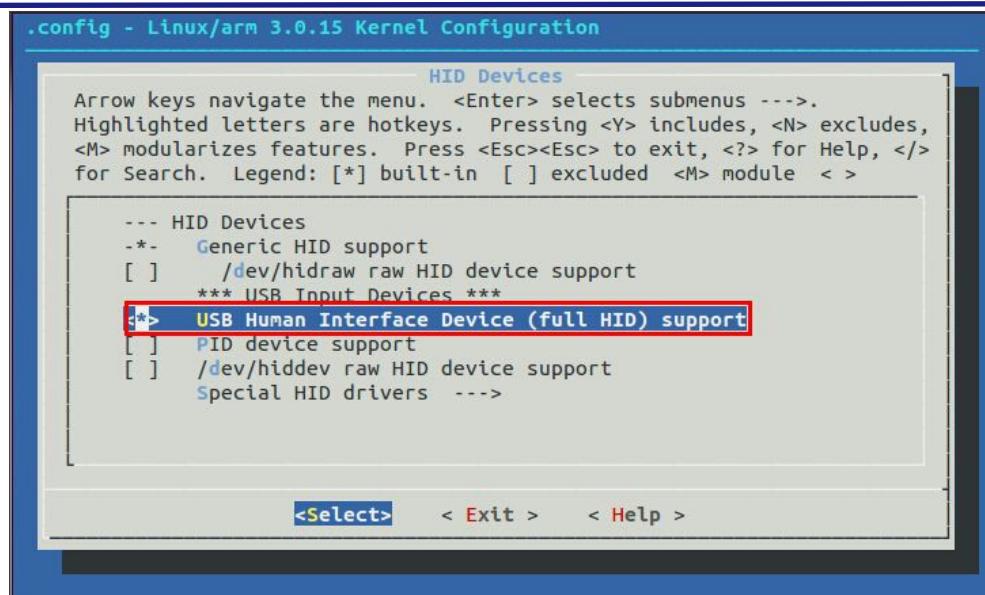
如下图，进入“Device Drivers --->”。



如下图，进入“HID Devices --->”。



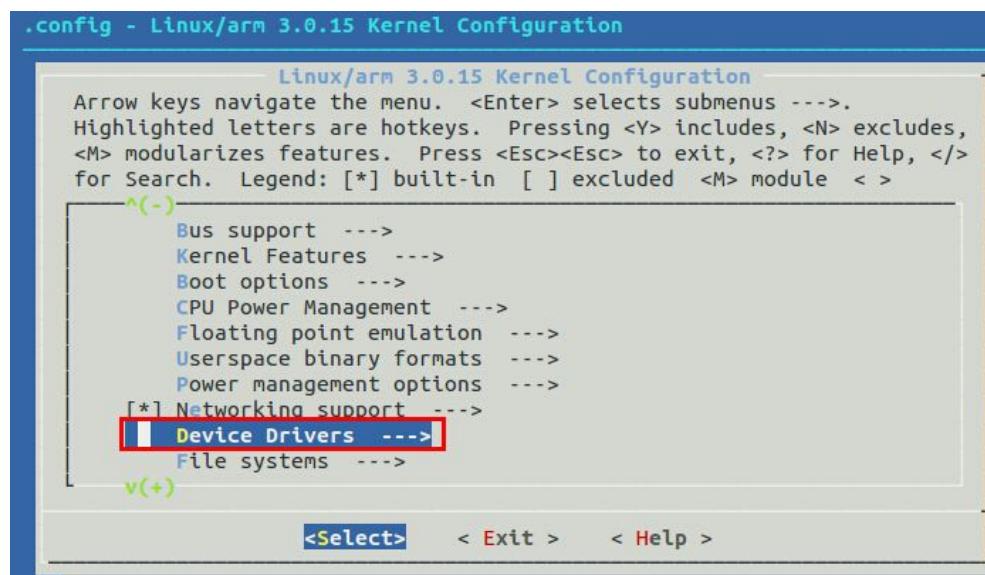
如下图，配置“USB Human Interface Device (full HID) support”。



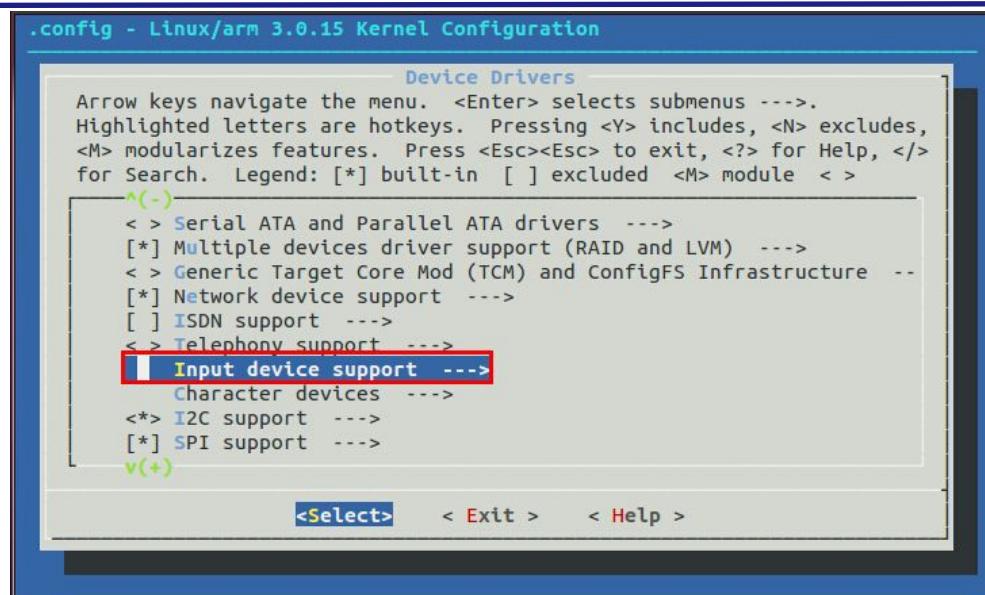
## 9.4.22 矩阵键盘 GPIO\_KEYS

矩阵键盘驱动对应源码为“drivers/input/keyboard/gpio\_keys.c”。

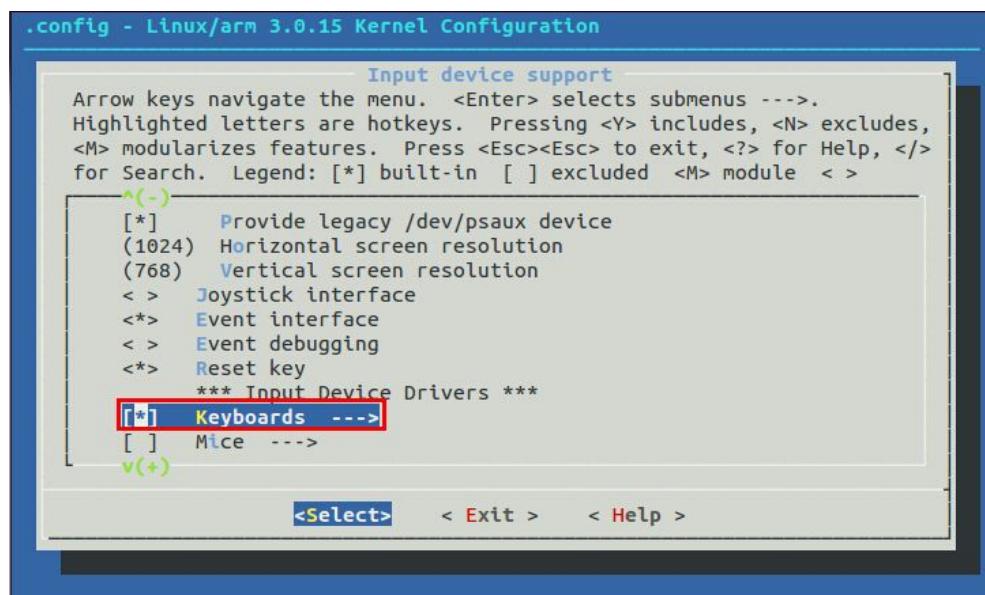
如下图，进入“Device Drivers --->”。



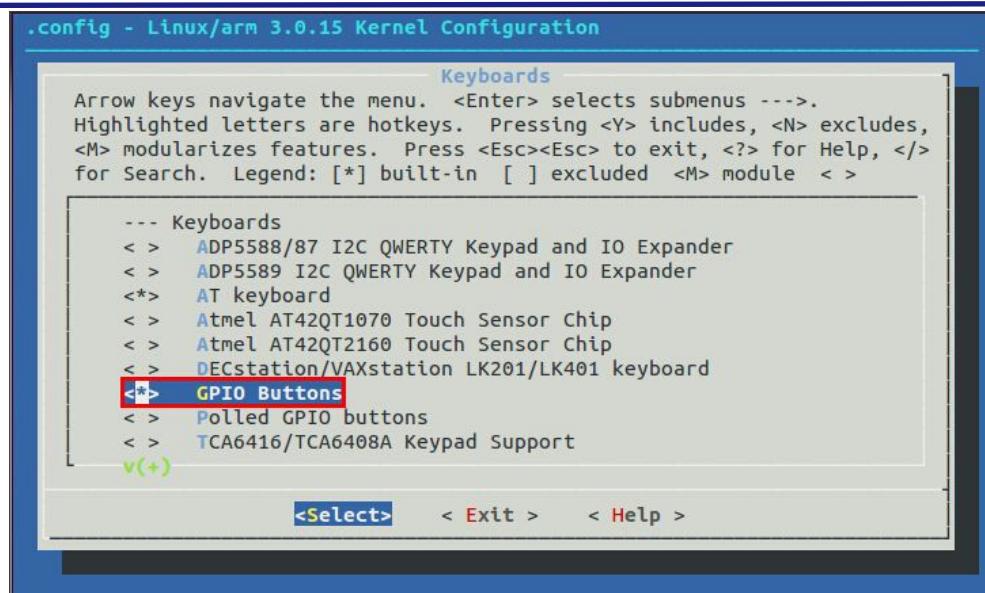
如下图，进入“Input device support --->”。



如下图，进入“Keyboards --->”。



如下图，设置“GPIO Buttons”。

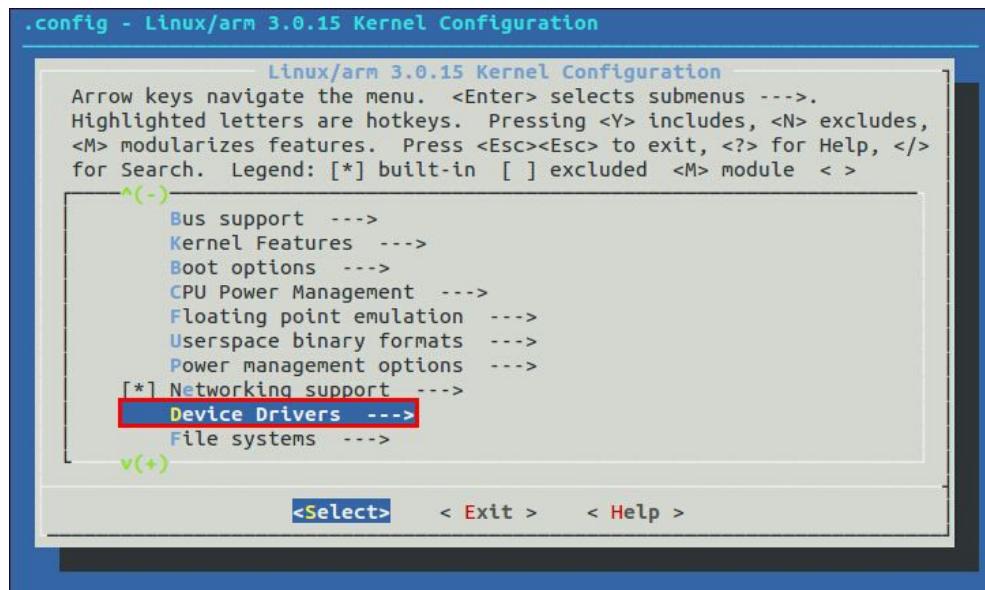


## 9.4.23 U 盘

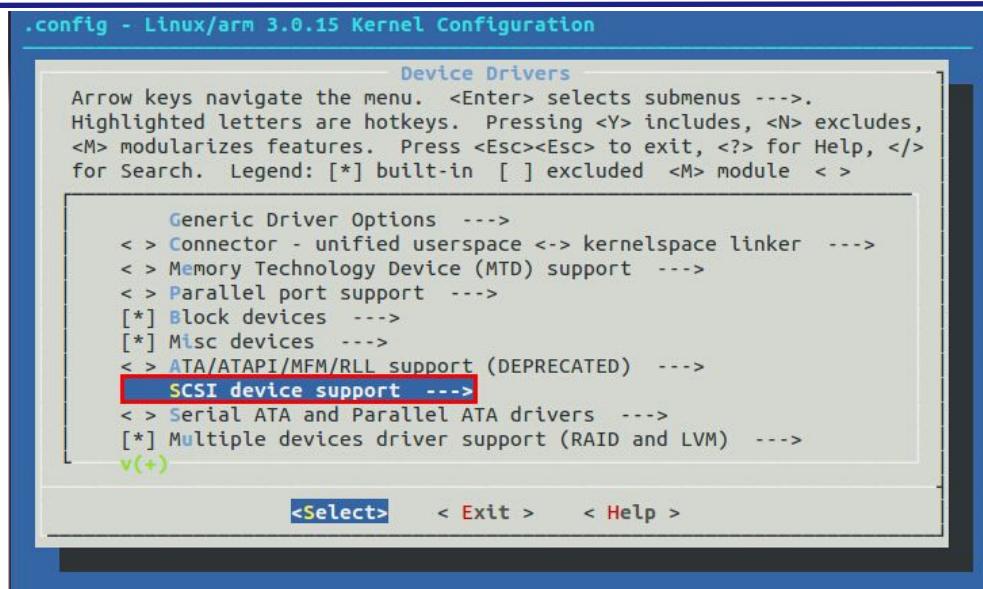
该驱动对应源码 “drivers/usb/storage/usb-storage.c” 。

U 盘首先需要增加 SCSI 支持。

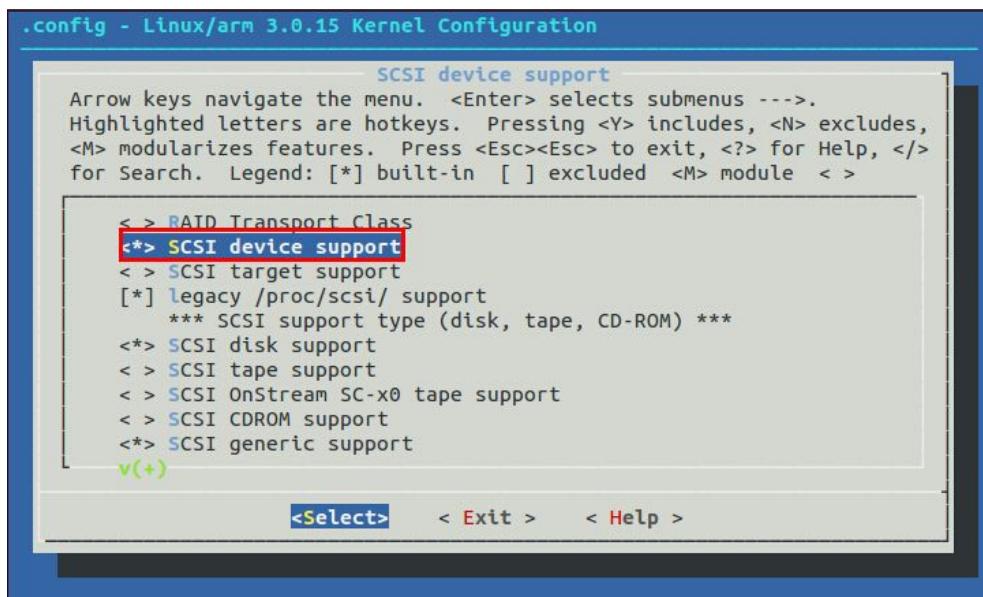
如下图，进入 “Device Drivers --->” 。



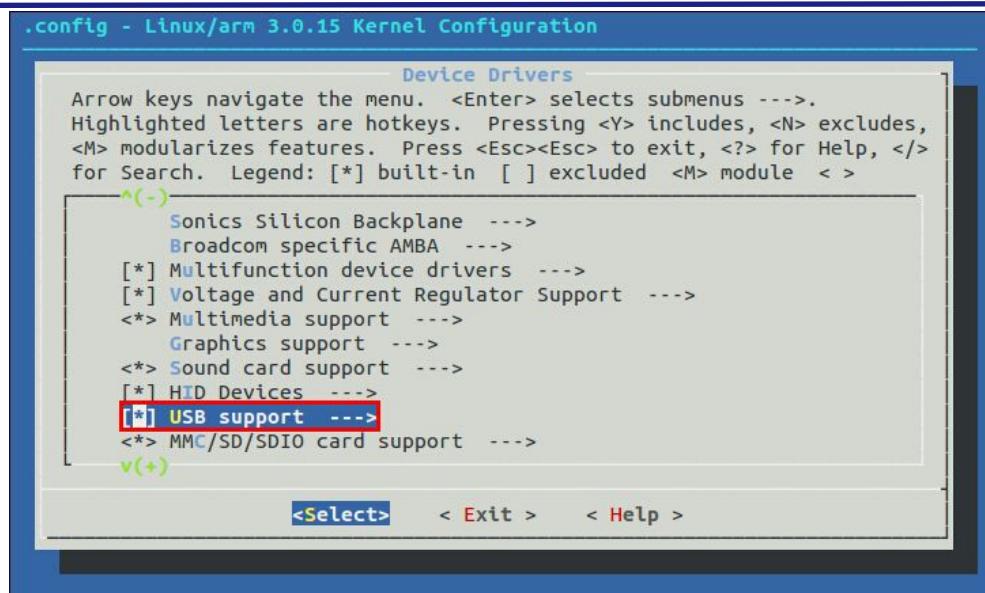
如下图，进入 “SCSI device support --->” 。



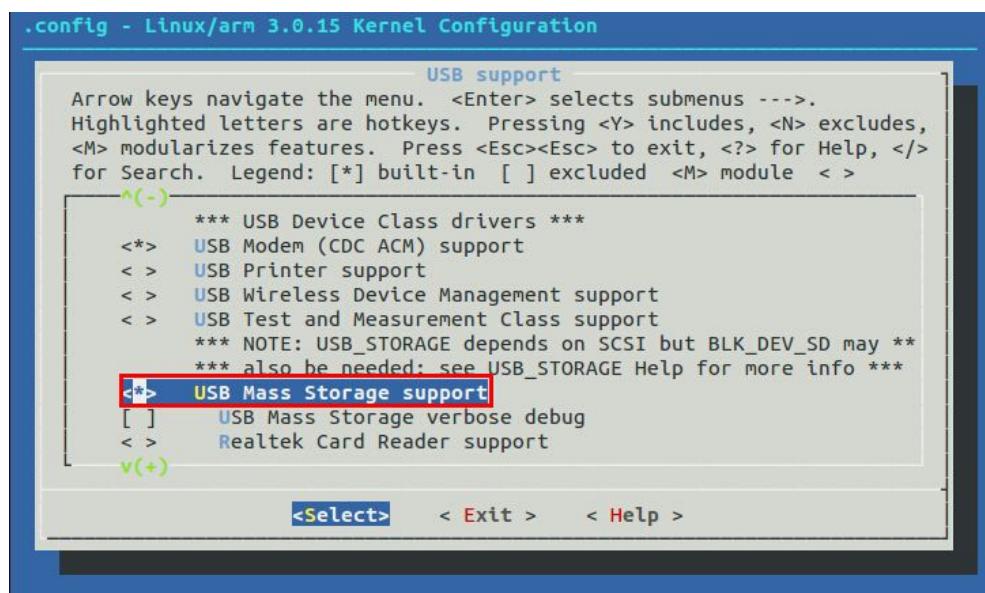
如下图，配置“SCSI device support”。



返回“Device Drivers”菜单，进入“USB support --->”。



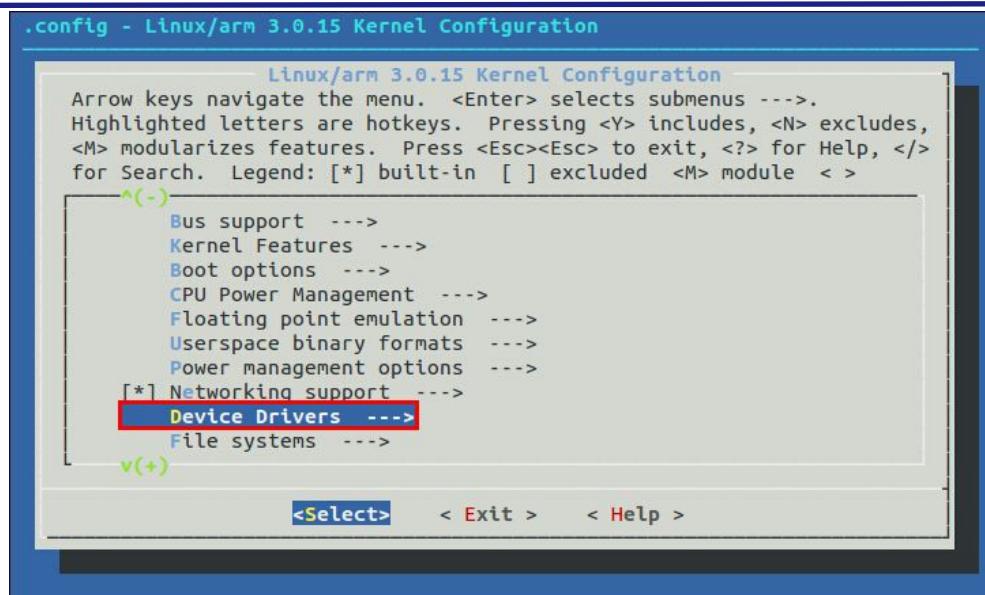
如下图，配置“USB Mass Storage support”。



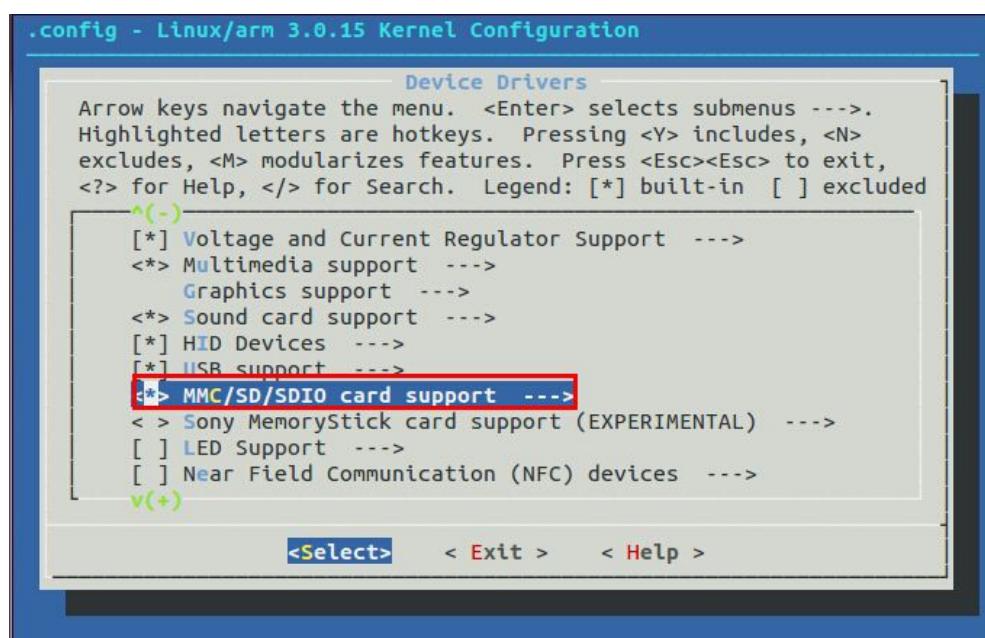
## 9.4.24 SD 卡/eMMC

该驱动对应源码“drivers/mmc/host/sdhci-s3c.c”。

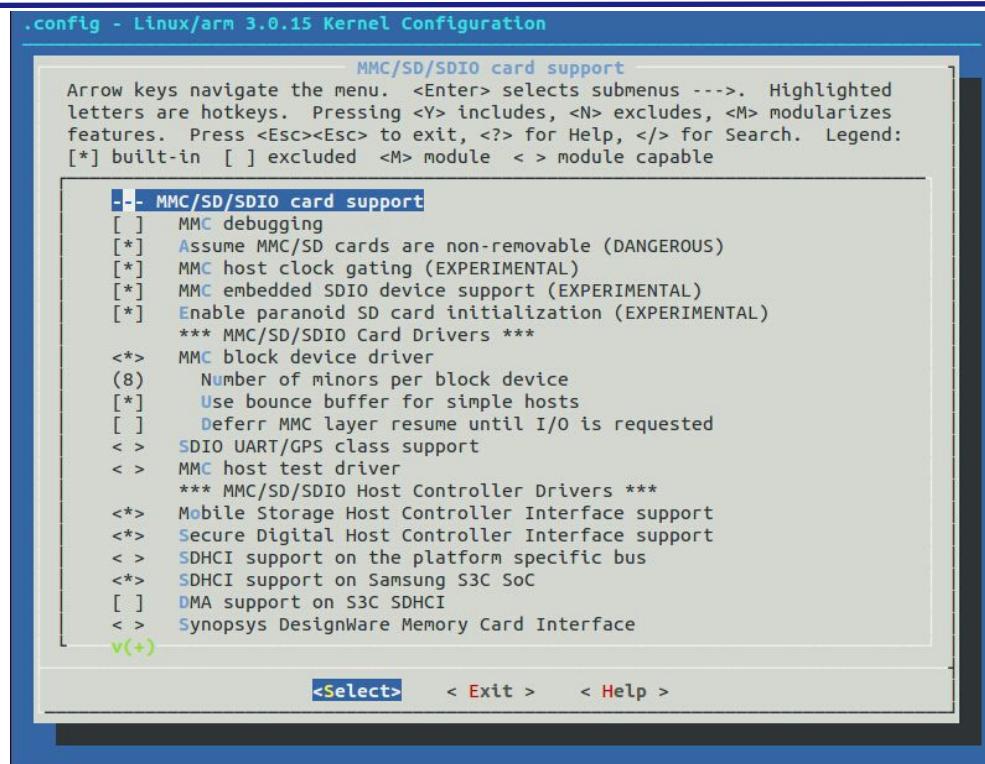
如下图，进入“Device Drivers --->”。



如下图，进入“MMC/SD/SDIO card support --->”。



配置如下图所示的所有参数即可。

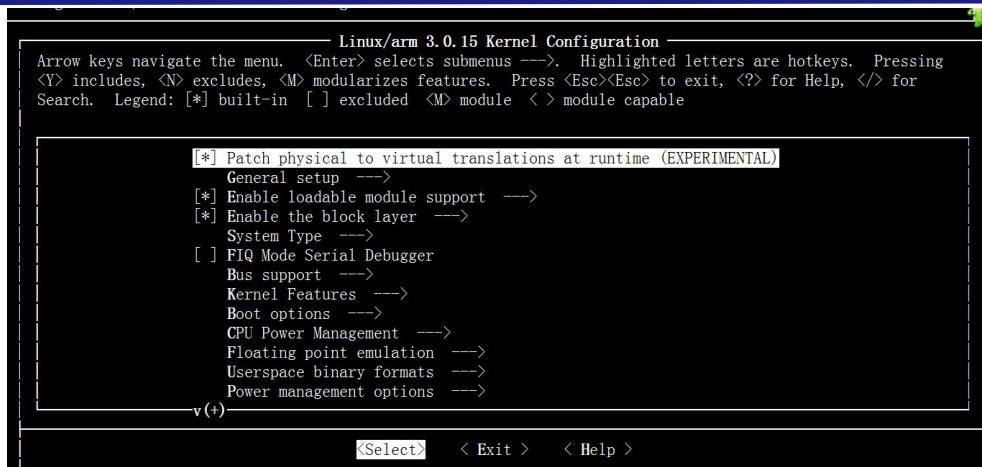


#### 9.4.25 AVIN 驱动

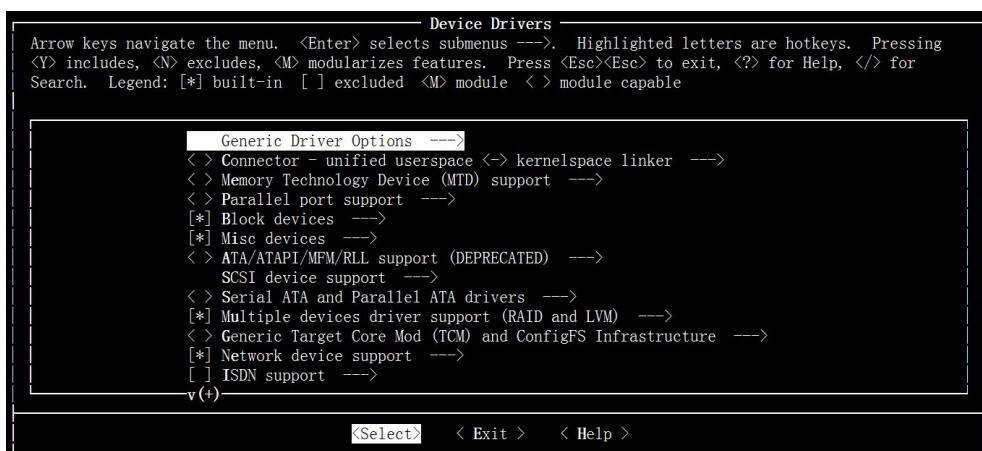
在内核目录下执行命令 “make menuconfig” , 如下图 :

```
iTop4412_Kernel_3.0#
iTop4412_Kernel_3.0#
iTop4412_Kernel_3.0# make menuconfig
```

将会打开内核的配置界面 , 如下图 :

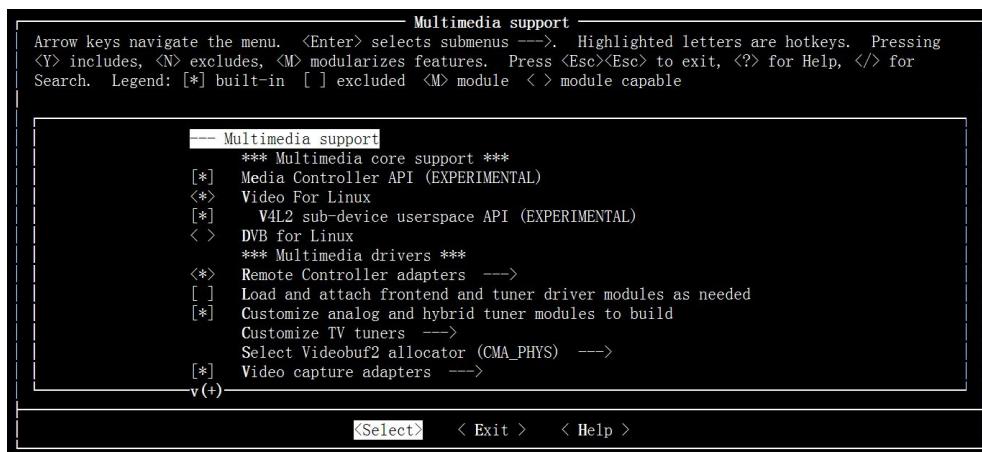


然后选择“Device Drivers”选项，进入“Device Drivers”配置界面，如下图：

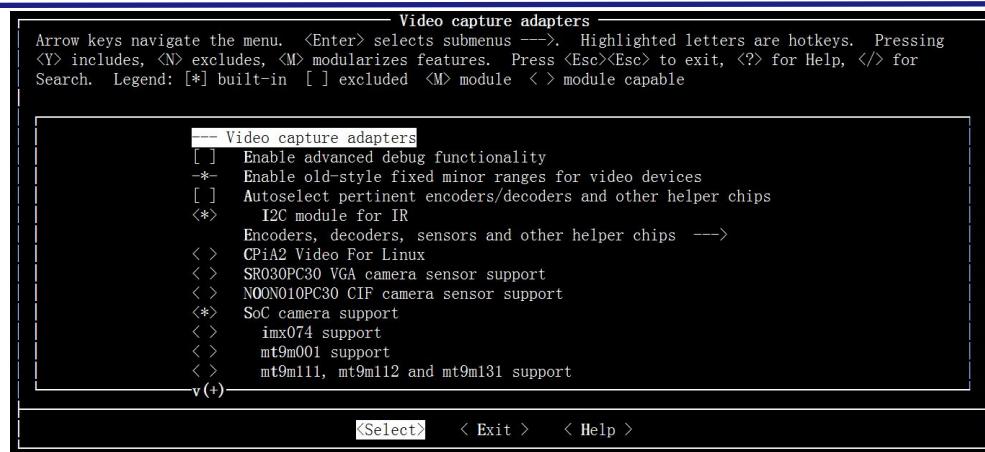


然后选择“Multimedia support”选项，进入“Multimedia support”配置界面，如下图：

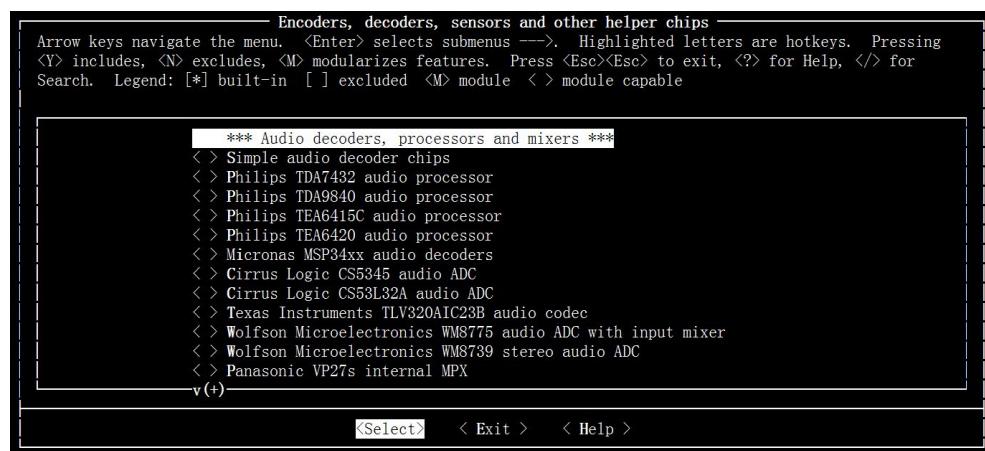
图：



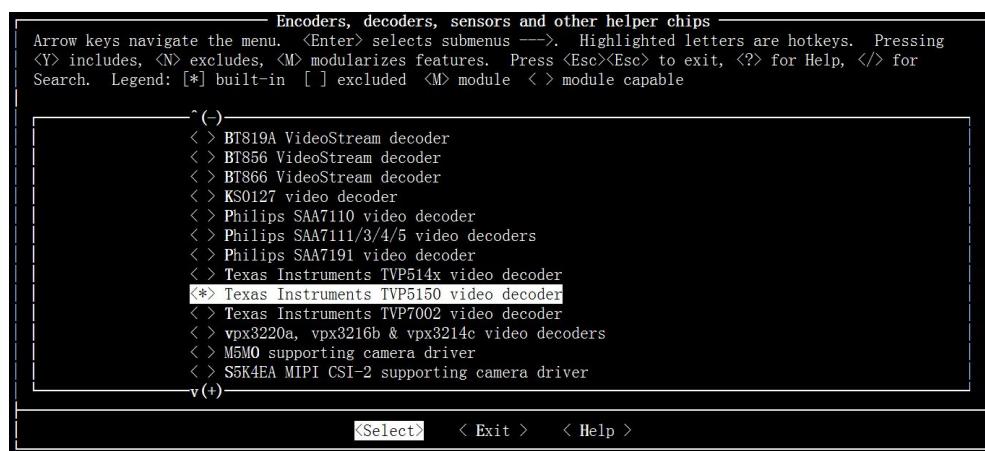
然后选择“Video capture adapters”选项，进入“Video capture adapters”配置界面，如下图：



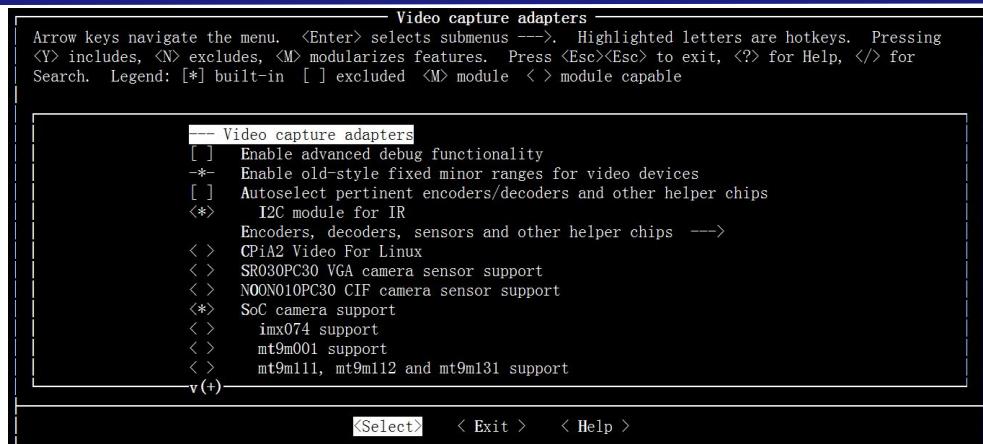
然后选择“Encoders, decoders, sensors and other helper chips”，进入“Encoders, decoders, sensors and other helper chips”配置界面，如下图：



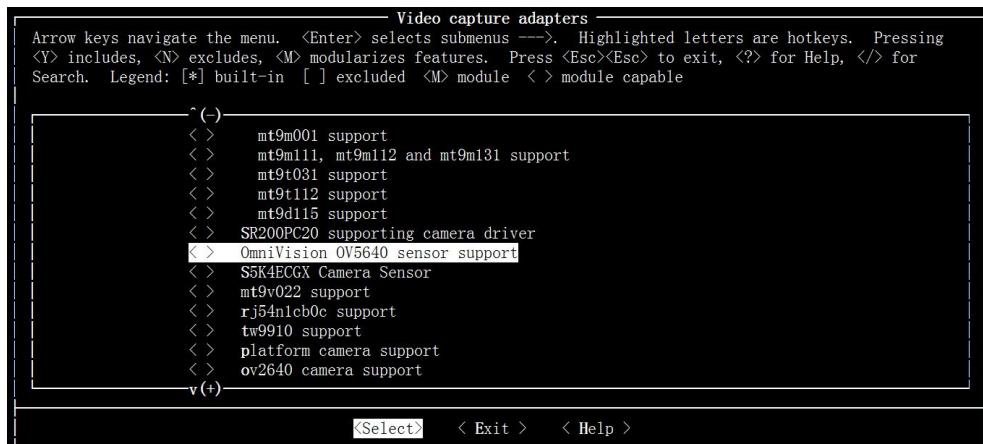
然后选择“Texas Instruments TVP5150 video decoder”选项，如下图：



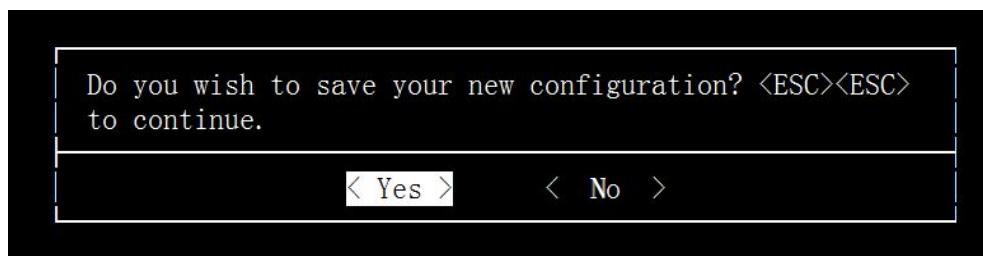
然后返回到“Video capture adapters”配置界面，如下图：



然后找到“OmniVision OV5640 sensor support”选项，取消掉“OmniVision OV5640 sensor support”的配置，如下图：



然后依次选择“Exit”退出配置界面，如下图：



然后在上图选择“Yes”，并按回车，退出配置界面。

最后在串口输入“make”开始编译内核，如下图：

```
iTop4412_Kernel_3.0#
/iTop4412_Kernel_3.0#
/iTop4412_Kernel_3.0# make
```

编译完成后，把生成的 zImage 烧写到 iTOP-4412 开发板就可以支持 AVIN 的摄像头了。

#### 9.4.26 修改开机 logo

iTIO-4412 开发板内核启动时 LCD 会显示 logo，关于这个 logo 是保存在 “drivers/video/samsung/ iTOP-4412.h” 文件，打开这个文件，会看到里面指示定义了一个数组 iBitmapData\_q，这个数组的内容就是要显示的 logo。我们修改 logo，就需要准备一张 480x640 的 bmp 图片然后使用 Image2LCD 软件转换成数组，把 iBitmapData\_q 里面的内容用新生成的数组替换掉。

可能我们自己制作的 logo 没有显示在屏幕的最中央，那我们需要修改下文件 “drivers/video/samsung/ s3cfb\_ops.c”，在这个文件找到函数：s3cfb\_draw\_logo  
修改这个函数里面的 top 和 left 就可以控制图片在屏幕显示的位置了。

#### 9.4.27 200W 和 500WCMOS ( ITU ) 摄像头的配置

开发板提供的内核默认使用的 500W 摄像头，如果大家使用迅为提供的 200W 摄像头，需要做如下的修改。

500W 摄像头的配置，我想用户可以通过 200W 摄像头的配置反推回来，这里就不再重复。

在 Ubuntu 的终端里面进入到内核的源码目录，然后执行 “make menuconfig” 命令，如下图所示：

```
root@ubuntu:/home/broswer/iTop4412_Kernel1_3.0#  
root@ubuntu:/home/broswer/iTop4412_Kernel1_3.0#  
root@ubuntu:/home/broswer/iTop4412_Kernel1_3.0#  
root@ubuntu:/home/broswer/iTop4412_Kernel1_3.0# make menuconfig
```

打开内核配置界面，然后选择 “Device Drivers”，如下图所示：

```
----- Linux/arm 3.0.15 Kernel Configuration -----
ow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys.
includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, <
rch. Legend: [*] built-in [ ] excluded <M> module <> module capable

^(-)
[*] Enable the block layer --->
    System Type --->
    [ ] FIQ Mode Serial Debugger
        Bus support --->
        Kernel Features --->
        Boot options --->
        CPU Power Management --->
        Floating point emulation --->
        Userspace binary formats --->
        Power management options --->
[*] Networking support --->
    Device Drivers --->
        File systems --->
v(+)

<Select> < Exit > < Help >
```

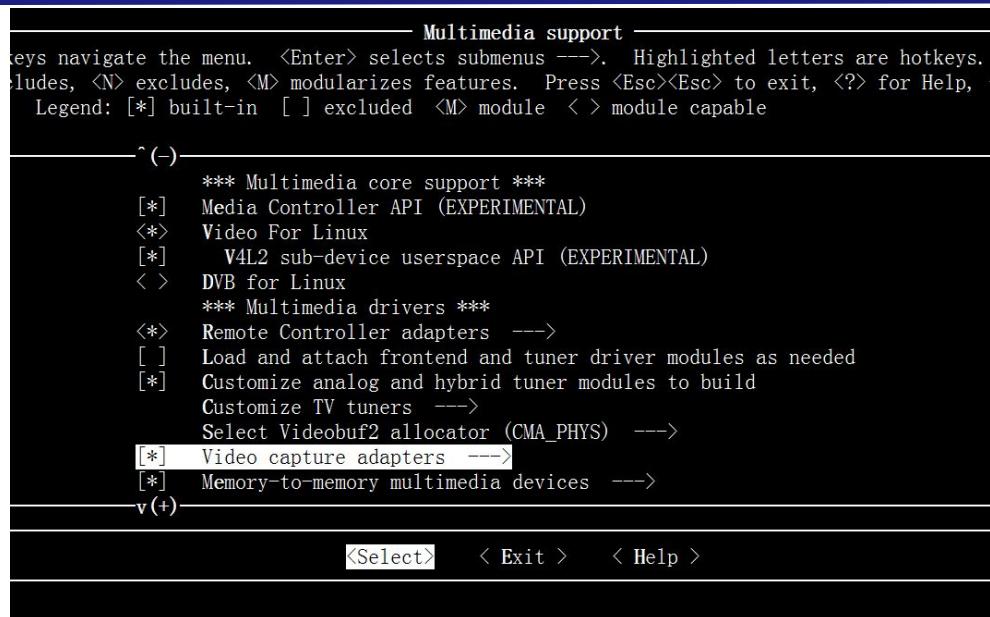
进入到“Device Drivers”界面然后选择“Multimedia support”，如下图所示：

```
----- Device Drivers -----
ys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkey
udes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help
Legend: [*] built-in [ ] excluded <M> module <> module capable

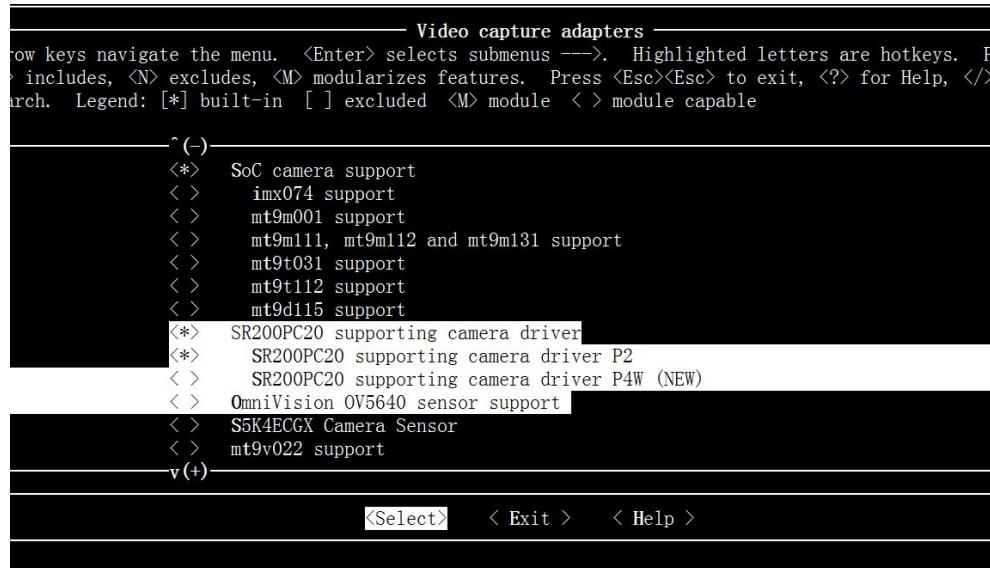
^(-)
    PTP clock support --->
    -*- GPIO Support --->
    < > Dallas's 1-wire support --->
    <*> Power supply class support --->
    < > Hardware Monitoring support --->
    < > Generic Thermal sysfs driver --->
[*] Watchdog Timer Support --->
    Sonics Silicon Backplane --->
    Broadcom specific AMBA --->
[*] Multifunction device drivers --->
[*] Voltage and Current Regulator Support --->
<*> Multimedia support --->
    Graphics support --->
v(+)

<Select> < Exit > < Help >
```

进入到“Multimedia support”界面，然后选择“Video capture adapters”，如下图所示：



进入到“Video capture adapters”界面，修改如下的三个地方，如下图所示：



按照上图高亮的三个地方修改，然后保存并退出内核配置界面，然后执行“make”命令编译内核，生成的 zImage 就支持迅为的 200W 摄像头了。

## 9.5 制作最小文件系统镜像

Exynos-4412 不仅可以跑 Android，还可以运行简单的 Linux 最小文件系统，下面我们来讲解一下这种文件系统的制作。

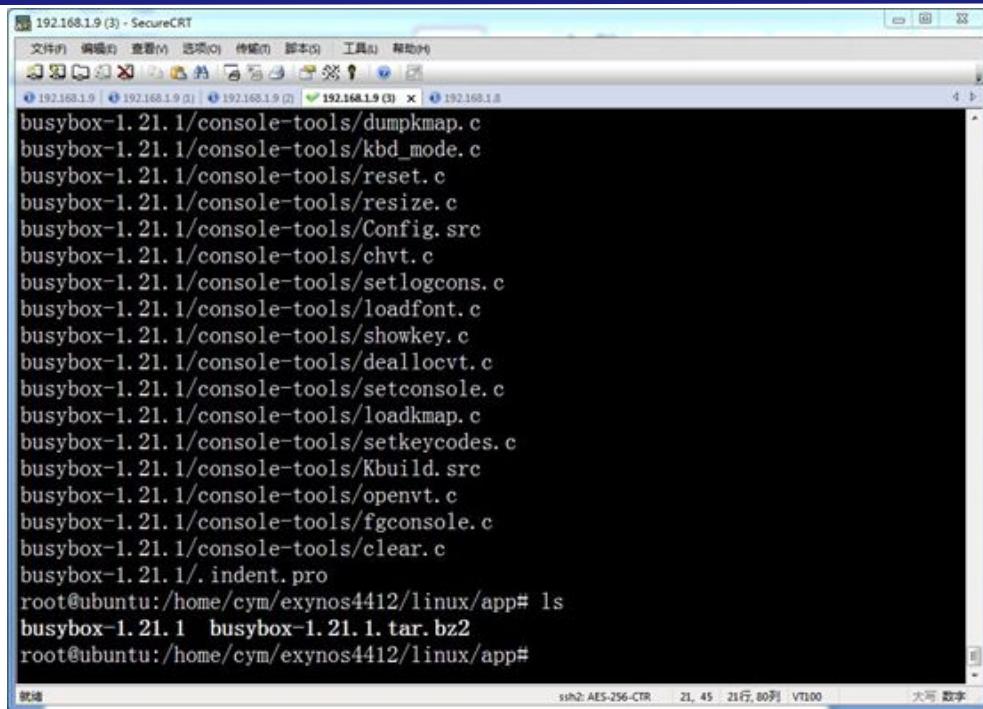
在制作文件系统的时候，我们需要使用“Busybox 工具”，即为网盘中的压缩包“busybox-1.21.1.tar.bz2”。 “BusyBox 工具”是一个集成了一百多个最常用 Linux 命令和工具的软件。

BusyBox 包含了一些简单的工具，例如 ls、cat 和 echo 命令等等，还包含了一些更大、更复杂的工具，例 grep、find、mount 以及 telnet 命令。有些人将 BusyBox 称为 Linux 工具里的瑞士军刀。简单的说 BusyBox 就好像是个大工具箱，它集成压缩了 Linux 的许多工具和命令，也包含了 Android 系统自带的 shell。

“Busybox 工具”的官方网址是“<http://www.busybox.net/>”，这是一个开源的程序，并且一直在更新中，我们使用的版本是“busybox-1.21.1.tar.bz2”。

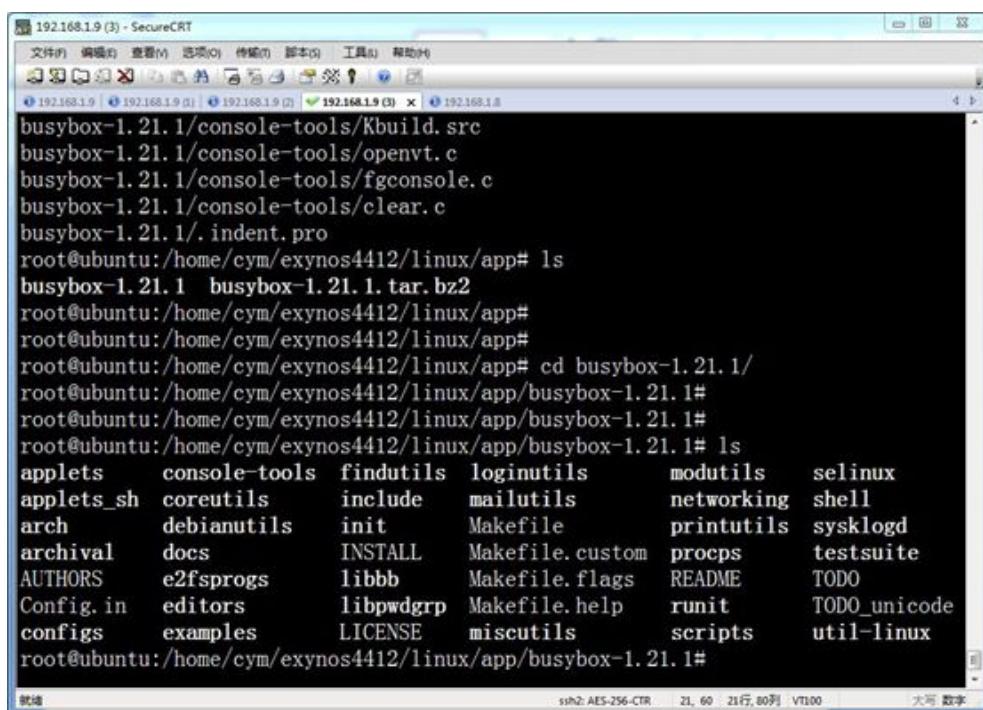
下面我们来讲解一下如何使用 BusyBox 制作最小文件系统：

(1) 首先拷贝“busybox-1.21.1.tar.bz2”到我们的虚拟机 Ubuntu 系统上，然后在 Ubuntu 命令行，执行“#tar -xvf busybox-1.21.1.tar.bz2”解压命令，解压完成后，如下图：



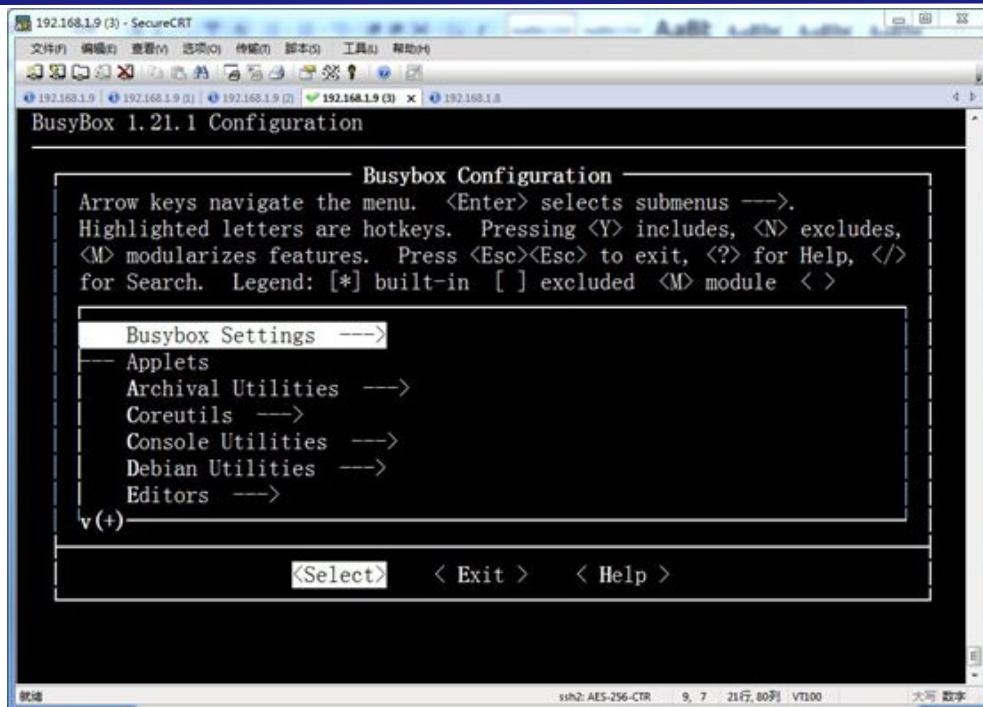
```
192.168.1.9 (3) - SecureCRT
busybox-1.21.1/console-tools/dumpkmap.c
busybox-1.21.1/console-tools/kbd_mode.c
busybox-1.21.1/console-tools/reset.c
busybox-1.21.1/console-tools/resize.c
busybox-1.21.1/console-tools/Config. src
busybox-1.21.1/console-tools/chvt.c
busybox-1.21.1/console-tools/setlogcons.c
busybox-1.21.1/console-tools/loadfont.c
busybox-1.21.1/console-tools/showkey.c
busybox-1.21.1/console-tools/deallocvt.c
busybox-1.21.1/console-tools/setconsole.c
busybox-1.21.1/console-tools/loadkmap.c
busybox-1.21.1/console-tools/setkeycodes.c
busybox-1.21.1/console-tools/Kbuild. src
busybox-1.21.1/console-tools/openvt. c
busybox-1.21.1/console-tools/fgconsole.c
busybox-1.21.1/console-tools/clear.c
busybox-1.21.1/. indent. pro
root@ubuntu:/home/cym/exynos4412/linux/app# ls
busybox-1.21.1 busybox-1.21.1.tar.bz2
root@ubuntu:/home/cym/exynos4412/linux/app#
```

( 2 ) 如下图 , 使用 “`#cd busybox-1.21.1`” 命令进入到前面解压出来的 “busybox-1.21.1” 文件夹中。

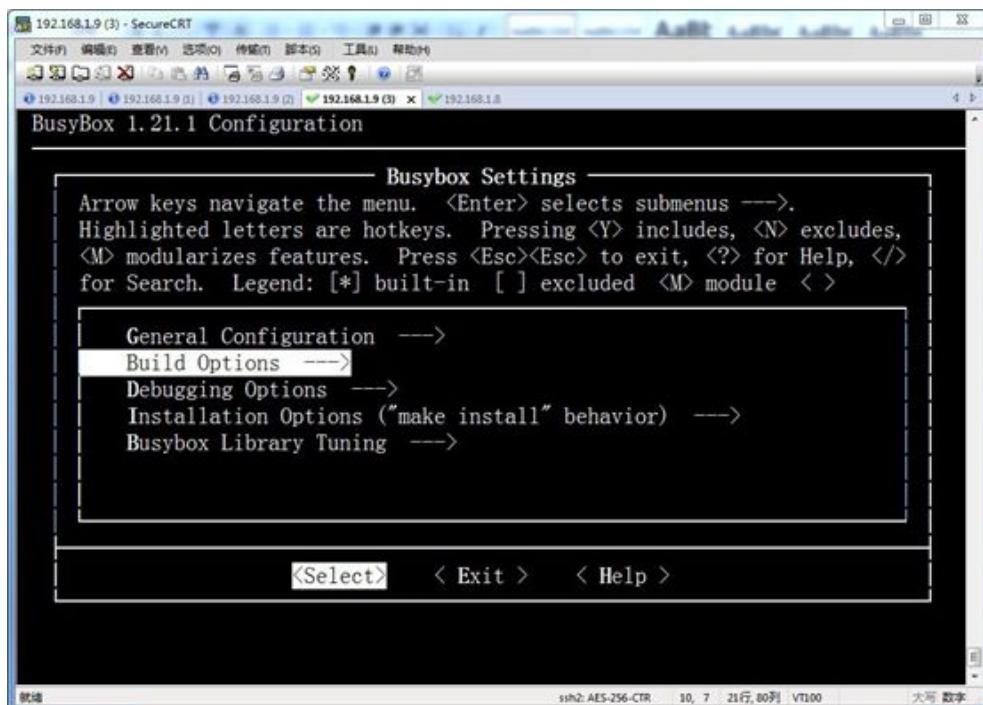


```
192.168.1.9 (3) - SecureCRT
busybox-1.21.1/console-tools/Kbuild. src
busybox-1.21.1/console-tools/openvt. c
busybox-1.21.1/console-tools/fgconsole.c
busybox-1.21.1/console-tools/clear.c
busybox-1.21.1/. indent. pro
root@ubuntu:/home/cym/exynos4412/linux/app# ls
busybox-1.21.1 busybox-1.21.1.tar.bz2
root@ubuntu:/home/cym/exynos4412/linux/app#
root@ubuntu:/home/cym/exynos4412/linux/app#
root@ubuntu:/home/cym/exynos4412/linux/app# cd busybox-1.21.1/
root@ubuntu:/home/cym/exynos4412/linux/app# ls
applets    console-tools  findutils  loginutils  modutils  selinux
applets_sh  coreutils     include    mailutils   networking  shell
arch       debianutils   init      Makefile    printutils sysklogd
archival   docs          INSTALL   Makefile.custom  procps   testsuite
AUTHORS   e2fsprogs    libbb    Makefile.flags  README   TODO
Config.in  editors       libpwdgrp Makefile.help   runit   TODO_unicode
configs   examples      LICENSE   miscutils   scripts  util-linux
root@ubuntu:/home/cym/exynos4412/linux/app# ls
applets    console-tools  findutils  loginutils  modutils  selinux
applets_sh  coreutils     include    mailutils   networking  shell
arch       debianutils   init      Makefile    printutils sysklogd
archival   docs          INSTALL   Makefile.custom  procps   testsuite
AUTHORS   e2fsprogs    libbb    Makefile.flags  README   TODO
Config.in  editors       libpwdgrp Makefile.help   runit   TODO_unicode
configs   examples      LICENSE   miscutils   scripts  util-linux
root@ubuntu:/home/cym/exynos4412/linux/app#
```

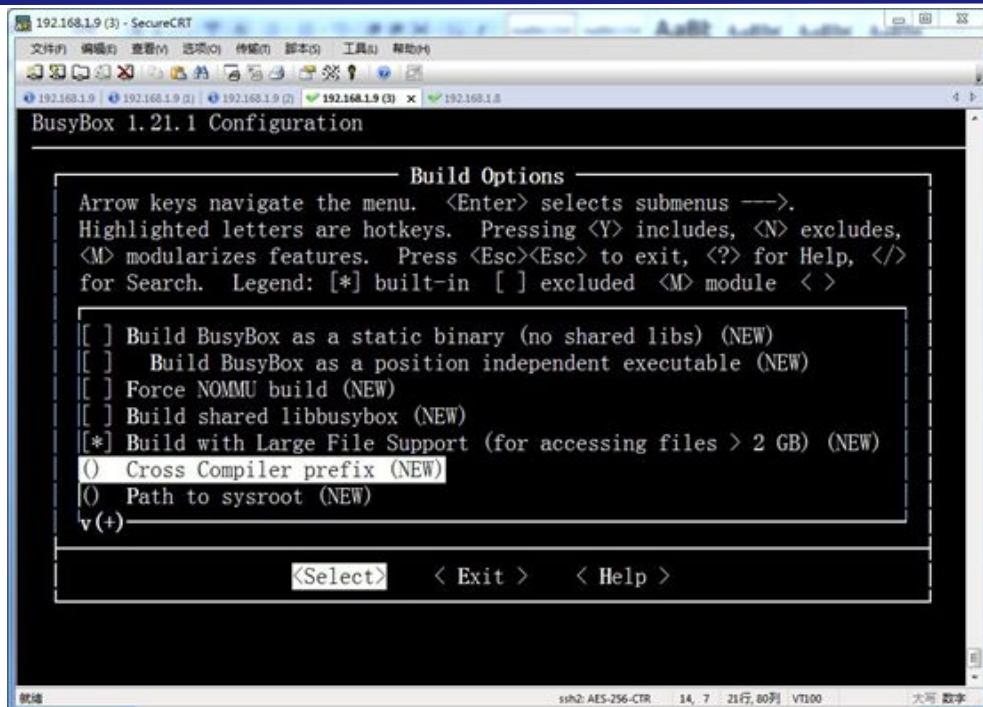
( 3 ) Busybox 的编译配置和 Linux 内核编译配置使用的命令是一样的 , 下面我们开始配置 Busybox。输入 “`#make menuconfig`” 命令 , 如下图 , 出现 “Busybox” 的配置界面。



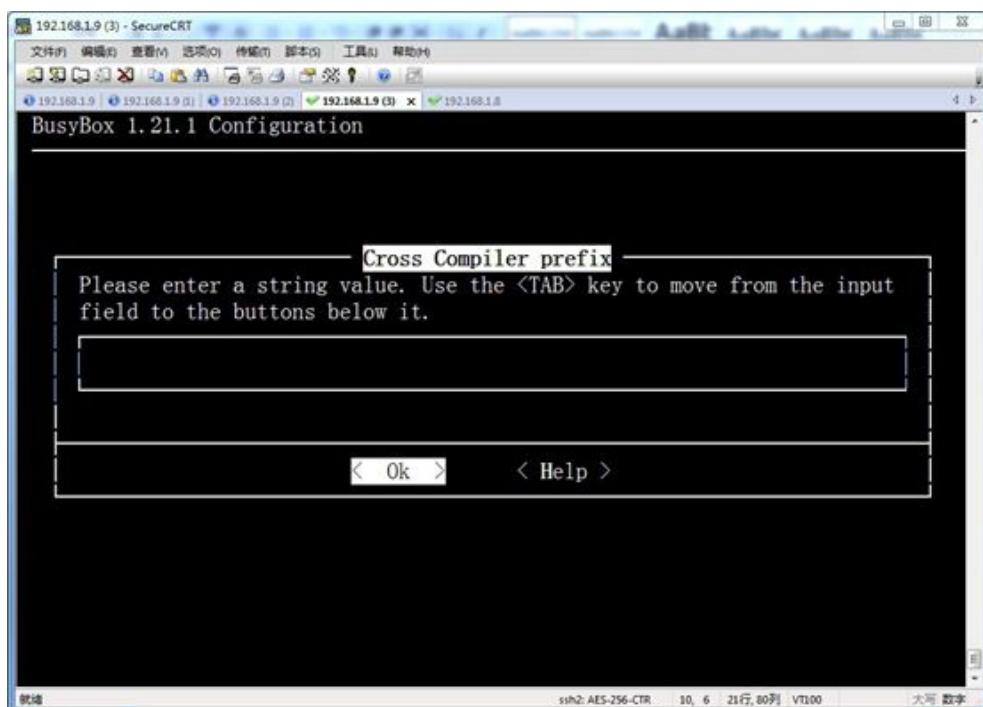
(4) 如上图，选中“Busybox Settings”，然后按“回车”进入到“Busybox Settings”界面，如下图：



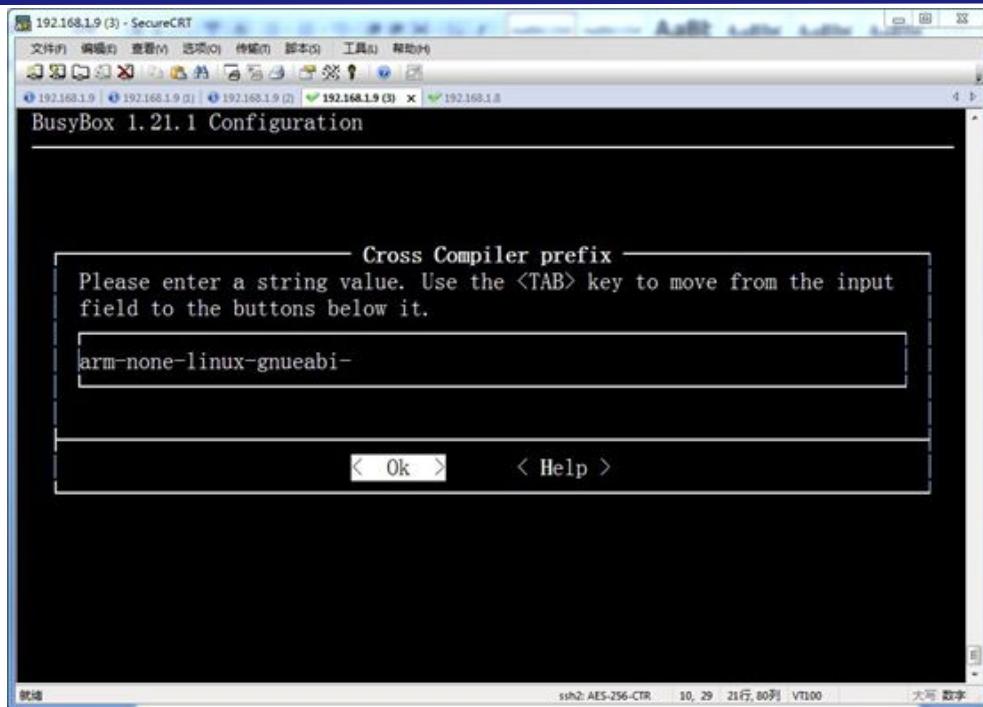
(5) 在“Busybox Settings”配置选项里面我们需要修改两个地方。第一个是“Build Options-> Cross Compiler prefix”，它是指定用什么编译器来编译 Busybox，如上图选中“Build Options”，按回车，进入到“Build Options”配置界面，如下图：



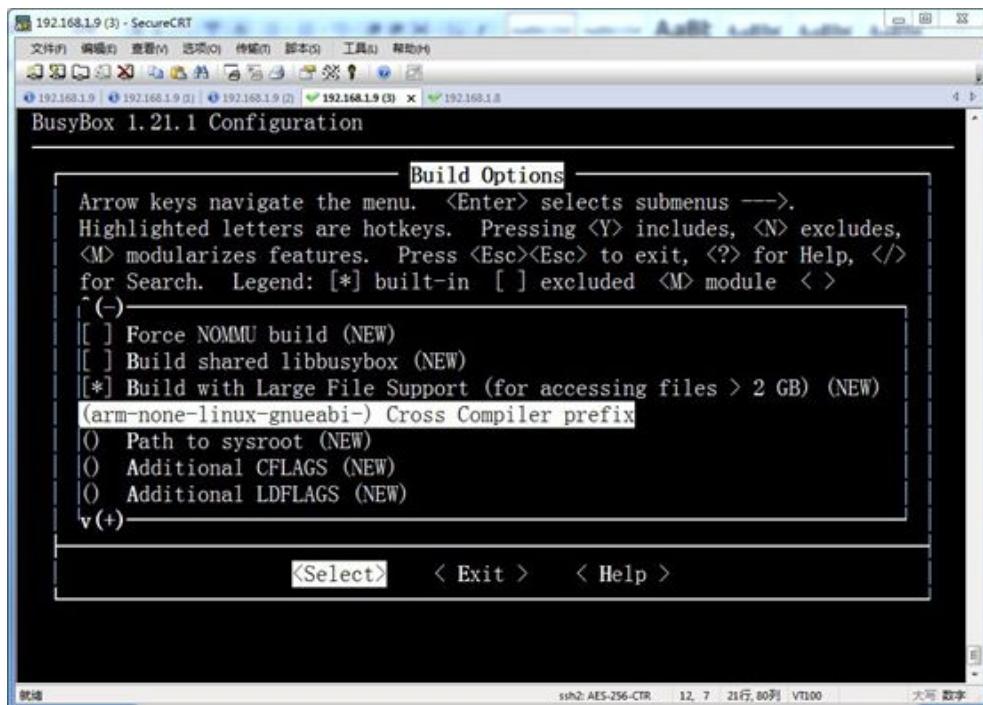
(6) 如上图，选中“Build Options”配置界面的“Cross Compiler prefix”，然后按“回车”，进入“Cross Compiler prefix”配置界面，如下图：



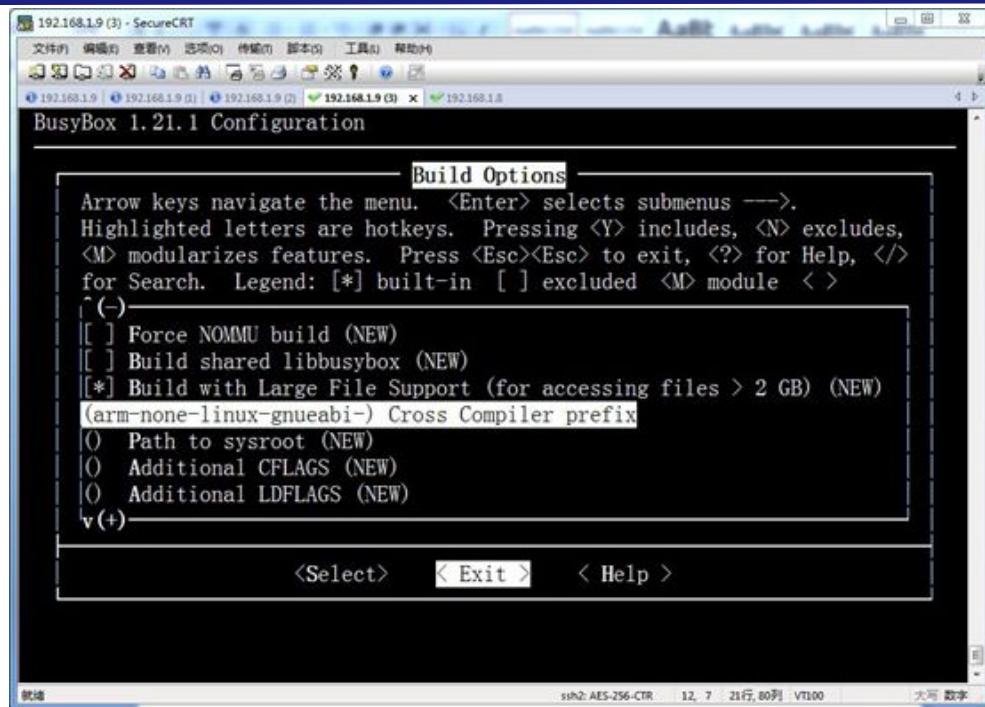
(7) 如下图，输入交叉编译工具“arm-none-linux-gnueabi-”。



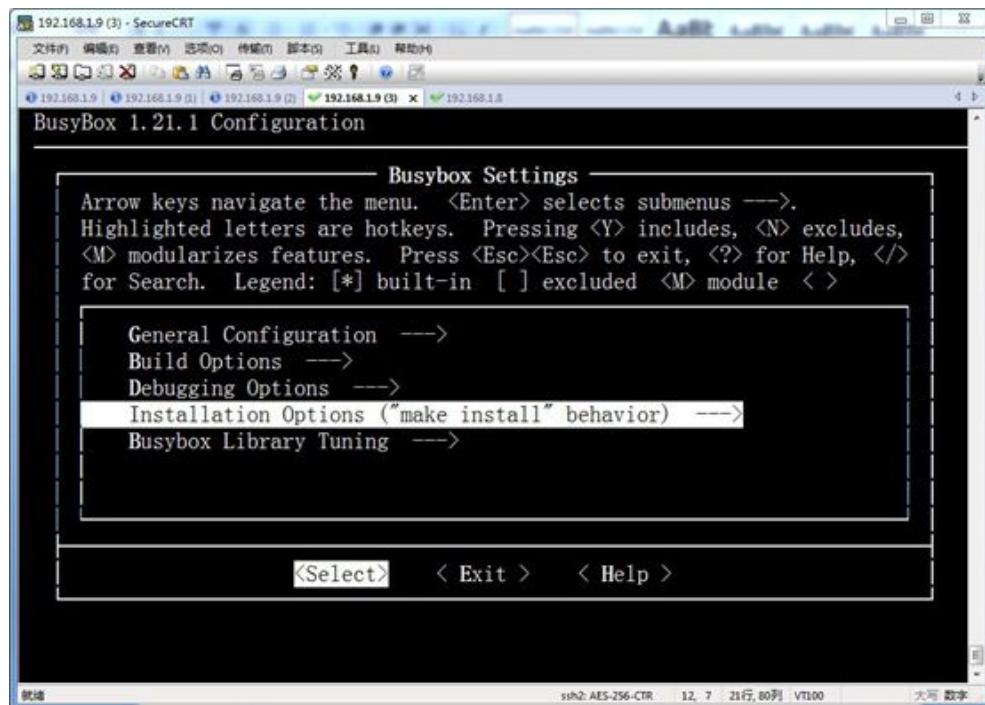
(8) 按“回车”返回到“Build Options”配置界面，这时可以看到刚才我们设置的交叉编译工具，如下图：



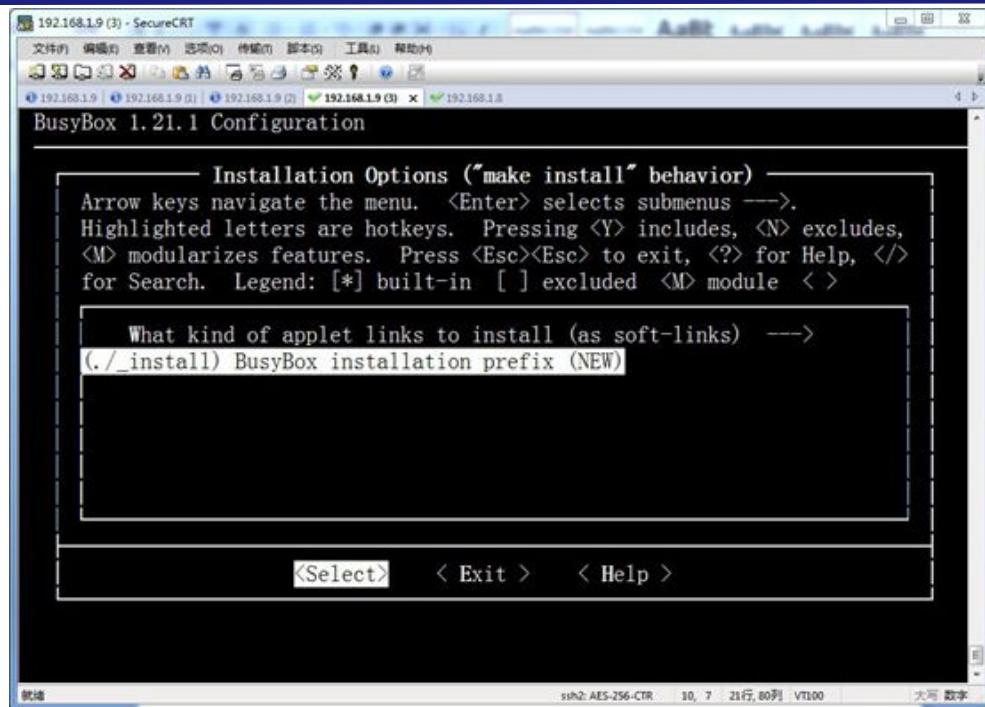
(9) 使用键盘的“左右”方向按键，选中“Exit”，如下图：



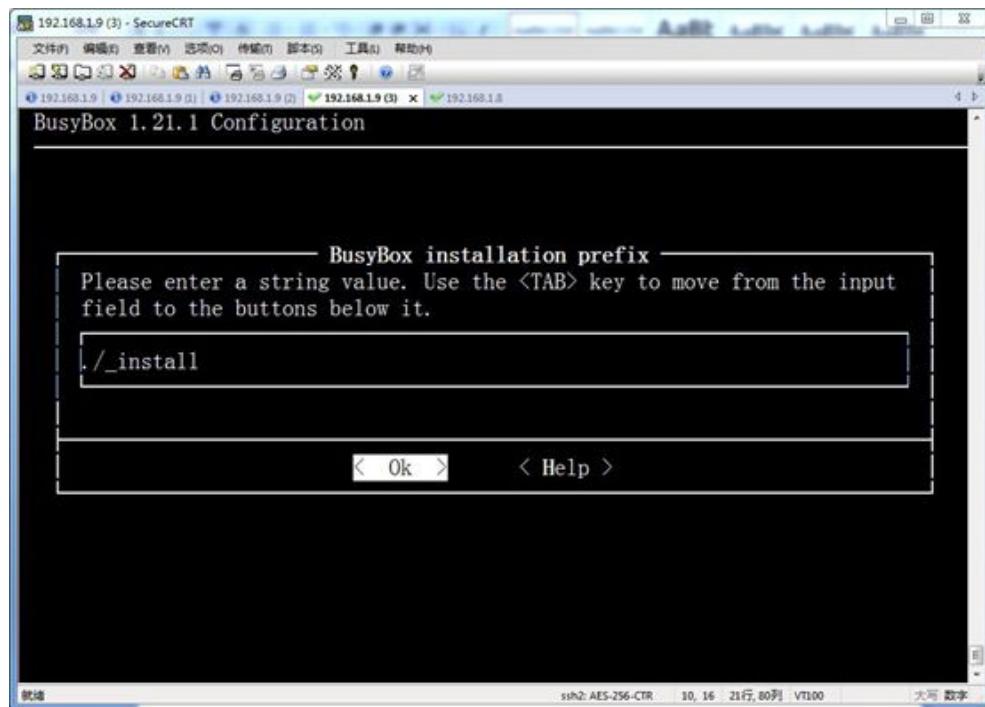
( 10 ) 然后按 “回车” , 回到 “Busybox Settings” 设置界面 , 如下图 :



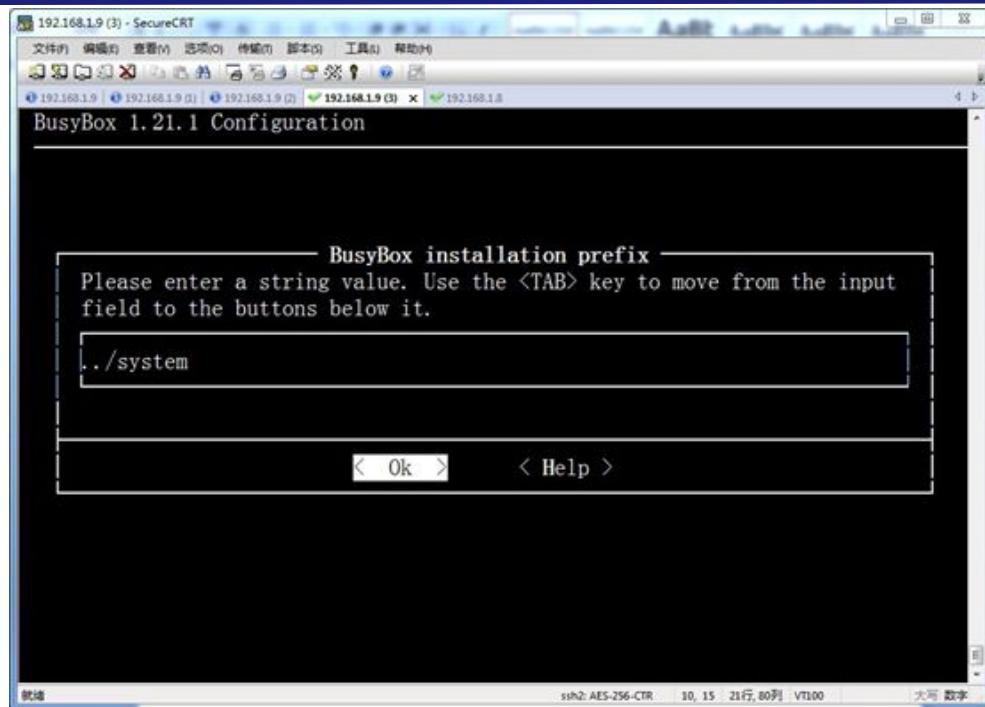
( 11 ) 如上图 , 选中 “Installation Options” , 然后按 “回车” , 进入 “Installation Options” 配置界面 , 如下图 :



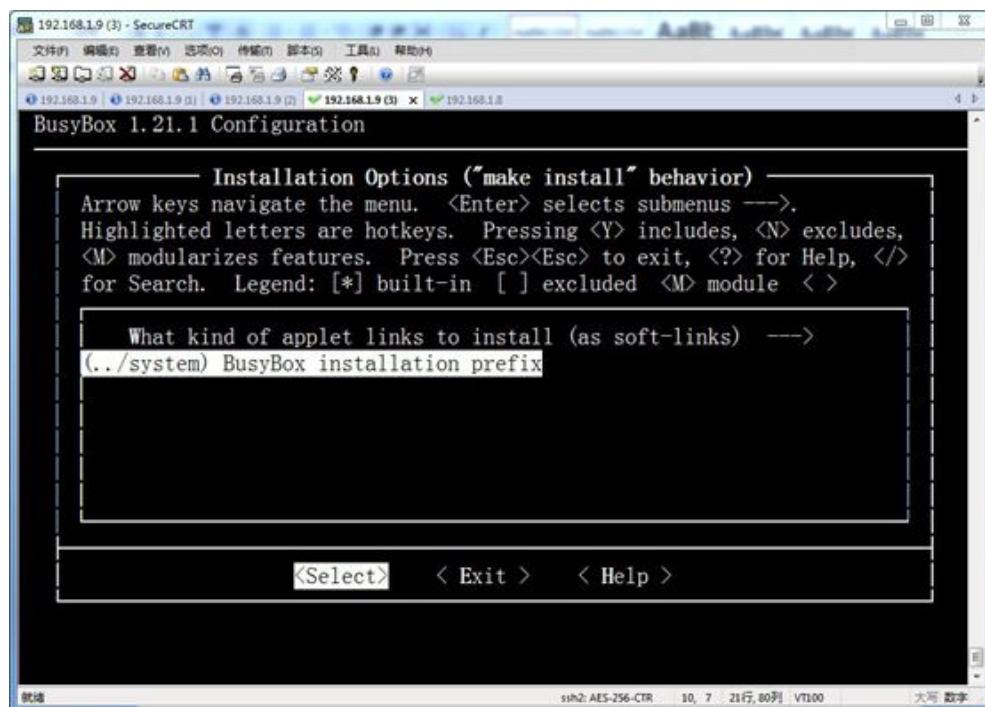
(12) 如上图所示，选中“BusyBox installation prefix”，然后按“回车”进入“BusyBox installation prefix”配置界面，这个界面是设置编译完“Busybox”之后，把最终生成的二进制文件安装到哪个目录下面，如下图：



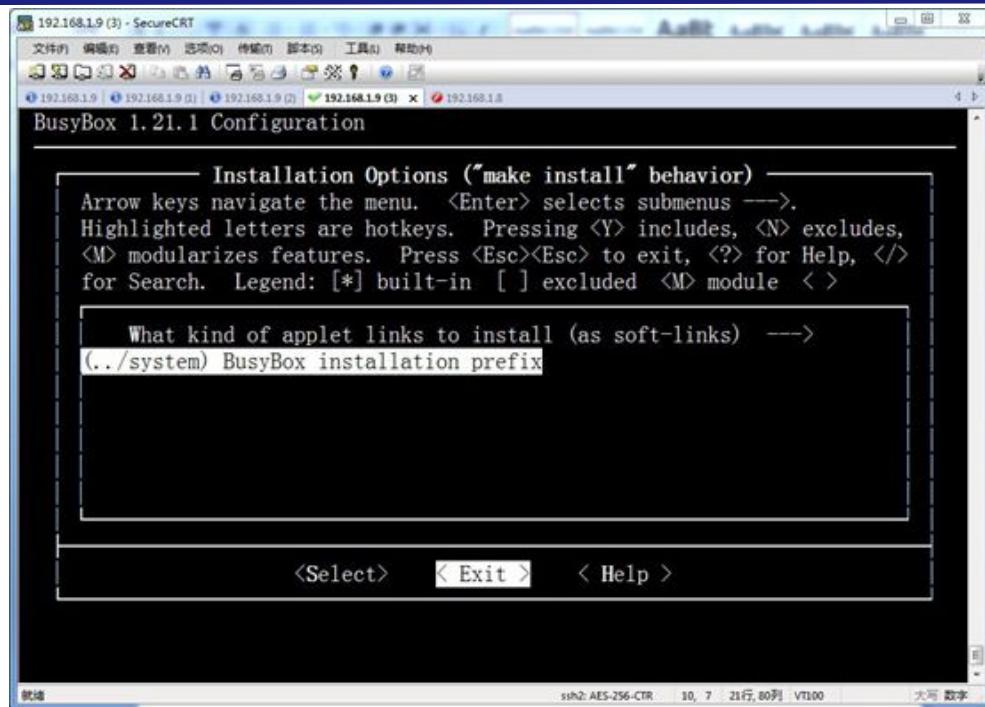
(13) 删除上图中的“./\_install”，然后输入“../system”，这样设置之后，最终生成的二进制文件会安装到“当前目录”的“上一级目录”下的“system”目录里面，如下图：



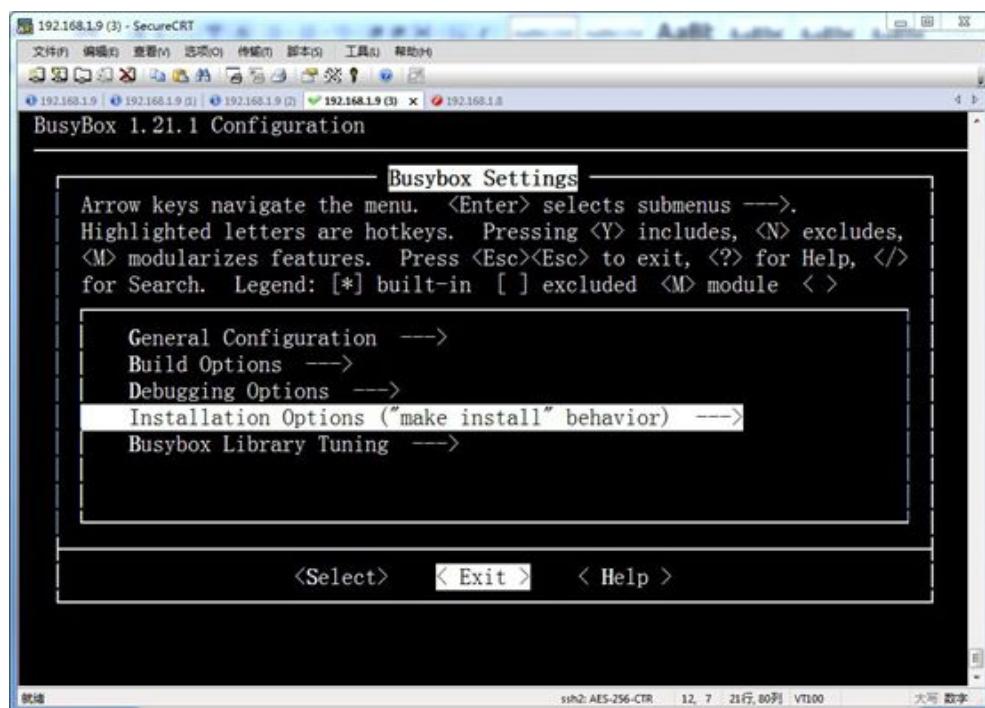
( 14 ) 按 “回车” , 回到 “Installation Options” , 如下图 :



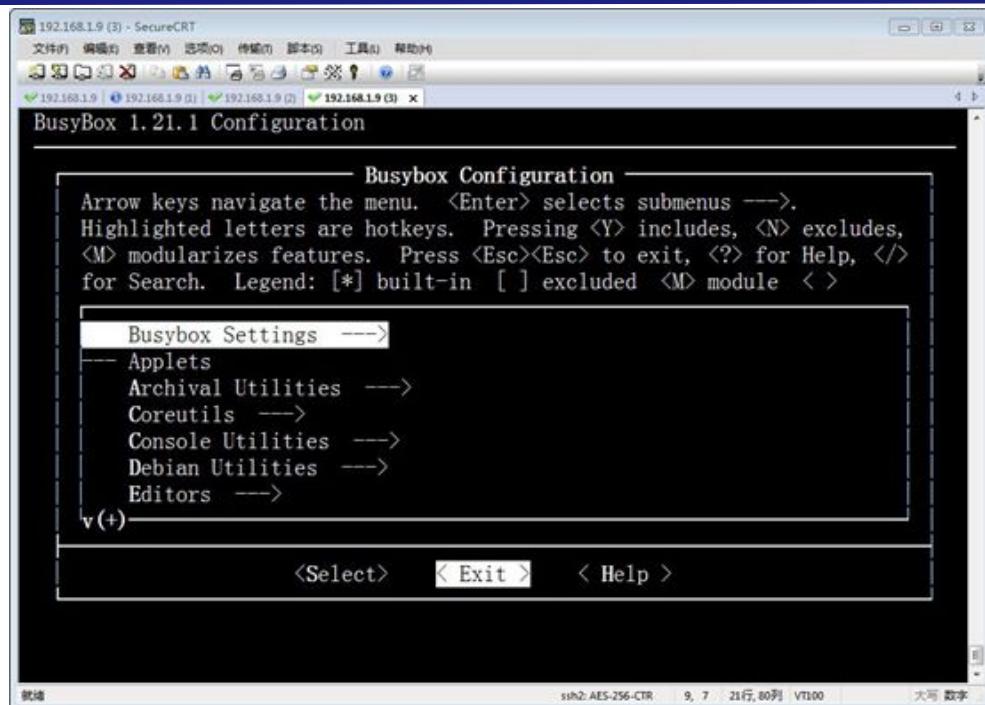
( 15 ) 使用键盘 “向右” 的方向键移动光标到 “Exit” , 如下图 :



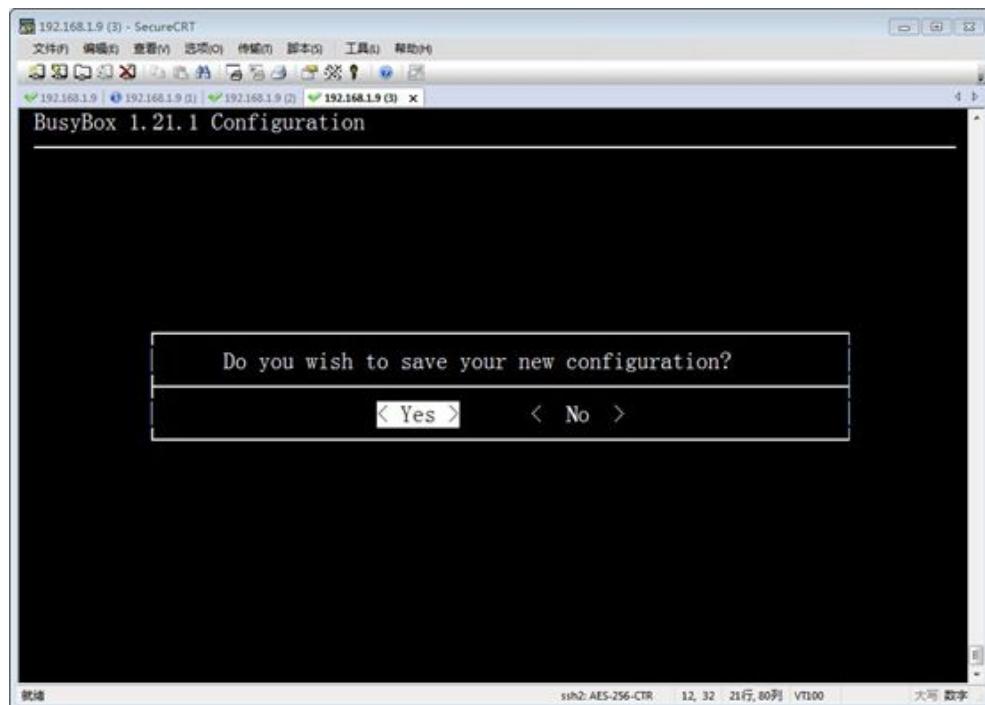
(16) 按“回车”，返回到“Busybox Settings”，使用键盘方向键，移动光标到“Exit”，如下图：



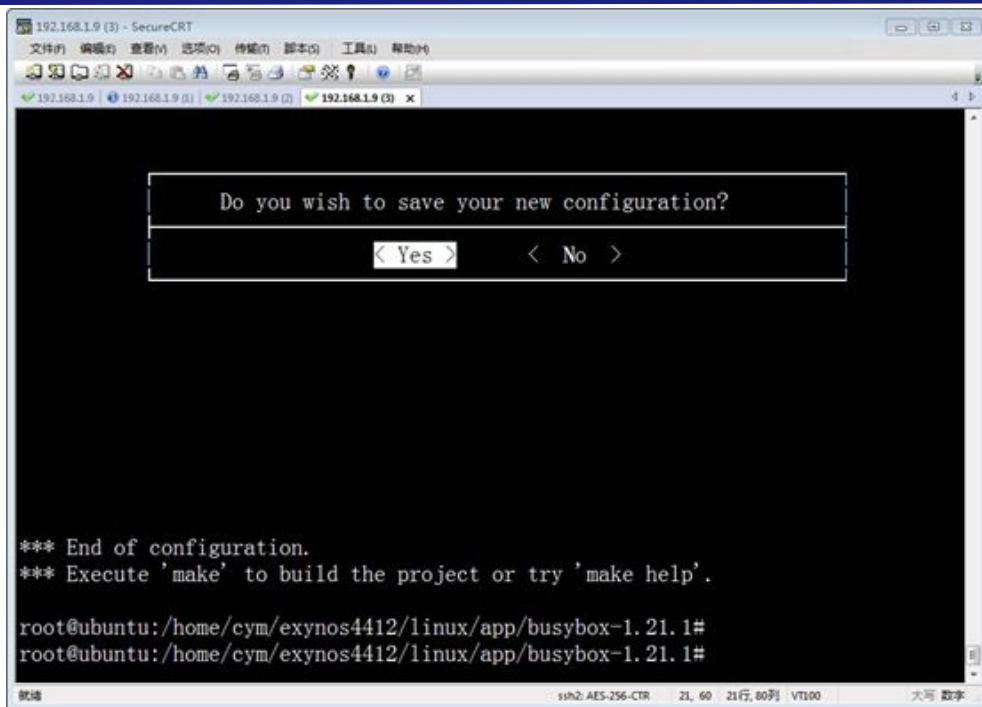
(17) 按“回车”，返回到“Busybox Configuration”，使用键盘方向键，移动光标到“Exit”，如下图：



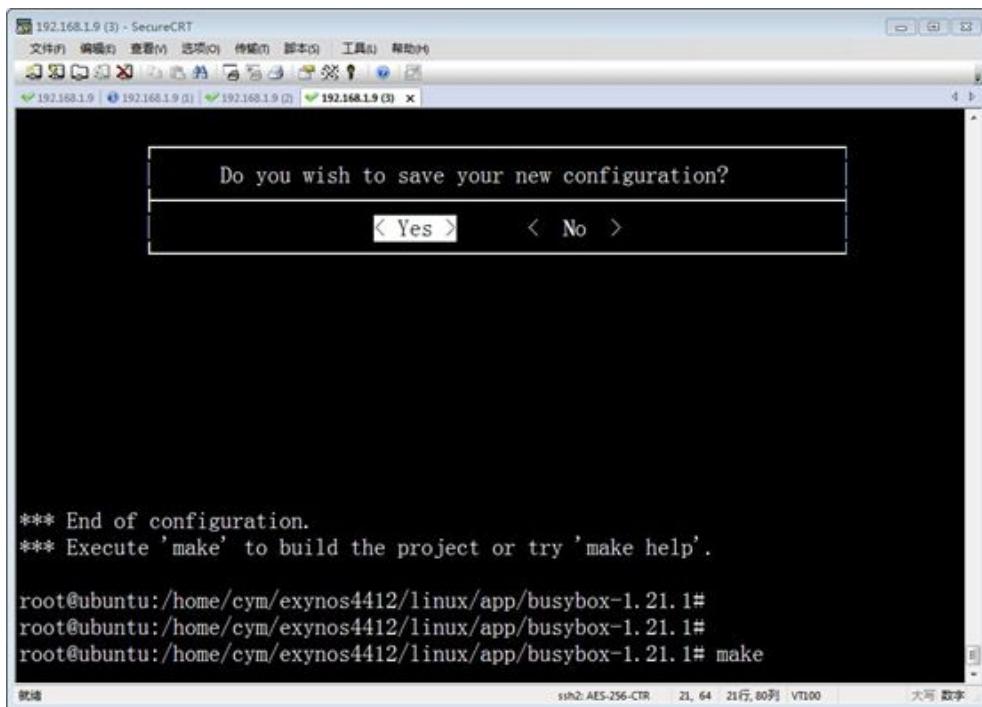
( 18 ) 输入 “回车” , 弹出保存配置界面 , 如下图 :



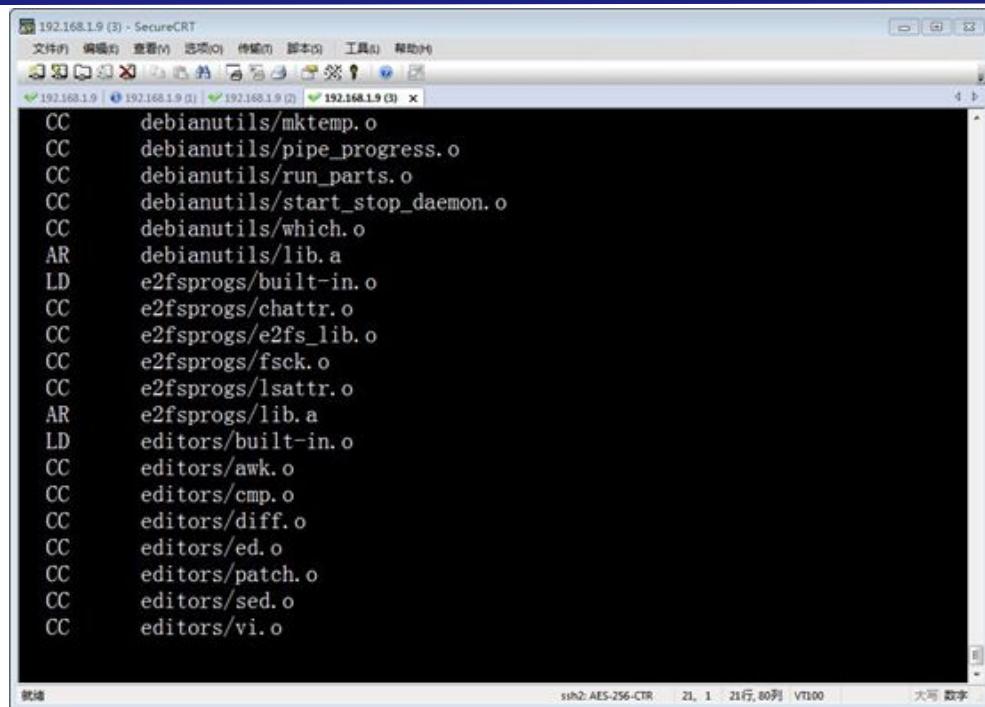
( 19 ) 使用键盘 “方向键” , 移动光标到 “Yes” , 然后按 “回车” 保存配置 , 退出配置界面 , 如下图 :



( 20 ) 现在 “Busybox” 的配置已经完成了，接下来我们开始编译 “Busybox” ，执行 “make” 命令，开始编译 “Busybox” ，如下图：

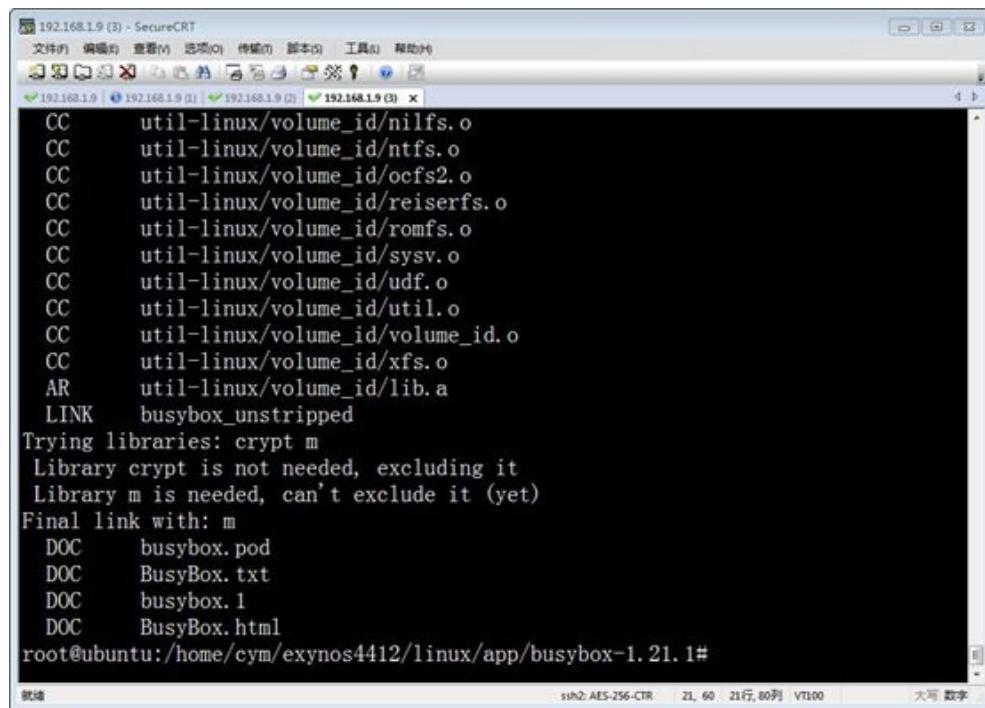


( 21 ) 下图为编译过程中的截图：



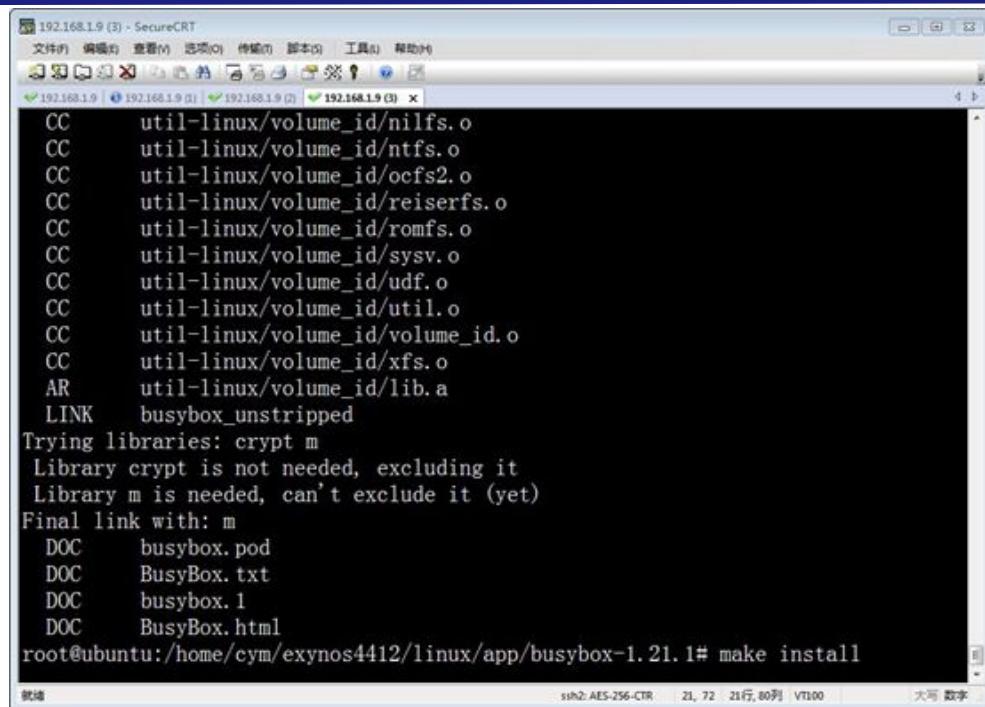
```
CC  debianutils/mktemp.o
CC  debianutils/pipe_progress.o
CC  debianutils/run_parts.o
CC  debianutils/start_stop_daemon.o
CC  debianutils/which.o
AR  debianutils/lib.a
LD  e2fsprogs/built-in.o
CC  e2fsprogs/chattr.o
CC  e2fsprogs/e2fs_lib.o
CC  e2fsprogs/fsck.o
CC  e2fsprogs/lsattr.o
AR  e2fsprogs/lib.a
LD  editors/built-in.o
CC  editors/awk.o
CC  editors/cmp.o
CC  editors/diff.o
CC  editors/ed.o
CC  editors/patch.o
CC  editors/sed.o
CC  editors/vi.o
```

( 22 ) 下图为编译完成的截图 :



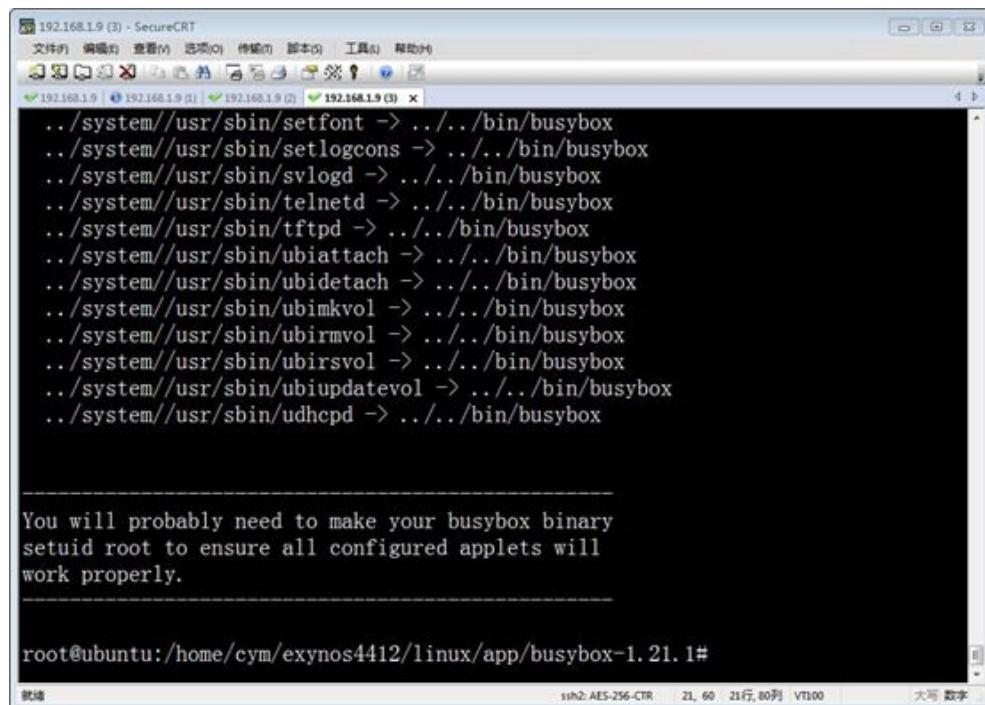
```
CC  util-linux/volume_id-nilfs.o
CC  util-linux/volume_id-ntfs.o
CC  util-linux/volume_id-ocfs2.o
CC  util-linux/volume_id/reiserfs.o
CC  util-linux/volume_id/romfs.o
CC  util-linux/volume_id/sysv.o
CC  util-linux/volume_id/udf.o
CC  util-linux/volume_id/util.o
CC  util-linux/volume_id/volume_id.o
CC  util-linux/volume_id/xfs.o
AR  util-linux/volume_id/lib.a
LINK  busybox_unstripped
Trying libraries: crypt m
Library crypt is not needed, excluding it
Library m is needed, can't exclude it (yet)
Final link with: m
DOC  busybox.pod
DOC  BusyBox.txt
DOC  busybox.1
DOC  BusyBox.html
root@ubuntu:/home/cym/exynos4412/linux/app/busybox-1.21.1#
```

( 23 ) 编译完成了 , 接下来我们需要把编译生成的 “二进制文件” 安装到刚才我们指定的 “`../system`” 目录里面 , 安装二进制文件到 “`../system`” 目录 , 如下图 , 输入命令 “`make install`” 。



```
CC util-linux/volume_id-nilfs.o
CC util-linux/volume_id-ntfs.o
CC util-linux/volume_id-ocfs2.o
CC util-linux/volume_id-reiserfs.o
CC util-linux/volume_id-romfs.o
CC util-linux/volume_id-sysv.o
CC util-linux/volume_id-udf.o
CC util-linux/volume_id-util.o
CC util-linux/volume_id/volume_id.o
CC util-linux/volume_id/xfs.o
AR util-linux/volume_id/lib.a
LINK busybox_unstripped
Trying libraries: crypt m
Library crypt is not needed, excluding it
Library m is needed, can't exclude it (yet)
Final link with: m
DOC busybox.pod
DOC BusyBox.txt
DOC busybox.1
DOC BusyBox.html
root@ubuntu:/home/cym/exynos4412/linux/app/busybox-1.21.1# make install
```

(24)下图为"make install"命令执行完成的截图：



```
../system//usr/sbin/setfont -> ../../bin/busybox
../system//usr/sbin/setlogcons -> ../../bin/busybox
../system//usr/sbin/svlogd -> ../../bin/busybox
../system//usr/sbin/telnetd -> ../../bin/busybox
../system//usr/sbin/tftpd -> ../../bin/busybox
../system//usr/sbin/ubiaattach -> ../../bin/busybox
../system//usr/sbin/ubidetach -> ../../bin/busybox
../system//usr/sbin/ubimkvol -> ../../bin/busybox
../system//usr/sbin/ubirmvol -> ../../bin/busybox
../system//usr/sbin/ubirsvol -> ../../bin/busybox
../system//usr/sbin/ubiupdatevol -> ../../bin/busybox
../system//usr/sbin/udhcpcd -> ../../bin/busybox

-----
You will probably need to make your busybox binary
setuid root to ensure all configured applets will
work properly.

-----
root@ubuntu:/home/cym/exynos4412/linux/app/busybox-1.21.1#
```

( 25 ) 现在我们使用 "cd .. /system" 命令 , 进入 ".. /system" 目录 , 看看里面安装的文件 , 如下图 :

The screenshot shows a terminal window titled "192.168.1.9 (3) - SecureCRT". The terminal displays a list of binary files from the "/system" directory, all pointing to "/bin/busybox". Below this, a message states: "You will probably need to make your busybox binary setuid root to ensure all configured applets will work properly." At the bottom, the command "ls" is run in the "/system" directory, showing subdirectories "bin", "linuxrc", "sbin", and "usr".

```
.../system//usr/sbin/telnetd -> ../../bin/busybox
.../system//usr/sbin/tftpd -> ../../bin/busybox
.../system//usr/sbin/ubiaattach -> ../../bin/busybox
.../system//usr/sbin/ubidetach -> ../../bin/busybox
.../system//usr/sbin/ubimkvol -> ../../bin/busybox
.../system//usr/sbin/ubirmvol -> ../../bin/busybox
.../system//usr/sbin/ubirsvol -> ../../bin/busybox
.../system//usr/sbin/ubiupdatevol -> ../../bin/busybox
.../system//usr/sbin/udhcpcd -> ../../bin/busybox

-----
You will probably need to make your busybox binary
setuid root to ensure all configured applets will
work properly.

root@ubuntu:/home/cym/exynos4412/linux/app/busybox-1.21.1# cd ../system/
root@ubuntu:/home/cym/exynos4412/linux/app/system# ls
bin  linuxrc  sbin  usr
root@ubuntu:/home/cym/exynos4412/linux/app/system#
```

( 26 ) 制作的文件系统还需要新建 “dev,etc,lib,mnt,proc,sys,tmp,var” 文件夹 , 使用命令 “mkdir dev etc lib mnt proc sys tmp var” , 如下图 :

The screenshot shows a terminal window titled "192.168.1.9 (3) - SecureCRT". The terminal shows the process of navigating to the "/system" directory and creating new subdirectories "dev", "etc", "lib", "mnt", "proc", "sys", "tmp", and "var". The final output shows the directory structure: "bin", "dev", "etc", "lib", "linuxrc", "mnt", "proc", "sbin", "sys", "tmp", "usr", and "var".

```
root@ubuntu:~#
root@ubuntu:~#
root@ubuntu:~# cd /home/cym/exynos4412/linux/app/
root@ubuntu:/home/cym/exynos4412/linux/app#
root@ubuntu:/home/cym/exynos4412/linux/app# ls
busybox-1.21.1  busybox-1.21.1.tar.bz2  system
root@ubuntu:/home/cym/exynos4412/linux/app#
root@ubuntu:/home/cym/exynos4412/linux/app# cd system/
root@ubuntu:/home/cym/exynos4412/linux/app/system#
root@ubuntu:/home/cym/exynos4412/linux/app/system# ls
bin  linuxrc  sbin  usr
root@ubuntu:/home/cym/exynos4412/linux/app/system#
root@ubuntu:/home/cym/exynos4412/linux/app/system#
root@ubuntu:/home/cym/exynos4412/linux/app/system#
root@ubuntu:/home/cym/exynos4412/linux/app/system#
root@ubuntu:/home/cym/exynos4412/linux/app/system#
root@ubuntu:/home/cym/exynos4412/linux/app/system# mkdir dev etc lib mnt proc sys
s tmp var
root@ubuntu:/home/cym/exynos4412/linux/app/system# ls
bin  dev  etc  lib  linuxrc  mnt  proc  sbin  sys  tmp  usr  var
root@ubuntu:/home/cym/exynos4412/linux/app/system#
```

( 27 ) 使用 “cd etc” 命令进入到刚才创建的 “etc” 文件夹 , 如下图 :

```
root@ubuntu:/home/cym/exynos4412/linux/app/system# 
root@ubuntu:/home/cym/exynos4412/linux/app/system# ls
bin dev etc lib linuxrc mnt proc sbin sys tmp usr var
root@ubuntu:/home/cym/exynos4412/linux/app/system# 
root@ubuntu:/home/cym/exynos4412/linux/app/system# 
root@ubuntu:/home/cym/exynos4412/linux/app/system# cd etc/
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc# 
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc# 
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc#
```

( 28 ) 在 “etc” 目录下 , 使用 “vi eth0-setting” 命令建立 “eth0-setting” 文件 , 并在 “eth0-setting” 文件里输入下面的内容 :

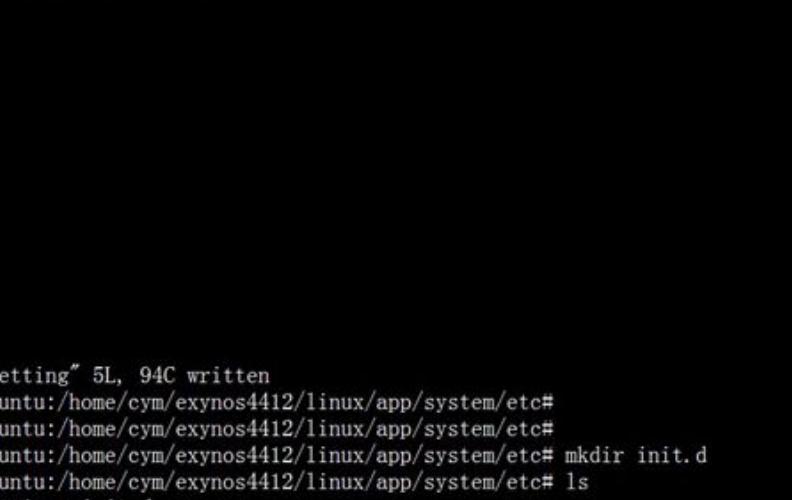
IP=192.168.1.230  
Mask=255.255.255.0  
Gateway=192.168.1.1  
DNS=192.168.1.1  
MAC=08:90:90:90:90:90

( 29 ) 如下图 :

```
192.168.1.9 (3) - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(W) 脚本(S) 工具(T) 帮助(H)
192.168.1.9 192.168.1.9 192.168.1.9 192.168.1.9
root@ubuntu:/home/cym/exynos4412/linux/app/system#
root@ubuntu:/home/cym/exynos4412/linux/app/system#
root@ubuntu:/home/cym/exynos4412/linux/app/system#
root@ubuntu:/home/cym/exynos4412/linux/app/system# cd etc/
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc#
IP=192.168.1.230
Mask=255.255.255.0
Gateway=192.168.1.1
DNS=192.168.1.1
MAC=08:90:90:90:90:90
~
```

( 30 ) 保存并退出 “eth0-setting” 文件 , 使用 “chmod 755 eth0-setting” 命令修改文件的权限 , 如下图 :

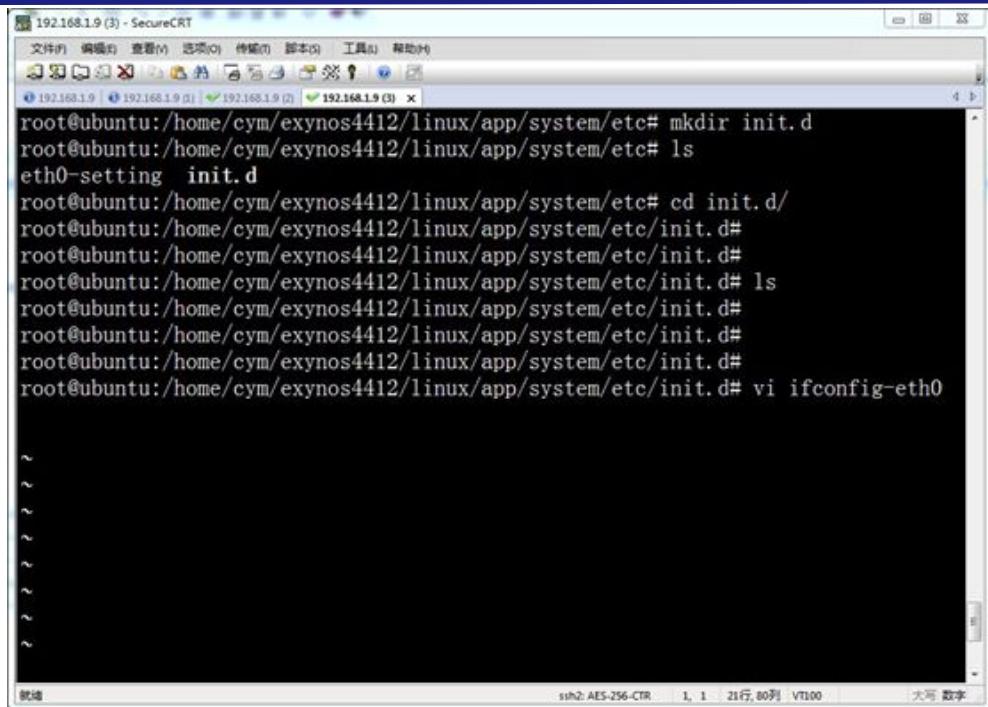
(31) 在 etc 目录下用 “mkdir init.d” 命令建立 “init.d” 文件夹，如下如：



```
"eth0-setting" 5L, 94C written
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc# mkdir init.d
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc# ls
eth0-setting  init.d
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc#
```

(32) 使用“cd init.d”命令进入到“init.d”文件夹，如下图：

( 33 ) 在 “init.d” 文件夹下面使用 “vi ifconfig-eth0” 命令建立 “ifconfig-eth0” 文件，如下图：



( 34 ) 然后在 “ifconfig-eth0” 文件中输入下面的内容 :

```
#!/bin/sh

echo -n Try to bring eth0 interface up.....>/dev/ttySAC2

if [ -f /etc/eth0-setting ] ; then
    source /etc/eth0-setting

    if grep -q "^/dev/root / nfs " /etc/mtab ; then
        echo -n NFS root ... > /dev/ttySAC2
    else
        ifconfig eth0 down
        ifconfig eth0 hw ether $MAC
        ifconfig eth0 $IP netmask $Mask up
        route add default gw $Gateway
    fi

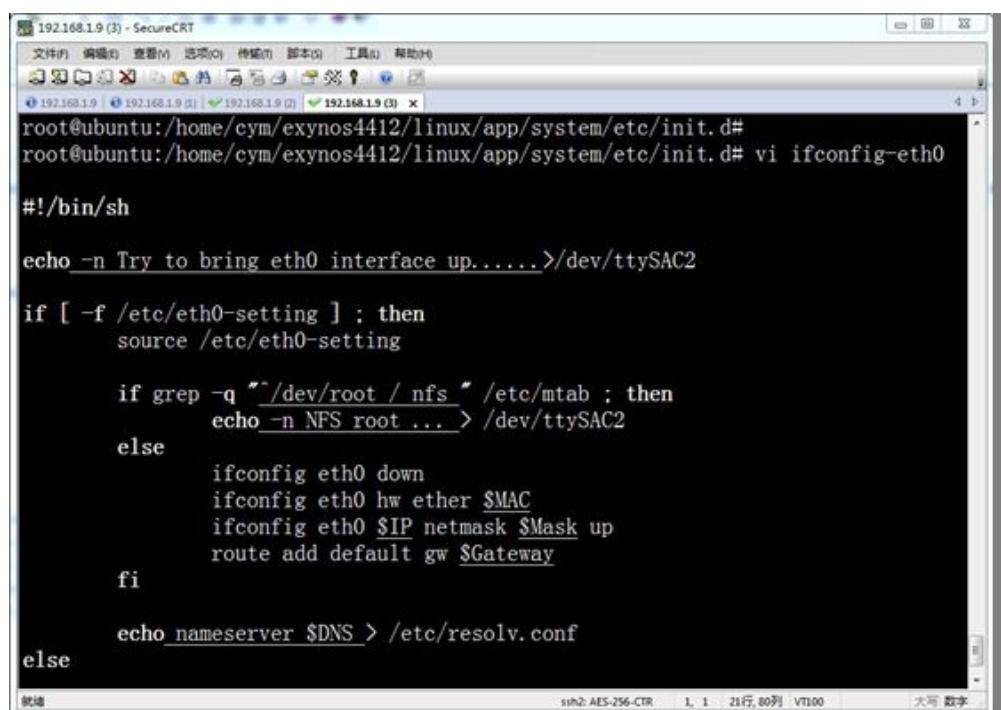
    echo nameserver $DNS > /etc/resolv.conf
```

```
else
```

```
    if grep -q "^/dev/root / nfs " /etc/mtab ; then
        echo -n NFS root ... > /dev/ttySAC2
    else
        /sbin/ifconfig eth0 192.168.253.12 netmask 255.255.255.0 up
    fi
fi

echo Done > /dev/ttySAC2
```

( 35 ) 如下图 :



```
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/init.d# vi ifconfig-eth0
#!/bin/sh

echo -n Try to bring eth0 interface up.....>/dev/ttySAC2

if [ -f /etc/eth0-setting ] ; then
    source /etc/eth0-setting

    if grep -q "/dev/root / nfs " /etc/mtab ; then
        echo -n NFS root ... > /dev/ttySAC2
    else
        ifconfig eth0 down
        ifconfig eth0 hw ether $MAC
        ifconfig eth0 $IP netmask $Mask up
        route add default gw $Gateway
    fi
else
    echo nameserver $DNS > /etc/resolv.conf
fi
```

( 36 ) 保存并退出 “ifconfig-eth0” 文件，使用 “chmod 755 ifconfig-eth0” 命令修改权限，如下图：

192.168.1.9 (3) - SecureCRT

文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(U) 帮助(H)

192.168.1.9 | 192.168.1.9 (1) | 192.168.1.9 (2) | 192.168.1.9 (3) |

```
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/init.d# chmod 755 ifconfig
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/init.d# ifconfig -eth0
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/init.d# root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/init.d#
```

(37) 然后在“init.d”文件夹下使用“vi rcS”命令建立“rcS”文件，如下图：

( 38 ) 然后在 “rcS” 文件输入下面的内容 :

```
#!/bin/sh  
PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/bin:  
runlevel=S
```

```
prevlevel=N
umask 022
export PATH runlevel prevlevel

#
#      Trap CTRL-C &c only in this shell so we can interrupt subprocesses.
#
trap ":" INT QUIT TSTP
/bin/hostname iTOP-4412

#
#/bin/mount -n -t proc none /proc
#/bin/mount -n -t sysfs none /sys
#/bin/mount -n -t usbfs none /proc/bus/usb
#/bin/mount -t ramfs none /dev
[ -e /proc/1 ] || /bin/mount -n -t proc none /proc
[ -e /sys/class ] || /bin/mount -n -t sysfs none /sys
[ -e /dev/tty ] || /bin/mount -t ramfs none /dev

echo /sbin/mdev > /proc/sys/kernel/hotplug
/sbin/mdev -s
#/bin/hotplug
# mounting file system specified in /etc/fstab
mkdir -p /dev/pts
mkdir -p /dev/shm
/bin/mount -n -t devpts none /dev/pts -o mode=0622
/bin/mount -n -t tmpfs tmpfs /dev/shm
#/bin/mount -n -t ramfs none /tmp
#/bin/mount -n -t ramfs none /var
mkdir -p /var/empty
mkdir -p /var/log
mkdir -p /var/log/boa
mkdir -p /var/lock
```

```
mkdir -p /var/run
mkdir -p /var/tmp

ln -sf /dev/ttYS2 /dev/tty2
ln -sf /dev/ttYS2 /dev/tty3
ln -sf /dev/ttYS2 /dev/tty4

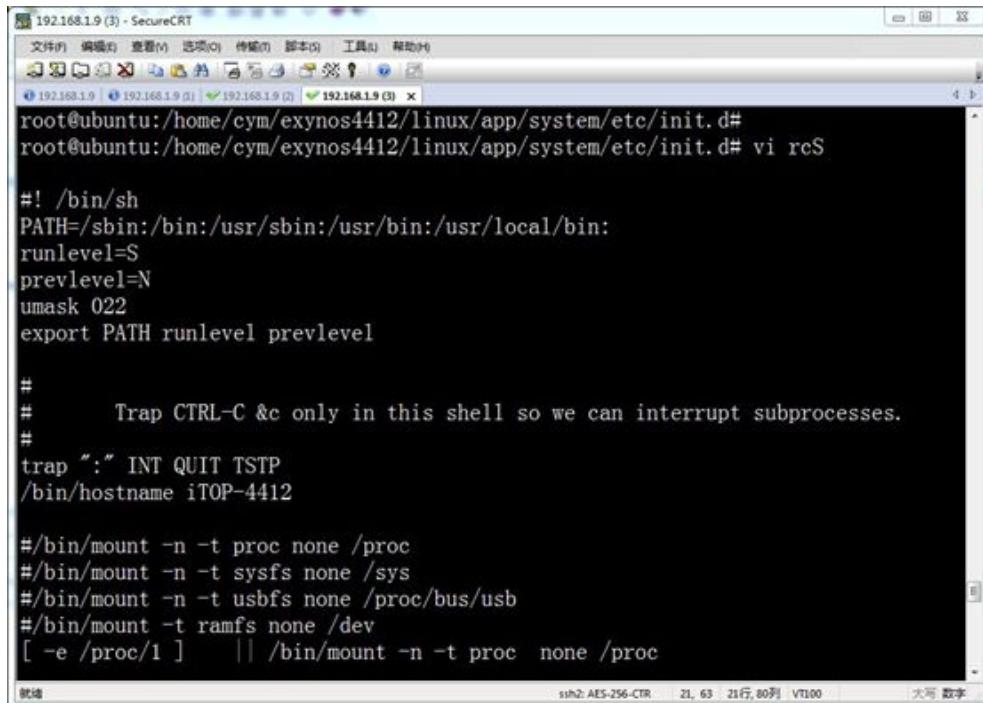
syslogd
/etc/rc.d/init.d/netd start
echo "                " > /dev/tty1
echo "Starting networking..." > /dev/tty1
#sleep 1
#/etc/rc.d/init.d/httpd start
#echo "                " > /dev/tty1
#echo "Starting web server..." > /dev/tty1
#sleep 1
#/etc/rc.d/init.d/leds start
#echo "                " > /dev/tty1
#echo "Starting leds service..." > /dev/tty1
#echo "
#sleep 1

#echo "*****" > /dev/ttYSAC2
#echo "      http://www.topeet.com.cn      " > /dev/ttYSAC2
#echo "*****" > /dev/ttYSAC2
#echo "*****"
#echo "      http://www.topeet.com.cn      "
#echo "*****"

mkdir /mnt/disk
```

```
sleep 1  
/sbin/ifconfig lo 127.0.0.1  
/etc/init.d/ifconfig-eth0
```

( 39 ) 如下图 :



The screenshot shows a SecureCRT window titled "192.168.1.9 (3) - SecureCRT". The terminal session is running as root on an Ubuntu system. The user is editing the "/etc/init.d/rcS" file using the vi editor. The script contains configuration for the system's runlevels, trap handlers, and mounting of various filesystems like proc, sysfs, and dev.

```
#!/bin/sh  
PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/bin:  
runlevel=S  
prevlevel=N  
umask 022  
export PATH runlevel prevlevel  
  
#  
#      Trap CTRL-C &c only in this shell so we can interrupt subprocesses.  
#  
trap ":" INT QUIT TSTP  
/bin/hostname iTOP-4412  
  
#/bin/mount -n -t proc none /proc  
#/bin/mount -n -t sysfs none /sys  
#/bin/mount -n -t usbfs none /proc/bus/usb  
#/bin/mount -t ramfs none /dev  
[ -e /proc/1 ] || /bin/mount -n -t proc none /proc
```

( 40 ) 然后保存并退出 “rcS” 文件 , 使用 “chmod 755 rcS” 命令修改 “rcS” 文件的权限 , 如下图 :

```
192.168.1.9 (3) - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(U) 帮助(H)
192.168.1.9 | 192.168.1.9 (3) | 192.168.1.9 (2) | 192.168.1.9 (3) x
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/init.d#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/init.d# chmod 755 rcS
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/init.d#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/init.d#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/init.d#
```

(41) 接下来输入“ls”命令可以看到新创建的两个文件“ifconfig-eth0”和“rcS”，

如下图：

(42) 接下来使用“cd ..”命令返回到“init.d”文件夹的上一级目录“etc”，如下图：

(43) 接下来在“etc”目录下使用“vi passwd”命令建立文件“passwd”，如下图：

```
192.168.1.9 (3) - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(W) 帮助(H)
192.168.1.9 192.168.1.9 (1) 192.168.1.9 (2) 192.168.1.9 (3) x
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/init.d#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/init.d#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/init.d#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/init.d#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/init.d#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/init.d#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/init.d# ls
ifconfig-eth0 rcS
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/init.d#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/init.d# cd ..
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc# vi passwd

~
~
```

( 44 ) 在新建立的 “passwd” 文件中输入下面的内容 :

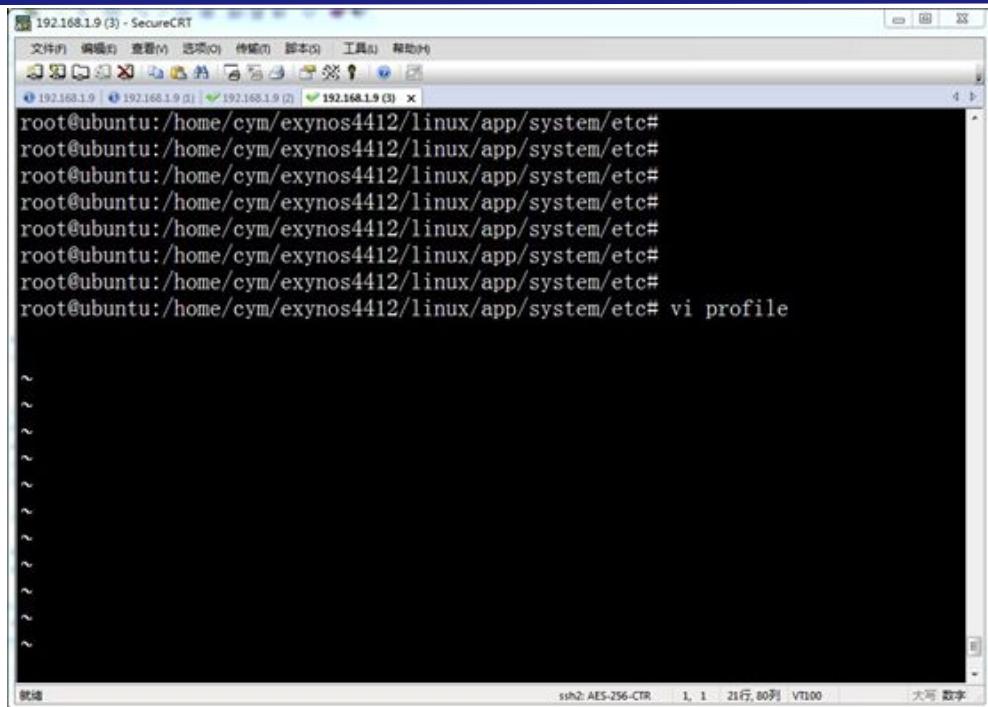
```
root::0:0:root:/bin/sh  
bin:*:1:1:bin:/bin:  
daemon:*:2:2:daemon:/sbin:
```

nobody:\*:99:99:Nobody:/:

( 45 ) 输入结果如下图 :

( 46 ) 然后保存并退出 “passwd” 文件，使用 “chmod 755 passwd” 命令修改 “passwd” 文件的权限，如下图：

( 47 ) 然后使用 “vi profile” 命令在 “etc” 目录建立 “profile” 文件 , 如下图 :



( 48 ) 然后在 “profile” 文件中输入下面的内容 :

```
# Ash profile
# vim: syntax=sh

# No core files by default
ulimit -S -c 0 > /dev/null 2>&1

USER=`id -un`
LOGNAME=$USER
PS1='[$USER@$HOSTNAME]# '
PATH=$PATH

HOSTNAME=`/bin/hostname`


export USER LOGNAME PS1 PATH
```

( 49 ) 输入结果如下图 :

192.168.1.9 (3) - SecureCRT

文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(U) 帮助(H)

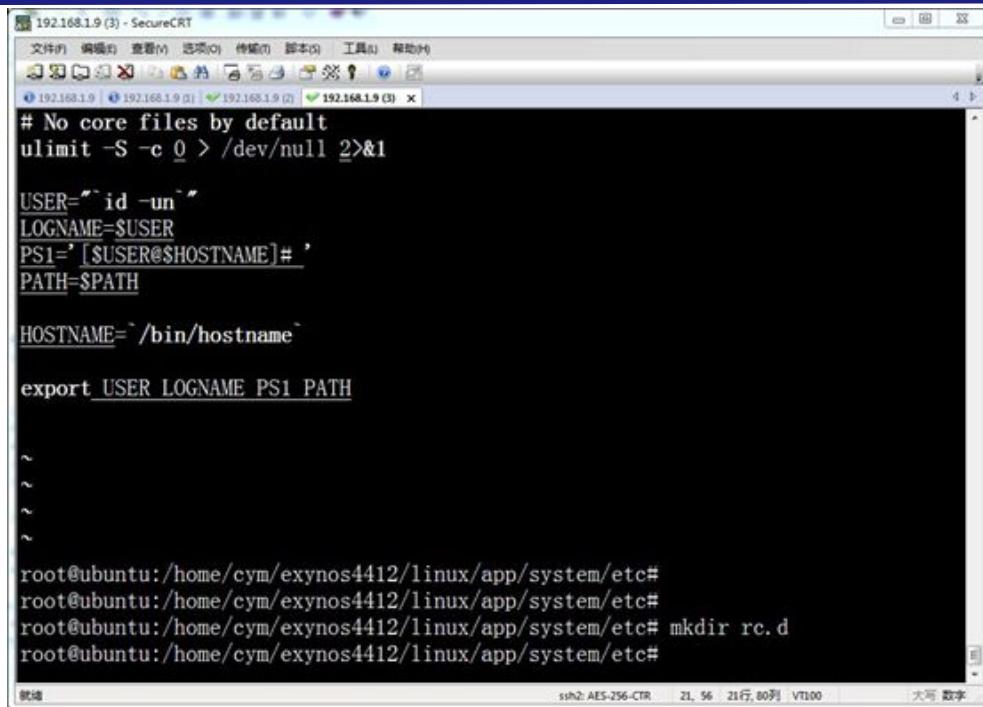
192.168.1.9 | 192.168.1.9 (1) | 192.168.1.9 (2) | 192.168.1.9 (3) x

```
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc#  
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc# vi profile
```

```
# Ash profile  
# vim: syntax=sh  
  
# No core files by default  
ulimit -S -c 0 > /dev/null 2>&1  
  
USER=`` id -un ``  
LOGNAME=$USER  
PS1='[$USER@$HOSTNAME]# '  
PATH=$PATH  
  
HOSTNAME=`/bin/hostname`  
  
export USER LOGNAME PS1 PATH  
  
~  
~
```

( 50 ) 然后保存并退出 “profile” 文件，使用 “chmod 755 profile” 命令修改 “profile” 文件的权限，如下图：

( 51 ) 接下来使用 “`mkdir rc.d`” 命令在 “`etc`” 目录建立文件夹 “`rc.d`” , 如下图 :



```
# No core files by default
ulimit -S -c 0 > /dev/null 2>&1

USER=`id -un`
LOGNAME=$USER
PS1='[$USER@$HOSTNAME]# '
PATH=$PATH

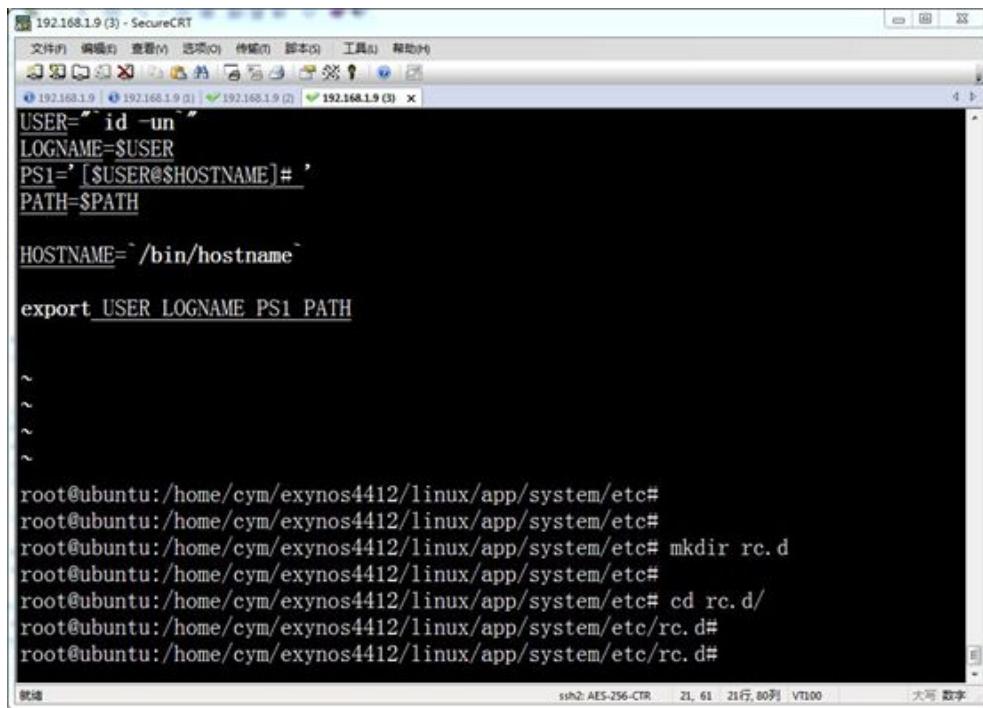
HOSTNAME=`/bin/hostname`

export USER LOGNAME PS1 PATH

~
~
~
~

root@ubuntu:/home/cym/exynos4412/linux/app/system/etc#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc# mkdir rc.d
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc#
```

( 52 ) 使用 “cd rc.d” 命令进入到刚才建立的 “rc.d” 文件夹 , 如下图 :



```
USER=`id -un`
LOGNAME=$USER
PS1='[$USER@$HOSTNAME]# '
PATH=$PATH

HOSTNAME=`/bin/hostname`

export USER LOGNAME PS1 PATH

~
~
~
~

root@ubuntu:/home/cym/exynos4412/linux/app/system/etc#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc# mkdir rc.d
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc# cd rc.d/
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/rc.d#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/rc.d#
```

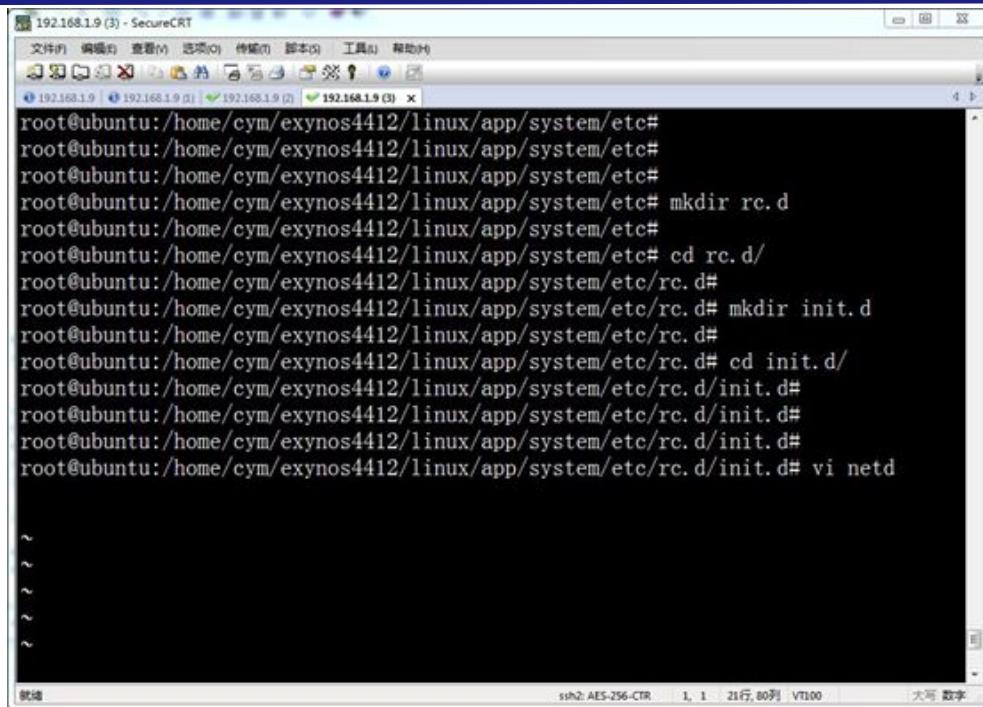
( 53 ) 接下来在 “rc.d” 目录下使用 “mkdir init.d” 命令建立 “init.d” 文件夹 , 如下

图 :

( 54 ) 然后使用 “cd init.d” 命令进入到刚才建立的 “init.d” 文件夹，如下图：

```
192.168.1.9 (3) - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(U) 帮助(H)
192.168.1.9 | 192.168.1.9 (1) | 192.168.1.9 (2) | 192.168.1.9 (3) x
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc# mkdir rc.d
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc# cd rc.d/
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/rc.d#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/rc.d# mkdir init.d
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/rc.d#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/rc.d# cd init.d/
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/rc.d/init.d#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/rc.d/init.d#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/rc.d/init.d#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/rc.d/init.d#
```

(55) 接着在“init.d”文件夹，使用“vi netd”命令建立“netd”文件，如下图：



```
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc# 
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc# 
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc# 
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc# mkdir rc.d
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc# 
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc# cd rc.d/
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/rc.d# 
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/rc.d# mkdir init.d
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/rc.d# 
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/rc.d# cd init.d/
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/rc.d/init.d# 
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/rc.d/init.d# 
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/rc.d/init.d# 
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/rc.d/init.d# vi netd

~
~
~
~
~
```

( 56 ) 然后在 “netd” 文件里面输入下面的内容 :

```
#!/bin/sh

base=inetd

# See how we were called.
case "$1" in
    start)
        /usr/sbin/$base
        ;;
    stop)
        pid=`/bin/pidof $base`
        if [ -n "$pid" ]; then
            kill -9 $pid
        fi
        ;;
esac
```

exit 0

( 57 ) 输入结果 , 如下图 :

```
192.168.1.9 (3) - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(U) 帮助(H)
192.168.1.9 | 192.168.1.9 (1) | 192.168.1.9 (2) | 192.168.1.9 (3) x
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/rc.d/init.d# vi netd

#!/bin/sh

base=inetd

# See how we were called.
case "$1" in
    start)
        /usr/sbin/$base
    ;;
    stop)
        pid=`/bin/pidof $base`
        if [ -n "$pid" ]; then
            kill -9 $pid
        fi
    ;;
esac

exit 0
```

( 58 ) 然后保存并退出 “netd” 文件，使用 “chmod 755 netd” 命令修改 “netd” 文件的权限，如下图：

( 59 ) 接着使用 “`cd ../../`” 命令返回到 “etc” 目录，如下图：

```
192.168.1.9 (3) - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(W) 脚本(S) 工具(T) 帮助(H)

root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/rc.d/init.d#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/rc.d/init.d# chmod 755 netd
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/rc.d/init.d#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/rc.d/init.d# cd ../../
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc#
```

( 60 ) 接着使用 “cd ..” 命令返回到 “system” 目录，如下图：

```
192.168.1.9 (3) - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(W) 脚本(S) 工具(T) 帮助(H)
192.168.1.9 192.168.1.9 (1) 192.168.1.9 (2) 192.168.1.9 (3) x

root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/rc.d/init.d#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/rc.d/init.d# chmod 755 netd
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/rc.d/init.d#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/rc.d/init.d# cd ../../
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc# cd ..
root@ubuntu:/home/cym/exynos4412/linux/app/system#
root@ubuntu:/home/cym/exynos4412/linux/app/system#
root@ubuntu:/home/cym/exynos4412/linux/app/system#
```

(61) 接着使用 “cd lib” 命令进入到 “lib” 目录，如下图：

```
192.168.1.9 (3) - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(U) 帮助(H)
192.168.1.9 192.168.1.9 (3) 192.168.1.9 (2) 192.168.1.9 (3) x

root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/rc.d/init.d#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/rc.d/init.d# cd ../../
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc# cd ..
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc# cd ..
root@ubuntu:/home/cym/exynos4412/linux/app/system# cd lib/
root@ubuntu:/home/cym/exynos4412/linux/app/system/lib#
```

( 62 ) 因为我们使用的交叉编译环境和编译内核是一样的，所以我们的编译器在文件夹 “/usr/local/arm/arm-2009q3” 中。Busybox 编译生成的二进制文件是以动态链接库的形式运行，所以我们需要拷贝编译器里面的库文件到 “lib” 目录，使用命令

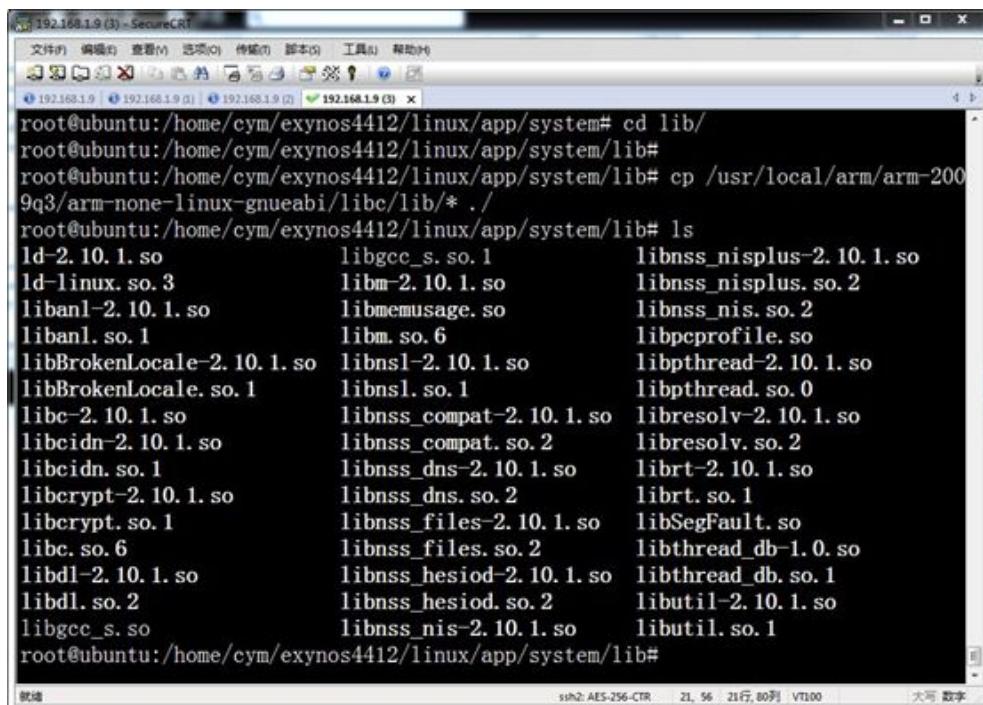
```
"cp /usr/local/arm/arm-2009q3/arm-none-Linux-gnueabi/libc/lib/* ./"
```

( 63 ) 执行结果如下图 :

```
192.168.1.9 (3) - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(U) 帮助(H)
192.168.1.9 192.168.1.9 (1) 192.168.1.9 (2) 192.168.1.9 (3) x

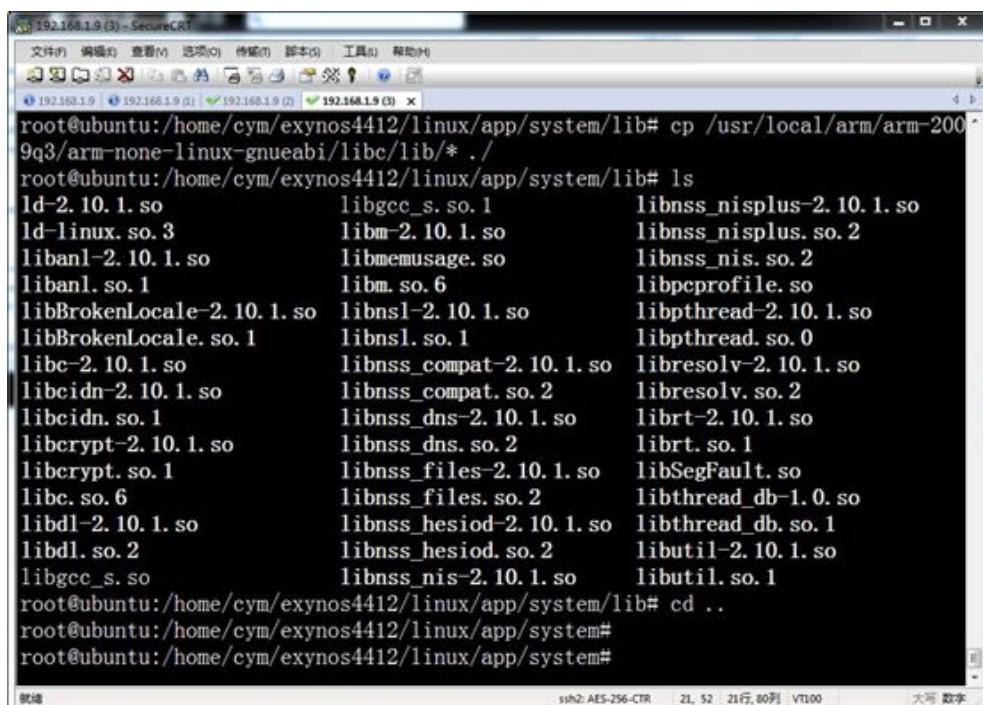
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc/rc.d/init.d#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc# cd ../../
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc#
root@ubuntu:/home/cym/exynos4412/linux/app/system/etc# cd ../
root@ubuntu:/home/cym/exynos4412/linux/app/system#
root@ubuntu:/home/cym/exynos4412/linux/app/system# cd lib/
root@ubuntu:/home/cym/exynos4412/linux/app/system/lib#
root@ubuntu:/home/cym/exynos4412/linux/app/system/lib# cp /usr/local/arm/arm-200
9q3/arm-none-linux-gnueabi/libc/lib/* ./
root@ubuntu:/home/cym/exynos4412/linux/app/system/lib#
```

( 64 ) 可以使用 “ls” 命令查看一下拷贝过来的库文件 , 如下图 :



```
root@ubuntu:/home/cym/exynos4412/linux/app/system# cd lib/
root@ubuntu:/home/cym/exynos4412/linux/app/system/lib#
root@ubuntu:/home/cym/exynos4412/linux/app/system/lib# cp /usr/local/arm/arm-200
9q3/arm-none-linux-gnueabi/libc/lib/*.so
root@ubuntu:/home/cym/exynos4412/linux/app/system/lib# ls
ld-2.10.1.so          libgcc_s.so.1           libnss_nisplus-2.10.1.so
ld-linux.so.3           libm-2.10.1.so          libnss_nisplus.so.2
libanl-2.10.1.so       libmemusage.so        libnss_nis.so.2
libanl.so.1             libm.so.6              libpcprofile.so
libBrokenLocale-2.10.1.so libnsl-2.10.1.so      libpthread-2.10.1.so
libBrokenLocale.so.1    libnsl.so.1            libpthread.so.0
libc-2.10.1.so         libnss_compat-2.10.1.so libresolv-2.10.1.so
libcidn-2.10.1.so      libnss_compat.so.2     libresolv.so.2
libcidn.so.1            libnss_dns-2.10.1.so   librt-2.10.1.so
libcrypt-2.10.1.so     libnss_dns.so.2       librt.so.1
libcrypt.so.1           libnss_files-2.10.1.so libSegFault.so
libc.so.6               libnss_files.so.2      libthread_db-1.0.so
libdl-2.10.1.so         libnss_hesiod-2.10.1.so libthread_db.so.1
libdl.so.2              libnss_hesiod.so.2    libutil-2.10.1.so
libgcc_s.so             libnss_nis-2.10.1.so   libutil.so.1
root@ubuntu:/home/cym/exynos4412/linux/app/system/lib#
```

( 65 ) 库文件拷贝完成后 , 使用 “cd ..” 命令返回到 “system” 目录 , 如下图 :



```
root@ubuntu:/home/cym/exynos4412/linux/app/system/lib# cp /usr/local/arm/arm-200
9q3/arm-none-linux-gnueabi/libc/lib/*.so
root@ubuntu:/home/cym/exynos4412/linux/app/system/lib# ls
ld-2.10.1.so          libgcc_s.so.1           libnss_nisplus-2.10.1.so
ld-linux.so.3           libm-2.10.1.so          libnss_nisplus.so.2
libanl-2.10.1.so       libmemusage.so        libnss_nis.so.2
libanl.so.1             libm.so.6              libpcprofile.so
libBrokenLocale-2.10.1.so libnsl-2.10.1.so      libpthread-2.10.1.so
libBrokenLocale.so.1    libnsl.so.1            libpthread.so.0
libc-2.10.1.so         libnss_compat-2.10.1.so libresolv-2.10.1.so
libcidn-2.10.1.so      libnss_compat.so.2     libresolv.so.2
libcidn.so.1            libnss_dns-2.10.1.so   librt-2.10.1.so
libcrypt-2.10.1.so     libnss_dns.so.2       librt.so.1
libcrypt.so.1           libnss_files-2.10.1.so libSegFault.so
libc.so.6               libnss_files.so.2      libthread_db-1.0.so
libdl-2.10.1.so         libnss_hesiod-2.10.1.so libthread_db.so.1
libdl.so.2              libnss_hesiod.so.2    libutil-2.10.1.so
libgcc_s.so             libnss_nis-2.10.1.so   libutil.so.1
root@ubuntu:/home/cym/exynos4412/linux/app/system/lib# cd ..
root@ubuntu:/home/cym/exynos4412/linux/app/system#
root@ubuntu:/home/cym/exynos4412/linux/app/system#
```

( 66 ) 接下来使用 “cd var” 命令进入到 “var” 目录 , 如下图 :

```

192.168.1.9 (3) - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(U) 帮助(H)
192.168.1.9 | 192.168.1.9 (1) | 192.168.1.9 (2) | 192.168.1.9 (3) x
root@ubuntu:/home/cym/exynos4412/linux/app/system/lib# ls
ld-2.10.1.so          libgcc_s.so.1           libnss_nisplus-2.10.1.so
libc-2.10.1.so         libm-2.10.1.so          libnss_nisplus.so.2
libanl-2.10.1.so       libmemusage.so        libnss_nis.so.2
libanl.so.1             libm.so.6              libpcprofile.so
libBrokenLocale-2.10.1.so libnsl-2.10.1.so      libpthread-2.10.1.so
libBrokenLocale.so.1    libnsl.so.1            libpthread.so.0
libc-2.10.1.so         libnss_compat-2.10.1.so libresolv-2.10.1.so
libcidn-2.10.1.so      libnss_compat.so.2     libresolv.so.2
libcidn.so.1            libnss_dns-2.10.1.so   librt-2.10.1.so
libcrypt-2.10.1.so     libnss_dns.so.2       librt.so.1
libcrypt.so.1           libnss_files-2.10.1.so libSegFault.so
libc.so.6               libnss_files.so.2      libthread_db-1.0.so
libdl-2.10.1.so         libnss_hesiod-2.10.1.so libthread_db.so.1
libdl.so.2              libnss_hesiod.so.2    libutil-2.10.1.so
libgcc_s.so             libnss_nis-2.10.1.so   libutil.so.1
root@ubuntu:/home/cym/exynos4412/linux/app/system/lib# cd ..
root@ubuntu:/home/cym/exynos4412/linux/app/system#
root@ubuntu:/home/cym/exynos4412/linux/app/system# cd var/
root@ubuntu:/home/cym/exynos4412/linux/app/system/var#
root@ubuntu:/home/cym/exynos4412/linux/app/system/var#

```

( 67 ) 然后使用 “mkdir lib lock log run tmp” 命令在 “var” 目录下建立 “lib , lock , log , run , tmp” 五个目录 , 如下图 :

```

192.168.1.9 (3) - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(U) 帮助(H)
192.168.1.9 | 192.168.1.9 (1) | 192.168.1.9 (2) | 192.168.1.9 (3) x
root@ubuntu:/home/cym/exynos4412/linux/app/system/lib# ls
libanl-2.10.1.so      libmemusage.so        libnss_nis.so.2
libanl.so.1             libm.so.6              libpcprofile.so
libBrokenLocale-2.10.1.so libnsl-2.10.1.so      libpthread-2.10.1.so
libBrokenLocale.so.1    libnsl.so.1            libpthread.so.0
libc-2.10.1.so         libnss_compat-2.10.1.so libresolv-2.10.1.so
libcidn-2.10.1.so      libnss_compat.so.2     libresolv.so.2
libcidn.so.1            libnss_dns-2.10.1.so   librt-2.10.1.so
libcrypt-2.10.1.so     libnss_dns.so.2       librt.so.1
libcrypt.so.1           libnss_files-2.10.1.so libSegFault.so
libc.so.6               libnss_files.so.2      libthread_db-1.0.so
libdl-2.10.1.so         libnss_hesiod-2.10.1.so libthread_db.so.1
libdl.so.2              libnss_hesiod.so.2    libutil-2.10.1.so
libgcc_s.so             libnss_nis-2.10.1.so   libutil.so.1
root@ubuntu:/home/cym/exynos4412/linux/app/system/lib# cd ..
root@ubuntu:/home/cym/exynos4412/linux/app/system#
root@ubuntu:/home/cym/exynos4412/linux/app/system# cd var/
root@ubuntu:/home/cym/exynos4412/linux/app/system/var#
root@ubuntu:/home/cym/exynos4412/linux/app/system/var# mkdir lib lock log run
tmp
root@ubuntu:/home/cym/exynos4412/linux/app/system/var#
root@ubuntu:/home/cym/exynos4412/linux/app/system/var#

```

( 68 ) 使用 “ls” 命令查看一下创建的这几个目录 , 如下图 :

```

192.168.1.9 (3) - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(W) 帮助(H)
192.168.1.9 | 192.168.1.9 (1) | 192.168.1.9 (2) | 192.168.1.9 (3) x
libBrokenLocale-2.10.1.so libnsl-2.10.1.so libpthread-2.10.1.so
libBrokenLocale.so.1 libnsl.so.1 libpthread.so.0
libc-2.10.1.so libnss_compat-2.10.1.so libresolv-2.10.1.so
libcidn-2.10.1.so libnss_compat.so.2 libresolv.so.2
libcidn.so.1 libnss_dns-2.10.1.so libert-2.10.1.so
libcrypt-2.10.1.so libnss_dns.so.2 libert.so.1
libcrypt.so.1 libnss_files-2.10.1.so libSegFault.so
libc.so.6 libnss_files.so.2 libthread_db-1.0.so
libdl-2.10.1.so libnss_hesiod-2.10.1.so libthread_db.so.1
libdl.so.2 libnss_hesiod.so.2 libutil-2.10.1.so
libgcc_s.so libnss_nis-2.10.1.so libutil.so.1
root@ubuntu:/home/cym/exynos4412/linux/app/system/lib# cd ..
root@ubuntu:/home/cym/exynos4412/linux/app/system#
root@ubuntu:/home/cym/exynos4412/linux/app/system# cd var/
root@ubuntu:/home/cym/exynos4412/linux/app/system/var#
root@ubuntu:/home/cym/exynos4412/linux/app/system/var# mkdir lib lock log run
tmp
root@ubuntu:/home/cym/exynos4412/linux/app/system/var#
root@ubuntu:/home/cym/exynos4412/linux/app/system/var# ls
lib lock log run tmp
root@ubuntu:/home/cym/exynos4412/linux/app/system/var#

```

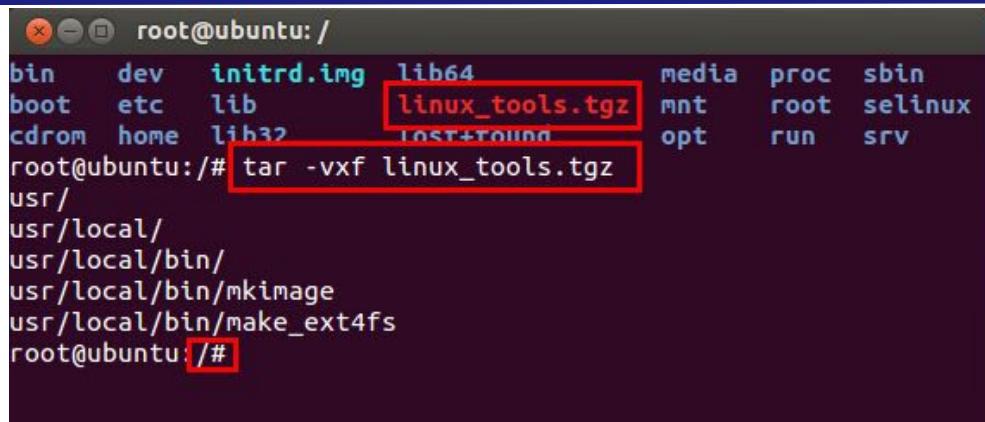
( 69 ) 至此 , 文件系统所需要的文件都已经创建好了 , 使用 “cd ../../” 命令返回到 “system” 文件夹的上一级目录 , 如下图 :

```

192.168.1.9 (3) - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(W) 帮助(H)
192.168.1.9 | 192.168.1.9 (1) | 192.168.1.9 (2) | 192.168.1.9 (3) x
libcidn-2.10.1.so libnss_compat.so.2 libresolv.so.2
libcidn.so.1 libnss_dns-2.10.1.so libert-2.10.1.so
libcrypt-2.10.1.so libnss_dns.so.2 libert.so.1
libcrypt.so.1 libnss_files-2.10.1.so libSegFault.so
libc.so.6 libnss_files.so.2 libthread_db-1.0.so
libdl-2.10.1.so libnss_hesiod-2.10.1.so libthread_db.so.1
libdl.so.2 libnss_hesiod.so.2 libutil-2.10.1.so
libgcc_s.so libnss_nis-2.10.1.so libutil.so.1
root@ubuntu:/home/cym/exynos4412/linux/app/system/lib# cd ..
root@ubuntu:/home/cym/exynos4412/linux/app/system#
root@ubuntu:/home/cym/exynos4412/linux/app/system# cd var/
root@ubuntu:/home/cym/exynos4412/linux/app/system/var#
root@ubuntu:/home/cym/exynos4412/linux/app/system/var# mkdir lib lock log run
tmp
root@ubuntu:/home/cym/exynos4412/linux/app/system/var#
root@ubuntu:/home/cym/exynos4412/linux/app/system/var# ls
lib lock log run tmp
root@ubuntu:/home/cym/exynos4412/linux/app/system/var#
root@ubuntu:/home/cym/exynos4412/linux/app/system/var# cd ../../..
root@ubuntu:/home/cym/exynos4412/linux/app#
root@ubuntu:/home/cym/exynos4412/linux/app#

```

(70)拷贝光盘 “tools” 文件夹下的压缩包 “Linux\_tools.tgz” 到 Ubuntu 的 “/” 目录下 , 并使用命令 “tar -vxf linux\_tools.tgz” 解压 , 注意 , 在搭建 “Linux-QT” 环境的时候也将这个工具放到这个目录了 , 如下图。

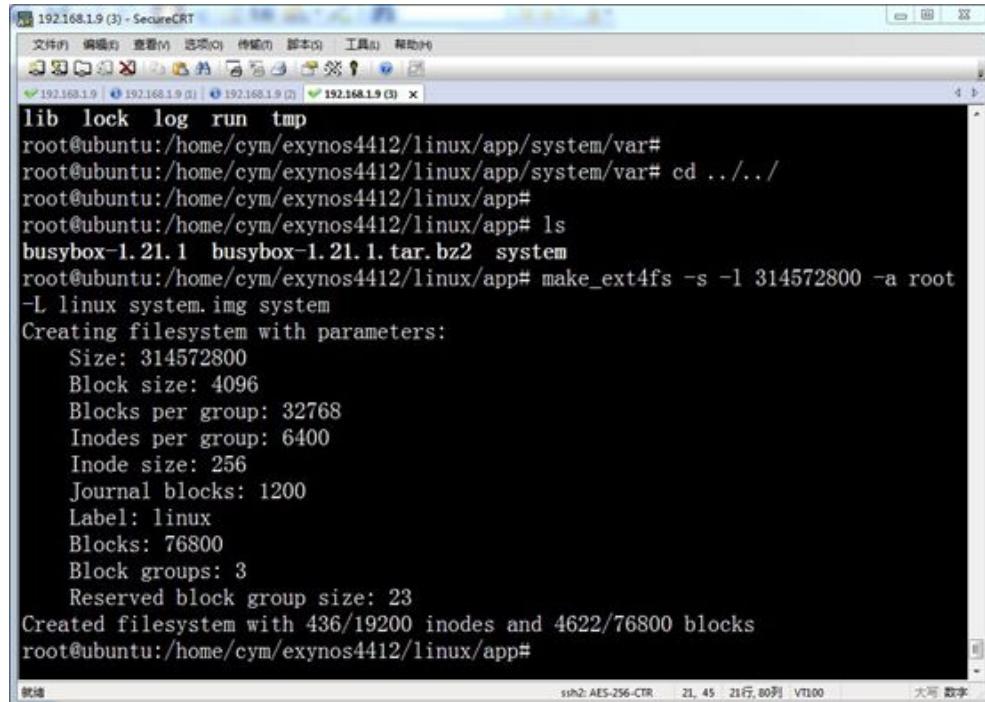


```
root@ubuntu: /  
bin dev initrd.img lib64 media proc sbin  
boot etc lib linux_tools.tgz mnt root selinux  
cdrom home lib32 LOST+FOUND opt run srv  
root@ubuntu:/# tar -vxf linux_tools.tgz  
usr/  
usr/local/  
usr/local/bin/  
usr/local/bin/mkimage  
usr/local/bin/make_ext4fs  
root@ubuntu:/#
```

( 71 ) 下面我们可以使用命令

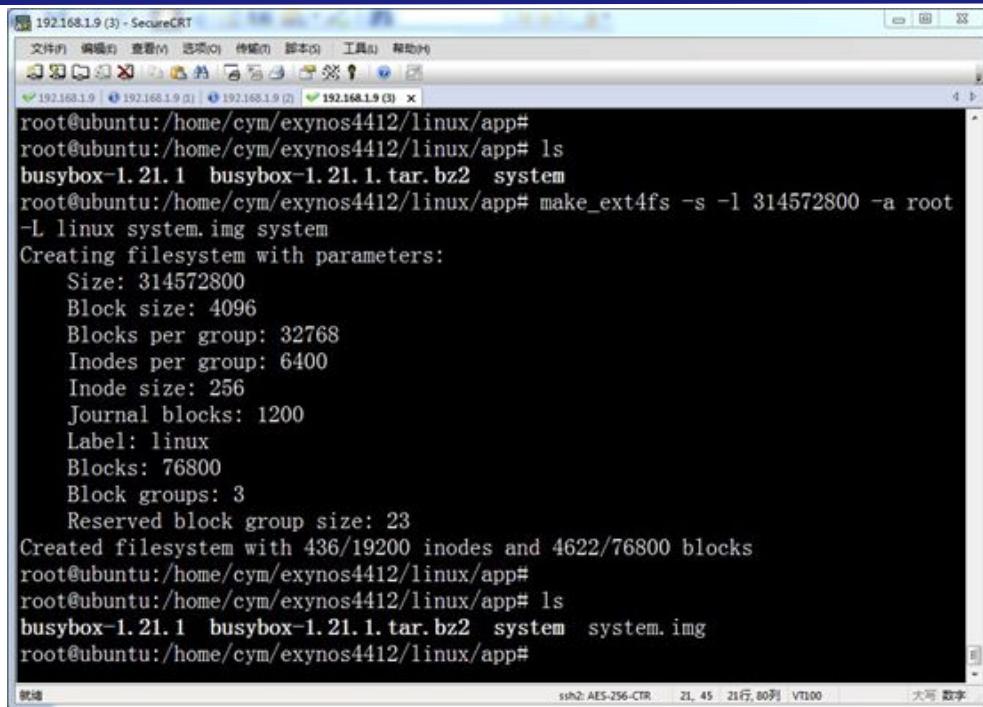
“make\_ext4fs -s -l 314572800 -a root -L Linux system.img system”

执行该命令后，会生成 “system.img” 文件系统镜像，如下图：



```
lib lock log run tmp  
root@ubuntu:/home/cym/exynos4412/linux/app/system/var#  
root@ubuntu:/home/cym/exynos4412/linux/app/system/var# cd ../../  
root@ubuntu:/home/cym/exynos4412/linux/app#  
root@ubuntu:/home/cym/exynos4412/linux/app# ls  
busybox-1.21.1 busybox-1.21.1.tar.bz2 system  
root@ubuntu:/home/cym/exynos4412/linux/app# make_ext4fs -s -l 314572800 -a root  
-L linux system. img system  
Creating filesystem with parameters:  
Size: 314572800  
Block size: 4096  
Blocks per group: 32768  
Inodes per group: 6400  
Inode size: 256  
Journal blocks: 1200  
Label: linux  
Blocks: 76800  
Block groups: 3  
Reserved block group size: 23  
Created filesystem with 436/19200 inodes and 4622/76800 blocks  
root@ubuntu:/home/cym/exynos4412/linux/app#
```

( 72 ) 使用 “ls” 命令查看一下生成的 “system.img”



The screenshot shows a terminal window titled "192.168.1.9 (3) - SecureCRT". The command history and output are as follows:

```
root@ubuntu:/home/cym/exynos4412/linux/app# ls
busybox-1.21.1 busybox-1.21.1.tar.bz2 system
root@ubuntu:/home/cym/exynos4412/linux/app# make_ext4fs -s -l 314572800 -a root
-L linux system. img system
Creating filesystem with parameters:
  Size: 314572800
  Block size: 4096
  Blocks per group: 32768
  Inodes per group: 6400
  Inode size: 256
  Journal blocks: 1200
  Label: linux
  Blocks: 76800
  Block groups: 3
  Reserved block group size: 23
Created filesystem with 436/19200 inodes and 4622/76800 blocks
root@ubuntu:/home/cym/exynos4412/linux/app# root@ubuntu:/home/cym/exynos4412/linux/app# ls
busybox-1.21.1 busybox-1.21.1.tar.bz2 system system. img
root@ubuntu:/home/cym/exynos4412/linux/app#
```

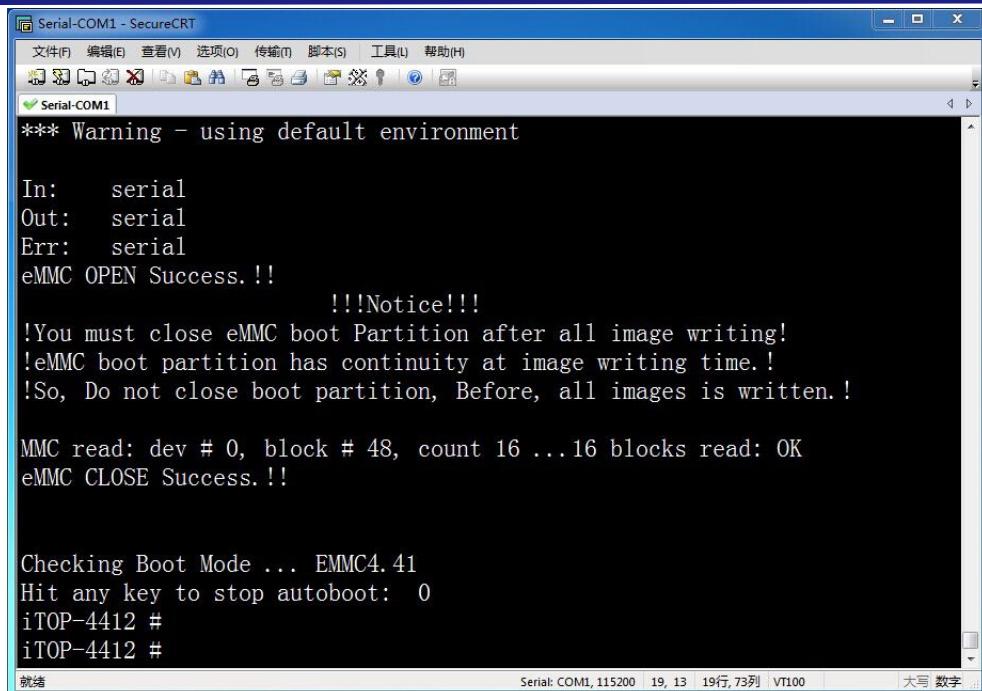
( 73 ) 现在已经完成了 Linux 文件系统的制作 , 可以把我们制作的 “system.img” 烧写到 iTOP-4412 开发板上 , 烧写的方式和 Linux-QT 文件系统的烧写一样。

还需要烧写其它三个文件 , 其中 “u-boot-iTOP-4412.bin” 文件不用烧写 , 该文件系统所需的其他两个镜像 “zImage” 、 "ramdisk-uboot.img" 和 Linux-QT 系统相同 , 它们在光盘 “image” --> “Linux” 文件夹中。

## 9.6 最小 linux 系统的存储空间修改

这里我们以修改成 1G 存储空间为例来讲解修改方法 , 如果需要改成其他大小的存储空间 , 参照此方法修改即可。

首先连接好 iTOP-4412 开发板的调试串口到 pc 上 , 在 pc 的 windows 系统下打开串口调试工具。开发板上电 , 在串口调试工具里按任意 pc 键盘的任意按键使开发板进入 uboot 命令行模式 , 如下图所示 :



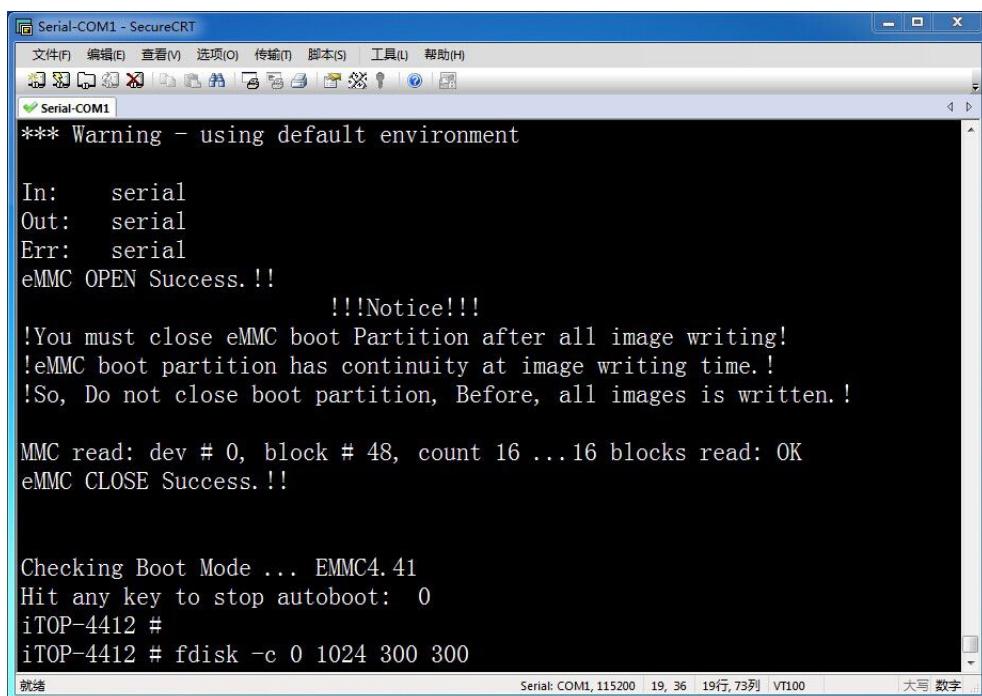
```
Serial-COM1 - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(I) 帮助(H)
Serial-COM1
*** Warning - using default environment

In:    serial
Out:   serial
Err:   serial
eMMC OPEN Success. !!
      !!!Notice!!!
!You must close eMMC boot Partition after all image writing!
!eMMC boot partition has continuity at image writing time. !
!So, Do not close boot partition, Before, all images is written. !

MMC read: dev # 0, block # 48, count 16 ... 16 blocks read: OK
eMMC CLOSE Success. !!

Checking Boot Mode ... EMMC4.41
Hit any key to stop autoboot: 0
iTOP-4412 #
iTOP-4412 #
```

然后在 uboot 输入分区命令：“fdisk -c 0 1024 300 300” , 如下图所示 :



```
Serial-COM1 - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(I) 帮助(H)
Serial-COM1
*** Warning - using default environment

In:    serial
Out:   serial
Err:   serial
eMMC OPEN Success. !!
      !!!Notice!!!
!You must close eMMC boot Partition after all image writing!
!eMMC boot partition has continuity at image writing time. !
!So, Do not close boot partition, Before, all images is written. !

MMC read: dev # 0, block # 48, count 16 ... 16 blocks read: OK
eMMC CLOSE Success. !!

Checking Boot Mode ... EMMC4.41
Hit any key to stop autoboot: 0
iTOP-4412 #
iTOP-4412 # fdisk -c 0 1024 300 300
```

上面图片里的命令是把 emmc 分区 , 其中的 1024 是 linux 的存储空间 , 单位是 MB , 也就是 1G 。如果想分配更大的空间修改这个值即可。

执行完上面的命令 , 如下图所示 :

```

Serial-COM1 - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(I) 脚本(S) 工具(L) 帮助(H)
Serial-Serial-COM1
!eMMC boot partition has continuity at image writing time. !
!So, Do not close boot partition, Before, all images is written. !

MMC read: dev # 0, block # 48, count 16 ... 16 blocks read: OK
eMMC CLOSE Success. !!

Checking Boot Mode ... EMMC4.41
Hit any key to stop autoboot: 0
iTOP-4412 #
iTOP-4412 # fdisk -c 0 1024 300 300
(fdisk is completed)

partition #      size(MB)    block start #    block count    partition_Id
    1            2064        3378474        4228686        0x0C
    2            1026         37290        2103156        0x83
    3             302         2140446        619014        0x83
    4             302         2759460        619014        0x83
iTOP-4412 #

```

然后在 uboot 命令行分别输入下面的命令，格式化分区：

`fatformat mmc 0:1`

`ext3format mmc 0:2`

`ext3format mmc 0:3`

`ext3format mmc 0:4`

至此 EMMC 的分区已经只做好了，下面我们开始制作 linux 文件系统，拷贝光盘

“linux/root\_xxxxxxxxx.tar.gz”（xxxxxxxx 是版本日期，）到 Ubuntu 虚拟机上，例如我这里拷贝到了 “/home/topeet/linux” 目录，如下图所示：

```

root@ubuntu:/home/topeet/linux#
root@ubuntu:/home/topeet/linux#
root@ubuntu:/home/topeet/linux# ls
root_20140912. tar. gz
root@ubuntu:/home/topeet/linux#

```

然后使用 “tar -xvf root\_20140912.tar.gz” 命令解压 linux 文件系统，如下图所示：

```

root@ubuntu:/home/topeet/linux#
root@ubuntu:/home/topeet/linux# ls
root_20140912. tar. gz
root@ubuntu:/home/topeet/linux# tar -xvf root_20140912. tar. gz

```

解压完成后，输入“ls”命令，可以看到生成了“root”文件夹，如下图所示：

```
root@ubuntu:/home/topeet/linux#
root@ubuntu:/home/topeet/linux# ls
root  root_20140912.tar.gz
root@ubuntu:/home/topeet/linux#
root@ubuntu:/home/topeet/linux#
```

接下来输入“make\_ext4fs -s -l 996147200 -a root -L linux system.img root”命令生成“system.img”，如下图所示：

```
root@ubuntu:/home/topeet/linux#
root@ubuntu:/home/topeet/linux# make_ext4fs -s -l 996147200 -a root -L linux system.img root
Creating filesystem with parameters:
  Size: 996147200
  Block size: 4096
  Blocks per group: 32768
  Inodes per group: 7600
  Inode size: 256
  Journal blocks: 3800
  Label: linux
  Blocks: 243200
  Block groups: 8
  Reserved block group size: 63
Created filesystem with 10561/60800 inodes and 53427/243200 blocks
root@ubuntu:/home/topeet/linux#
```

注意：使用“make\_ext4fs”命令前，确认已经安装好编译linux文件系统需要的软件包了，安装方法可以参照使手册的“6.3.5 生成 system.img”小节。

下面来看一下命令“make\_ext4fs -s -l 996147200 -a root -L linux system.img root”，这个命令里面的“996147200”就是指定了linux存储空间的大小了，即：

996x1024x1024=996MB (在前面的分区里我们分配的是1G的空间，这里我们需要预留几兆的空间，所以设置为996MB)

然后把生成的“system.img”烧写到iTOP-4412开发板，开发板启动进入到linux系统，输入“df”命令，可以看到linux存储空间变成996MB了，如下图所示：

```
[root@iTOP-4412]#
[root@iTOP-4412]# df
Filesystem      1K-blocks      Used Available Use% Mounted on
/dev/root        957496   198404    759092  21% /
tmpfs            307104         0    307104  0% /dev/shm
[root@iTOP-4412]#
```

通过上面的讲解我们已经清楚了怎么扩展存储空间，例如把存储空间改成 2G，那我们只需要修改下两个地方：

1 ) fdisk -c 0 **2048** 300 300

2) make\_ext4fs -s -l **2092957696** -a root -L linux system.img root

其中的 2092957696 是  $1996 \times 1024 \times 1024 = 1996\text{MB}$ 。

## 9.7 以模块的方式编译内核驱动

本节讲解如何在 linux 下实现以模块的方式加载内核驱动。我们以内核里面蜂鸣器的驱动为例来讲解。

1 ) 首先打开内核的源码，如下图所示：

```
root@ubuntu:/home/broswer/iTop4412_Kernel_3.0# 
root@ubuntu:/home/broswer/iTop4412_Kernel_3.0# ls
arch          config_for_ubuntu      drivers   Kbuild      Makefile      README      tools
binary        config_for_ubuntu_hdmi  firmware Kconfig     mm          REPORTING-BUGS  usr
block         COPYING             fs       kernel     modem.patch  samples    virt
config_for_android  CREDITS        include  kernel_readme.txt Module.symvers scripts
config_for_android_2M crypto        init     lib        net          security
config_for_linux Documentation  ipc     MAINTAINERS pull_log.bat  sound
```

2 ) 使用命令 “cd drivers/char/” ，进入到蜂鸣器驱动所在的目录，如下图所示：

```
root@ubuntu:/home/broswer/iTop4412_Kernel_3.0# cd drivers/char/
root@ubuntu:/home/broswer/iTop4412_Kernel_3.0/drivers/char#
root@ubuntu:/home/broswer/iTop4412_Kernel_3.0/drivers/char# ls
agg           dsp56k.c      hw_random      max485_ctl.c   nvram.c      raw.c      tb0219.c
apm-emulation.c dtlk.c       i8k.c        mbcs.c      nwbutton.c   rtc.c      tlc1k.c
applicom.c    efirte.c     ipmi        mbcs.h      nwbutton.h   s3c_mem.c  toshiba.c
applicom.h    exynos_mem.c itop4412_adc.c  mem.c      nwflash.c   s3c_mem.h  tpm
bfin-opt.c    generic_nvram.c itop4412_buzzer.c misc.c      pc8736x_gpio.c scc.h      ttyprintk.c
briq_panel.c  genrtc.c     itop4412_leds.c mmtimer.c   pcmcia      scx200_gpio.c uv_mmtimer.c
bsr.c         gps.c        itop4412_relay.c msm_smd_pkt.c ppdev.c    sns.c      viotape.c
dec_tty.c     gps.h        Kconfig      mspice.c    ps3flash.c  sns_event.c virtio_console.c
ds1302.c      hangcheck-timer.c lp.c        mwave      ramoops.c  sns.h      xilinx_hwicap
ds1620.c      hpet.c       Makefile    nsc_gpio.c random.c  sonypi.c
```

3 ) 然后使用命令 “vi Kconfig” 打开当前目录下的内核配置文件，如下图所示：

```
root@ubuntu:/home/broswer/iTop4412_Kernel_3.0/drivers/char# vi Kconfig

#
# Character device configuration
#
menu "Character devices"
source "drivers/tty/Kconfig"
config DEVMEM
    bool "Memory device driver"
    default y
    help
```

4 ) 然后找到 “config BUZZER\_CTL” 所在的位置，如下图所示：

```
config BUZZER_CTL
    bool "Enable BUZZER config"
    default n
    help
        Enable BUZZER config
```

5 ) 然后把 “bool "Enable BUZZER config" ”一行改成 “tristate "Enable BUZZER config" ”，如下图所示：

```
config BUZZER_CTL
    tristate "Enable BUZZER config"
    default n
    help
        Enable BUZZER config
```

6 ) 然后保存并退出，如下图所示：

```
lcomfig 090L, 23409C written
root@ubuntu:/home/broswer/iTop4412_Kernel_3.0/drivers/char#
root@ubuntu:/home/broswer/iTop4412_Kernel_3.0/drivers/char#
```

7 ) 然后回到内核源码的根目录下，如下图所示：

```
root@ubuntu:/home/broswer/iTop4412_Kernel_3.0#
root@ubuntu:/home/broswer/iTop4412_Kernel_3.0#
root@ubuntu:/home/broswer/iTop4412_Kernel_3.0#
```

8 ) 然后输入命令“ make menuconfig ”配置内核，如下图所示：

```
root@ubuntu:/home/broswer/iTop4412_Kernel_3.0# make menuconfig
scripts/kconfig/mconf Kconfig

.config - Linux/arm 3.0.15 Kernel Configuration

-- Linux/arm 3.0.15 Kernel Configuration --
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing
<Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for
Search. Legend: [*] built-in [ ] excluded <M> module < > module capable

[*] Patch physical to virtual translations at runtime (EXPERIMENTAL)
    General setup --->
[*] Enable loadable module support --->
[*] Enable the block layer --->
System Type --->
[ ] FIQ Mode Serial Debugger
    Bus support --->
    Kernel Features --->
    Boot options --->
    CPU Power Management --->
    Floating point emulation --->
    Userspace binary formats --->
    Power management options --->

v(+)
```

9 ) 选择“ Device Drivers ”->“Character devices”->“Enable BUZZER config” , 如下图所示 :

```
-- Character devices --
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing
<Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for
Search. Legend: [*] built-in [ ] excluded <M> module < > module capable

^(-)
< > Siemens R3964 line discipline
< > RAW driver (/dev/raw/rawN)
< > TPM Hardware Support --->
< > DCC tty driver
< > Log panic/oops to a RAM buffer
[*] Support for /dev/s3c-mem
[*] Support for /dev/exynos-mem
[*] Enable GPS PM
[*] Enable MAX485 pin config
[*] Enable LEDS config
<*> Enable BUZZER config
[*] ADC driver for iTOP4412
[*] Enable RELAY config

<Select> < Exit > < Help >
```

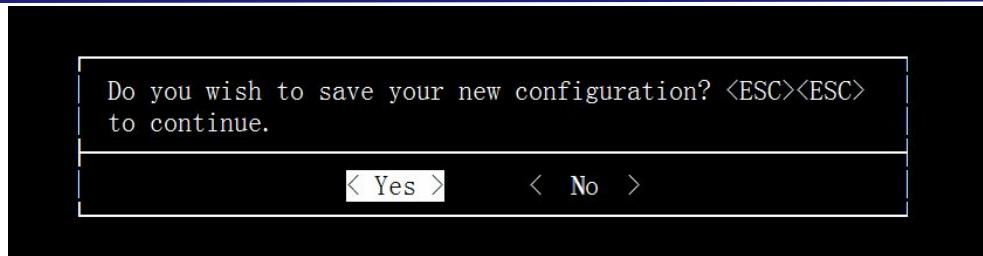
10 ) 然后把“ Enable BUZZER config ”左边的“ \* ”改成“ M ” , 如下图所示 :

```
-- Character devices --
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing
<Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for
Search. Legend: [*] built-in [ ] excluded <M> module < > module capable

^(-)
< > Siemens R3964 line discipline
< > RAW driver (/dev/raw/rawN)
< > TPM Hardware Support --->
< > DCC tty driver
< > Log panic/oops to a RAM buffer
[*] Support for /dev/s3c-mem
[*] Support for /dev/exynos-mem
[*] Enable GPS PM
[*] Enable MAX485 pin config
[*] Enable LEDS config
<M> Enable BUZZER config
[*] ADC driver for iTOP4412
[*] Enable RELAY config

<Select> < Exit > < Help >
```

11 ) 然后保存并退出配置界面 , 如下图 :



12 ) 然后使用命令“ vi arch/arm/mach-exynos/mach-itop4412.c ”打开 mach-itop4412.c , 如下图所示 :

```
root@ubuntu:/home/broswer/iTop4412_Kernel_3.0# vi arch/arm/mach-exynos/mach-itop4412.c

/* linux/arch/arm/mach-exynos/mach-itop4412.c
 *
 * Copyright (c) 2011 Topeet Electronics Co., Ltd.
 * http://www.topeetboard.com
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2 as
 * published by the Free Software Foundation.
 */

#include <linux/platform_device.h>
#include <linux/serial_core.h>
#include <linux/spi/spi.h>
#include <linux/spi/spi_gpio.h>
#include <linux/clk.h>
#include <linux/lcd.h>
```

13 ) 然后找到“ struct platform\_device s3c\_device\_buzzer\_ctl ”一行 , 如下图所示 :

```
#ifdef CONFIG_BUZZER_CTL
struct platform_device s3c_device_buzzer_ctl = {
    .name    = "buzzer_ctl",
    .id      = -1,
};

#endif
```

14 ) 把这一行前面的“ #ifdef CONFIG\_BUZZER\_CTL ”改成“ #if defined(CONFIG\_BUZZER\_CTL) || defined(CONFIG\_BUZZER\_CTL\_MODULE) ” , 如下图所示 :

```
#if defined(CONFIG_BUZZER_CTL) || defined(CONFIG_BUZZER_CTL_MODULE)
struct platform_device s3c_device_buzzer_ctl = {
    .name    = "buzzer_ctl",
    .id      = -1,
};

#endif
```

15 ) 然后找到“ &s3c\_device\_buzzer\_ctl ”一行 , 如下图所示 :

```
#ifdef CONFIG_BUZZER_CTL  
    &s3c_device_buzzer_ctl,  
#endif
```

16 ) 把这一行前面的“ #ifdef CONFIG\_BUZZER\_CTL ”改成“ #if  
defined(CONFIG\_BUZZER\_CTL) || defined(CONFIG\_BUZZER\_CTL\_MODULE) ” , 如下图  
所示 :

```
#if defined(CONFIG_BUZZER_CTL) || defined(CONFIG_BUZZER_CTL_MODULE)  
    &s3c_device_buzzer_ctl,  
#endif
```

17 ) 然后保存并退出 , 返回到 linux 内核源码的根目录下 , 如下图所示 :

```
root@ubuntu:/home/broswer/iTop4412_Kernel_3.0#  
root@ubuntu:/home/broswer/iTop4412_Kernel_3.0#  
root@ubuntu:/home/broswer/iTop4412_Kernel_3.0# █
```

18 ) 然后输入命令“ make ” , 开始编译内核 , 如下图所示 :

```
root@ubuntu:/home/broswer/iTop4412_Kernel_3.0#  
root@ubuntu:/home/broswer/iTop4412_Kernel_3.0#  
root@ubuntu:/home/broswer/iTop4412_Kernel_3.0# make █
```

19 ) 编译完成后会在内核的“ arch/arm/boot ”目录下生成镜像文件“ zImage ” , 如  
下图所示 :

```
root@ubuntu:/home/broswer/iTop4412_Kernel_3.0/arch/arm/boot#  
root@ubuntu:/home/broswer/iTop4412_Kernel_3.0/arch/arm/boot# ls  
bootp compressed Image install.sh Makefile zImage  
root@ubuntu:/home/broswer/iTop4412_Kernel_3.0/arch/arm/boot#
```

20 ) 在内核的“ drivers/char ”目录下生成了蜂鸣器的驱动模块“ itop4412\_buzzer.ko  
” , 如下图所示 :

```
root@ubuntu:/home/broswer/iTop4412_Kernel_3.0/drivers/char# ls
agp          exynos_mem.o      itop4412_buzzer.mod.c    mem.o      pcmcia      sns.h
apm-emulation.c generic_nvram.c  itop4412_buzzer.mod.o   misc.c     ppdev.c     sonypi
applicom.c   genrtc.c        itop4412_buzzer.o       misc.o     ps3flash.c   tb0219
applicom.h   gps.c          itop4412_leds.c       mmtimer.c  ramoops.c   tlclk.
bfin-otp.c   gps.h          itop4412_leds.o       modules.builtin random.c  toshib
briq_panel.c gps.o          itop4412_relay.c     modules.order  random.o   tpm
bsr.c        hangcheck-timer.c itop4412_relay.o     msm_smd_pkt.c raw.c      ttypri
built-in.o   hpet.c         Kconfig           mspec.c    rtc.c      uv_mmt
dcc_tty.c   hw_random      lp.c              mwave      s3c_mem.c   viotap
ds1302.c    i8k.c          Makefile          nsc_gpio.c s3c_mem.h   virtio
ds1620.c    ipmi          max485_ctl.c     nvram.c    s3c_mem.o   xilinx
dsp56k.c    itop4412_adc.c  max485_ctl.o     nwbutton.c scc.h      -
dtlk.c      itop4412_adc.o  mbcs.c          nwbutton.h scx200_gpio.c
efirtc.c   itop4412_buzzer.c mbcs.h          nwflash.c  sns.c      -
exynos_mem.c itop4412_buzzer.ko mem.c          pc8736x_gpio.c sns_event.c
root@ubuntu:/home/broswer/iTop4412_Kernel_3.0/drivers/char#
```

21 ) 下一步我们烧写生成的 zImage 到开发板上 , 然后开发板启动进入到 android 系统。

22 ) 然后通过 adb 把蜂鸣器的驱动模块传到开发板的"/data"目录下 , 如下图所示 :

```
D:\bsp\4412_DUT\USB_fastboot_tool\platform-tools>
D:\bsp\4412_DUT\USB_fastboot_tool\platform-tools>adb.exe push itop4412_buzzer.ko
/data
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
903 KB/s <101732 bytes in 0.110s>

D:\bsp\4412_DUT\USB_fastboot_tool\platform-tools>
```

23 ) 然后再串口输入命令“ cd /data ” , 进入到开发板的"/data"目录 , 如下图所示 :

```
root@android:/ #
root@android:/ # cd /data
root@android:/data #
root@android:/data #
root@android:/data #
```

24 ) 然后输入“ ls ”命令 , 可以看到通过 adb 传过来的蜂鸣器驱动模块“ itop4412\_buzzer.ko ” , 如下图所示 :

```
root@android:/ # cd /data/
root@android:/data #
root@android:/data #
root@android:/data # ls
app
app-private
backup
bluetooth
dalvik-cache
data
dontpanic
drm
itop4412_buzzer.ko
local
lost+found
misc
property
resource-cache
system
tombstones
user
wifi
root@android:/data #
```

25 ) 接着在串口输入命令“ insmod itop4412\_buzzer.ko ”，加载蜂鸣器的驱动，如下图所示：

```
root@android:/data # insmod itop4412_buzzer.ko
[ 414.652986] buzzer_ctl      initialized
root@android:/data #
root@android:/data #
```

26 ) 通过上图可以看到蜂鸣器的驱动已经加载到内核里面了，接着输入命令“cd /dev” ，如下图所示：

```
root@android:/data # cd /dev
root@android:/dev # ll
crwxrwxrwx root      root    10,   61 2012-01-01 05:20 AGPS
crw-rw---- system    system   10,  243 2012-01-01 05:20 HPD
crwxrwxrwx root      root    10,   58 2012-01-01 05:20 adc
crw-rw-r-- system    radio    10,   52 2012-01-01 05:20 alarm
crw-rw--- adb       adb     10,   56 2012-01-01 05:20 android_adb
crw-rw-rw- root      root    10,   62 2012-01-01 05:20 ashmem
crw-rw-rw- root      root    10,   51 2012-01-01 05:20 binder
drwxr-xr-x root      root        2012-01-01 05:20 block
drwxr-xr-x root      root        2012-01-01 05:20 bus
crw----- root      root    10,   42 2012-01-01 05:26 buzzer_ctl
drwxr-xr-x root      root        2012-01-01 05:21 com.qihoo.permmgr.shellservice
crw----- root      root     5,   1 2012-01-01 05:27 console
```

27 ) 通过上面的图片，可以看到生成了 “buzzer\_ctl” 蜂鸣器的节点了，我们需要输入 “chmod 777 buzzer\_ctl” 命令来修改下 buzzer\_ctl 的权限，如下图所示：

```
root@android:/dev #
root@android:/dev # chmod 777 buzzer_ctl
root@android:/dev #
root@android:/dev #
```

28 ) 然后我们可以使用 android 自带的蜂鸣器小程序来控制蜂鸣器了，至此以模块的方式加载驱动就完成了。

## 单独编译驱动模块

上面我们的驱动是放在了内核源码的目录下来实现的编译成驱动模块，很多时候我们都是拿到一个驱动源码，不需要把他放到内核源码里面，而是直接把他编译成驱动模块，下面我们就来讲下实现方法，我们还是以蜂鸣器的驱动为例来讲解。

1 ) 因为开发板带的内核默认是把蜂鸣器直接编译到内核里面了，所以我们要重复前边的步骤 “1” 到步骤 “19” ，完成这些步骤以后，把生成的镜像 “zImage” 烧写到开发板。

2 ) 然后在虚拟机的目录下用命令 mkdir 建立文件夹 “module” ，如下图所示：

```
root@ubuntu:/home/broswer#
root@ubuntu:/home/broswer#
root@ubuntu:/home/broswer# mkdir module
```

3 ) 然后用命令 cd 进入到建立的 “module” 文件夹，如下图所示：

```
root@ubuntu:/home/broswer#
root@ubuntu:/home/broswer# cd module/
root@ubuntu:/home/broswer/module#
root@ubuntu:/home/broswer/module# █
```

4 ) 拷贝内核里面的蜂鸣器驱动到 module 文件夹 , 如下图所示 :

```
root@ubuntu:/home/broswer/module#
root@ubuntu:/home/broswer/module# cp /home/broswer/iTop4412_Kernel_3.0/drivers/char/itop4412_buzzer.c ./
root@ubuntu:/home/broswer/module# █
```

5 ) 然后在 module 文件夹建立文件 “Makefile” , 如下图所示 :

```
root@ubuntu:/home/broswer/module#
root@ubuntu:/home/broswer/module# vi Makefile
```

6 ) 然后在"Makefile"文件输入下面的信息 :

obj-m += itop4412\_buzzer.o

KDIR := /home/broswer/iTop4412\_Kernel\_3.0

PWD = \$(shell pwd)

all:

make -C \$(KDIR) M=\$(PWD) modules

clean:

rm -rf \*.o

其中的 “KDIR := /home/broswer/iTop4412\_Kernel\_3.0” 是指定我们内核源码所在的目录 , 我这里内核源码是放在了 “ /home/broswer/ ” 目录下面 , 需要根据自己的存放位置来修改这个目录。

7 ) 然后保存并退出 Makefile , 在终端输入“ make ”命令 , 编译蜂鸣器驱动 , 如下图所示 :

```
root@ubuntu:/home/broswer/module# make
make -C /home/broswer/iTop4412_Kernel_3.0 M=/home/broswer/module modules
make[1]: Entering directory `/home/broswer/iTop4412_Kernel_3.0'
  CC [M]  /home/broswer/module/itop4412_buzzer.o
  /home/broswer/module/itop4412_buzzer.c:90: warning: initialization from incompatible pointer type
  Building modules, stage 2.
    MODPOST 1 modules
      CC      /home/broswer/module/itop4412_buzzer.mod.o
      LD [M]  /home/broswer/module/itop4412_buzzer.ko
make[1]: Leaving directory `/home/broswer/iTop4412_Kernel_3.0'
root@ubuntu:/home/broswer/module#
```

8 ) 编译完成后 , 可以看下在“ module ”文件夹下面生成了“ itop4412\_buzzer.ko ” , 如下图所示 :

```
root@ubuntu:/home/broswer/module#
root@ubuntu:/home/broswer/module# ls
itop4412_buzzer.c  itop4412_buzzer.mod.c  itop4412_buzzer.o  modules.order
itop4412_buzzer.ko  itop4412_buzzer.mod.o  Makefile           Module.symvers
root@ubuntu:/home/broswer/module#
```

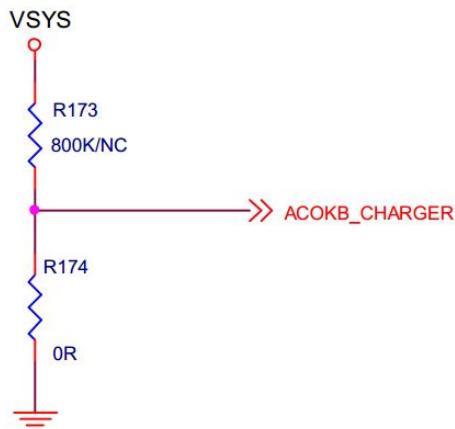
9 ) 然后可以参照前面的步骤“ 22 ”来测试我们编译的蜂鸣器驱动了。

## 9.8 一键实现开关机唤醒和休眠

本节介绍以开发板上的 reset 按键来实现一键开关机 , 休眠唤醒的功能。

### 9.8.1 设置启动方式 :

首先我们通过 reset 按键实现开机功能 , 在开发板的底板原理图上找到如下原理 :



如上图所示，通过电阻 R173 和 R174 可以设置开发板的启动方式：上电启动

## 2. 通过按 reset 按键来启动

默认是上电启动，所以我们焊了电阻 R174（把 ACOKB\_CHARGER 这个网络下拉），如果改成按键启动，需要把电阻 R174 去掉，电阻 R173 焊上（通过 800K 电阻上拉）。

（注意：因为我这里的截图是精英版的，所以这两个电阻的标号是 R173 和 R174，在全功能板上这两个电阻有可能是其他的标号，所以大家主要是搞明白了原理，根据自己的实际需要来选择就可以了，不要纠结于电阻的标号和我上面的截图不一样。）

### 9.8.2 设置一键休眠唤醒关机

我们可以通过 reset 按键实现休眠唤醒和关机功能。

首先用压缩包里的“samsung-keypad.c”（该文件可以再技术支持的群共享中找到）替换内核源码的“iT0p4412\_Kernel\_3.0/drivers/input/keyboard/samsung-keypad.c”。

然后修改内核“iT0p4412\_Kernel\_3.0/arch/arm/mach-exynos/mach-it0p4412.c”文件，在这个文件找到：

```
#else
    KEY(8, 0, KEY_1), KEY(8, 3, KEY_2), KEY(8, 5, KEY_3), KEY(8, 6, KEY_4),
    KEY(6, 0, KEY_5), KEY(6, 3, KEY_6), KEY(6, 5, KEY_7), KEY(6, 6, KEY_8),
    KEY(7, 0, KEY_9), KEY(7, 3, KEY_0), KEY(7, 5, KEY_A), KEY(7, 6, KEY_B),
    KEY(13, 0, KEY_C), KEY(13, 3, KEY_D), KEY(13, 5, KEY_E), KEY(13, 6, KEY_F)
#endif
```

然后添加 “KEY(0, 5, KEY\_POWER)”，修改成如下图：

```
KEY(0, 5, KEY_POWER),
    KEY(8, 0, KEY_1), KEY(8, 3, KEY_2), KEY(8, 5, KEY_3), KEY(8, 6, KEY_4),
    KEY(6, 0, KEY_5), KEY(6, 3, KEY_6), KEY(6, 5, KEY_7), KEY(6, 6, KEY_8),
    KEY(7, 0, KEY_9), KEY(7, 3, KEY_0), KEY(7, 5, KEY_A), KEY(7, 6, KEY_B),
    KEY(13, 0, KEY_C), KEY(13, 3, KEY_D), KEY(13, 5, KEY_E), KEY(13, 6, KEY_F)
```

修改完成后，保存并退出，然后输入“make”命令开始编译内核，编译完成，烧写生成的 zImage 到 iTOP-4412 开发板，就可以实现一键休眠唤醒和关机功能了。

# 十 Android 应用开发入门指南

## 10.1 搭建 Android 应用的开发环境

在不同的平台下，Android 应用的开发环境，需要的软件各不相同，但是安装方法和使用方法都是一样。这里给大家详细介绍在 Win7-64 位操作系统下搭建开发环境的方法，其它平台可以参考下面的搭建方法。

这里需要注意的是，这里介绍的是用户如何下载最新的编译环境。

在网盘中我们有提供了一套编译环境，用户可以从网盘中直接下载，可以直接使用。  
"JDK\_Win7-64"中有更新后的 Android4.0.3 的工具，将 ADT 解压后得到的文件覆盖就可以直接编译我们提供的例程。

文件名	大小	修改时间
ADT_Win7-64	-	2014-12-08 10:37
JDK_Win7-64	-	2014-12-08 10:36
编译安卓应用so文件所用NDK	-	2014-11-24 14:46

### 10.1.1 下载和安装 JDK

#### 10.1.1.1 下载 JDK

下载网址如下

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

打开网页，如下图所示，单击红色框的 JDK 下载链接。

The screenshot shows the Java SE Downloads page. At the top, there are tabs for Overview, Downloads (which is selected), Documentation, Community, Technologies, and Training. Below the tabs, there are two main download sections:

- Java Platform (JDK) 8u25**: Includes the Java logo and a "DOWNLOAD" button.
- NetBeans with JDK 8**: Includes the NetBeans logo and a "DOWNLOAD" button.

Below these sections, a box labeled "Java Platform, Standard Edition" contains information about Java SE 8u25:

- Java SE 8u25**: A release note stating it includes important security fixes, with a link to "Learn more".
- A list of links:
  - Installation Instructions
  - Release Notes
  - Oracle License
  - Java SE Products
  - Third Party Licenses
  - Certified System Configurations
  - Readme Files
    - JDK ReadMe
    - JRE ReadMe

On the right side of the "Java Platform, Standard Edition" box, there are three download buttons:

- JDK**: A red-bordered button with a "DOWNLOAD" button.
- Server JRE**: A "DOWNLOAD" button.
- JRE**: A "DOWNLOAD" button.

如下图所示，选择红色方框中的圆圈，接受下载协议。

## Java SE Development Kit 8 Downloads

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications, applets, and components using the Java programming language.

The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

See also:

- Java Developer Newsletter (tick the checkbox under Subscription Center > Oracle Technology News)
- Java Developer Day hands-on workshops (free) and other events
- Java Magazine

JDK MD5 Checksum

**Looking for JDK 8 on ARM?**

JDK 8 for ARM downloads have moved to the [JDK 8 for ARM download page](#).

### Java SE Development Kit 8u25

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Accept License Agreement  Decline License Agreement

Product / File Description	File Size	Download
Linux x86	135.24 MB	<a href="#">jdk-8u25-linux-i586.rpm</a>
Linux x86	154.88 MB	<a href="#">jdk-8u25-linux-i586.tar.gz</a>
Linux x64	135.6 MB	<a href="#">jdk-8u25-linux-x64.rpm</a>
Linux x64	153.42 MB	<a href="#">jdk-8u25-linux-x64.tar.gz</a>
Mac OS X x64	209.13 MB	<a href="#">jdk-8u25-macosx-x64.dmg</a>
Solaris SPARC 64-bit (SVR4 package)	137.01 MB	<a href="#">jdk-8u25-solaris-sparcv9.tar.Z</a>
Solaris SPARC 64-bit	97.14 MB	<a href="#">jdk-8u25-solaris-sparcv9.tar.gz</a>
Solaris x64 (SVR4 package)	137.11 MB	<a href="#">jdk-8u25-solaris-x64.tar.Z</a>

根据平台选择 jdk 的下载，这里使用的是 Win7-64 位，所以选择如下图所示的版本。目前使用的版本是 jdk1.8。

Java SE Development Kit 8u25		
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.		
Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.		
Product / File Description	File Size	Download
Linux x86	135.24 MB	<a href="#">jdk-8u25-linux-i586.rpm</a>
Linux x86	154.88 MB	<a href="#">jdk-8u25-linux-i586.tar.gz</a>
Linux x64	135.6 MB	<a href="#">jdk-8u25-linux-x64.rpm</a>
Linux x64	153.42 MB	<a href="#">jdk-8u25-linux-x64.tar.gz</a>
Mac OS X x64	209.13 MB	<a href="#">jdk-8u25-macosx-x64.dmg</a>
Solaris SPARC 64-bit (SVR4 package)	137.01 MB	<a href="#">jdk-8u25-solaris-sparcv9.tar.Z</a>
Solaris SPARC 64-bit	97.14 MB	<a href="#">jdk-8u25-solaris-sparcv9.tar.gz</a>
Solaris x64 (SVR4 package)	137.11 MB	<a href="#">jdk-8u25-solaris-x64.tar.Z</a>
Solaris x64	94.24 MB	<a href="#">jdk-8u25-solaris-x64.tar.gz</a>
Windows x86	157.26 MB	<a href="#">jdk-8u25-windows-i586.exe</a>
Windows x64	169.62 MB	<a href="#">jdk-8u25-windows-x64.exe</a>

### 10.1.1.2 AndroidJDK 和修改 JDK 环境变量

下载完成后，双击下载的可执行文件“jdk-8u25-windows-x64.exe”，根据向导完成安装即可，安装路径选择默认就可以，否则后面的环境变量需要用户根据情况修改。

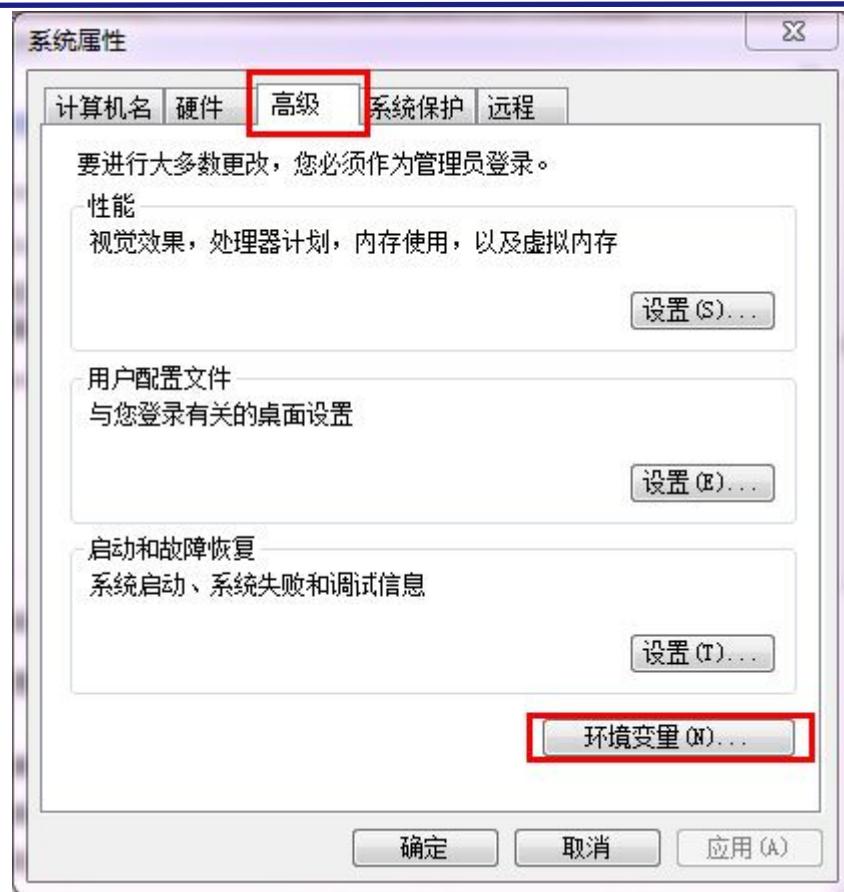


安装完成后，下面来修改一下环境变量。

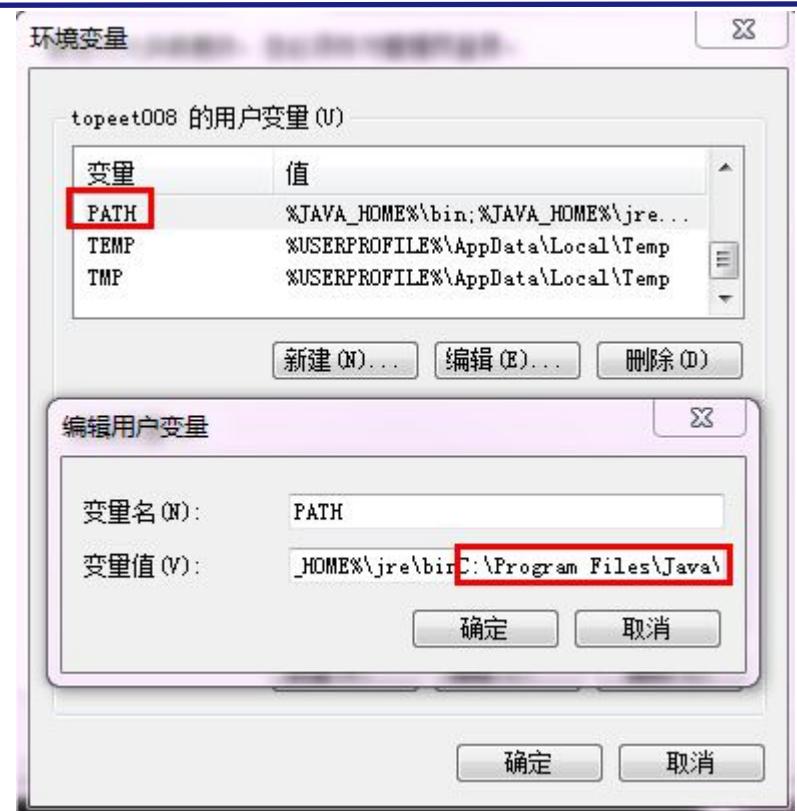
右键单击“我的电脑”，进入“属性”，如下图，单击“高级系统设置”。



如下图，打开环境变量设置窗口。

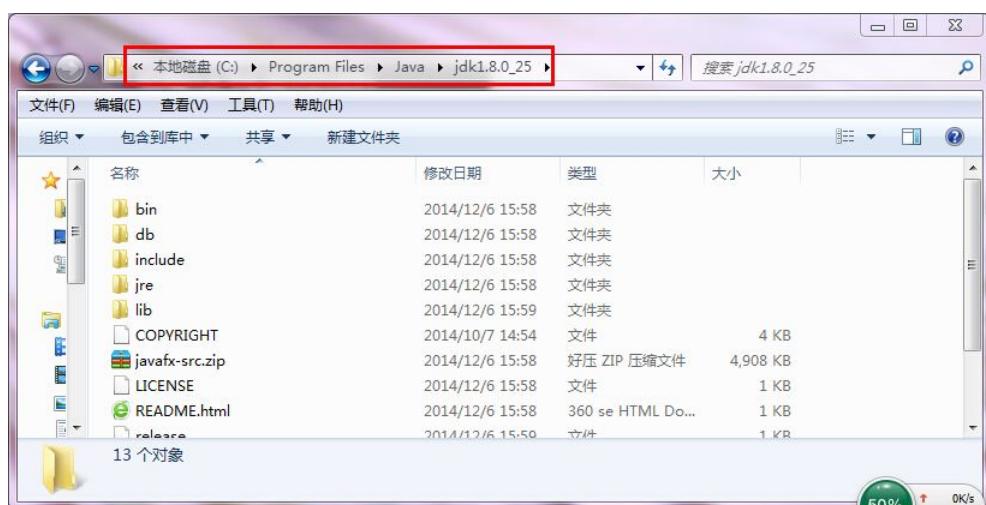


在后面添加环境变量“C:\Program Files\Java\jdk1.8.0\_25\;”，注意后面要带分号。



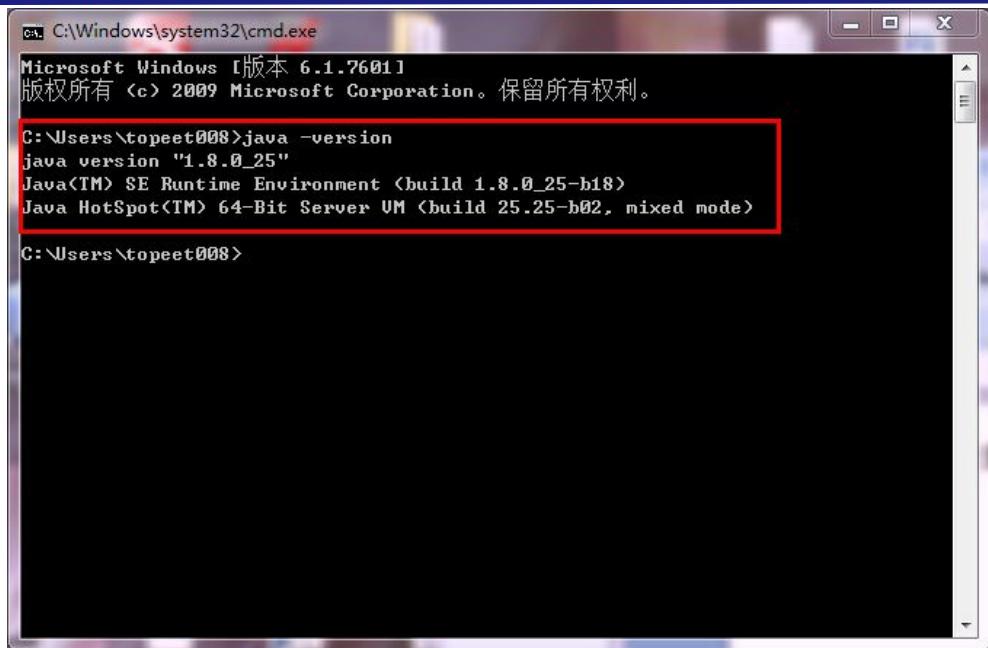
点击“确定”完成 JDK 环境变量的设置。

如下图，是 JDK 安装的默认路径，如果用户 JDK 安装的路径不一样，用户需要稍微改一下路径。



环境变量设置完成后，接下来测试一下。

进入 DOS 命令行，如下图，输入命令 “#java -version” ，出现 java 版本，则表明环境变量设置正确。



### 10.1.2 下载和安装 ADT 集成开发环境以及 SDK 管理器

下载地址为：

<http://developer.android.com/sdk/>

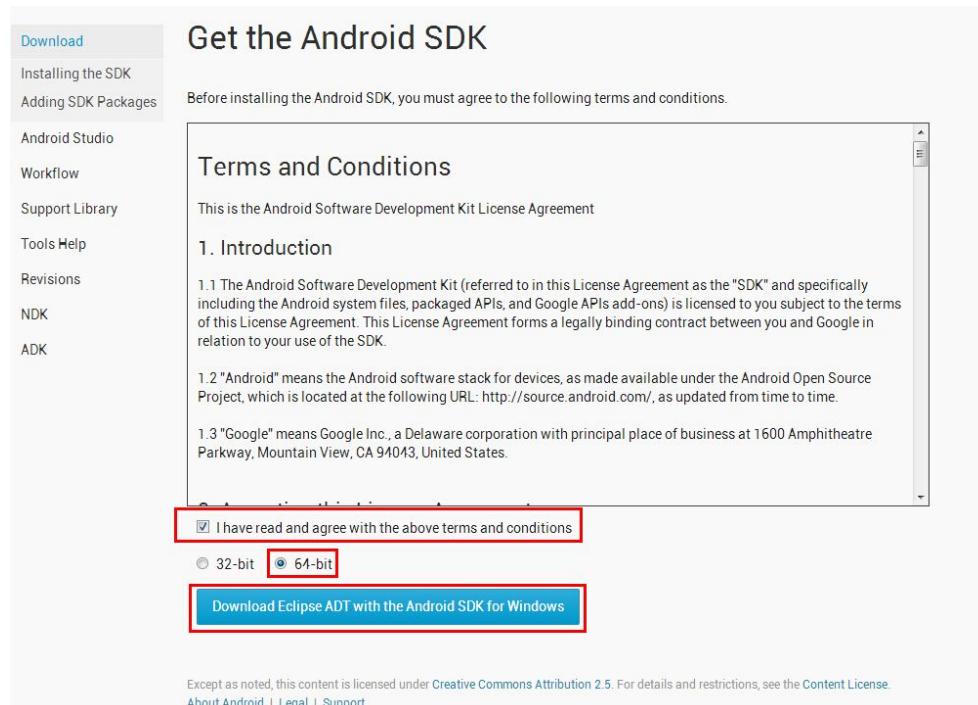
如下图所示，进入下载页面，

The page features two main sections:

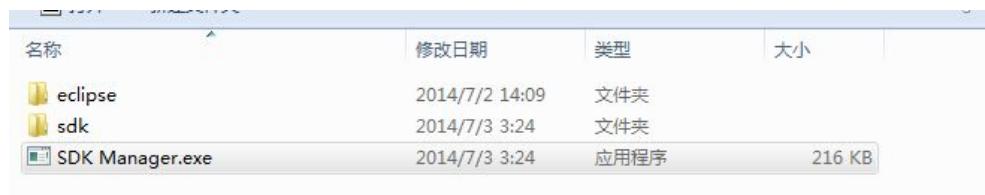
- Get the Android SDK**: Describes the Android SDK and provides a large blue button labeled "Download Eclipse ADT with the Android SDK for Windows".
- Get Android Studio Beta**: Describes Android Studio and provides a link to "Learn more about Android Studio".

Both sections include images of a smartphone and a computer monitor displaying the Android development environment.

接受下载协议，选择 64 位，开始下载。这里根据用户实际情况来选择是 32 位还是 64 位。

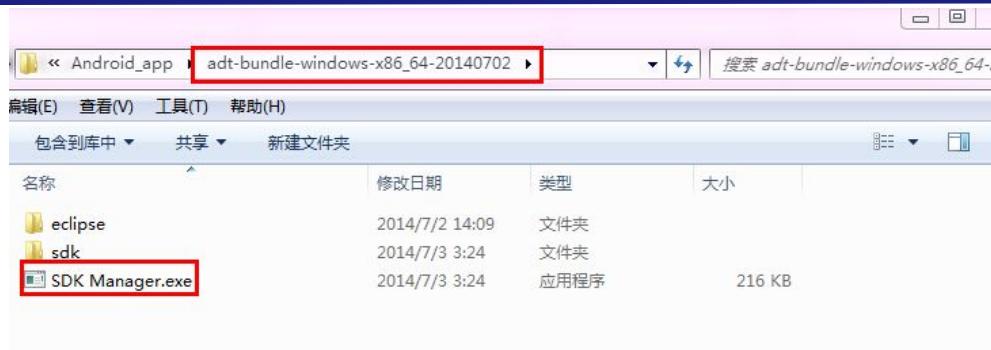


下载完成后，解压得到“adt-bundle-windows-x86\_64-20140702”文件夹（Google 更新后文件夹的后缀会改变，不过更新的工具都是向下兼容的），里面已经包含了 eclipse 和 SDK Manager.exe。

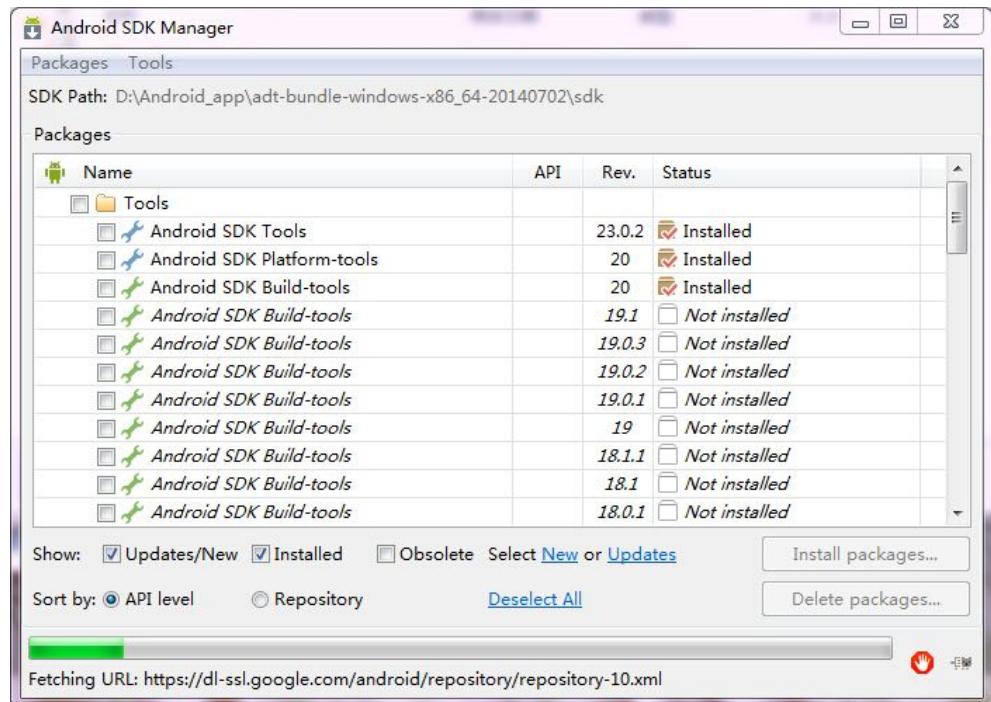


### 10.1.3 下载 SDK

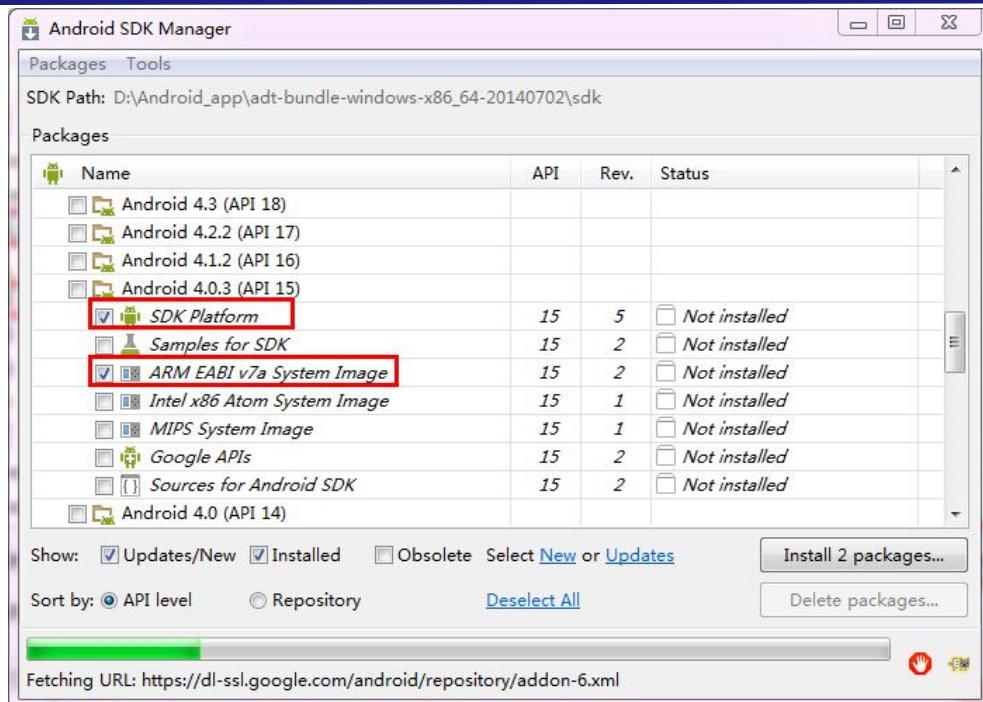
进入解压的文件夹，如下图，打开可执行程序“SDK Manager.exe”。



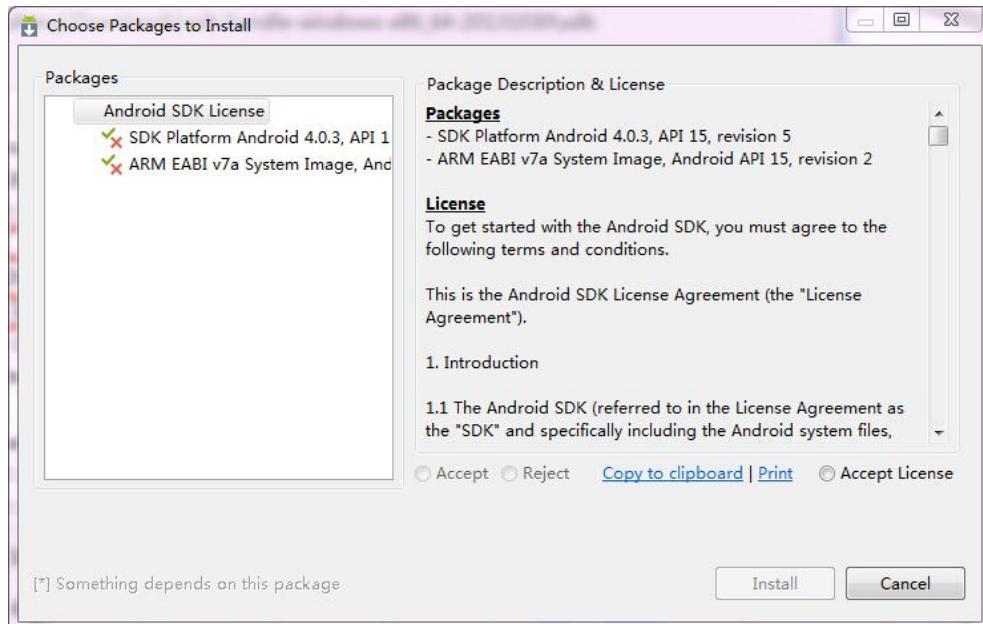
打开后，“SDK Manager.exe”程序如下图所示。



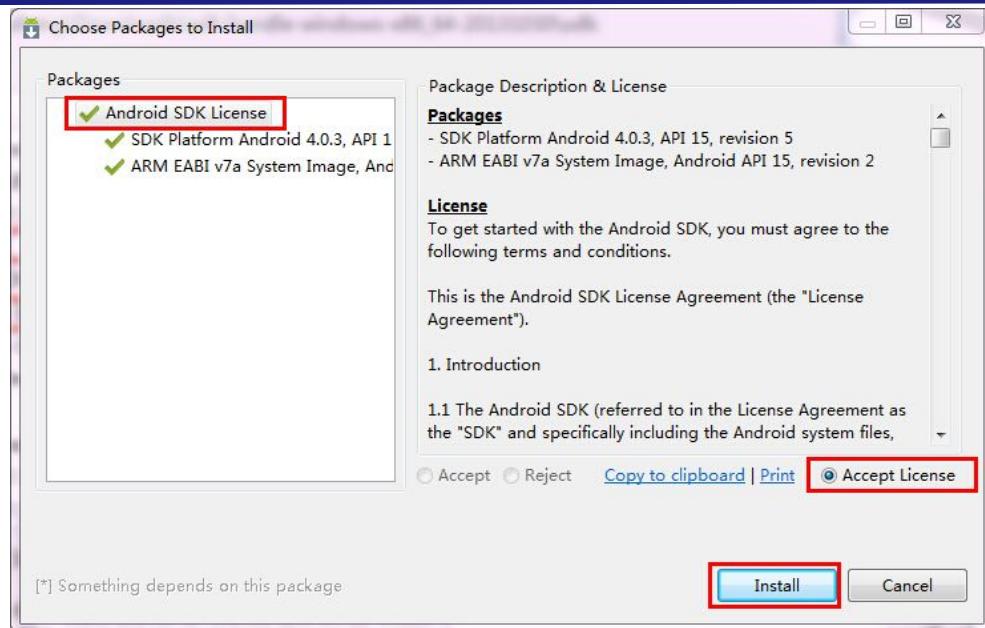
iTop-4412 的 Android 版本是 4.0.3，所以需要下载 Android4.0.3 的 SDK。如下图，选择 Android4.0.3 的工具。



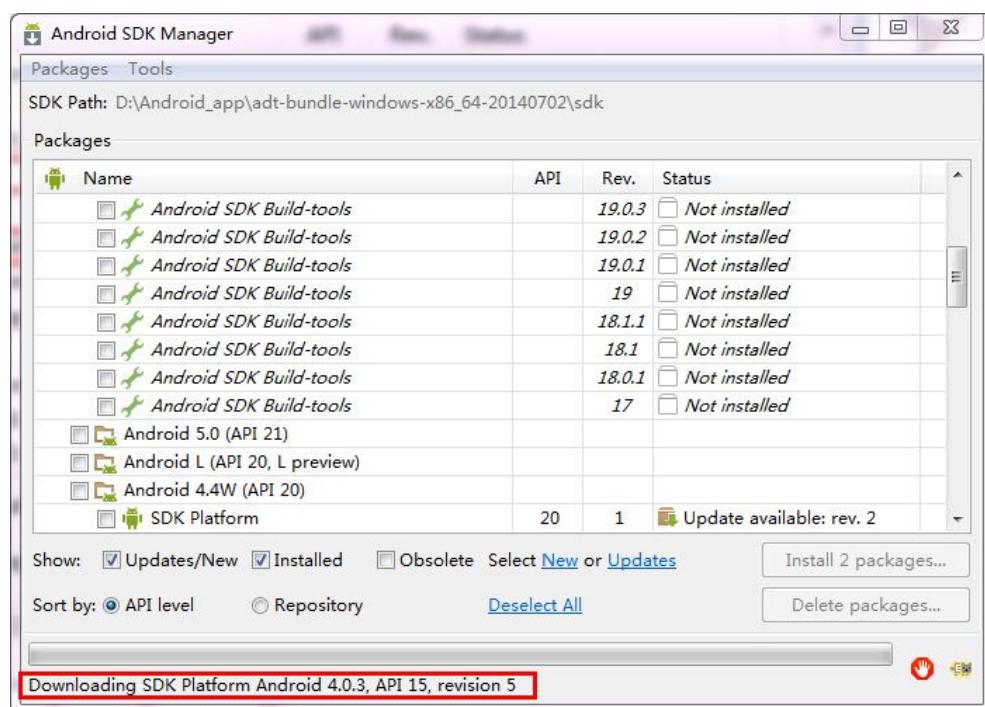
单击按钮 “Install 2 packages” , 弹出下图所示窗口。



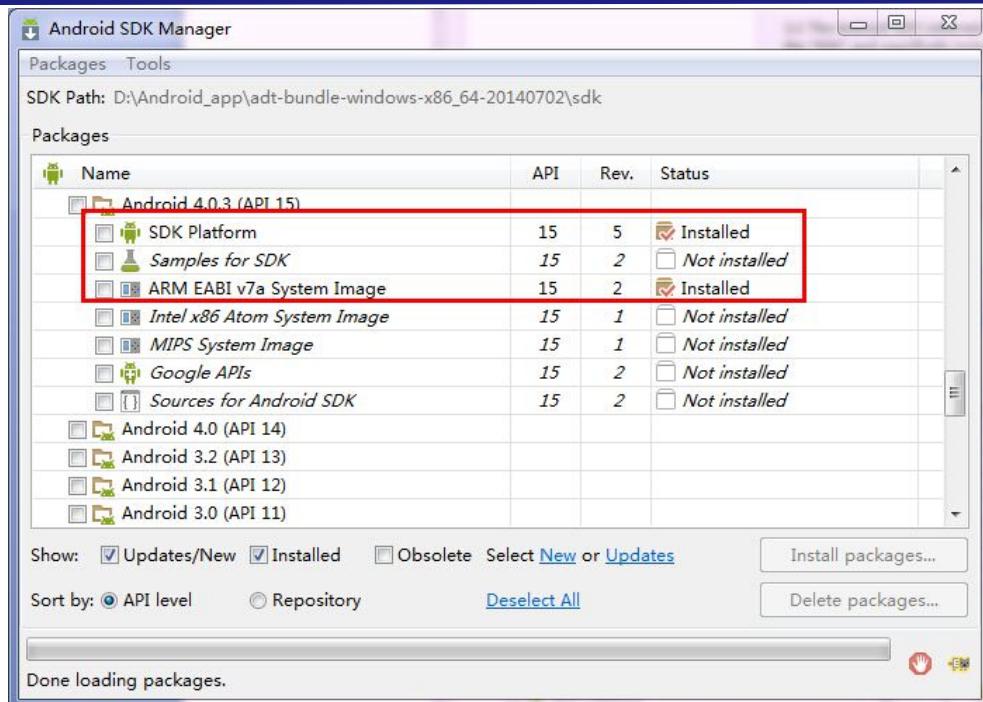
如下图 , 接受下载协议 , 单击按钮 “Install” , 开始下载。



如下图所示，开始下载。

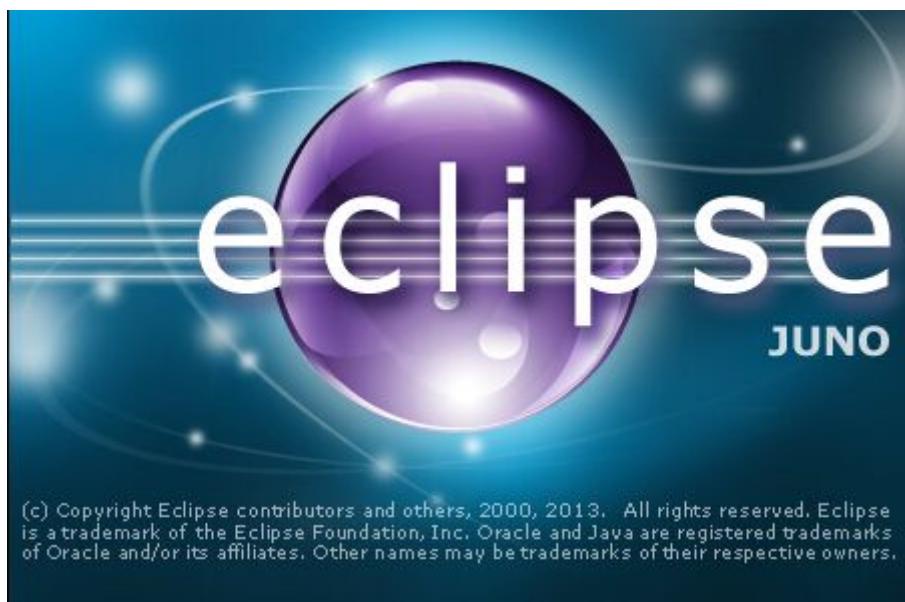


如下图所示，安装完成。

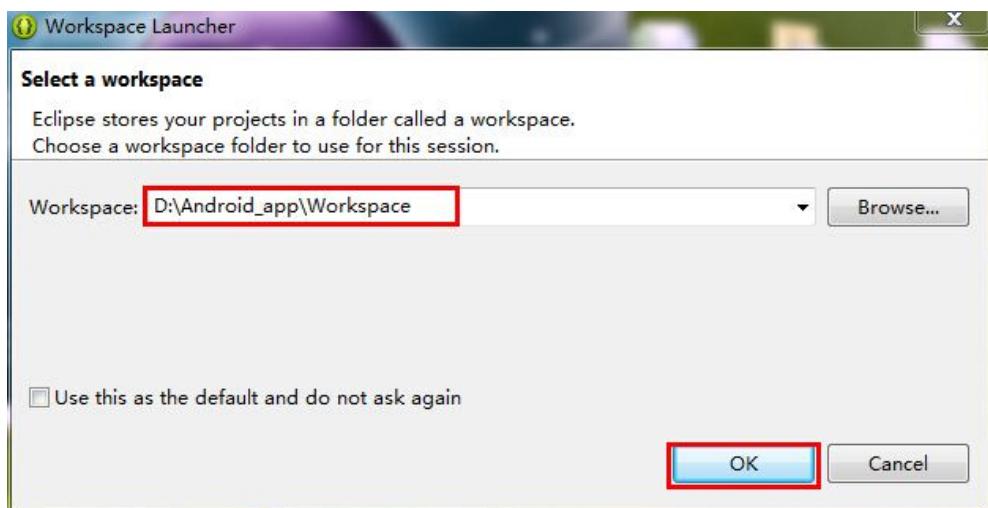


#### 10.1.4 ADT 集成开发环境

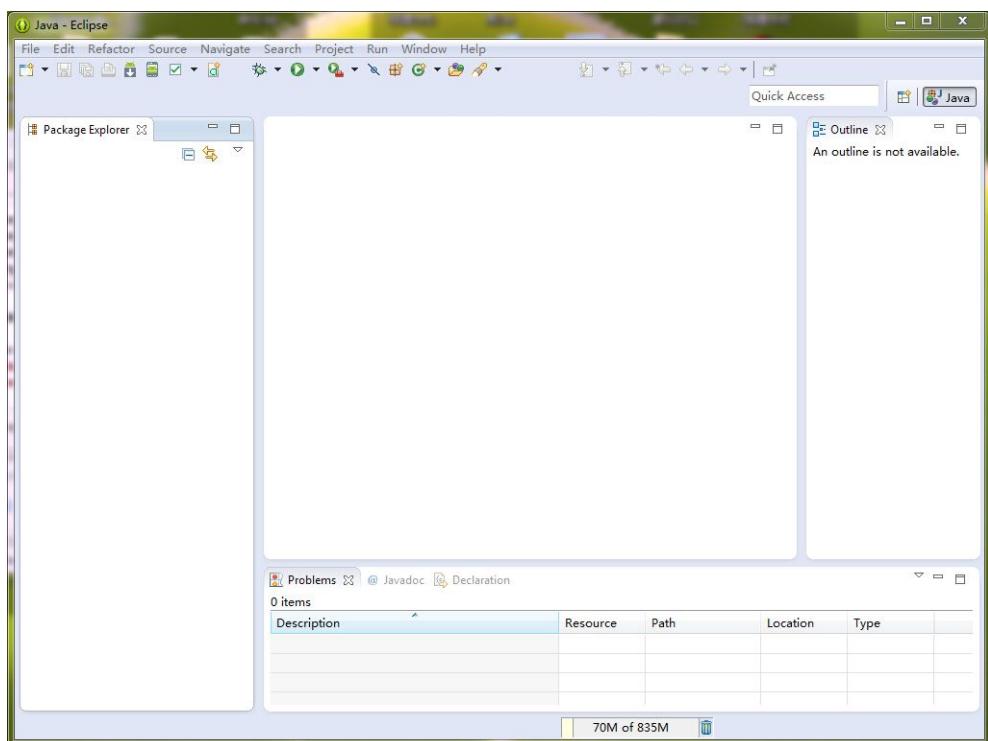
进入“adt-bundle-windows-x86\_64-20140702\ eclipse”打开“eclipse.exe”应用程序。



如下图所示，设置 Workspace 的路径。设置好之后点击 OK。如果选择默认也可以，用户根据实际情况设定。



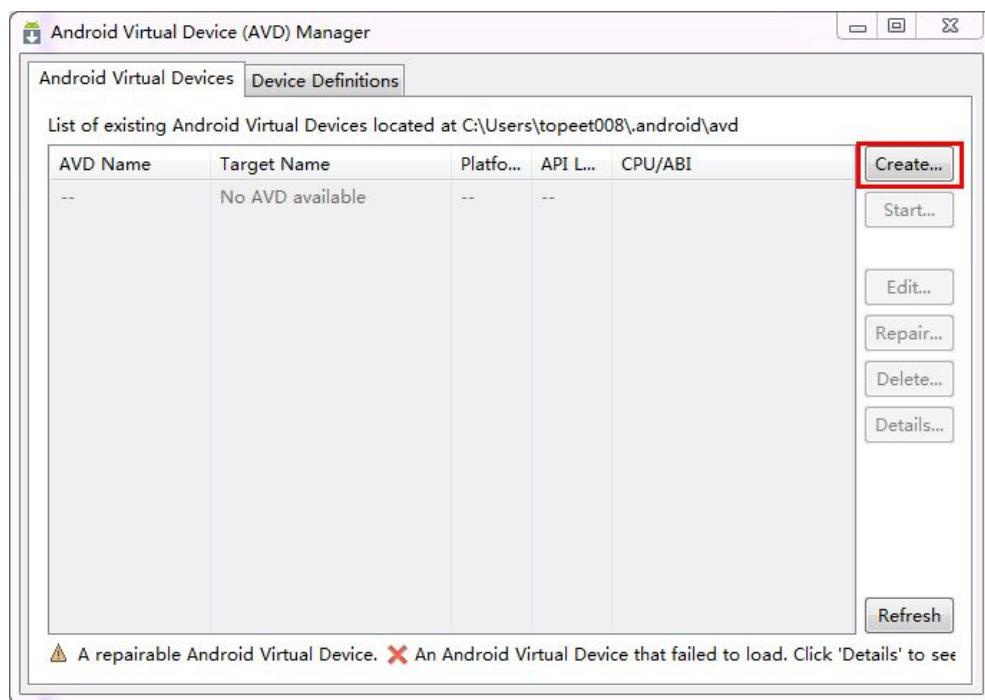
进入 Eclipse 的主界面。



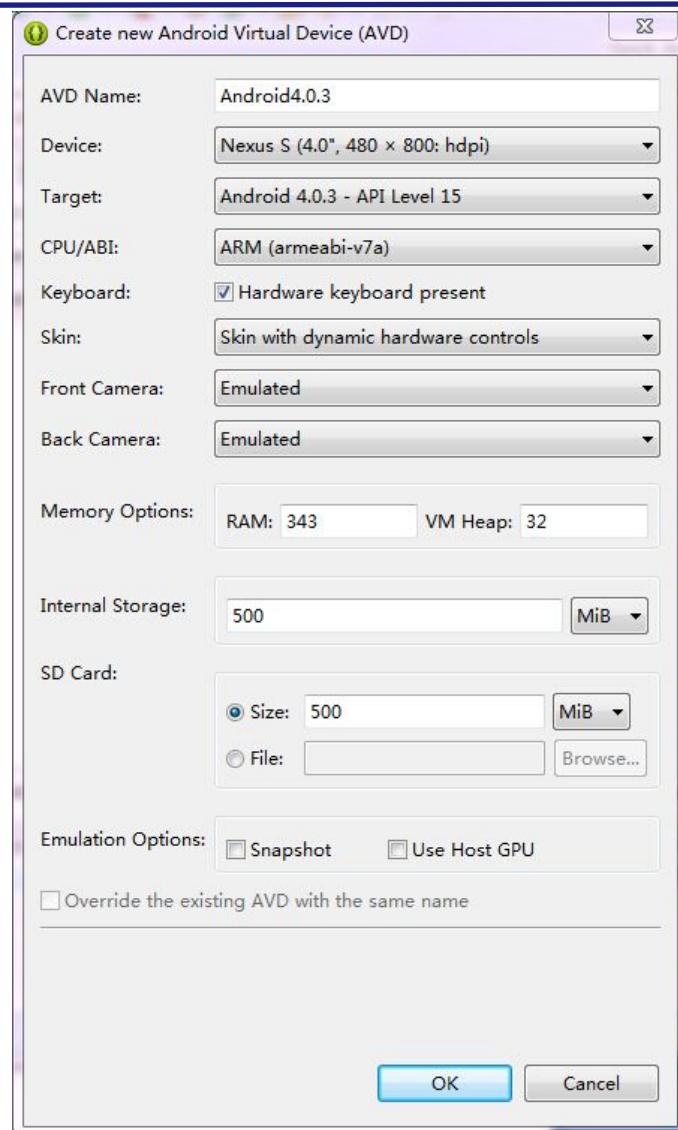
Eclipse 是一个开放源代码的、基于 Java 的可扩展开发平台。就其本身而言，它只是一个框架和一组服务，用于通过插件组件构建开发环境。但是 Android 应用程序的开发环境（ADT——Android Developer Tools）都是以 Eclipse 为骨架建立的，所以 Eclipse 在这里可以代指 ADT。

### 10.1.5 创建 Android 模拟器

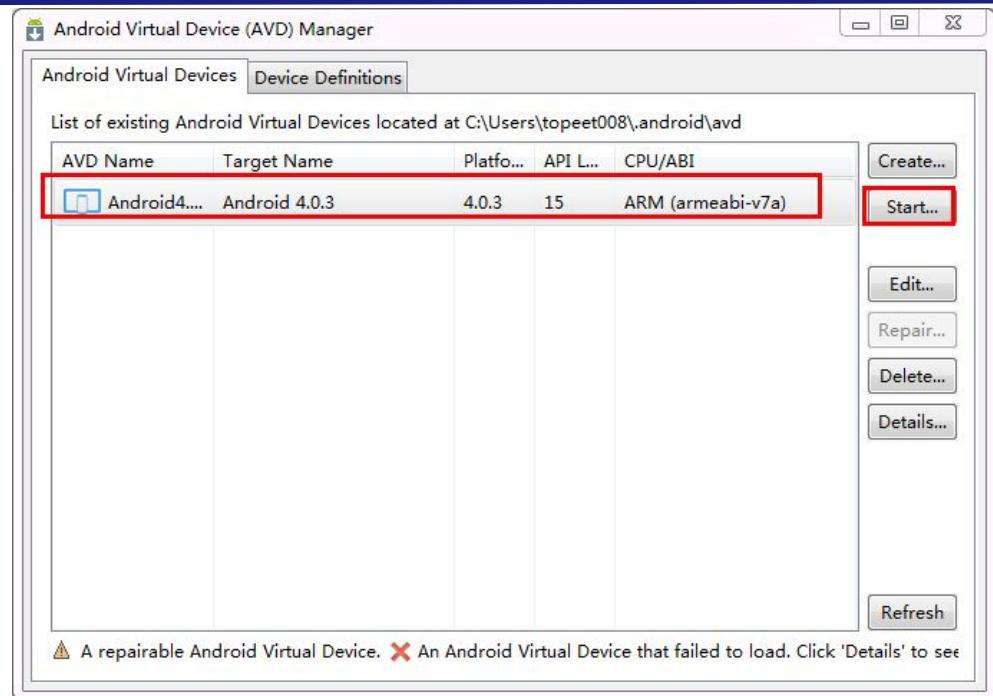
在 Eclipse 中，单击 “Windows” 菜单，选择 “Android Virtual Device Manager” 启动模拟器管理插件。然后如下图，单击 “Create.....” 。



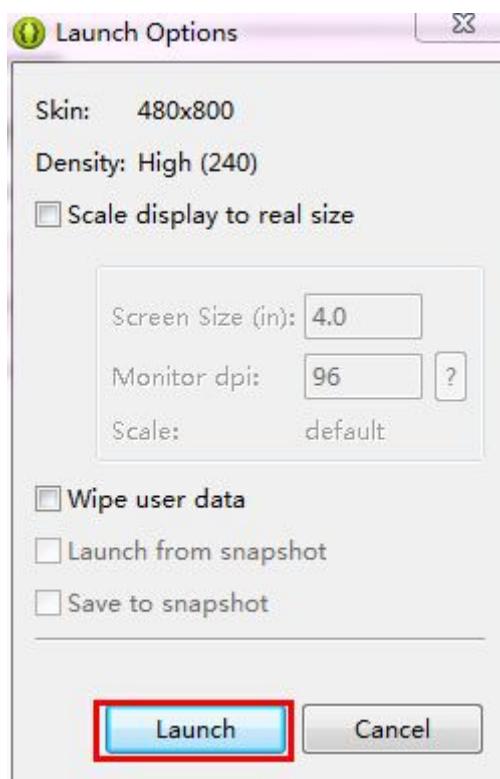
如下图所示，设置参数，单击 “OK” ，完成。



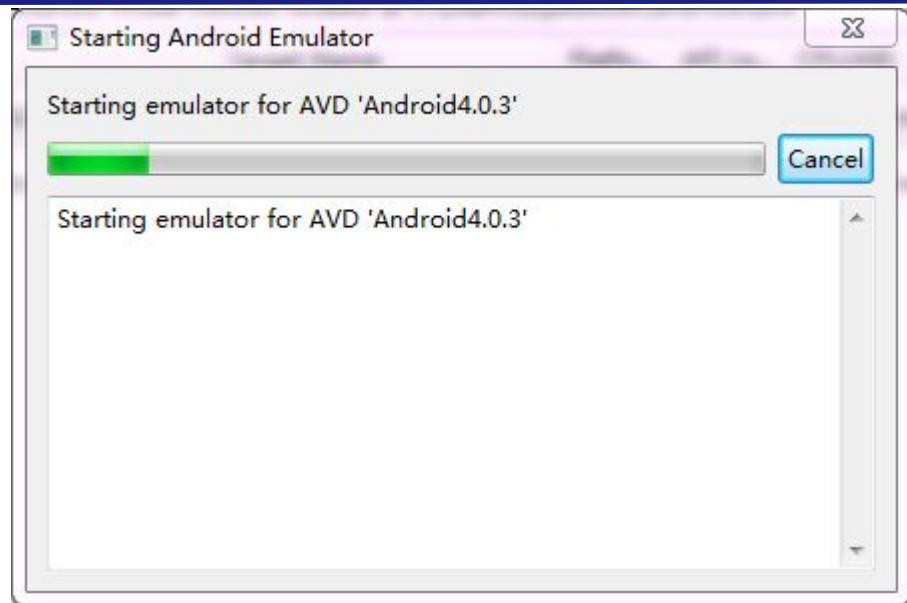
如下图，选择 Android4.0.3，单击“Start”启动。



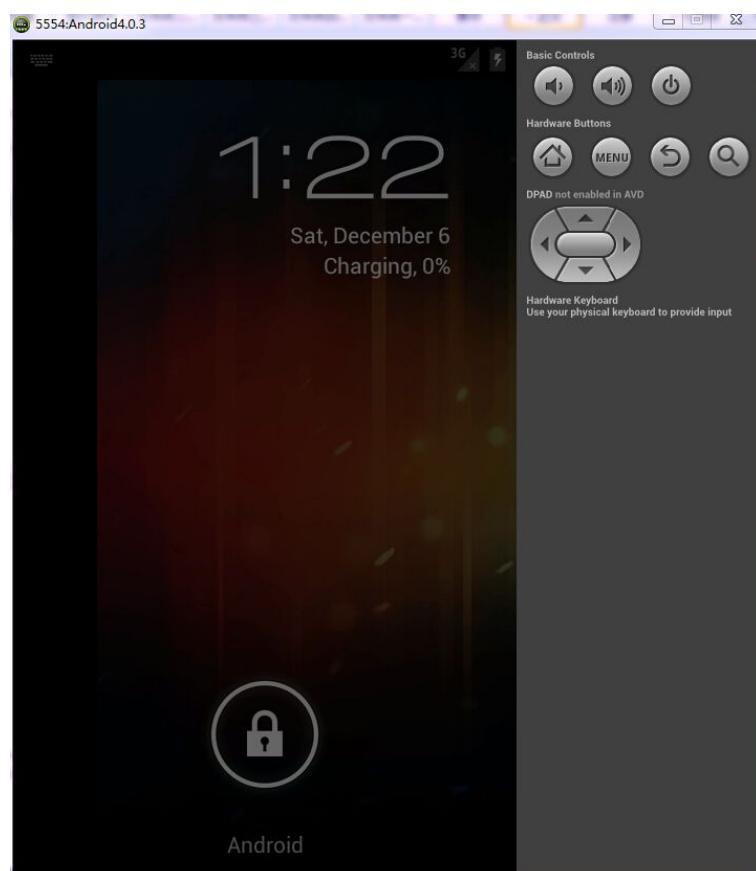
如下图，单击“Launch”，启动模拟器。



如下图，启动中，可能需要 1 到 5 分钟。

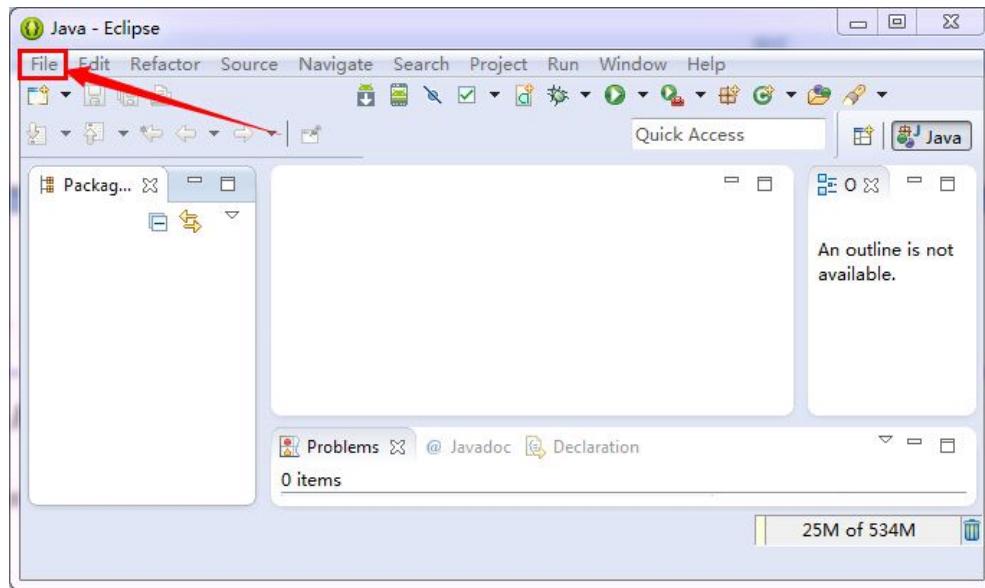


如下图，模拟器启动完成。

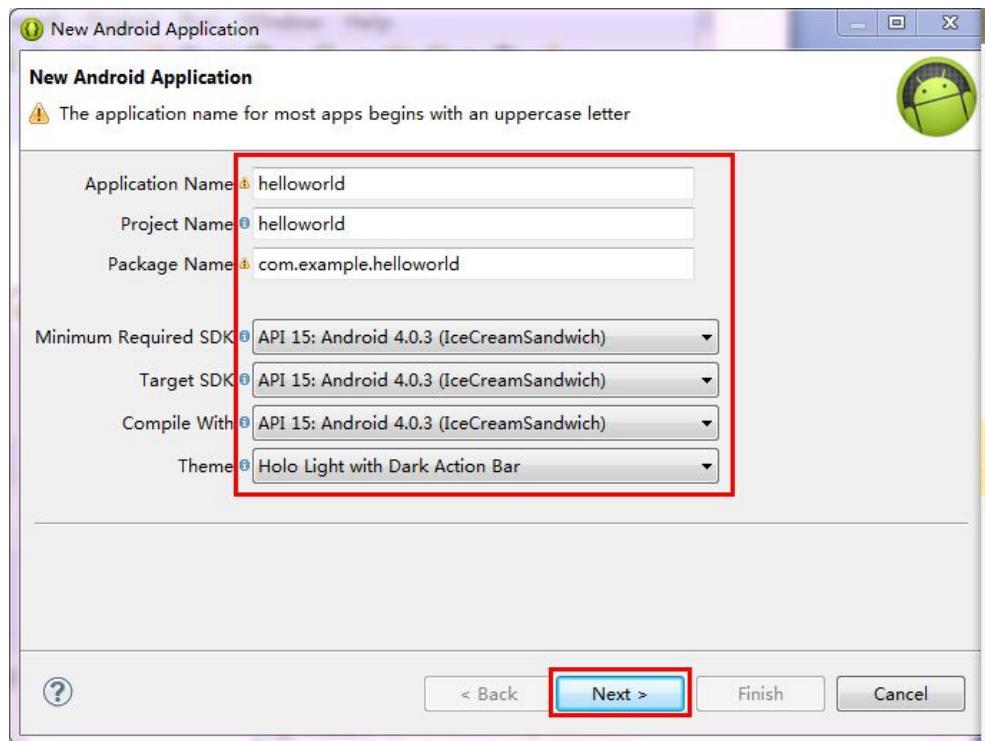


### 10.1.6 创建第一个 Android 应用程序 helloworld

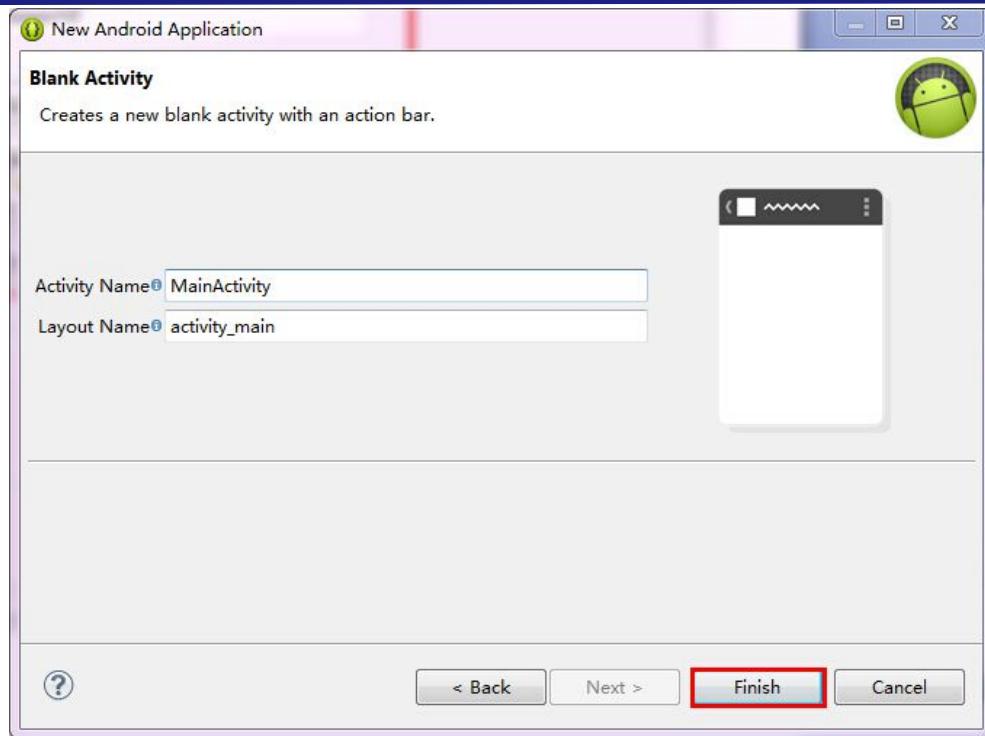
打开 Eclipse，选择菜单 “File->New->Android Application Project” ，



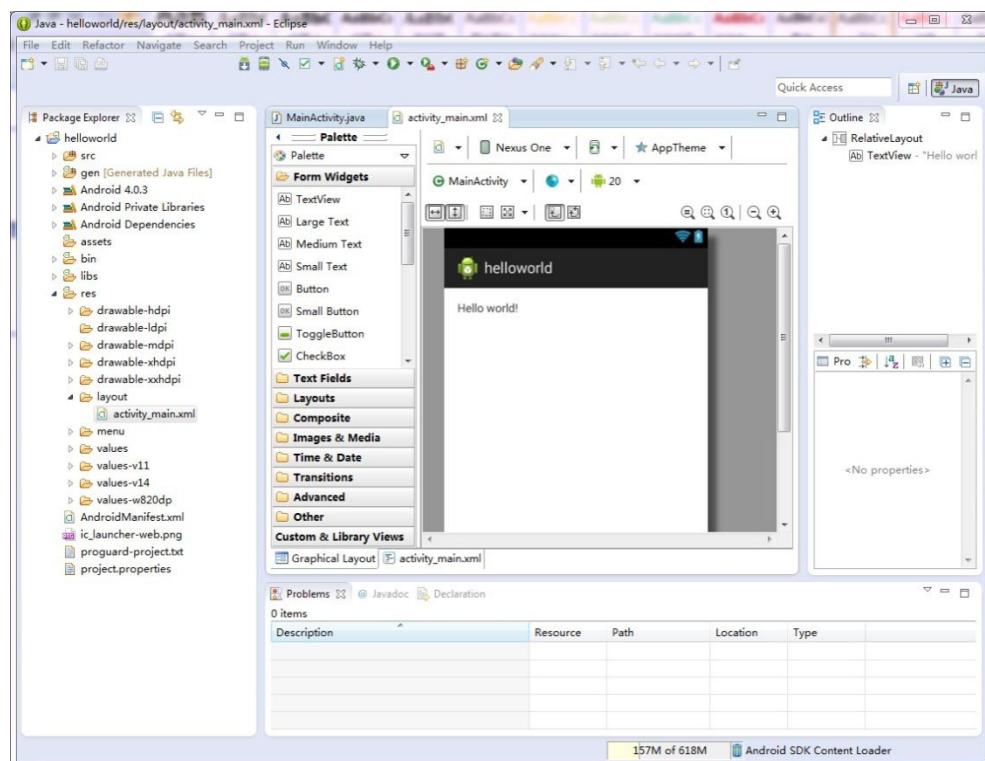
如下图，弹出对话框 “New Android Project” 。如下图所示设置。



单击上图中的按钮 “Next” ，后面的设置可以直接默认，直到出现如下图所示界面，单击 “Finish” ，回到 Eclipse 界面。

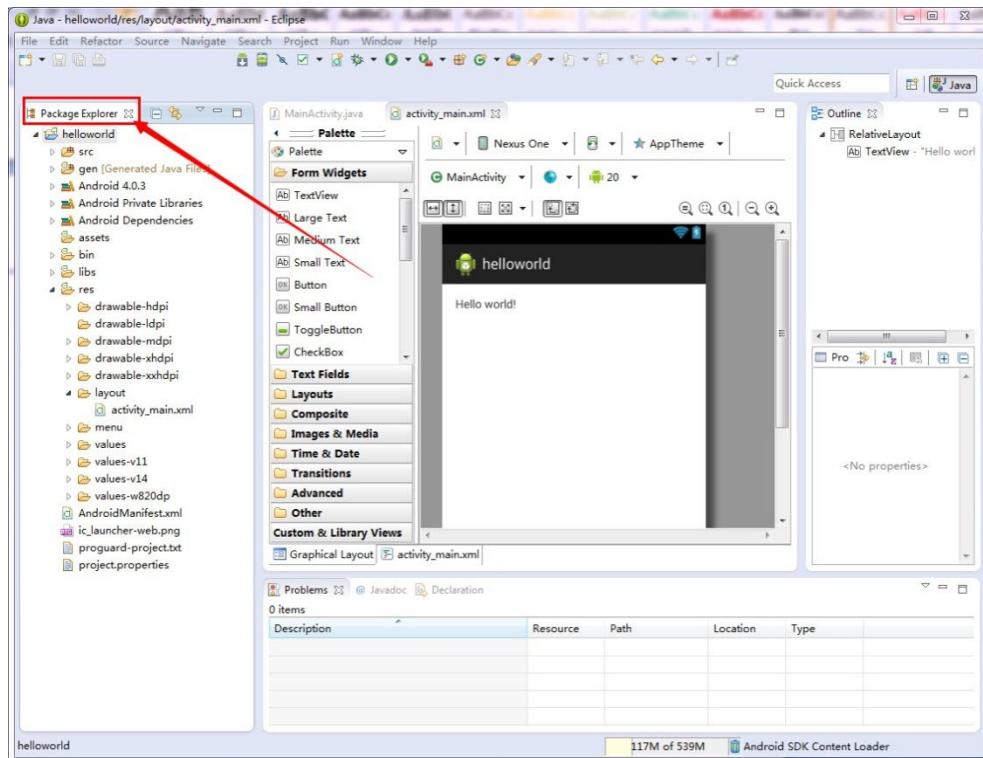


回到 Eclipse 界面，如下图所示。



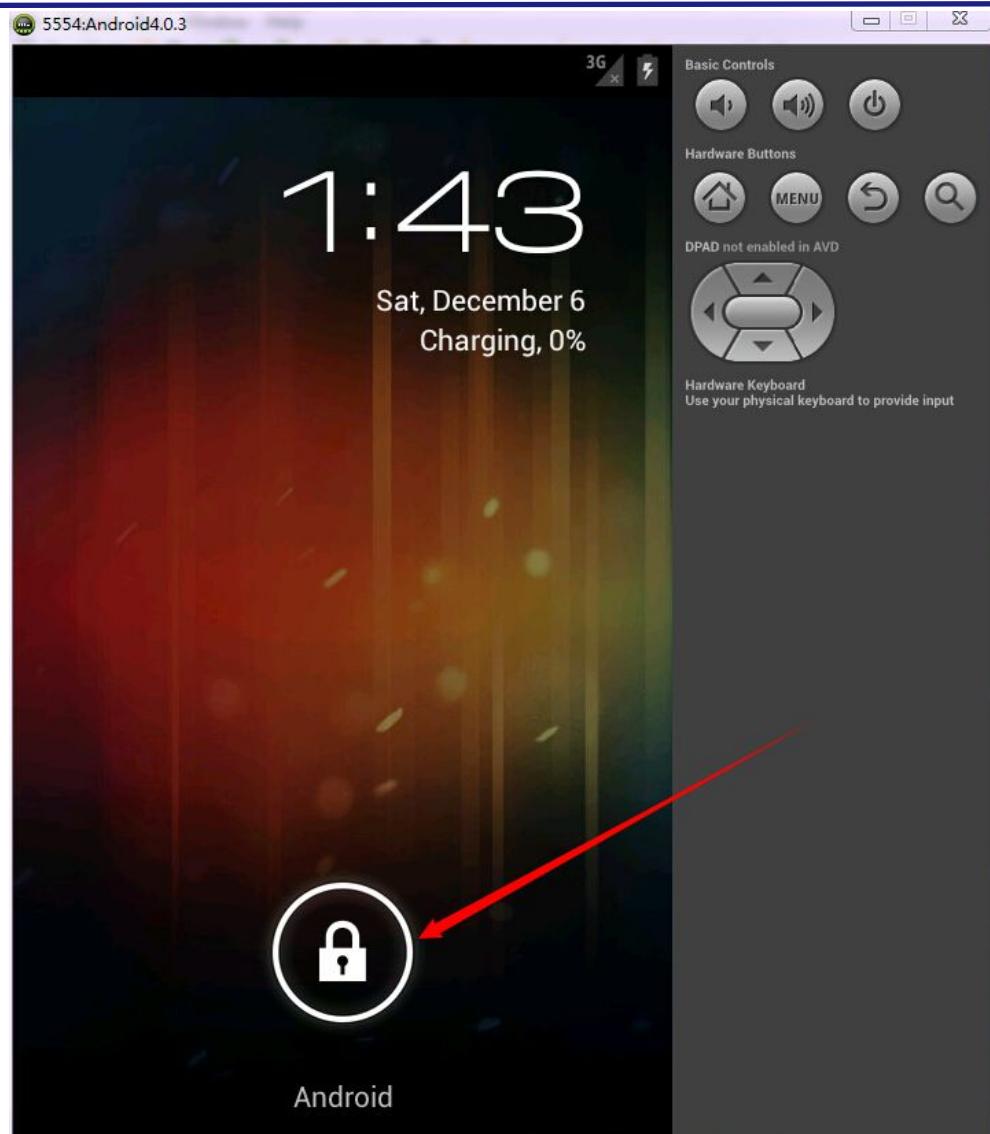
### 10.1.7 在模拟器上运行 hellworld

接上一小节，如下图，在“Package Explorer”中选中 hellworld 工程。

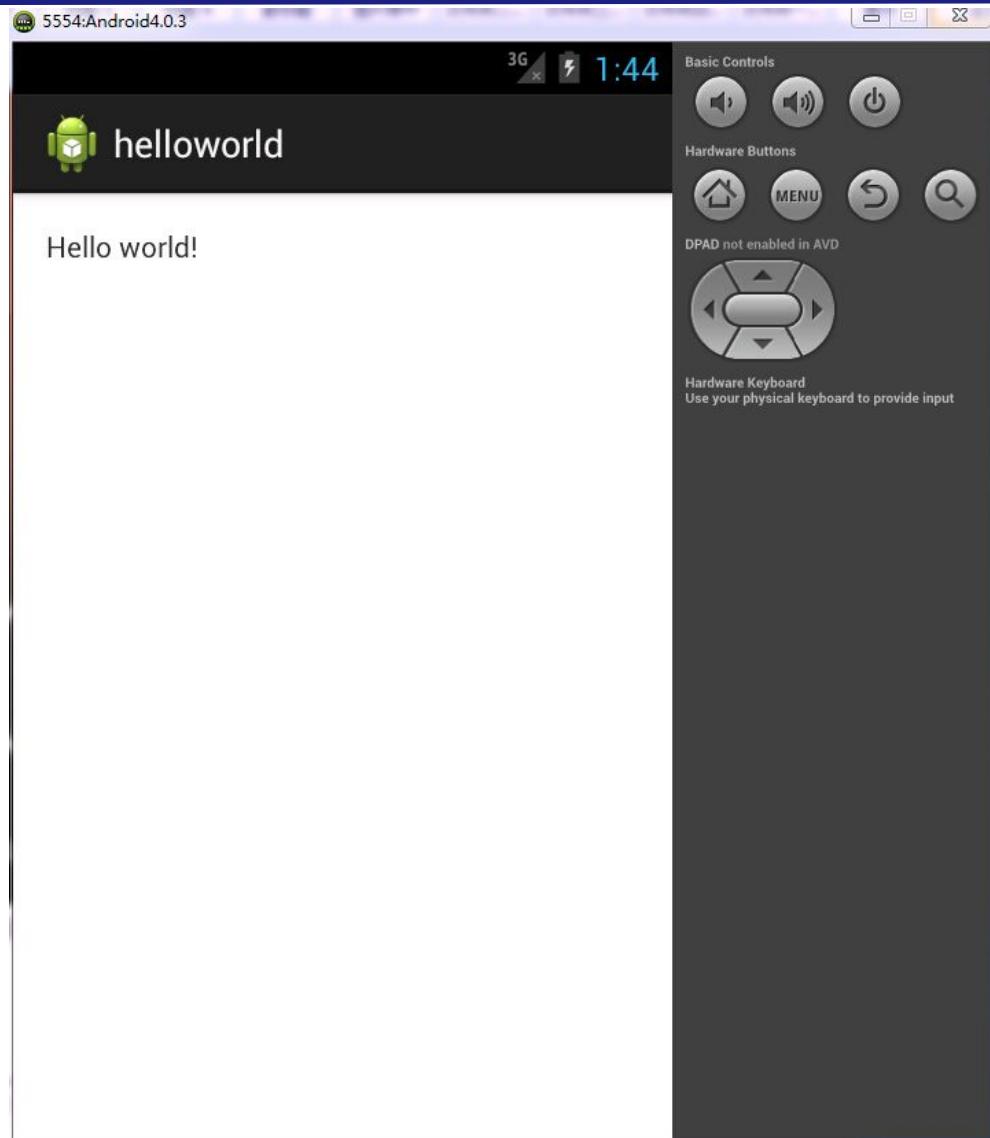


选择 Eclipse 菜单 “Run->Run As->Android Application” ，运行 hellworld 程序。

Eclipse 会自动启动 Android 的模拟器，这个过程和用户机器的配置有关，一般 1-5 分钟之间。启动后，如下图所示，给模拟器屏幕解锁。



如下图所示，hellworld 程序在模拟器上运行起来了。



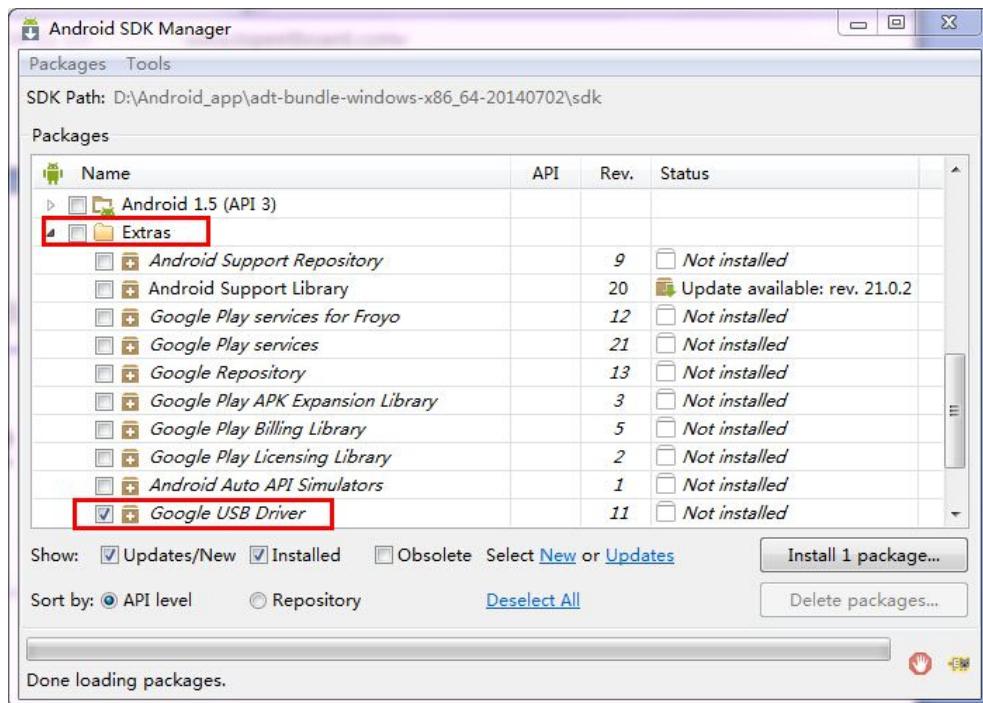
## 10.2 在 iTOP-4412 开发板上调试 helloworld 应用

### 10.2.1 安装 ADB 驱动

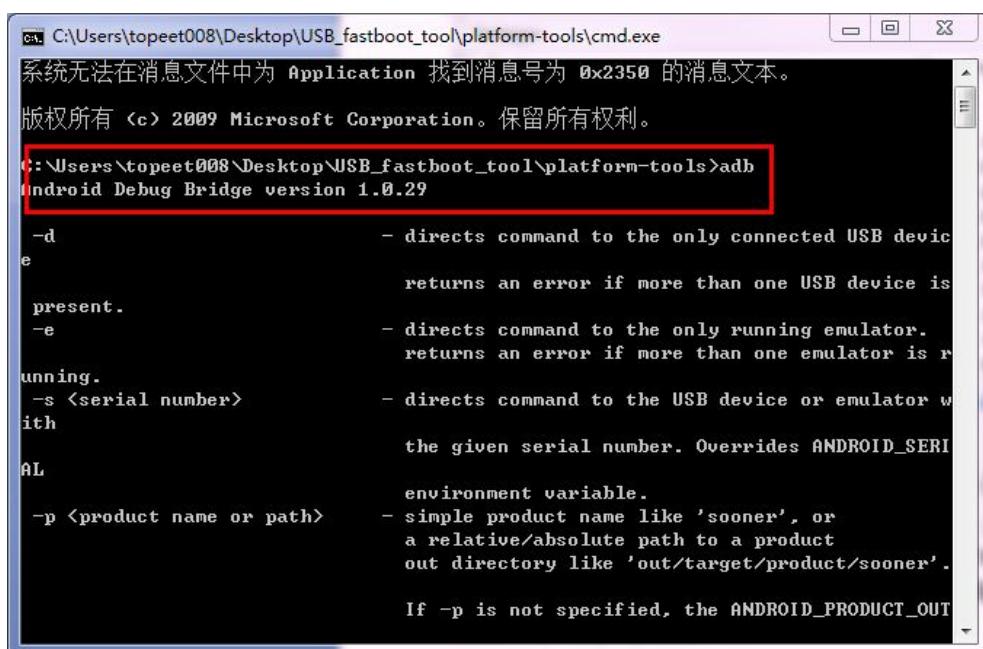
在开发板上调试 Android 应用，首先要安装 ADB 驱动。

用户可以参考 3.6 小节手动安装，在使用 fastboot 烧写镜像的时候已经安装过 ADB 驱动了。

也可以通过“SDK Manager.exe”来安装。如下图所示。另外需要注意的是，如果要使用 SDK Manager 安装软件，需要将 Eclipse 关闭。

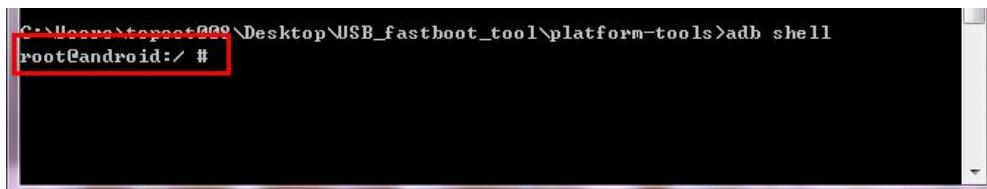


安装完成后，打开文件夹中“USB\_fastboot\_tool\platform-tools”的命令行 cmd.exe，如下图所示，输入命令“#adb”，然后回车。这里集成了 adb 命令，不需要用户去设置环境变量。如果有疑问参考参考 3.6 小节。



## 10.2.2 测试 ADB 驱动

启动 iTOP-4412 开发板，然后使用 OTG 线和电脑的 USB 接口相连接。在命令行中，输入命令 “#adb shell” 。如下图所示，表明 ADB 已经连接成功。



这里需要注意的是，ADB 全称是“Android Debug Bridge”，它是安卓的调试的一个程序。无论使用迅为电子提供的驱动，还是通过上面小节的程序“SDK Manager.exe”来安装驱动，本质上也是一个 USB，只是这个 USB 接口比较小，我们把它叫做 OTG 接口。OTG 接口，这是硬件上的一个称呼。

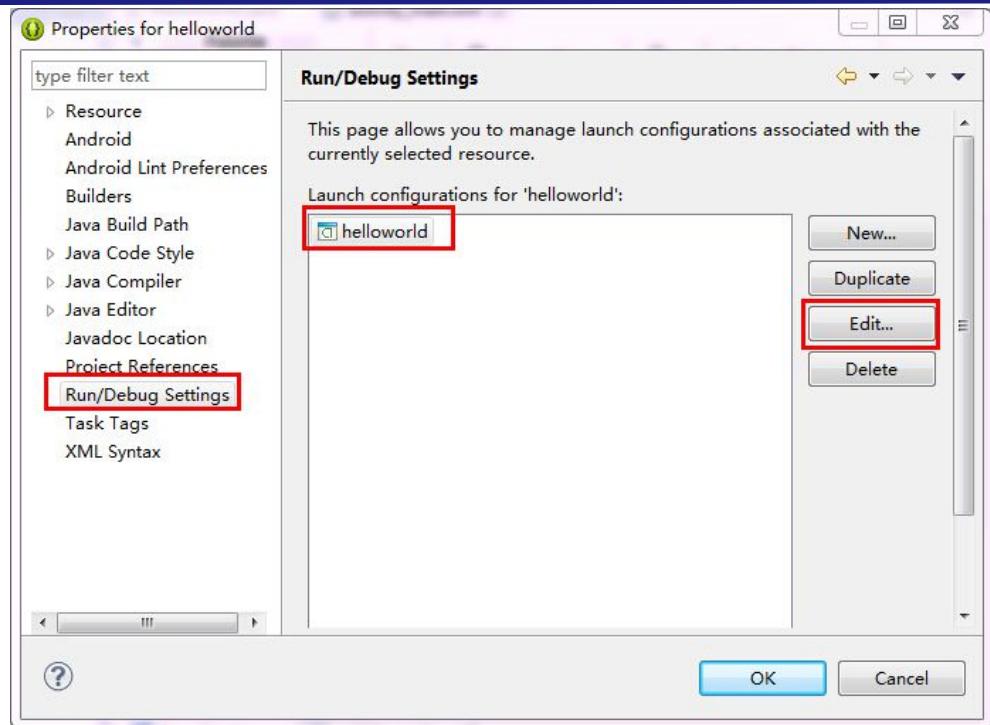
因为它需要在命令行中输入命令，有时候在用到这个接口的时候，也叫它 Windows 命令行。由于在 Windows 下的命令行是从 DOS 系统延续而来，所以也可以叫 DOS 命令行。我们这里所用的 ADB 只是增加了“adb.exe”这个可执行程序，有了这个程序，我们就可以输入 ADB 命令，所以有时候也叫它 ADB 命令。

## 10.2.3 通过 OTG 接口调试 helloworld 应用

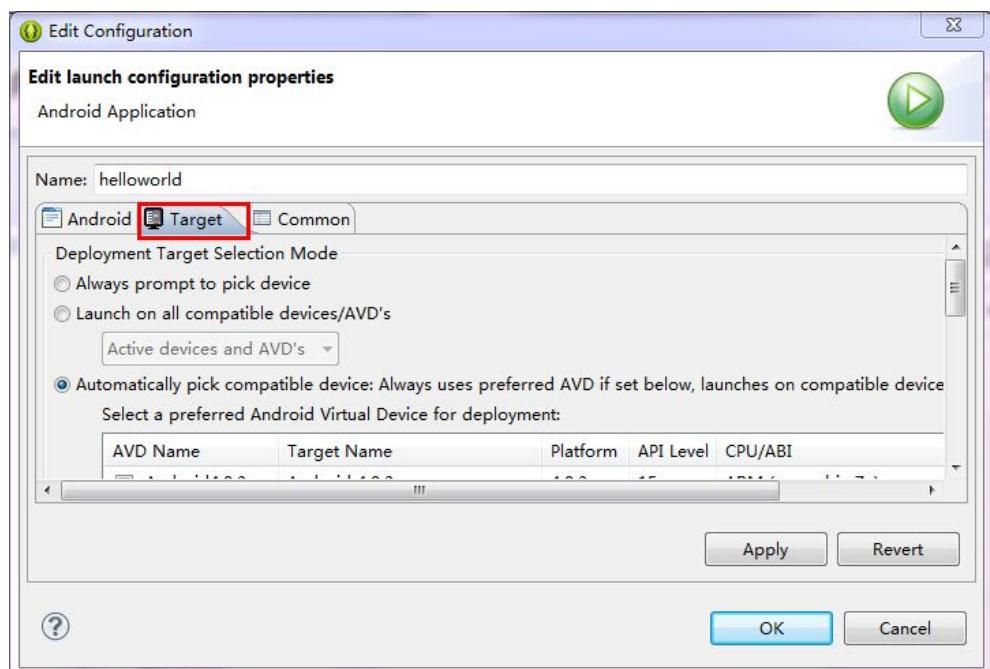
开启 iTOP-4412 开发板，使用 OTG 线连接电脑。

打开 Eclipse，打开 helloworld 工程。

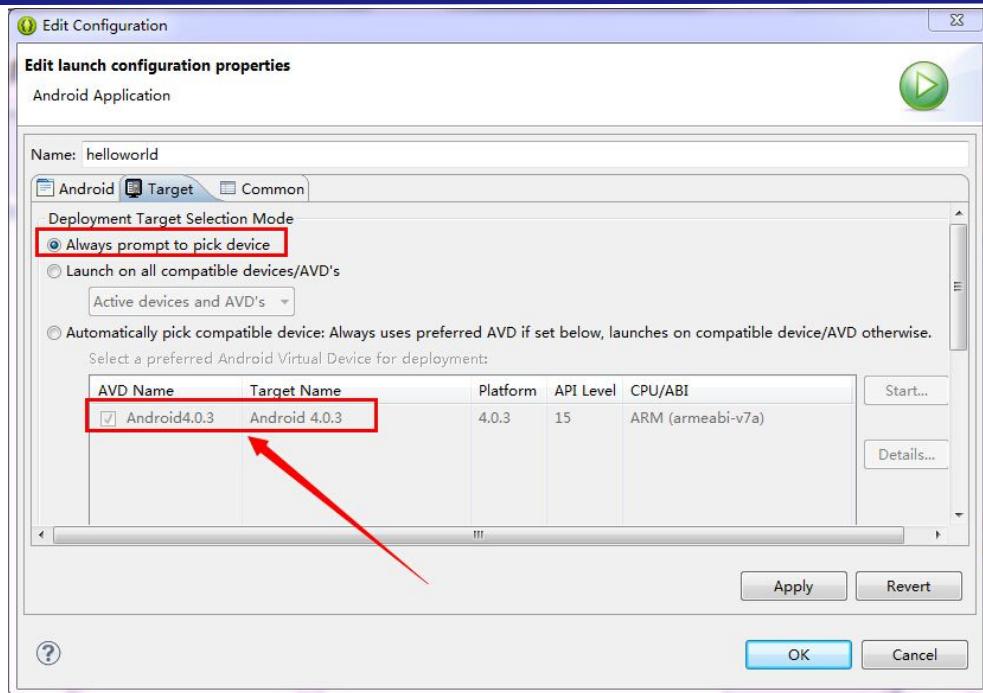
在界面的左侧，右击 helloworld 工程，点 Properties，弹出如下图所示的“Properties for helloworld”窗口。



如上图，选择“Run/Debug Settings”以及“helloworld”,单击“Edit...”。弹出如下图所示窗口，单击“Target”。

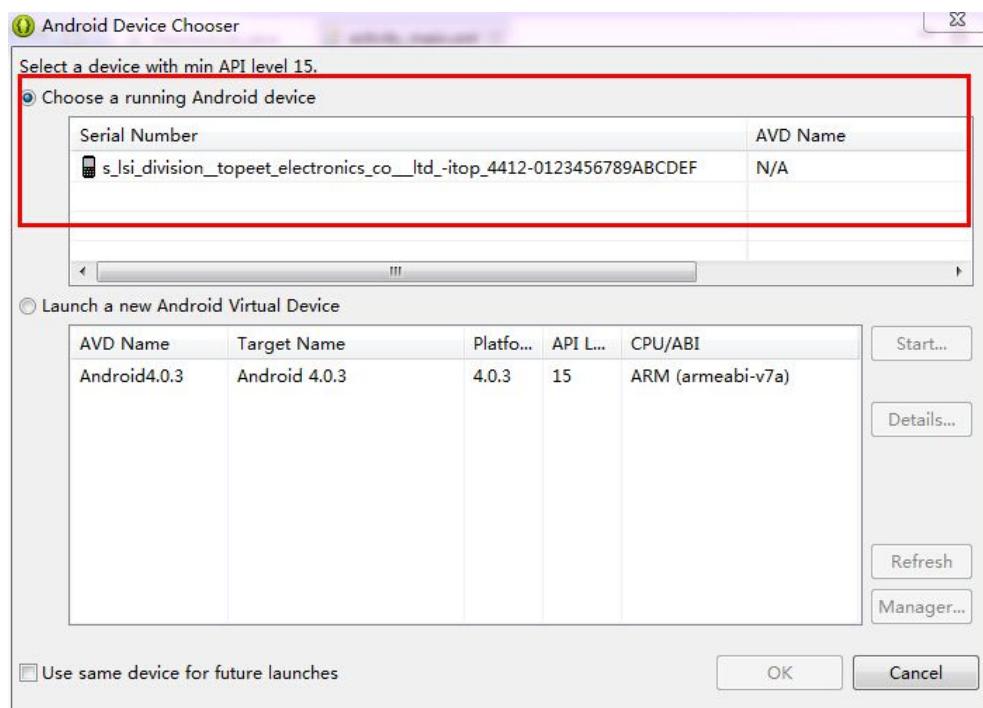


如下图所示，先选择红色箭头指的“Android4.0.3”，然后选择“Always prompt to pick device”。

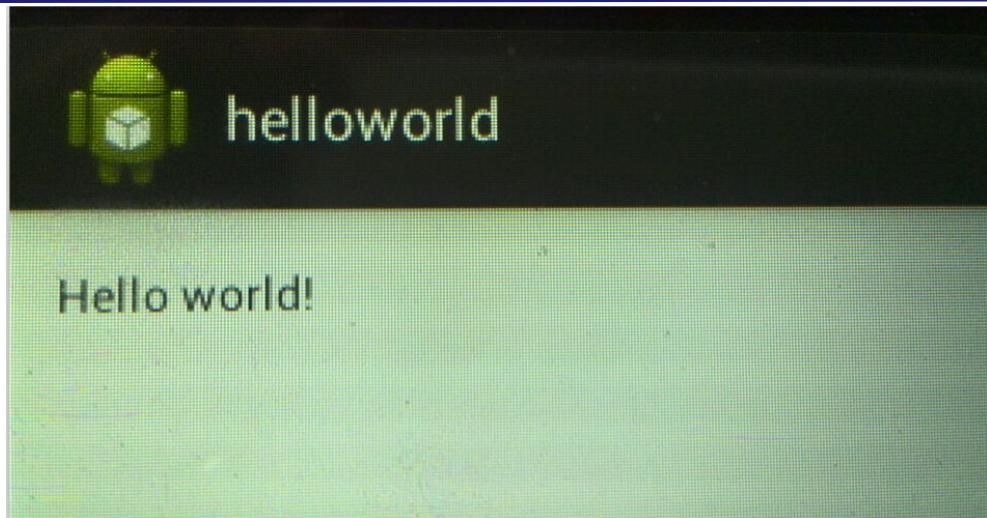


单击 OK 保存退出。

选中 helloworld 工程，选择菜单 “Run->Run” ，弹出如下图所示的 “Android Device Chooser” 设备选择框。



如上图所示，选择实体设备，也就是 iTOP-4412 开发板，选择后单击按钮 “OK” 。稍等一会，helloworld 应用程序就在开发板上运行起来了。



## 10.3 Led 应用程序

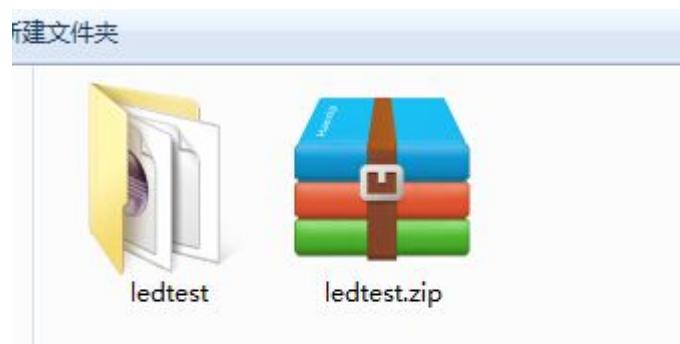
用户可以在网盘中下载“ledtest.zip”工程文件。

本节以 Led 为例详细讲解如何使用迅为电子提供的例程。

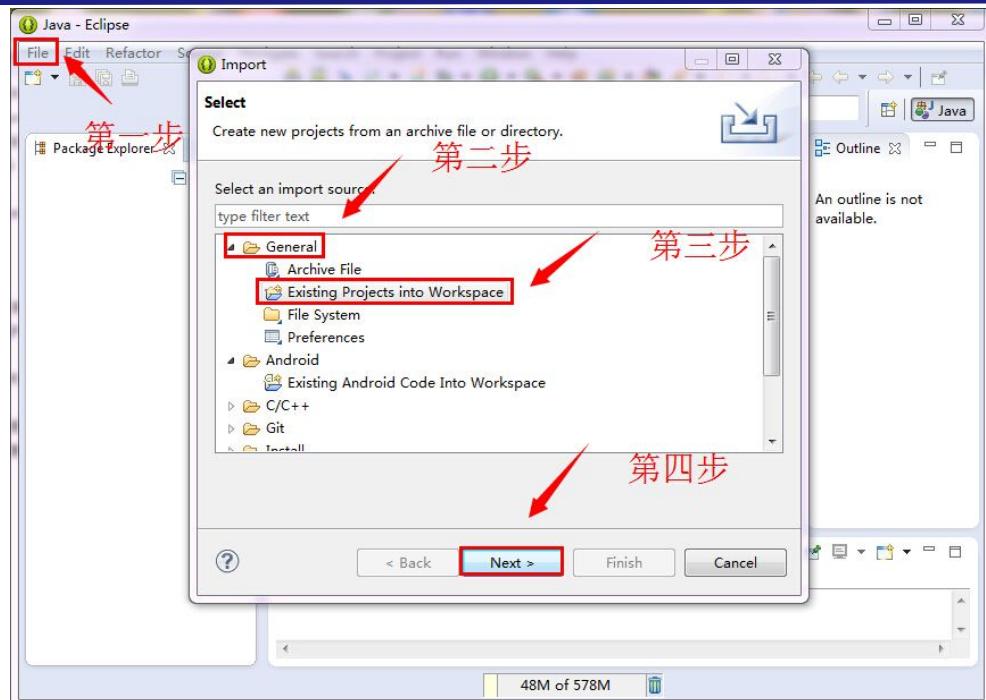
### 10.3.1 导入 Led 应用程序工程

本小节给大家详细讲解如何导入 Android 应用的工程文件。

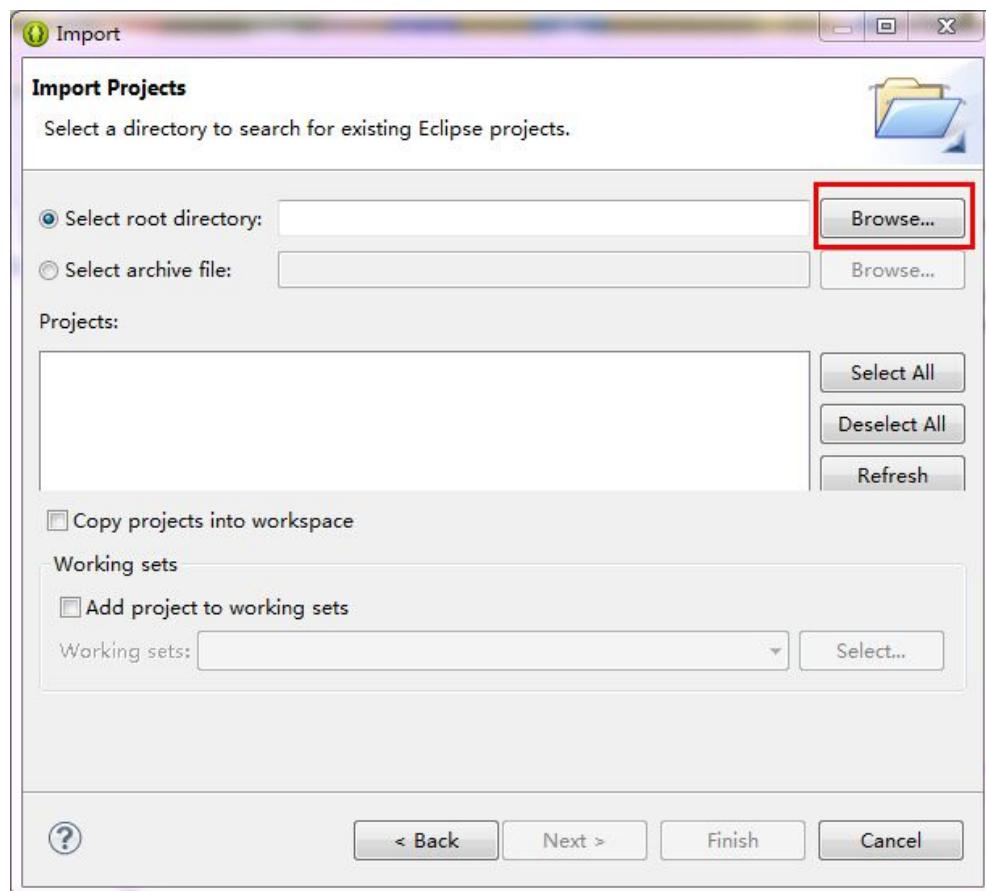
先解压“ledtest.zip”压缩包。如下图所示，解压出 ledtest 文件夹



然后，如下图所示，打开 Eclipse，单击“File”菜单，选择“Import.....”。

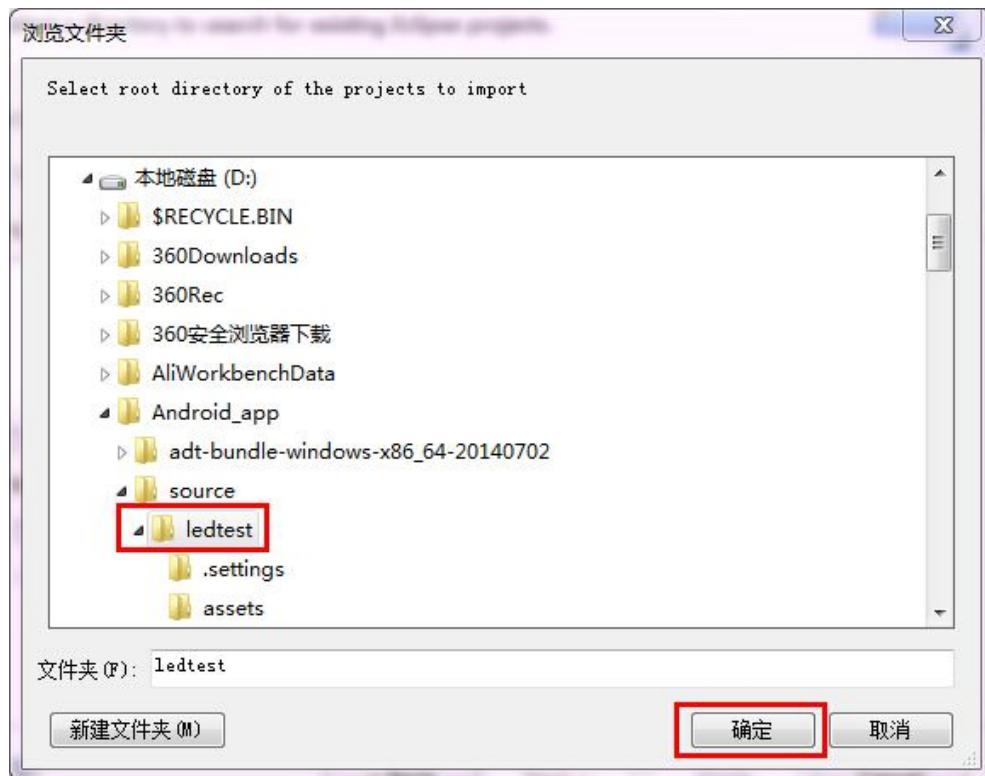


如上图所示，选择弹出窗口 “Import” 中的 “General/Existing Project into Workspace” ,单击按钮 “Next” 。弹出如下图所示界面。

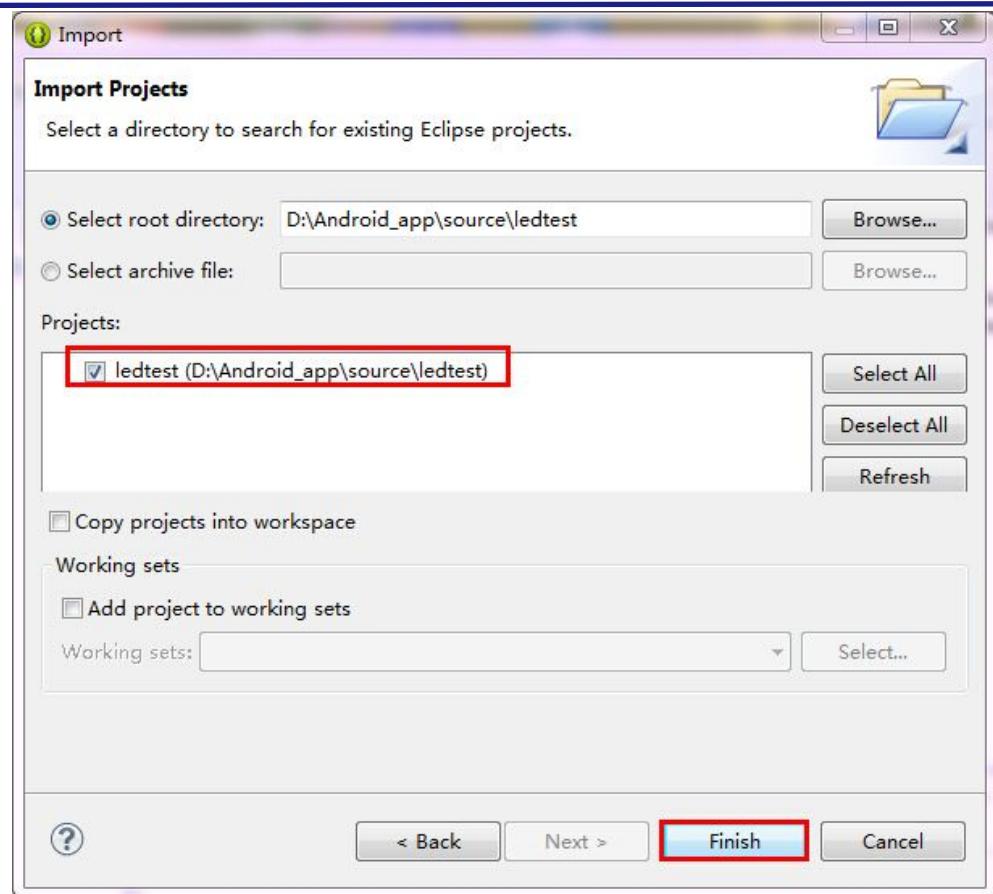


如上图，单击红色框中的按钮“Browse”。

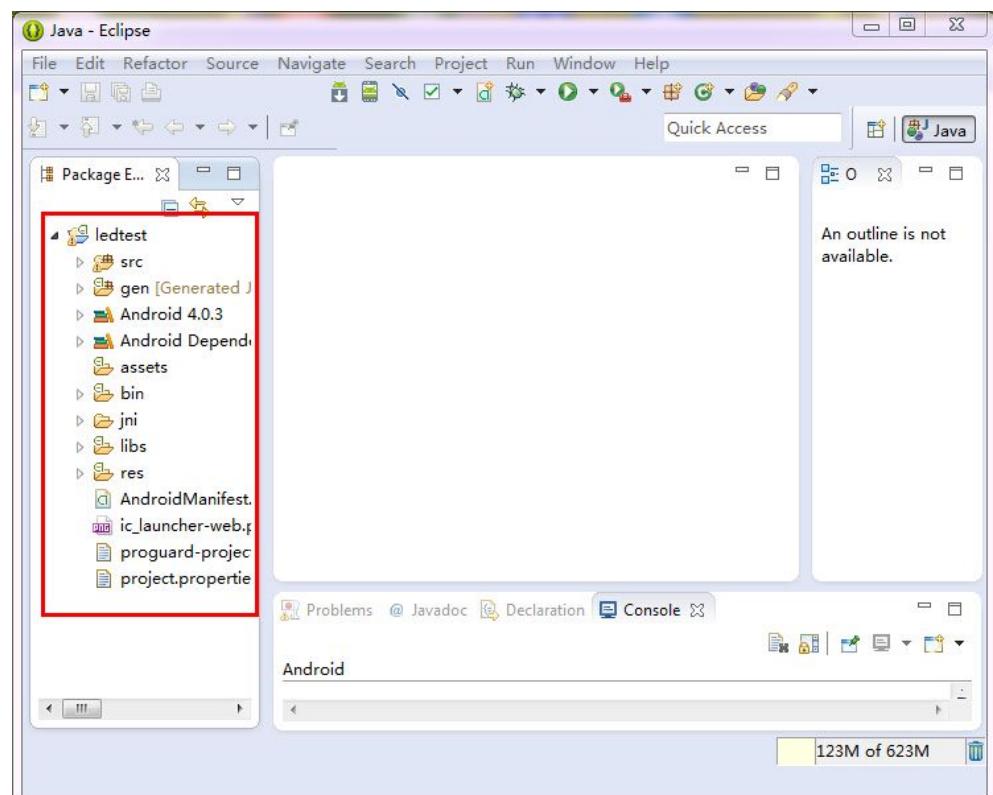
如下图，选上解压 ledtest 出的文件夹，单击确定。



如下图所示，返回 “Import” 弹窗，选上 ledtest 工程，然后单击按钮 “Finish” 。



如下图所示，导入成功。



### 10.3.2 导入工程常见问题

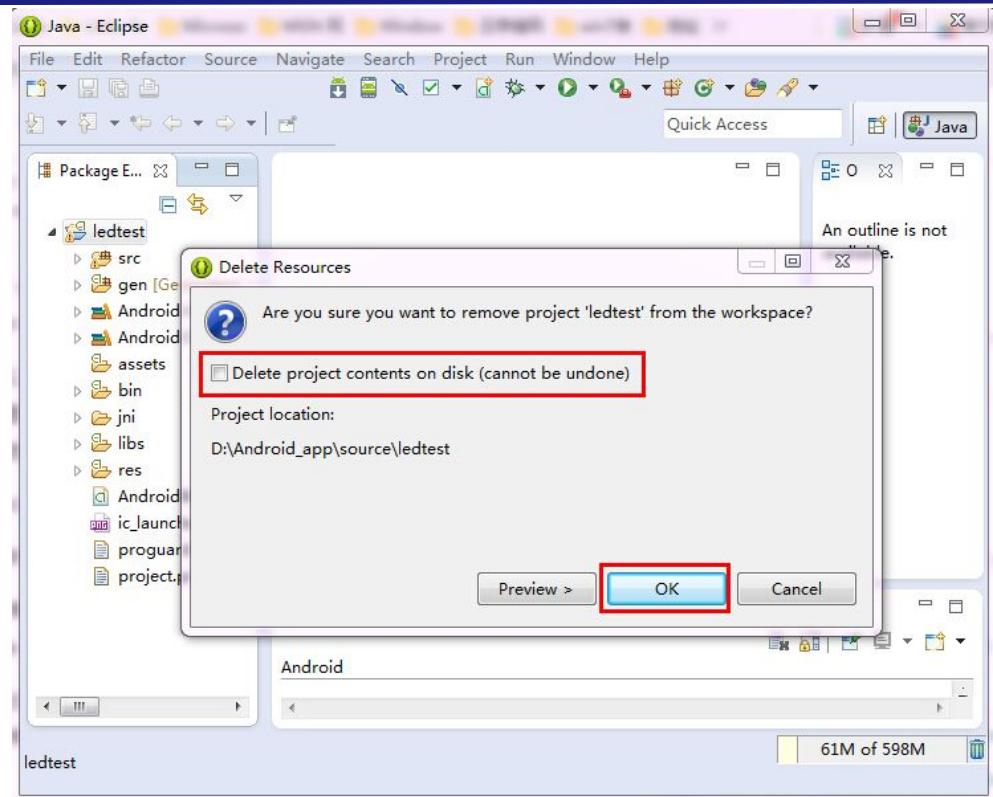
如果 Eclipse 使用不当，在导入工程的时候，可能会出现问题。本小节，介绍导入产生的原因以及解决办法。

从外部导入工程 workspace 目录的时候，在 workspace 目录中明明没有改工程，却提示“Some projects cannot be imported because they already exist in the workspace”。

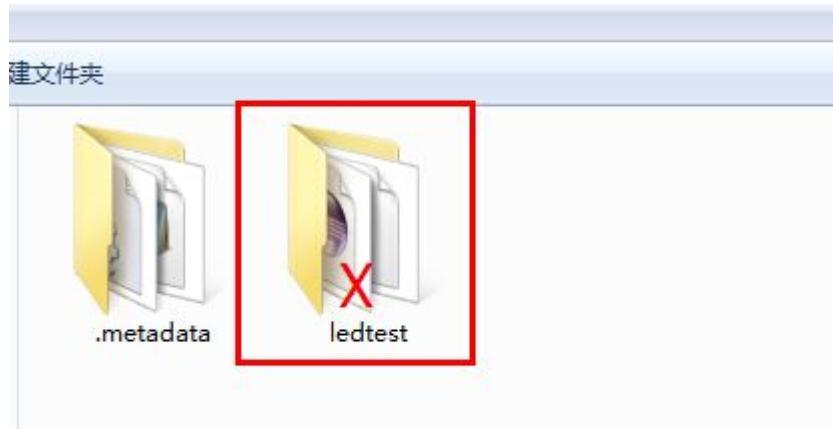
#### 10.3.2.1 导入工程失败的原因

出现上述错误的原因主要有两种，这里以已经导入的 ledtest 工程为例。

第一种情况，直接在 Eclipse 中删除工程，但是没有选择下图中方框的选项。这种删除方式导致工程文件删除了，但是在工作站中的文件夹 “.metadata” 中可能还有一些二进制残存的文件。就会导致 Eclipse 认为在工作站中，仍然有 ledtest 工程。

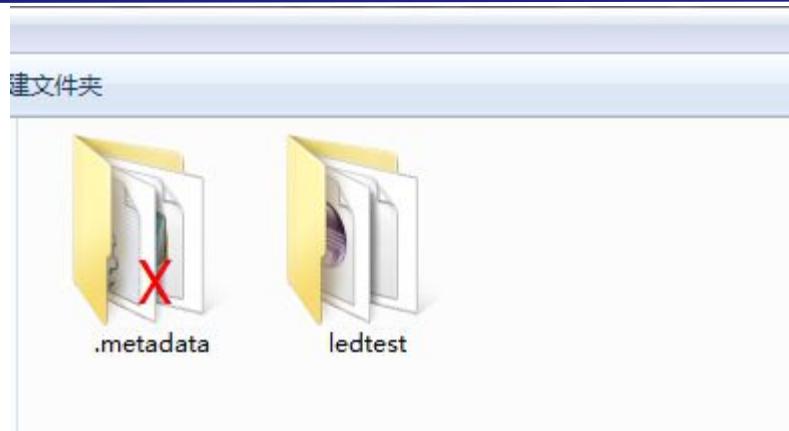


第一种情况，如下图所示，直接删除在工作站中的文件夹 “.metadata” ，原因同上。



### 10.3.2.2 解决办法

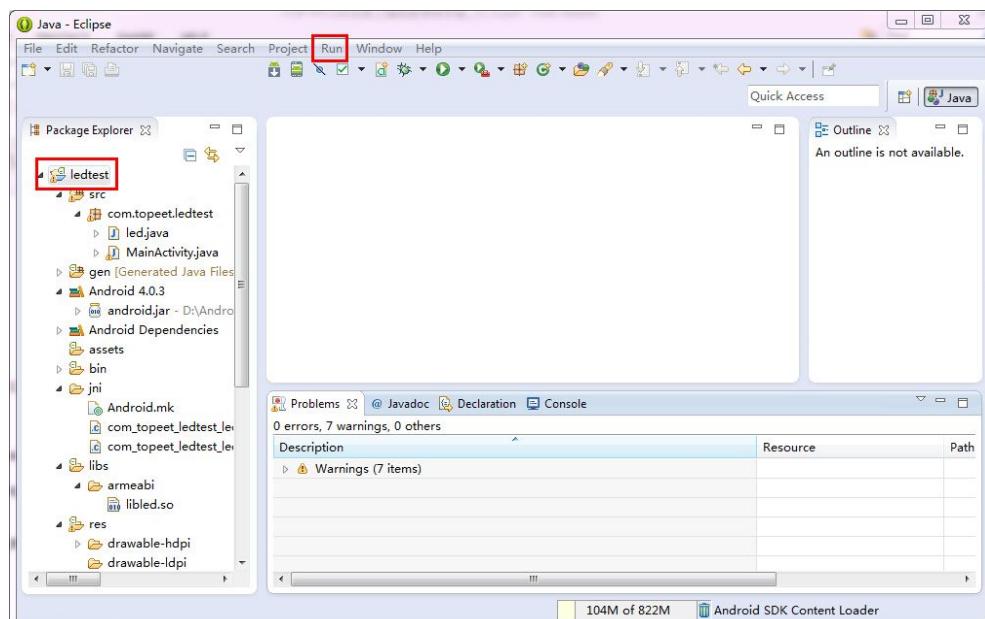
如下图所示，进入工作站中，删除文件夹 “.metadata” ，然后重新建立一个工作站，将新建工作站中的文件夹 “.metadata” 拷贝到该工作站文件夹中，重新导入。



#### 10.3.4 在模拟器上调试

在调试前，需要建立虚拟设备，参考“10.1.6 创建 Android 模拟器”。Android4.0.3 的模拟器建好了之后，将默认模拟器设置为 Android4.0.3 版本。

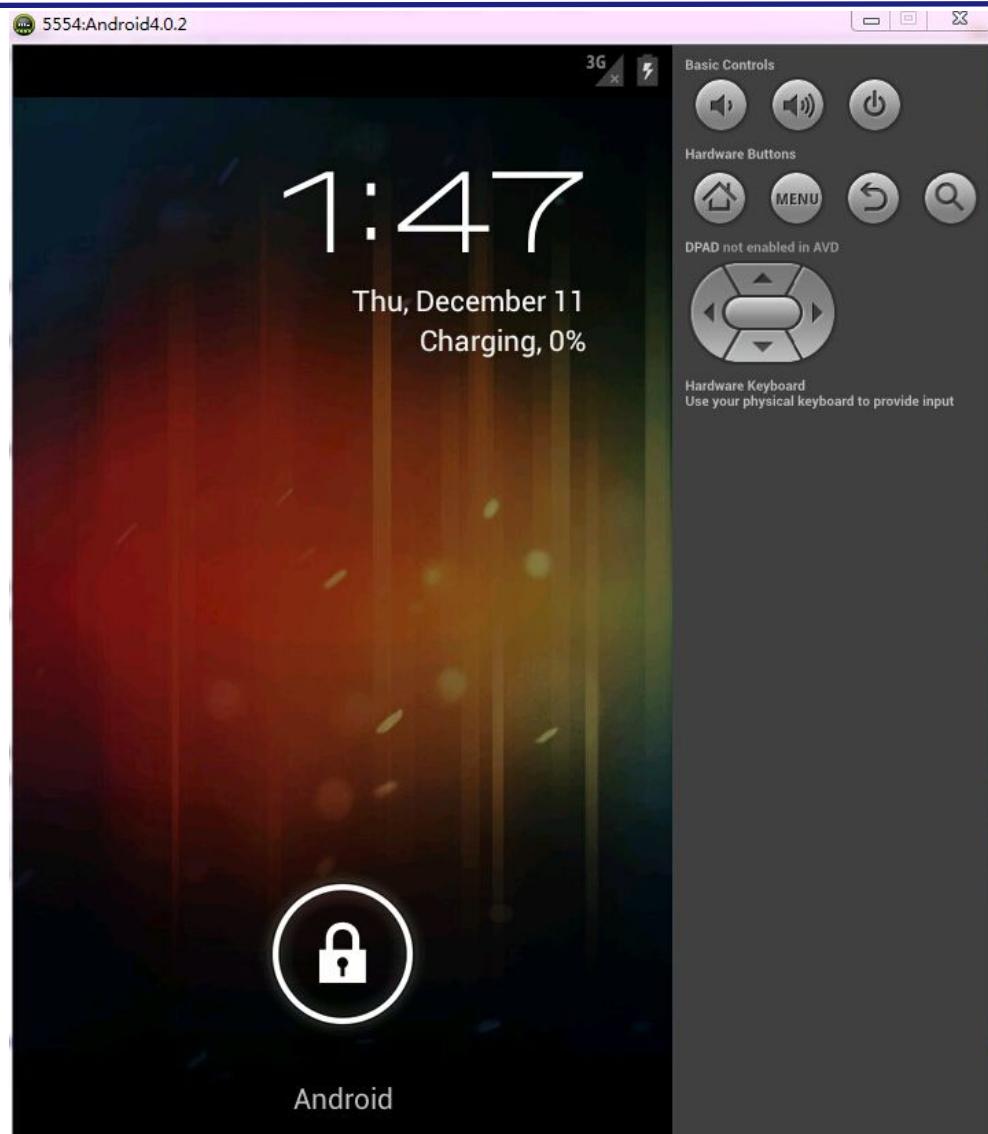
打开 Eclipse，选上工程 ledtest，然后单击菜单“run→run”。



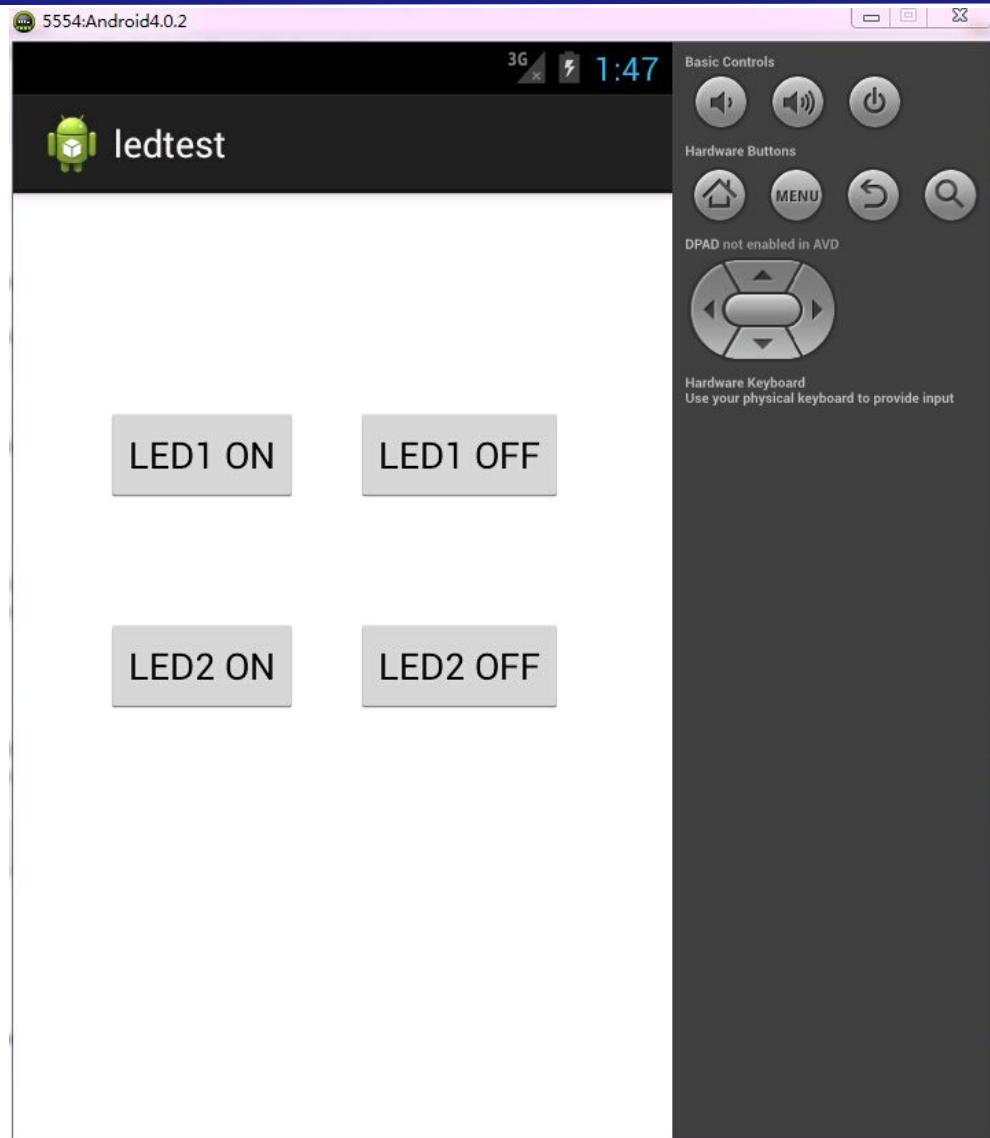
如下图，模拟器加载中。



如下图所示，加载完成。



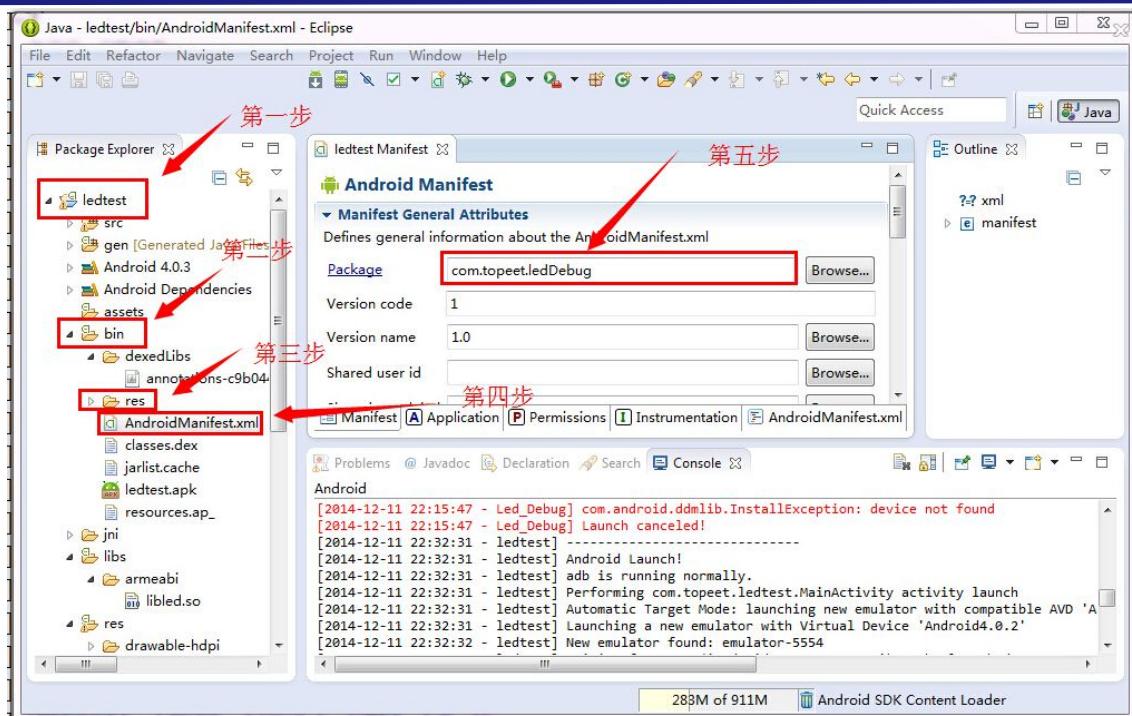
如下图，将屏幕解锁之后，ledtest 会在模拟器上直接运行起来。



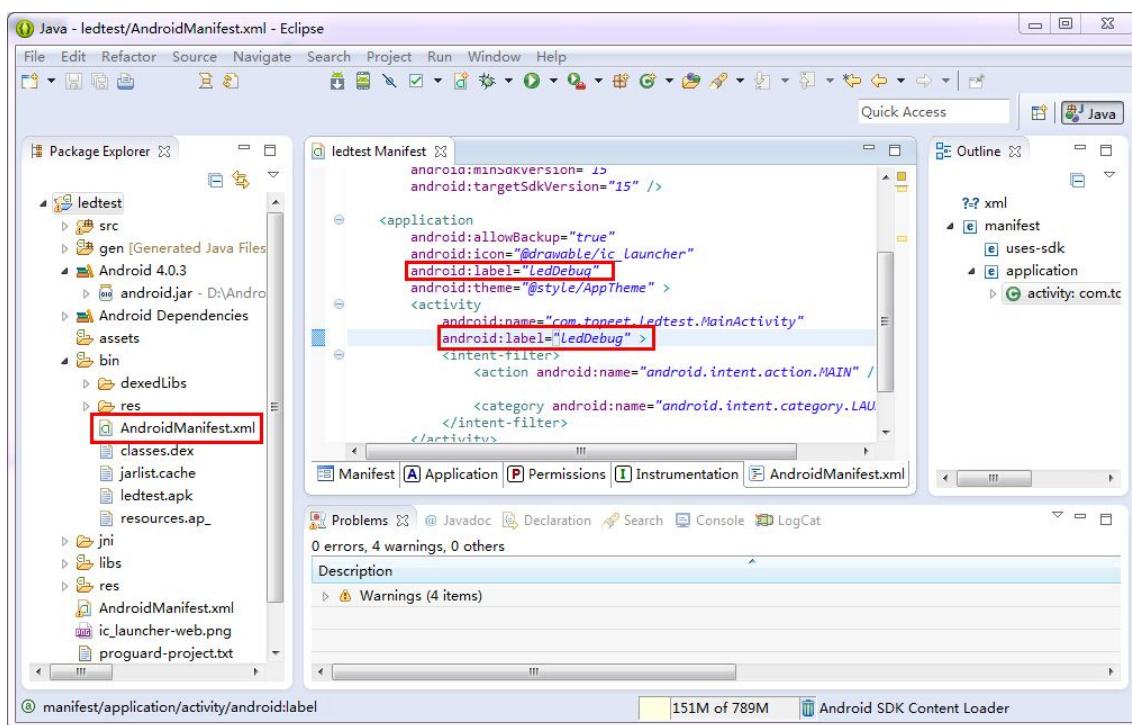
### 10.3.5 在开发板上调试

由于在开发板上 ledtest 应用已经默认安装了，所以在开发板上调试已安装应用的时候，需要做一下处理才能够正常连接。

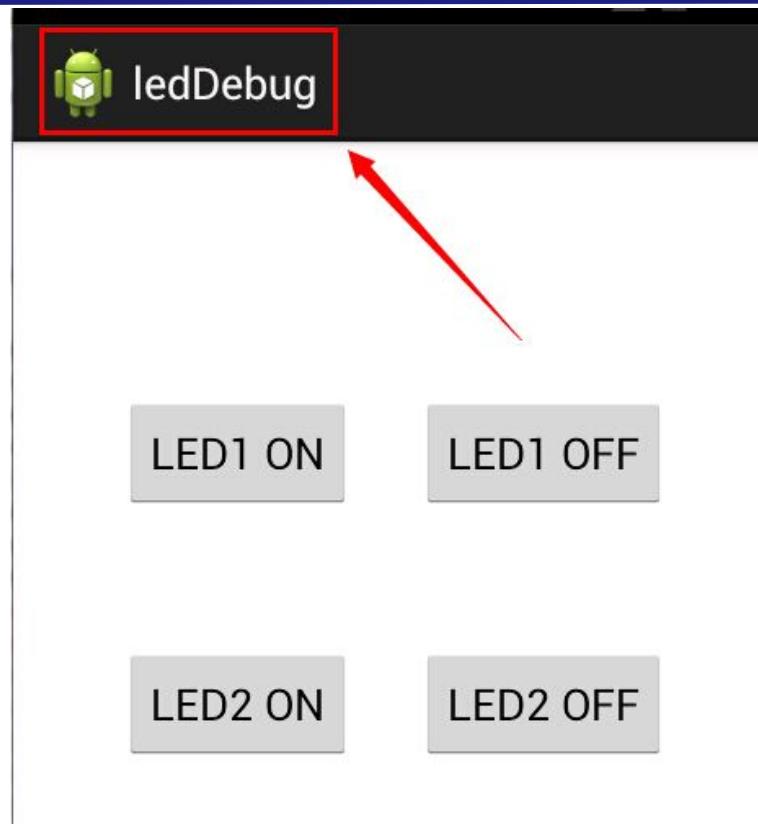
如下图所示，在打开工程“ledtest”，打开“bin”→“res”→“AndroidManifest.xml”文件。将 Package 的名称“com.topeet.ledtest”改为“com.topeet.ledDebug”。



如果想修改应用图标名称和应用程序窗口名称，则将下面的两个红色框中的参数修改成下图对应的名称。



如下图所示，应用程序窗体名称改变为 ledDebug。



如下图所示，应用程序 APK 名称改变为 ledDebug。



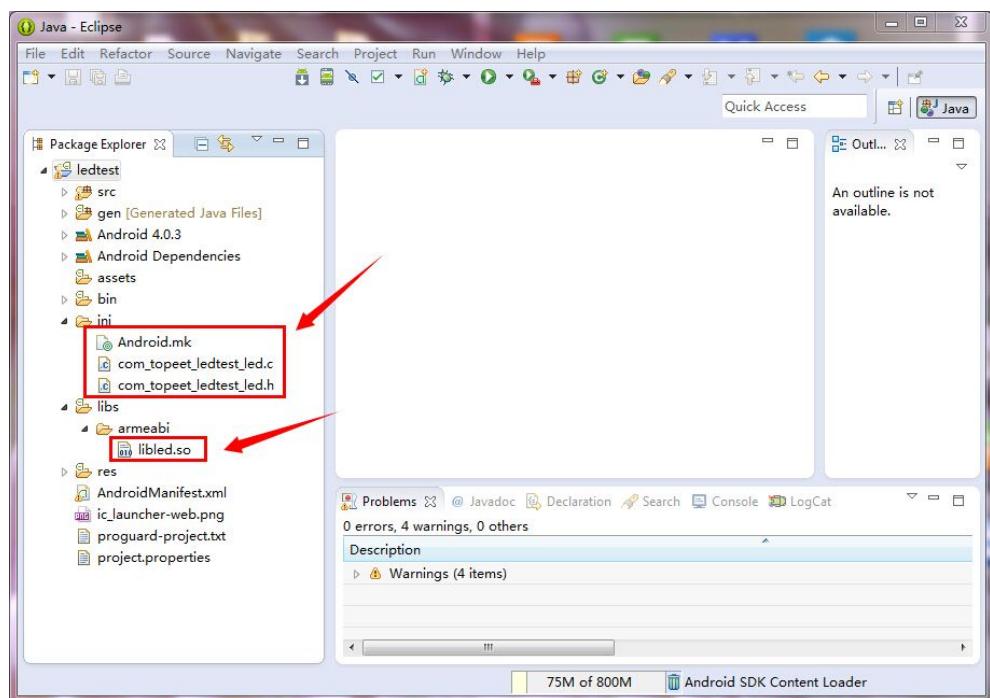
## 10.4 JNI 基础概念

Java 程序可以通过 JNI ( Java Native Interface , Java 本地调用 ) 访问本地的动态链接库 , 从而扩展其功能、保护关键代码、提高运行效率。

用户可以在网盘下载 “android-ndk-r8b-linux-x86.tar.bz2” 。

### 10.4.1 ledtest 工程中的 JNI

打开 ledtest 工程 , 如下图所示 , 在工程文件中 , 会发现 jni 文件夹中有 c 文件以及在 libs 文件夹中有 libled.so 库文件。这个就是 JNI 的一个具体应用。



### 10.4.2 什么情况下需要使用 JNI

Java 本机接口 ( Java Native Interface (JNI) ) 是一个本机编程接口 , 它是 Java 软件开发工具箱 ( Java Software Development Kit (SDK) ) 的一部分 , JNI 它提供了若干的 API , 实现了和 Java 和其他语言的通信 ( 主要是 C&C++ ) 。

JNI 允许 Java 代码使用以其它语言（譬如 C 和 C++）编写的代码和代码库。注意本使用手册以及例程都是 C 的代码和库。

Invocation API ( JNI 的一部分 ) 可以用来将 Java 虚拟机 ( JVM ) 嵌入到本机应用程序中，从而允许程序员从本机代码内部调用 Java 代码。

我们知道 Java 是一种平台无关性的语言，平台对于上层的 java 代码来说是透明的，所以在 Android 上层应用中，大部分情况是不需要 JNI 的。

但是在嵌入式 Android 开发中，以下两种情况肯定需要用到 JNI。

第一种情况，用户需要直接调用开发板底层中的驱动实现特定功能，那么怎么实现？因为底层 Kernel 中都是用 C 编写的。Java 是没有办法直接调用底层的，那么这种情况下就需要用到 JNI。

第二种情况，用户有现成的应用，但是都是基于 C 写的，那么想要尽快的完成任务，那么就需要用到 JNI。将 C 的程序移植到 JNI 中，然后制作好 API 接口，供上层 Java 的应用程序调用。

当然，还有其它很多情况下都需要用到 JNI，例如你有 C 写的算法，以及一些 C 的动态库想直接调用等等，但是那不是属于纯粹的嵌入式 Android 了，这里就不详细介绍。

### 10.4.3 与 JNI 相关的文件

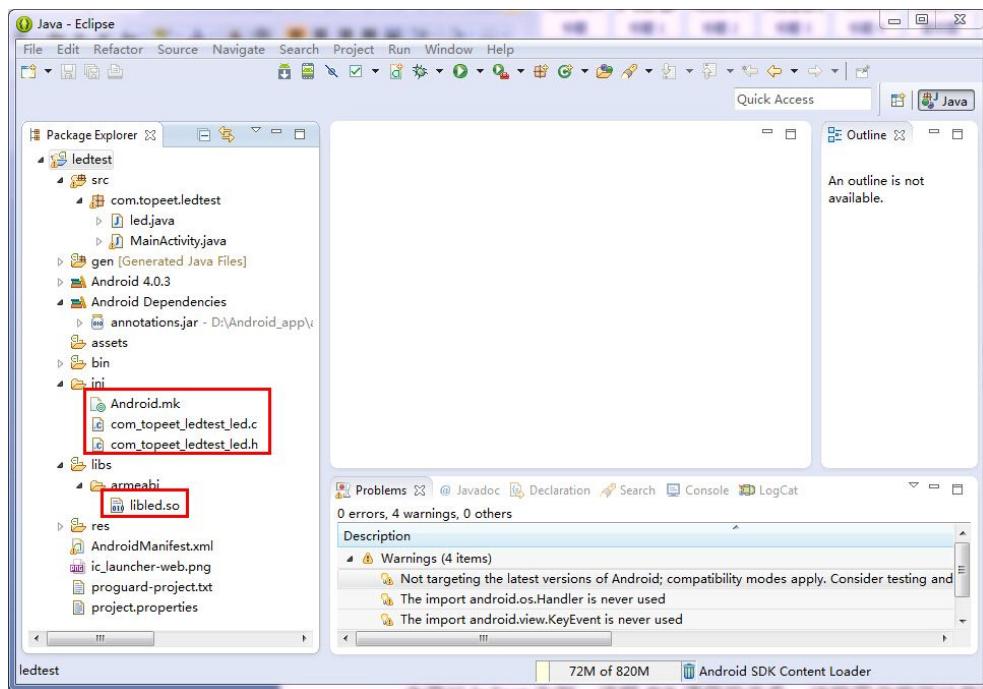
如下图所示，如果需要使用 JNI 功能和 Kernel 通信，那么就需要有以下几个文件：

jni.h→com.topeet.ledtest.h

jni.c→com.topeet.ledtest.c

lib.so→libled.so

## Android.mk→Android.mk



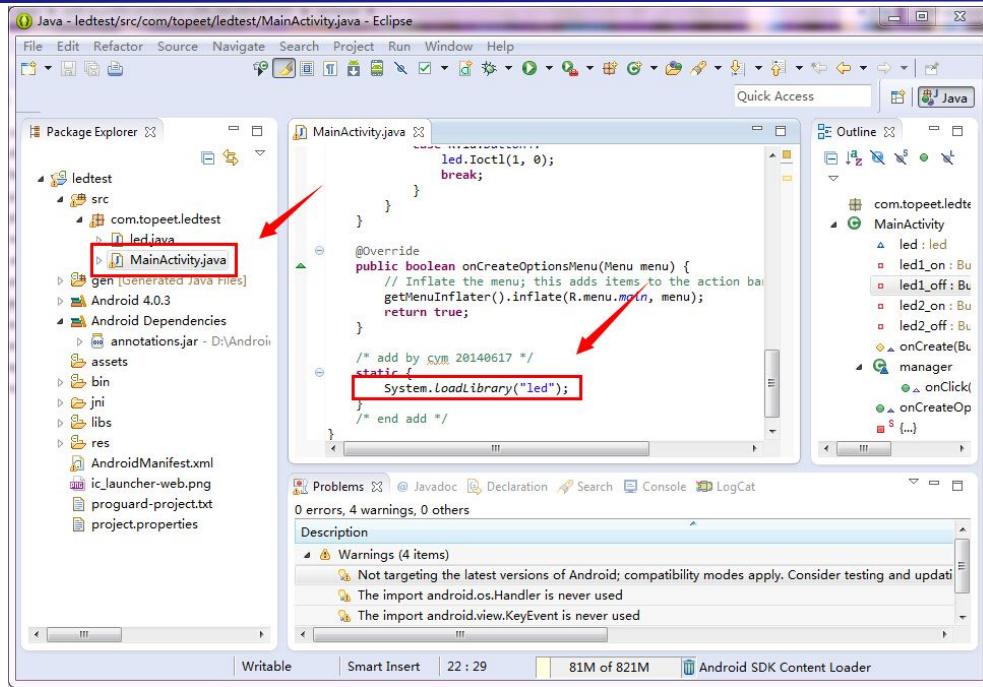
## 10.5 Java 程序调用 JNI 的方法和步骤

本节以 ledtest 为例，讲解 JNI 调用的关系，这样用户就可以仿写自己的 JNI。

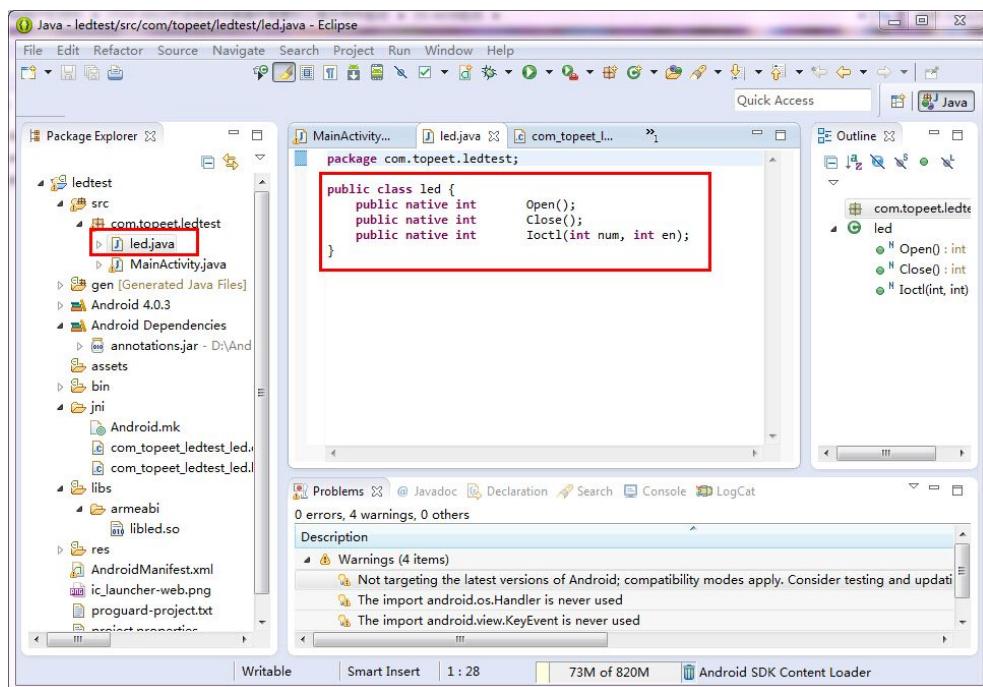
### Java 应用调用库

如下图所示，找到 “MainActivity.java” 文件，找到代码

“System.loadLibrary("led");” 这个代码就是包含了 led 库，也就是 C 的库。



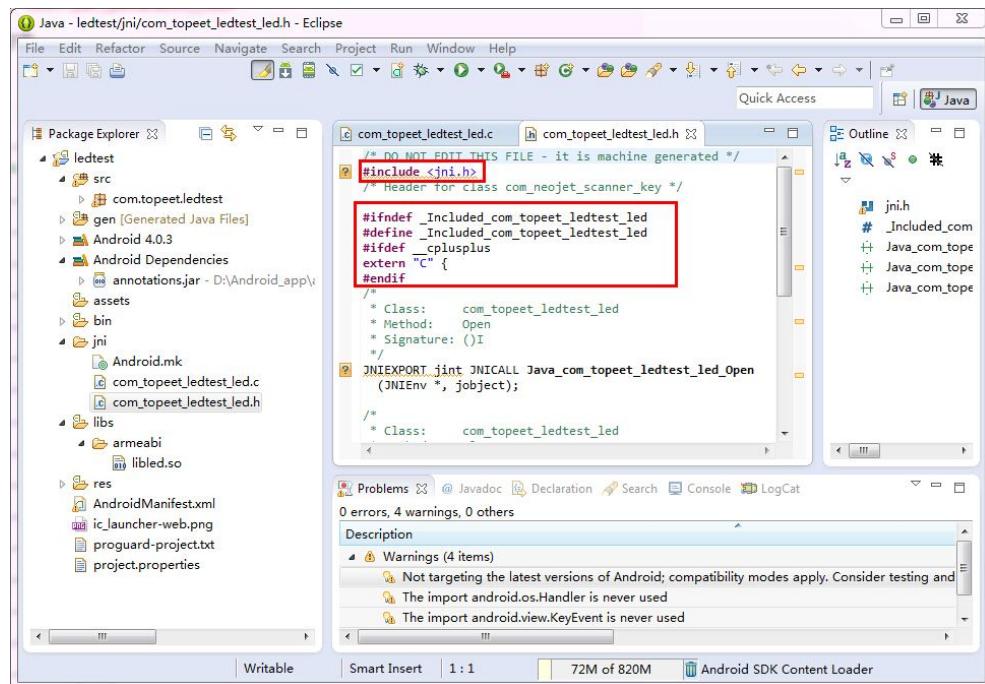
然后，如下图所示，找到文件“led.java”，led 类中包含了 Open、Close 以及 Ioctl，这里对应对底层字符驱动的打开、关闭以及输入数据的操作。



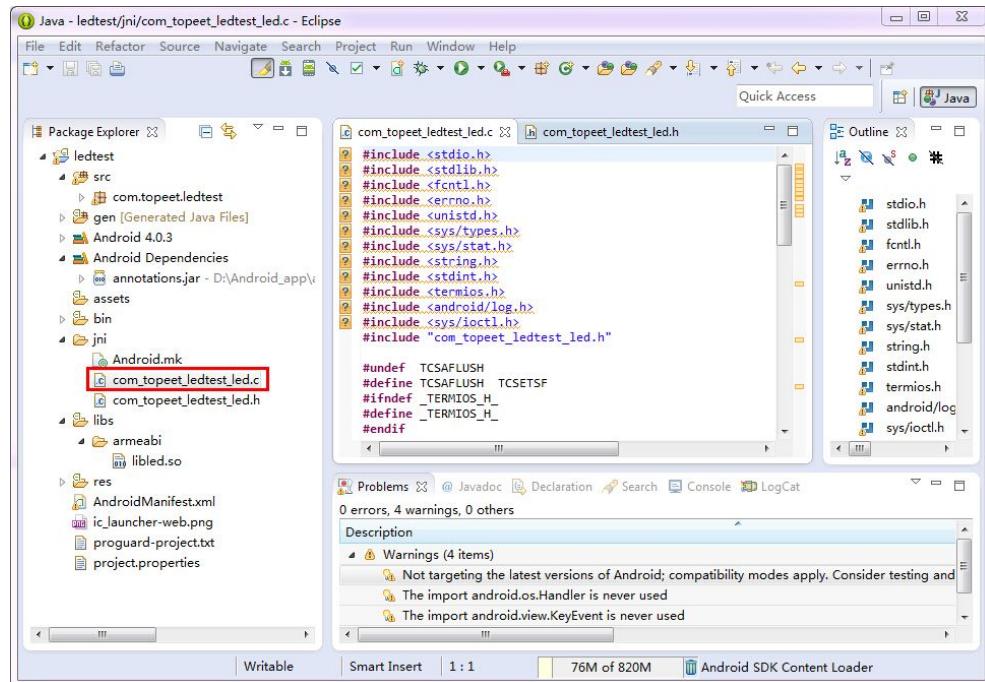
上面是属于本机文件的处理，下面介绍一下和 linux 相关的文件。

如下图所示，打开"com.topeet.ledtest\_led.h"文件。首先需要包含 jni.h 文件，实现 JNI 功能必须包含这个头文件。然后就是大的红色方框中，这里需要通知 Java 程序，这里是 C 的

程序，用户可以模仿者写。然后就是上图中 Open、Close 以及 ioctl 的定义，这里也是有特定的格式要求，用户如果想了解更多，可以看看 jni 头文件，里面对语法的规则都有定义。



最后给大家详细分析一下" com.topeet.ledtest\_led.c"文件，如下图所示。



如下图所示，头文件"com\_topeet\_ledtest\_led.h"上面已经分析过了，必须按照 jni.h 中定义的语法来写。包含了头文件 "android/log.h"，这样我们在和开发板上调试的时候就可以使用 android 的 printk，方便调试。

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <string.h>
#include <stdint.h>
#include <termios.h>
#include <android/log.h>
#include <sys/ioctl.h>
#include "com_topeet_ledtest_led.h"
```

再来介绍一下剩下的头文件，剩下的头文件全部是 linux 中标准的库文件，包括字符处理、报错、数学库以及调用设备节点需要的头文件等等。如果是做上层应用的用户，这些头文件简单的了解一下就可以。这里提醒一句，我们提供了很多 C 的应用例程，应用例程中的一些头文件也是实现对应的调用底层功能的，用户直接使用就可以了，不需要弄的太清楚。最后把自己需要的函数打包给上层调用就成了。

### 最后来看一下定义的三个 Open、Close 以及 ioctl

先来分析一下 Open 函数，JNI 中的 "Java\_com\_topeet\_ledtest\_led\_Open"，主要功能是调用 "/dev/leds" 设备节点。

```
/*
 * Class:      com_topeet_ledtest_led
 * Method:    Open
 * Signature: ()I
 */
JNIEXPORT jint JNICALL Java_com_topeet_ledtest_led_Open
(JNIEnv *env, jobject obj)
{
    if(fd<=0)fd = open("/dev/leds", O_RDWR|O_NDELAY|O_NOCTTY);
    if(fd <=0 )__android_log_print(ANDROID_LOG_INFO, "serial", "open /dev/leds Error");
    else __android_log_print(ANDROID_LOG_INFO, "serial", "open /dev/leds Sucess fd=%d",fd);
}
```

然后，就是 JNI 中的 “Java\_com\_topeet\_ledtest\_led\_Close” ，主要功能是释放 “/dev/leds” 设备节点。

```
/*
 * Class:      com_topeet_ledtest_led
 * Method:     Close
 * Signature:  ()I
 */
JNIEXPORT jint JNICALL Java_com_topeet_ledtest_led_Close
(JNIEnv *env, jobject obj)
{
    if(fd > 0)close(fd);
}
```

然后，就是 JNI 中的 “Java\_com\_topeet\_ledtest\_led\_Ioctl” ，主要功能是向底层输入数据。

```
/*
 * Class:      com_topeet_ledtest_led
 * Method:     Ioctl
 * Signature:  ()I
 */
JNIEXPORT jint JNICALL Java_com_topeet_ledtest_led_Ioctl
(JNIEnv *env, jobject obj, jint num, jint en)
{
    ioctl(fd, en, num);
}
```

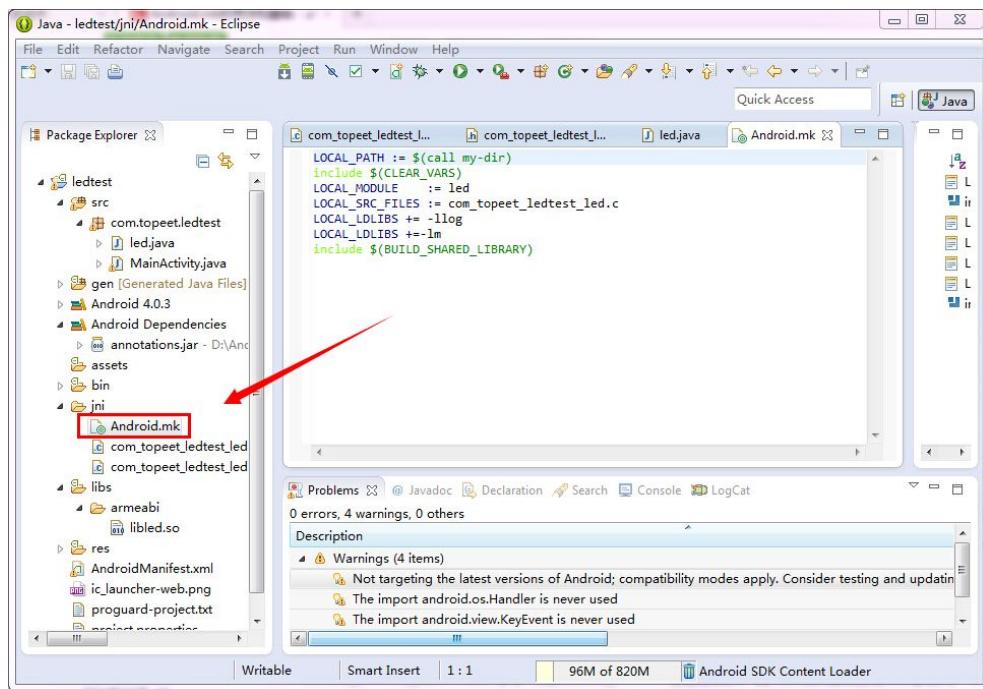
最后说明一下 “/dev/leds” 驱动源码在内核中的目录

“iTop4412\_Kernel\_3.0/drivers/char/itop4412\_leds.c” 。

## 10.6 Android.MK 文件

前面的一小节已经介绍了 “com\_topeet\_ledtest\_led.h” 和 “com\_topeet\_ledtest\_led.c” ，但是源码还需要编译成 lib.so 库才能使用。编译成库就涉及到 Makefile 文件，本节给大家简单介绍一下 Android.mk 的编写。

这里还是以 ledtest 为例，如下图。



下面是 Android.mk 文件中的代码

```

LOCAL_PATH := $(call my-dir)
include $(CLEAR_VARS)
LOCAL_MODULE      := led
LOCAL_SRC_FILES := com_topeet_ledtest_led.c
LOCAL_LDLIBS += -llog
LOCAL_LDLIBS += -lm
include $(BUILD_SHARED_LIBRARY)

```

下面我们来解析一下这几行代码。

1、LOCAL\_PATH := \$(call my-dir)

一个 Android.mk file 首先必须定义好 LOCAL\_PATH 变量。它用于查找源文件。在这个例子中，宏函数 ‘my-dir’ ,由编译系统提供，用于返回当前路径（即包含 Android.mk file 文件的目录）。

## 2、include \$(CLEAR\_VARS)

CLEAR\_VARS 由编译系统提供（可以在 android 安装目录下的/build/core/config.mk 文件看到其定义，为 CLEAR\_VARS:= \$(BUILD\_SYSTEM)/clear\_vars.mk），指定让 GNU MAKEFILE 为你清除许多 LOCAL\_XXX 变量（例如 LOCAL\_MODULE, LOCAL\_SRC\_FILES, LOCAL\_STATIC\_LIBRARIES, 等等...），除 LOCAL\_PATH。这是必要的，因为所有的编译控制文件都在同一个 GNU MAKE 执行环境中，所有的变量都是全局的。

## 3、LOCAL\_MODULE := led

LOCAL\_MODULE 变量必须定义，以标识你在 Android.mk 文件中描述的每个模块。名称必须是唯一的，而且不包含任何空格。注意编译系统会自动产生合适的前缀和后缀，换句话说，一个被命名为'led'的共享库模块，将会生成 “libled.so” 文件（也可以直接使用 libled 命名好）。

## 4、LOCAL\_SRC\_FILES := com\_topeet\_ledtest\_led.c

LOCAL\_SRC\_FILES 变量必须包含将要编译打包进模块中的 C 或 C++ 源代码文件。注意，你不用在这里列出头文件和包含文件，因为编译系统将会自动为你找出依赖型的文件；仅仅列出直接传递给编译器的源代码文件就好。

## 5、LOCAL\_LDLIBS += -llog

## 6、LOCAL\_LDLIBS += -lm

编译模块时要使用的附加的链接器选项。使用 “-l” 前缀传递指定库的名字是有用的。

LOCAL\_LDLIBS := -llog 表示告诉链接器生成的模块要在加载时刻链接到 “/system/lib/liblog.so”

可查看 docs/STABLE-APIS.TXT 获取使用 NDK 发行版能链接到的开放的系统库列表。这里表示生成的是动态库。

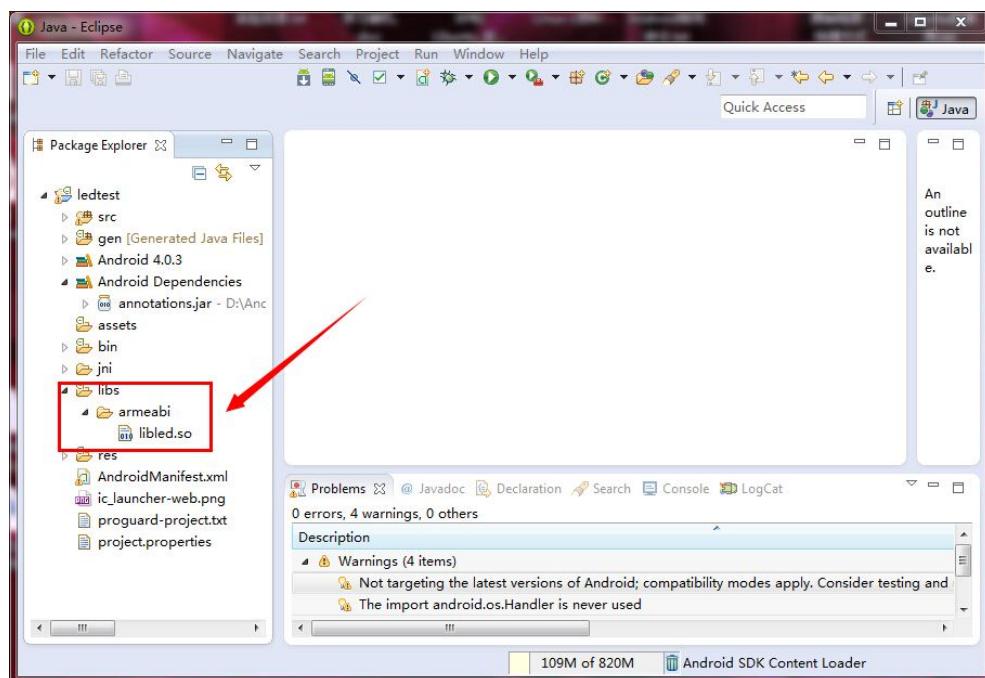
## 7、include \$(BUILD\_SHARED\_LIBRARY)

会生成依赖关系，当库不存在时会去编译这个库。

这里主要介绍的是基于 C 的 Makefile 文件的编写，如果用户使用的是 C++ 代码，Makefile 文件中的变量以及宏定义就会有所不同。

## 10.7 安装 NDK 编译器以及编译 JNI 库文件

这里给大家介绍一下如何生成基于 Linux 的 libxx.so 库。如下图所示，最后用户需要放在“libs/armeabi” 中的库文件。



### 10.7.1 安装 NDK 编译器

这里编译的方法是基于 Ubuntu12.04.2 的，首先需要安装 NDK 编译器。这个编译器压缩包 “android-ndk-r8b-linux-x86.tar.bz2” 在网盘中有提供。

将压缩包拷贝到 Ubuntu12.04.2 的 “/usr/local/ndk” 文件夹中，如果没有 ndk 文件夹，则需要用户自己建立一个 ndk 文件夹。如下图所示。

```
root@ubuntu:/usr/local/ndk# ls  
android-ndk-r8b-linux-x86.tar.bz2
```

进入新建的 ndk 目录，使用命令 “#tar -vxf android-ndk-r8b-linux-x86.tar.bz2” 将编译器解压到 ndk 目录中。如下图所示，解压完成。

```
android-ndk-r8b/build/tools/toolchain-symbols/x86/  
android-ndk-r8b/build/tools/toolchain-symbols/x86/libgcc.a.functions.txt  
android-ndk-r8b/RELEASE.TXT  
android-ndk-r8b/README.TXT  
android-ndk-r8b/GNUmakefile  
root@ubuntu:/usr/local/ndk# ls  
android-ndk-r8b  android-ndk-r8b-linux-x86.tar.bz2  
root@ubuntu:/usr/local/ndk#
```

如下图所示，使用 “#cd” 命令，确认修改的是 root 用户的环境变量，然后使用命令 “#vim .bashrc” 打开设置环境变量的文件。

```
root@ubuntu:/usr/local/ndk# ls  
android-ndk-r8b  android-ndk-r8b-linux-x86.tar.bz2  
root@ubuntu /usr/local/ndk# cd  
root@ubuntu ~# vim .bashrc
```

进入文件 “.bashrc” 的最后一行。添加环境变量

“export PATH=\$PATH:/usr/local/ndk/android-ndk-r8b”

如下图所示。

```
# enable programmable completion features (you don't need to enable  
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile  
# sources /etc/bash.bashrc).  
#if [ -f /etc/bash_completion ] && ! shopt -oq posix; then  
#  . /etc/bash_completion  
#fi  
  
export PATH=$PATH:/usr/local/ndk/android-ndk-r8b
```

保存退出，然后使用命令 “##source .bashrc” 更新环境变量。

```
root@ubuntu:/usr/local/ndk# ls
android-ndk-r8b  android-ndk-r8b-linux-x86.tar.bz2
root@ubuntu:/usr/local/ndk# cd
root@ubuntu:~# vim .bashrc
root@ubuntu:~# source .bashrc
root@ubuntu:~#
```

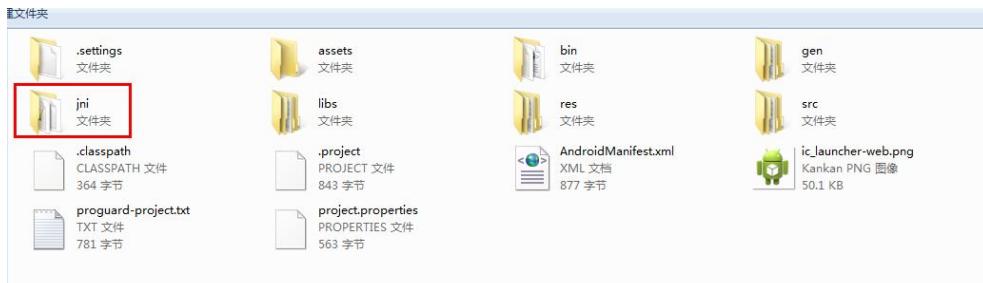
然后测试一下环境变量是否安装成功。输入 “#ndk” , 然后按 “Tab” 键 , 如下图所示 , 则设置成功。

```
root@ubuntu:~# source .bashrc
root@ubuntu:~# ndk-
ndk-build  ndk-gdb    ndk-stack
root@ubuntu:~# ndk-
```

## 10.7.2 编译 Android 动态链接库

下面以 Android 应用程序的 “ledtest” 工程为例来讲解生成动态链接库的方法。

如下图所示 , 工程 “ledtest” 中的 JNI 文件夹中有编译动态链接库需要用到的源代码以及编译脚本。



将工程中的 “jni” 文件夹拷贝到 Ubuntu12.04.2 的 “/home/topeet/Android-app” 目录中 , 如果没有 “Android-app” 目录 , 则可以新建一个 , 如下图所示。

```
root@ubuntu:/home/topeet/Android-app# ls
jni
root@ubuntu:/home/topeet/Android-app#
```

如下图所示 , 进入 jni 目录 , 使用命令 “#ndk-build” 编译。

```
root@ubuntu:/home/topeet/Android-app# cd jni  
root@ubuntu:/home/topeet/Android-app/jni# ndk-build  
Compile thumb : led <= com_topeet_ledtest_led.c  
SharedLibrary : libled.so  
Install       : libled.so => libs/armeabi/libled.so  
root@ubuntu:/home/topeet/Android-app/jni#
```

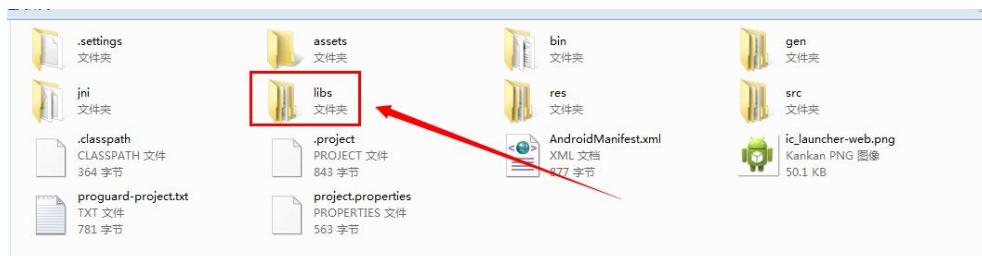
返回上一级目录，使用命令 “#ls” 查看，可以看到生成了文件夹 “libs” 和 “obj”。

```
root@ubuntu:/home/topeet/Android-app/jni# cd ..  
root@ubuntu:/home/topeet/Android-app# ls  
jni  libs  obj  
root@ubuntu:/home/topeet/Android-app#
```

使用命令 “#cd libs” 进入 “libs” 目录，然后使用 “#ls” 命令查看，可以发现生成了文件夹 “armeabi” ，里面有库文件 “libled.so”。

```
root@ubuntu:/home/topeet/Android-app# cd libs/  
root@ubuntu:/home/topeet/Android-app/libs# ls  
armeabi  
root@ubuntu:/home/topeet/Android-app/libs#
```

编译生成的 “libs” 和工程 “ledtest” 中的 “libs” 文件夹是对应的。



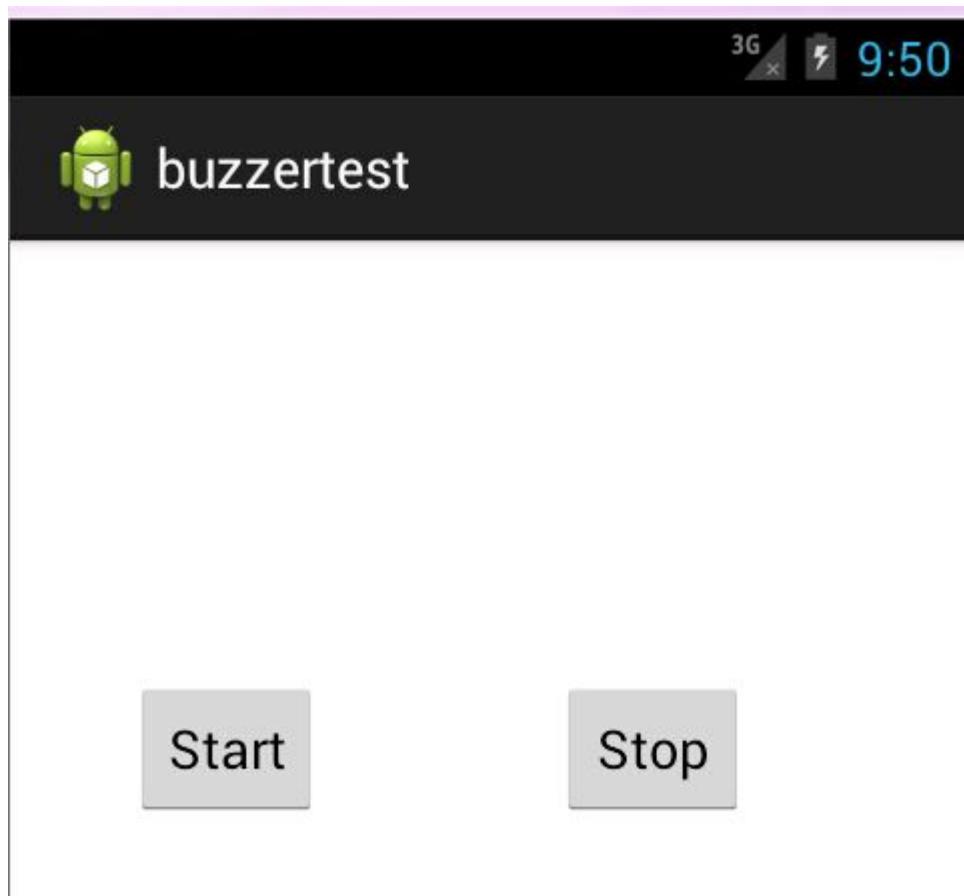
## 10.8 其他常用 Android 应用程序

本节着重介绍其它几个常用的嵌入式 Android 涉及到的应用程序，大家要注意的是，Android 已经有很多集成了的应用，这里就不再做介绍了。用户可以在“第三章”中找到更多集成的应用程序。

### 10.8.1 Buzzer 应用程序

用户可以参考 ledtest 的方法，导入、分析以及移植 buzzertest 工程。

和开发板调试时，屏幕界面如下图所示。



上层应用调用 JNI 接口为 “class buzzer ,” 在

"src/com/topeet/buzzertest/MainActivity/buzzer" , 如下图所示。

```
class buzzer {  
    public native int      Open();  
    public native int      Close();  
    public native int      Ioctl(int num, int en);  
}
```

调用的 buzzer 设备节点为 “/dev/buzzer\_ctl” , 打开文件

"jni/ com.topeet.buzzertest\_buzzer.c" , 如下图所示。

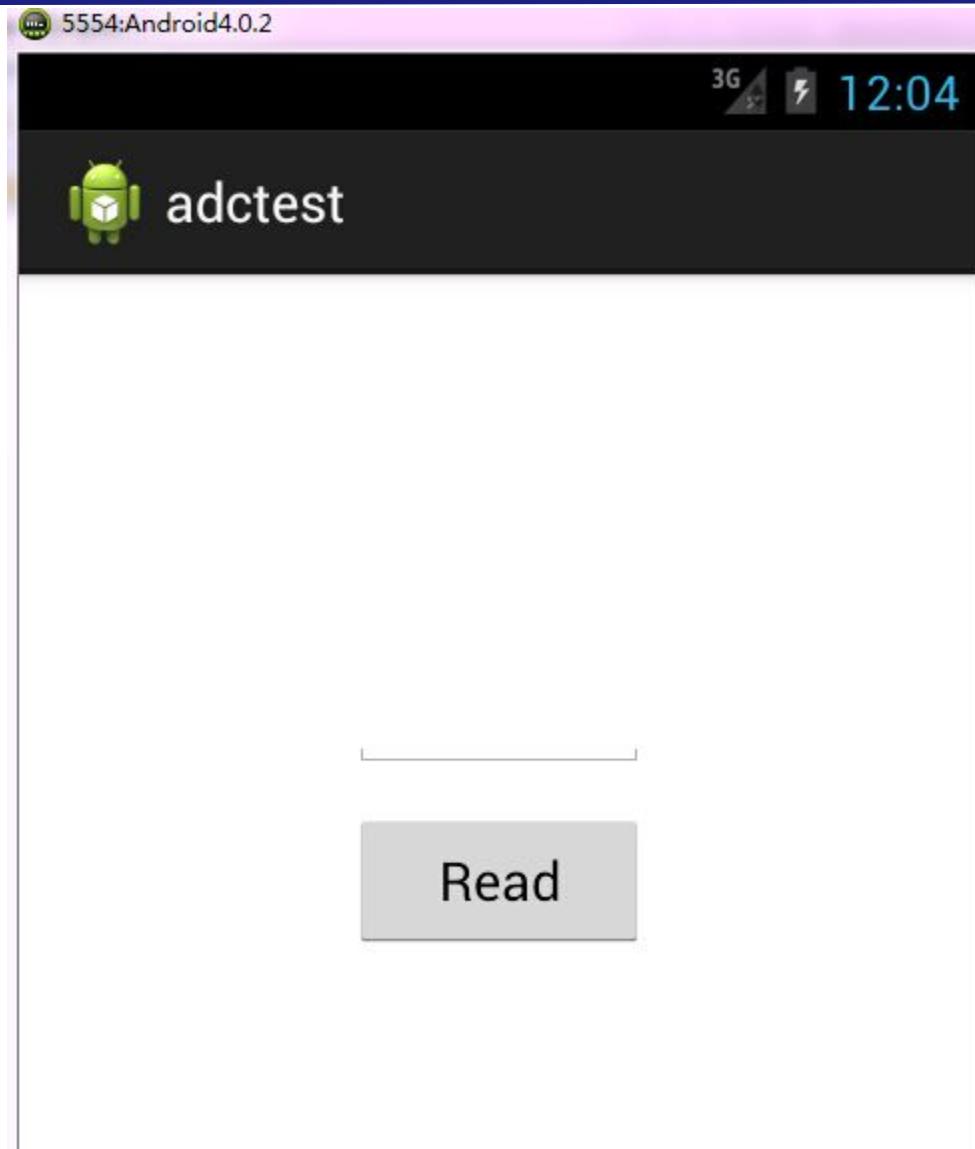
```
/*  
 * JNIEEXPORT jint JNICALL Java_com_topeet_buzzertest_buzzer_Open  
 * (JNIEnv *env, jobject obj)  
{  
 *     if(fd<=0)fd = open("/dev/buzzer_ctl", O_RDWR|O_NDELAY|O_NOCTTY);  
 *     if(fd <=0 )_android_log_print(ANDROID_LOG_INFO, "serial", "open /dev/buzzer_ctl Error");  
 *     else _android_log_print(ANDROID_LOG_INFO, "serial", "open /dev/buzzer_ctl Sucess fd=%d",fd);  
 * }  
 */
```

内核驱动源码位置为 “/drivers/char/itop4412\_buzzer.c” 。

### 10.8.2 ADC 应用程序

用户可以参考 ledtest 的方法，导入、分析以及移植 adctest 工程。

和开发板调试时，屏幕界面如下图所示。



上层应用调用 JNI 接口为 “class adc” ,在"src/com.topeet.adctest/adc.java" , 如下图所示。

```
public class adc {  
    public native int      Open();  
    public native int      Close();  
    public native int      Ioctl(int num, int en);  
    public native int[]   Read();  
}
```

调用的 buzzer 设备节点为 “/dev/adc” , 打开文件" jni/com.topeet.adc\_adc.c" , 如下图所示。

```
JNIEXPORT jint JNICALL Java_com_topeet_adctest_adc_Open
(JNIEnv *env, jobject obj)
{
    if(fd<=0)fd = open("/dev/adc" O_RDWR|O_NDELAY|O_NOCTTY);
    if(fd <=0 )__android_log_print(ANDROID_LOG_INFO, "serial", "open /dev/adc Error");
    else __android_log_print(ANDROID_LOG_INFO, "serial", "open /dev/adc Sucess fd=%d",fd);
}
```

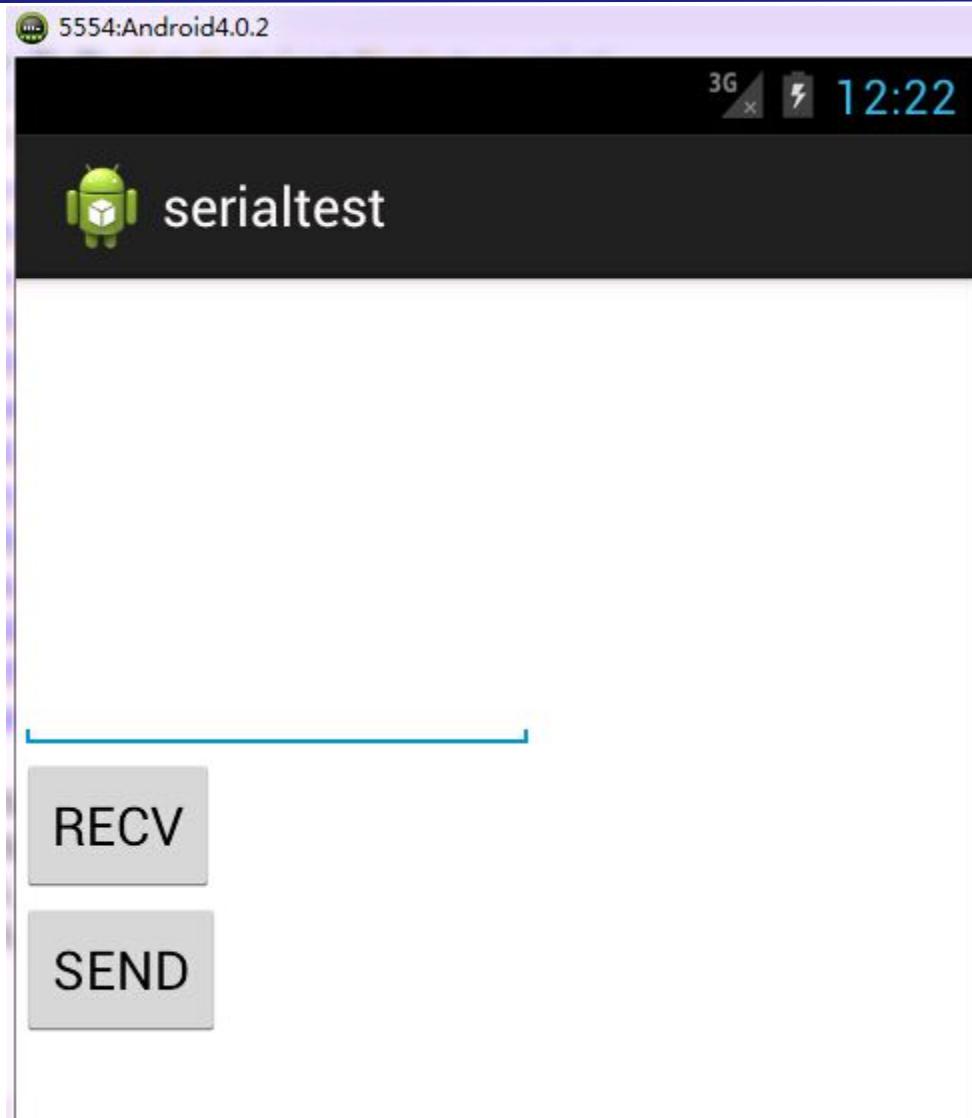
内核驱动源码位置为 “/drivers/char/itop4412\_adc.c”

## 10.8.3 串口应用程序

### 10.8.3.1 串口应用程序

用户可以参考 ledtest 的方法，导入、分析以及移植 serialtest 工程。

和开发板调试时，屏幕界面如下图所示。



上层应用调用 JNI 接口为 “class serial” , 在  
"src/com/topeet/serialtest/serial.java" , 如下图所示。

```
public class serial {  
  
    public native int Open(int Port,int Rate);  
    public native int Close();  
    public native int[] Read();  
    public native int Write(int[] buffer,int len);  
  
}
```

调用的串口设备节点为 “/dev/ttySAC3” , 打开文件

“serialtest\src\com\topeet\serialtest” 的文件 “MainActivity.java” , 如下图所示。

```
ETT1 = (EditText) findViewById(R.id.edit1);  
RECV = (Button) findViewById(R.id.recv1);  
SEND = (Button) findViewById(R.id.send1);
```

```
com3.Open(3, 115200);
```

```
RECV.setOnClickListener(new manager());
```

```
SEND.setOnClickListener(new manager());
```

### 10.8.3.2 串口应用移植需要注意的问题

下面给大家介绍一下在 Android 操作系统中串口的移植需要注意的几个方面：

1 ) 系统启动后，在超级终端中输入命令 "#ls dev/tty\*" , 如下图，红色方框中就是开发板的四个串口设备节点，这个测试程序中使用的是 “/dev/ttySAC2” 。

```
root@android:/ # ls /dev/tt*  
/dev/tty  
/dev/ttyGS0  
/dev/ttyGS1  
/dev/ttyGS2  
/dev/ttyGS3  
/dev/ttyS0  
/dev/ttyS1  
/dev/ttyS2  
/dev/ttyS3  
/dev/ttySAC0  
/dev/ttySAC1  
/dev/ttySAC2  
/dev/ttySAC3  
root@android:/ #
```

2 ) 如果想使用别的串口，则需要修改 Android 串口应用的源码。打开应用的源码文件夹 “serialtest\src\com\topeet\serialtest” 的文件 “MainActivity.java” , 搜索 “com3.Open” , 这里使用的串口设备节点是 “/dev/ttySAC3” , 对应的是 CON2 接口。如果需要使用别的串口，则需要根据实际情况，修改代码。

```

ET1 = (EditText) findViewById(R.id.edit1);
RECV = (Button) findViewById(R.id.recv1);
SEND = (Button) findViewById(R.id.send1);

```

```
com3.Open(3, 115200);
```

```

RECV.setOnClickListener(new manager());
SEND.setOnClickListener(new manager());

```

3 ) 打开文件夹 “serialtest\jni” 中的文件 “com\_topeet\_serialtest\_serial.c” , 如下图 , 找到函数 “JNIEXPORT jint JNICALL Java\_com\_topeet\_serialtest\_serial\_Open” 。在 “#if 1”

到 “#endif” 中的代码 , 有需要设置的参数 , 这些参数都是通用的 , 可以直接根据需要设置。不过在这个例程中 , 除了波特率 , 其它都是缺省状态。

```

        return -1;
    }
#if 1
    if(fd > 0)
    {
        __android_log_print(ANDROID_LOG_INFO, "serial", "serial open fd=%d", fd);
        // disable echo on serial ports
        struct termios ios;
        tcgetattr(fd, &ios);
        ios.c_oflag &=~(INLCR|IGNCR|ICRNL);
        ios.c_oflag &=~(ONLCR|OCRNL);
        ios.c_iflag &=~(ICRNL | IXON);
        ios.c_iflag &=~(INLCR|IGNCR|ICRNL);
        ios.c_iflag &=~(ONLCR|OCRNL);
        tcflush(fd,TCIFLUSH);

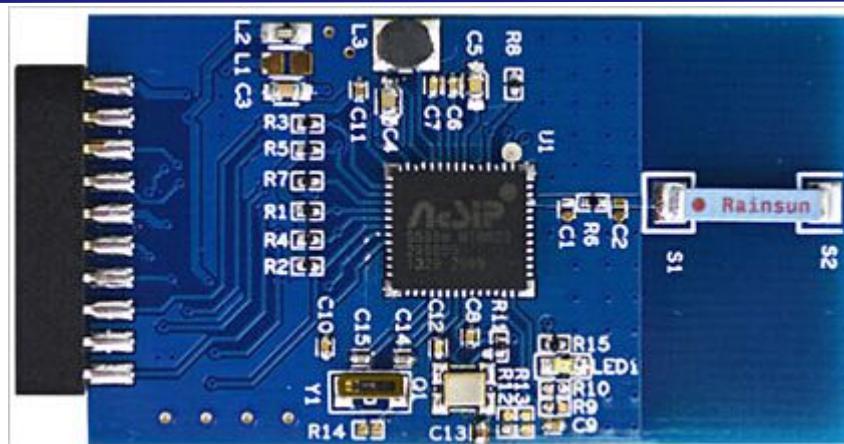
        if(Rate == 2400){cfsetospeed(&ios, B2400); cfsetispeed(&ios, B2400);}
        if(Rate == 4800){cfsetospeed(&ios, B4800); cfsetispeed(&ios, B4800);}
        if(Rate == 9600){cfsetospeed(&ios, B9600); cfsetispeed(&ios, B9600);}
        if(Rate == 19200){cfsetospeed(&ios, B19200); cfsetispeed(&ios, B19200);}
        if(Rate == 38400){cfsetospeed(&ios, B38400); cfsetispeed(&ios, B38400);}
        if(Rate == 57600){cfsetospeed(&ios, B57600); cfsetispeed(&ios, B57600);}
        if(Rate == 115200){cfsetospeed(&ios, B115200); cfsetispeed(&ios, B115200);}

        ios.c_cflag |= (CLOCAL | CREAD);
        ios.c_cflag &= ~PARENB;
        ios.c_cflag &= ~CSTOPB;
        ios.c_cflag &= ~CSIZE;
        ios.c_cflag |= CS8;
        ios.c_lflag = 0;
        tcsetattr(fd, TCSANOW, &ios );
    }
#endif
}

```

## 10.8.4 蓝牙应用

配置了蓝牙模块后 , 接到开发板上 , 就可以使用蓝牙功能了。



进入 Android 文件系统，如下图所示（此图是 AVD 的界面，用户以实际开发板屏幕为准），在“设置”中，可以设置蓝牙开关。



如果用户需要操作蓝牙，则只需要掌握对应的蓝牙命令即可。

下面给大家介绍几个常用的命令，并演示使用命令进行连接。下面介绍的方法是，启动开发板之后，在超级终端中输入命令。

### 查询命令

所有的指令都可以在找到。

#hciconfig --help

```
root@android:/data # hciconfig --help
hciconfig - HCI device configuration utility
Usage:
      hciconfig
      hciconfig [-a] hciX [command ...]
Commands:
      up                  Open and initialize HCI device
      down                Close HCI device
      reset               Reset HCI device
      rstat               Reset statistic counters
      auth                Enable Authentication
      noauth              Disable Authentication
      encrypt             Enable Encryption
      noencrypt           Disable Encryption
      piscan              Enable Page and Inquiry scan
```

### 开启/关闭蓝牙

#hciconfig hciX up(down)

这里蓝牙标号为 hci0

开启蓝牙# hciconfig hci0 up

```
root@android:/data # hciconfig hci0 up
[ 3992.253811] [HCI-STP] [I]hci_stp_open:hci0(0xd4994800)
[ 3992.257533] [WMT-EXP] [I]mtk_wcn_wmt_func_ctrl:OPID(3) type(0)
start
[ 3992.263752] [STP-C] [W]mtk_wcn_stp_psm_disable:STP Not Ready,
Dont do Sleep/Wakeup
```

关闭蓝牙# hciconfig hci0 down

```
root@android:/data # hciconfig hci0 down
[ 4075.133584] [STP-C] [I]mtk_wcn_stp_send_data:#####Type = 0, to
inform WMT to wakeup chip, ret = 4
[ 4075.141398] [PSM] [I]_stp_psm_set_state: work_state = INACT --
> INACT_ACT
[ 4075.147746] and = COMBO_TF_HADM to option down idle=0
```

准备一个带有 Android 蓝牙接口的手机，开启手机的蓝牙功能（下图显示的是 U980），同时也开启开发板上的蓝牙。

### 使用查询命令

```
#hcitool scan
```

```
root@android:/data # hcitool scan
Scanning ...[ 4392.184369] [STP-C][I]mtk_wcn_stp_send_data:#####
Type = 0, to inform WMT to wakeup chip, ret = 9
[ 4392.192585] [PSM][I]_stp_psm_set_state: work_state = INACT --
> INACT_ACT
[ 4392.199217] src = COMBO_IF_UART, to enter deep idle? 0
```

### 查询结果

```
[ 4402.483207] [PSM][I]_stp_psm_set_state: work_state = INACT_AC
T --> ACT
AC:E2:15:13:F3:5A      U9508
root@android:/data # [ 4407.500099] [PSM][I]_stp_psm_stp_is_idle
: **IDLE is over 5000 msec, go to sleep!!!**
```

这里需要注意的是，蓝牙开启后只能在一段时间内检测到以及连接。

### 绑定手机蓝牙的硬件地址

#### 使用绑定命令

```
#rfcomm bind /dev/rfcomm0 AC:E2:15:13:F3:5A
```

```
root@android:/data # rfcomm bind /dev/rfcomm0  AC:E2:15:13:F3:5A
root@android:/data #
```

#### 解绑命令

```
#rfcomm release /dev/rfcomm0 AC:E2:15:13:F3:5A
```

```
root@android:/data # rfcomm release /dev/rfcomm0 AC:E2:15:13:F3:5A
```

## 10.8.5 485 应用

用户在网盘可以下载 Android 的 485 源码应用，具体使用方法和串口类似，这里不再重复。

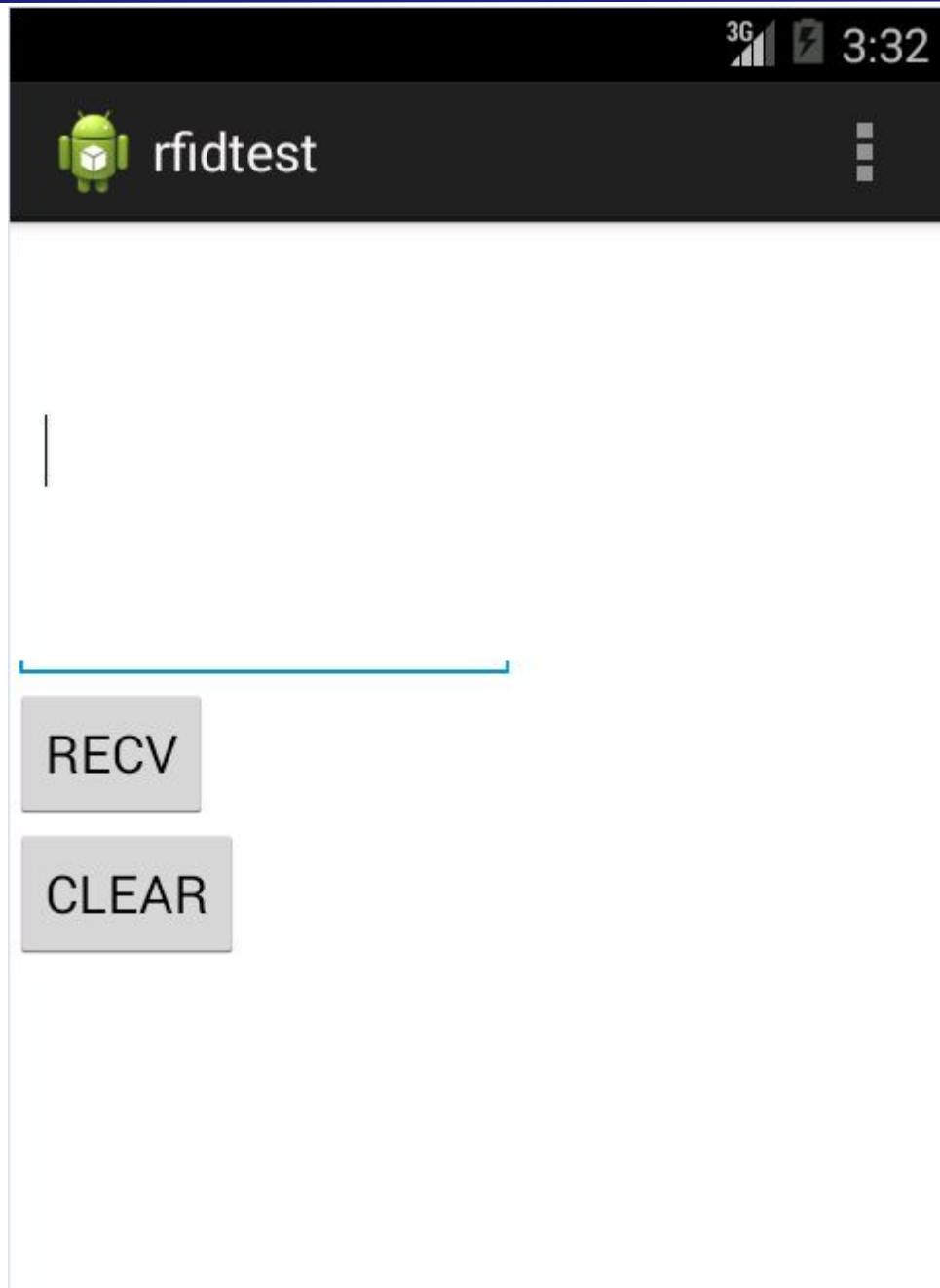
### 10.8.6 RFID 的应用

用户在网盘下载 Android 的 RFID 源码。

如下图所示，是迅为电子的 RFID 模块。模块和底板连接的时候，通过一个简单的转接板，地板上对应是 GPIO 口（靠近耳机插座的那个接口），模块和转接板都有丝印，注意不要插反。



连接好之后，Android 的测试界面如下，将提供的白卡放到 RFID 模块的无线识别区，然后按下图中的“RECV”，即可读出卡号。



## 10.9 Android 文件系统源码修改

有些 Android 功能配置需要在源码中修改，这里介绍几个常见的。

### 10.9.1 更改默认休眠时间

Android 默认休眠时间是可以修改的，下面以将开发板休眠时间修改成无限长为例来讲解具体怎么操作。

如下图，打开 Android 源码中的文件

“frameworks/base/packages/SettingsProvider/res/values/defaults.xml”

```
root@ubuntu:/home/topeet/Android/iTop4412_ICS# vim frameworks/base/packages/SettingsProvider/res/values/defaults.xml
```

找到 “<integer name="def\_screen\_off\_timeout">600000</integer>” 这一行，这里单位是 ms，下图中是 10 分钟后休眠，如果改成-1 则不休眠。

```
<resources>
    <bool name="def_dim_screen">true</bool>
    <integer name="def_screen_off_timeout">600000</integer>
    <bool name="def_airplane_mode_on">false</bool>
    <!-- Comma-separated list of bluetooth, wifi, and cell. -->
```

### 10.9.2 去掉默认安装的 APK

在 Android 文件系统中，默认安装了一些应用程序，如果有不需要的可以将其去掉，本小节和下一小节结合了来使用。

如下图所示，在 Android 源码目录 “device/samsung/smdk4x12/apk” 中，有一些默认的应用程序。

```
root@ubuntu:/home/topeet/Android/iTop4412_ICS# ls device/samsung/smdk4x12/apk/topeet/
360_assistant_1.9.160.apk  buzzer  sina_weibo          tencent_brower  weather
adc                      ledtest  storm             tencent_video
angrybirds                qq      tencent_2.8.0.apk  UvcWebCam
```

### 10.9.3 将 APK 编译到 Android 镜像中

上一小节中，讲到如何去掉默认安装的 APK，如果用户需要将 APK 编译到 Android 镜像中，除了需要将 APK 拷贝到“device/samsung/smdk4x12/apk”目录中，还需要添加安装脚本。

如下图所示，安装 APK 的脚本是“device/samsung/smdk4x12/device.mk”，我们以 led 为例

```
root@ubuntu:/home/topeet/Android/iTop4412_ICS# vim device/samsung/smdk4x12/device.mk
```

如下图所示，这两行脚本是与 led 相关的。首先将 ledtest.apk 放到对应的文件夹中。其次，ledtest 需要一个库文件，这个库文件也要放到对应的文件夹中。

如果是其它的 APK，可以参照 ledtest 的脚本。

```
#Install LedTest
PRODUCT_COPY_FILES += \
    device/samsung/smdk4x12/apk/topeet/ledtest/ledtest.apk:system/app/ledt.apk
PRODUCT_COPY_FILES += \
    device/samsung/smdk4x12/apk/topeet/ledtest/libled.so:system/lib/libled.o
```

### 10.9.4 手机和平板模式（横屏竖屏）

在 Android 系统中，有手机和平板两种模式。可以通过修改 Android 代码，改变开发板的模式。

如下图所示，切换手机模式和平板模式的文件是

“device/samsung/smdk4x12/BoardConfig.mk” , 打开该文件。

```
root@ubuntu:/home/topeet/Android/iTop4412_ICS# vim device/samsung/smdk4x12/BoardConfig.mk
```

找到文件中的参数 “BOARD\_USES\_HIGH\_RESOLUTION\_LCD” , 如下图所示 , 如果变量设置为 “true” , 则是对应平板模式 , 如果是 “false” , 则对应手机模式

```
BOARD_USE_S3D_SUPPORT := true  
BOARD_USE_CSC_FIMC := false  
  
BOARD_USES_HIGH_RESOLUTION_LCD := true  
  
BOARD_USES_FFMPEG := false
```

## 10.9.5 设置有线网

Android 默认是使用 WIFI , 如果想使用有线网络 , 则需要配置一下。

具体参考 “3.4 小节”

## 10.9.6 设置 Android 的 GPS

开发板支持两种 GPS 模块 , 分别为 GNS7560 和 UBLOX , 下面讲解一下代码里面怎么选择对应的驱动 :

这两种 GPS 模块都是通过串口来传输数据的 , linux 内核里面已经支持串口的驱动了 , 所以我们需要修改的只是 android 代码 , 通过配置对应的宏来选择支持我们使用的 GPS , 进入到 “iTop4412\_ICS” android 的源码目录 , 如下图 :

```
root@ubuntu:/home/broswer/iTop4412_ICS#  
root@ubuntu:/home/broswer/iTop4412_ICS# ls  
abi      build_android.sh  device    hardware  out       setenv  
bionic   cts              docs      libcore   packages  system  
bootable dalvik           external  Makefile  prebuilt v8.log  
build    development      frameworks ndk       sdk  
root@ubuntu:/home/broswer/iTop4412_ICS#  
root@ubuntu:/home/broswer/iTop4412_ICS#  
root@ubuntu:/home/broswer/iTop4412_ICS#
```

然后输入 “vi device/samsung/smdk4x12/BoardConfig.mk” 命令，如下图：

```
root@ubuntu:/home/broswer/iTop4412_ICS#  
root@ubuntu:/home/broswer/iTop4412_ICS#  
root@ubuntu:/home/broswer/iTop4412_ICS#  
root@ubuntu:/home/broswer/iTop4412_ICS#  
root@ubuntu:/home/broswer/iTop4412_ICS# vi device/samsung/smdk4x12/BoardC  
onfig.mk
```

然后在 BoardConfig.mk 文件里面找到 “BOARD\_HAVE\_GNS7560 := false” ，如下

图：

```
#add by cym 20150305  
BOARD_HAVE_GNS7560 := false  
#end add
```

如果使用 GNS7560 模块，需要把这行改成：

```
BOARD_HAVE_GNS7560 := true
```

如果使用 UBLOX 模块，需要把这行改成：

```
BOARD_HAVE_GNS7560 := false
```

修改完成以后，保存并退出。然后在终端依次输入下面的两条编译命令：

make clobber

./build\_android.sh

重新编译 android 就可以了，最后会生成的 Android 镜像就可以支持对应的 GPS。

### 10.9.7 设置 Android 的 HDMI 转 VGA

在 Android 源码目录 ( iTop4412\_ICS ) 下打开文件 :

device/samsung/smdk4x12/BoardConfig.mk , 找到里面的 “BOARD\_HDMI\_STD := STD\_1080P” , 如下图 :

```
BOARD_USES_HDMI_SUBTITLES := false
BOARD_USES_HDMI := true
BOARD_HDMI_STD := STD_1080P
BOARD_HDMI_DDC_CH := DDC_CH_I2C_0
BOARD_USES_FIMGAPI := true
```

然后把这一行修改为 : ” BOARD\_HDMI\_STD := STD\_480P\_TOPEET ” , 如下图所示 :

```
BOARD_USES_HDMI_SUBTITLES := false
BOARD_USES_HDMI := true
BOARD_HDMI_STD := STD_480P_TOPEET
BOARD_HDMI_DDC_CH := DDC_CH_I2C_0
BOARD_USES_FIMGAPI := true
```

然后退出并保存。如果以前编译过 Android , 需要执行下面的命令删除编译生成的一些中间文件 , 如下图所示 :

```
iTop4412_ICS#
iTop4412_ICS# rm -rf out/target/product/smdk4x12/obj
```

然后执行 “./build\_android.sh” 编译 Android , 如下图所示 :

```
iTop4412_ICS#
iTop4412_ICS# ./build_android. sh
```

编译完成后 , 最后把生成的镜像烧写到开发板上。

## 10.9.8 设备权限的修改

在 Android 下调用设备内核驱动的设备节点的时候，会遇到没有权限的问题，这个时候需要修改一下设备权限。下面以串口为例介绍一下，其它的设备修改方法也类似。

我们在 android 下操作串口有时会遇到没有权限的问题，这就需要修改下 android 的启动脚本，在里面修改下串口的权限。

具体修改方法是：

在 android 源码目录下输入 “vi device/samsung/smdk4x12/conf/init.smdk4x12.rc” ，如下图所示：

```
root@ubuntu:/home/broswer/iTop4412_ICS#  
root@ubuntu:/home/broswer/iTop4412_ICS# vi device/samsung/smdk4x12/conf/init.smdk4x12.rc
```

在里面找到修改权限的地方，如下图所示：

```
#add by cym 20130305  
    chmod 0777 /linuxrc  
    chmod 0777 /dev/adc  
    chmod 0777 /dev/buzzer_ctl  
    chmod 0777 /dev/max485_ctl_pin  
    chmod 0777 /dev/leds  
    chmod 0777 /dev/rc522
```

上面的 “chmod 777 xxxx” 就是修改设备节点的权限，比如我们现在想修改串口 0 ( /dev/ttYSAC0 ) 的权限，那我们在这下面输入 “chmod 777 /dev/ttYSAC0” 就可以了，如下图所示：

```
#add by cym 20130305
    chmod 0777 /linuxrc
    chmod 0777 /dev/adc
    chmod 0777 /dev/buzzer_ctl
    chmod 0777 /dev/max485_ctl_pin
    chmod 0777 /dev/leds
    chmod 0777 /dev/rc522
    chmod 0777 /dev/ttySAC0
```

其他几个串口的修改方法也是这样的，修改完以后，保存并退出，回到 android 源码的目录下面，如下图所示：

```
root@ubuntu:/home/broswer/iTop4412_ICS#
root@ubuntu:/home/broswer/iTop4412_ICS#
root@ubuntu:/home/broswer/iTop4412_ICS#
root@ubuntu:/home/broswer/iTop4412_ICS#
root@ubuntu:/home/broswer/iTop4412_ICS#
```

然后输入 “./build\_android.sh ” 开始编译 android，如下图所示：

```
root@ubuntu:/home/broswer/iTop4412_ICS#
root@ubuntu:/home/broswer/iTop4412_ICS#
root@ubuntu:/home/broswer/iTop4412_ICS#
root@ubuntu:/home/broswer/iTop4412_ICS# ./build_android.sh
```

编译完成后，把生成的 “ramdisk-uboot.img” 和 “system.img” 烧到开发板里面，重新启动 android，就可以看到串口 0 ( /dev/ttySAC0 ) 的权限修改了。

# 十一 QtE 应用开发入门指南

Qt 是一个跨平台应用程序和 UI 开发框架。使用 Qt 只需要一次性的开发应用程序，不需要重新编写源代码，便可跨不同桌面和嵌入式操作系统运行这些应用程序。

Qt Creator 是全新的跨平台 Qt IDE，可以单独使用，也可以与 Qt 库和开发工具组成一套完整的 SDK。其中包括：高级 C++ 代码编辑器、项目和生成管理工具、集成的上下文相关的帮助系统等等。目前迅为电子使用的版本是“qtcreator-3.2.2”，用户可以去迅为的网盘下载对应的“.run”格式安装包，也可以去官网下载最新更新的版本。

本章节，给用户介绍一下基础软件的安装和 QtE 应用的入门。注意，开发环境是基于“qtcreator-3.2.2”（Ubuntu12.04.2），库文件是“Qt/E4.7.1”。

## 11.1 Qt 的下载和安装

这一节不会涉及到代码，从 Qt 和 Qt Creator 的下载与安装介绍，开始学习 Qt。迅为电子通过提供给用户的开发平台是 Ubuntu12.04.2，所以下面主要讲解 Ubuntu 版本的 Qt Creator。

### 11.1.1 下载软件

为了避免由于开发环境的版本差异导致的不必要的问题，推荐用户下载和使用手册中安装的相同版本“qtcreator-3.2.2”。用户有两种方式可以下载。这里需要注意的是，如果用户想要在上位机上调试，则需要下载对应环境的插件。我们网盘提供的是“qt-opensource-linux-

x64-android-5.3.2.run" , 里面包含了在 Ubuntu 中运行的插件，也就是编译出来的应用程序，可以在 Ubuntu12.04.2 中运行。然后经过简单的移植，就可以在开发板上运行。

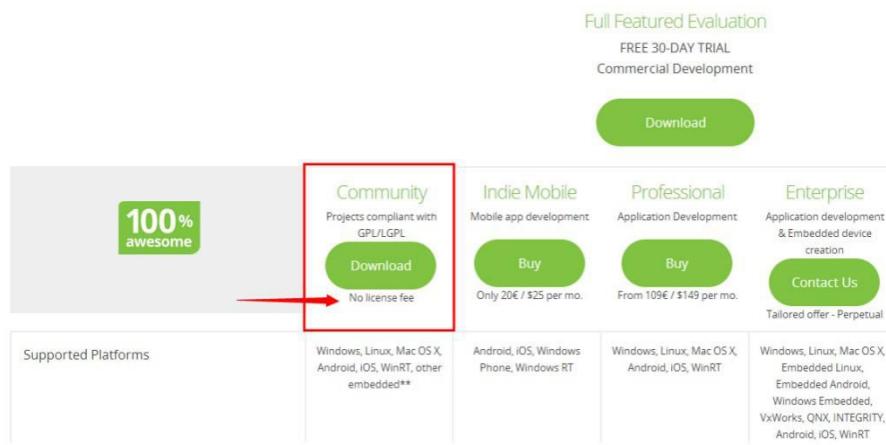
### 11.1.1.1 网盘下载

一种方法，在百度网盘 “iTOP-4412 开发板搭建编译环境所需要的工具包以及补丁包” → “07-Qt\_Creator” 中下载文件夹 “QtE\_IDE3.2.2” 。该文件夹中有 “qt-opensource-linux-x64-android-5.3.2.run” ，经过测试，可以在 Ubuntu12.04.2 中正常运行。

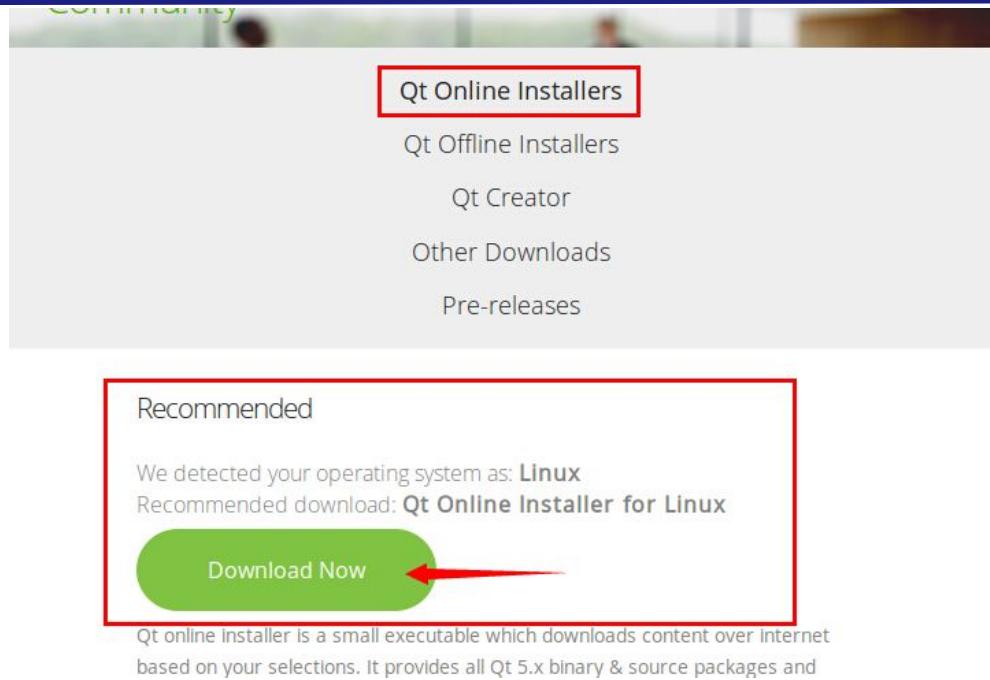
### 11.1.1.2 官网下载

另外一种方法，去官网下载。下载载地址 “<http://qt-project.org/downloads#qt-creator>” 。

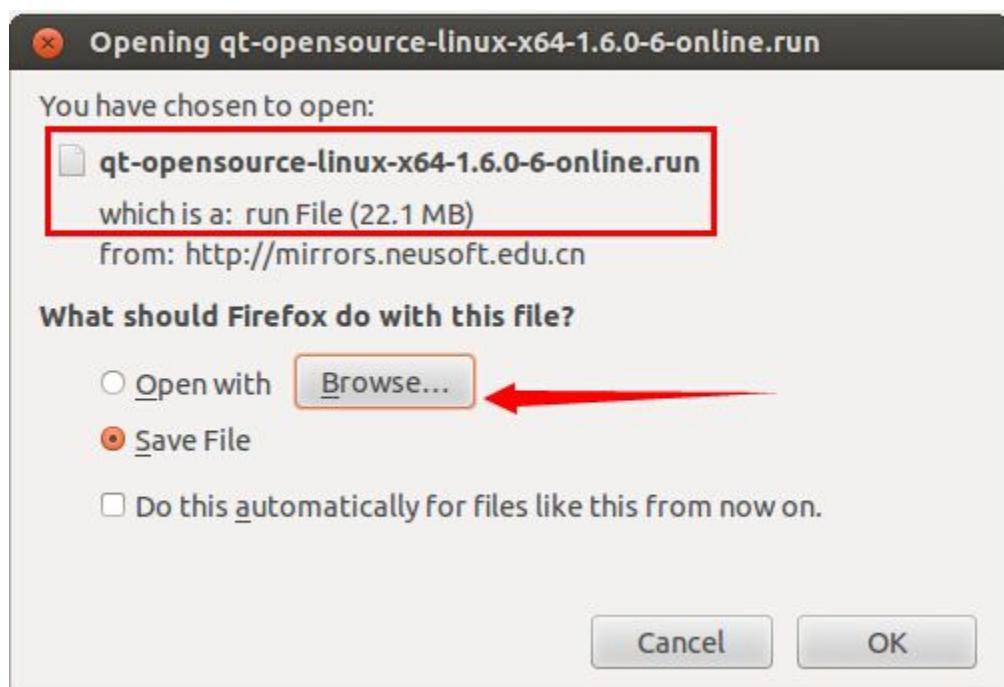
下面先介绍一下，如何使用官网链接下载。使用 Ubuntu 的浏览器登录上面的下载地址



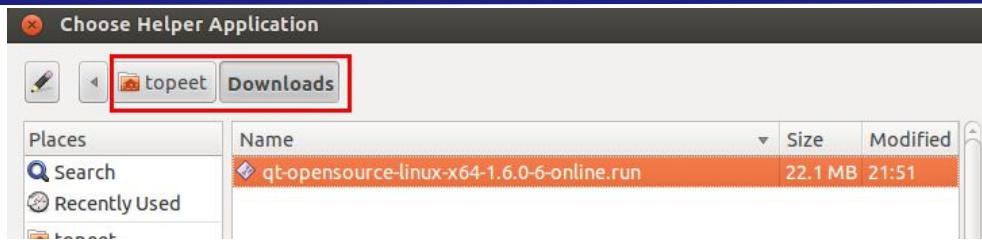
进入如下图所示页面，选择在线自动安装。也可以选择 Qt Offline Installers，把安装软件下载后，就和网盘下载的安装方法一模一样了。



如下图，下载完毕，查看下载后文件存储的路径。



如下图，在目录“topeet/Downloads”下。如果离线安装，软件的大小应该不止  
22MB。



### 11.1.2 安装 Qt Creator

下面介绍一下 Qt Creator 的安装。

这里需要注意的是，本小节介绍的是离线安装的方法。由于外网不太稳定，不推荐在线安装。

如下图，打开终端，进入 “/home/topeet/Downloads” 目录。注意，这里是以笔者的系统为例，把 “qt-opensource-linux-x64-android-5.3.2.run” 放在 “/home/topeet/Downloads” 目录了。

```
root@ubuntu:/home/topeet# cd Downloads  
root@ubuntu:/home/topeet/Downloads# ls  
qt-opensource-linux-x64-android-5.3.2.run  
root@ubuntu:/home/topeet/Downloads#
```

修改该文件的权限

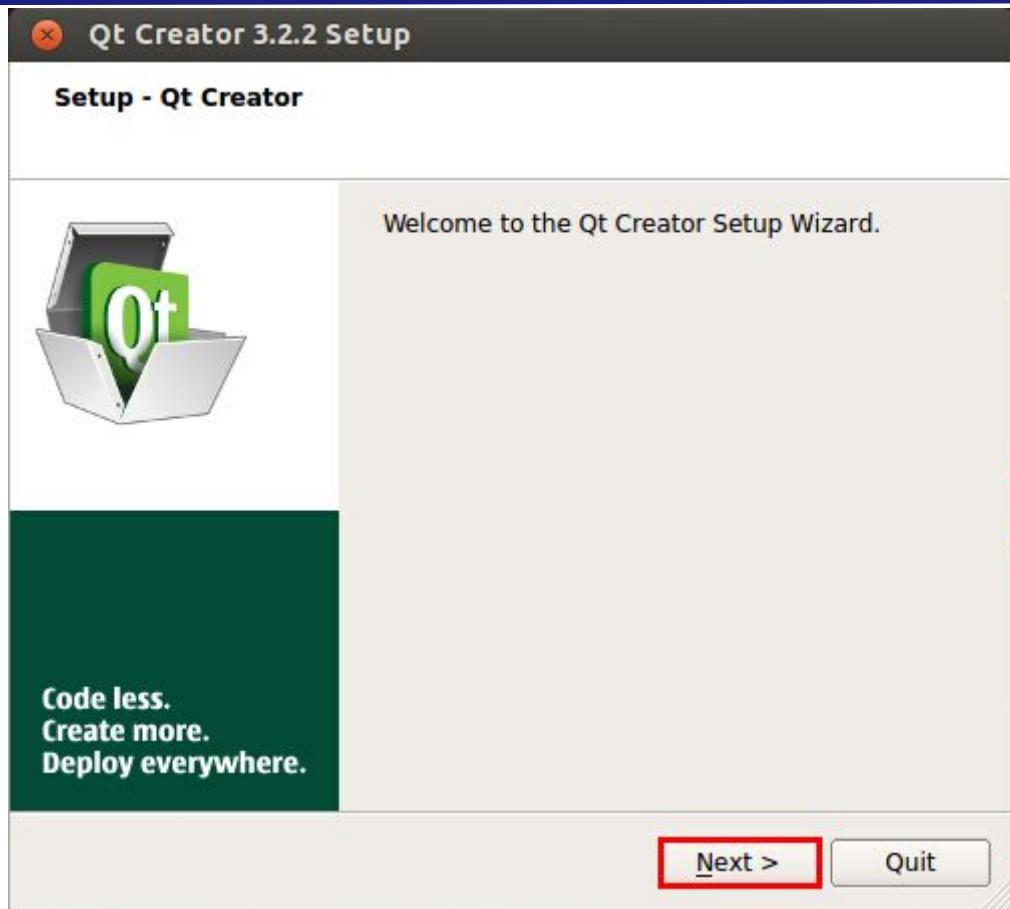
```
#chmod 777 qt-opensource-linux-x64-android-5.3.2.run
```

```
root@ubuntu:/home/topeet/Downloads#  
root@ubuntu:/home/topeet/Downloads# chmod 777 qt-opensource-li  
nux-x64-android-5.3.2.run  
root@ubuntu:/home/topeet/Downloads# ls  
qt-opensource-linux-x64-android-5.3.2.run
```

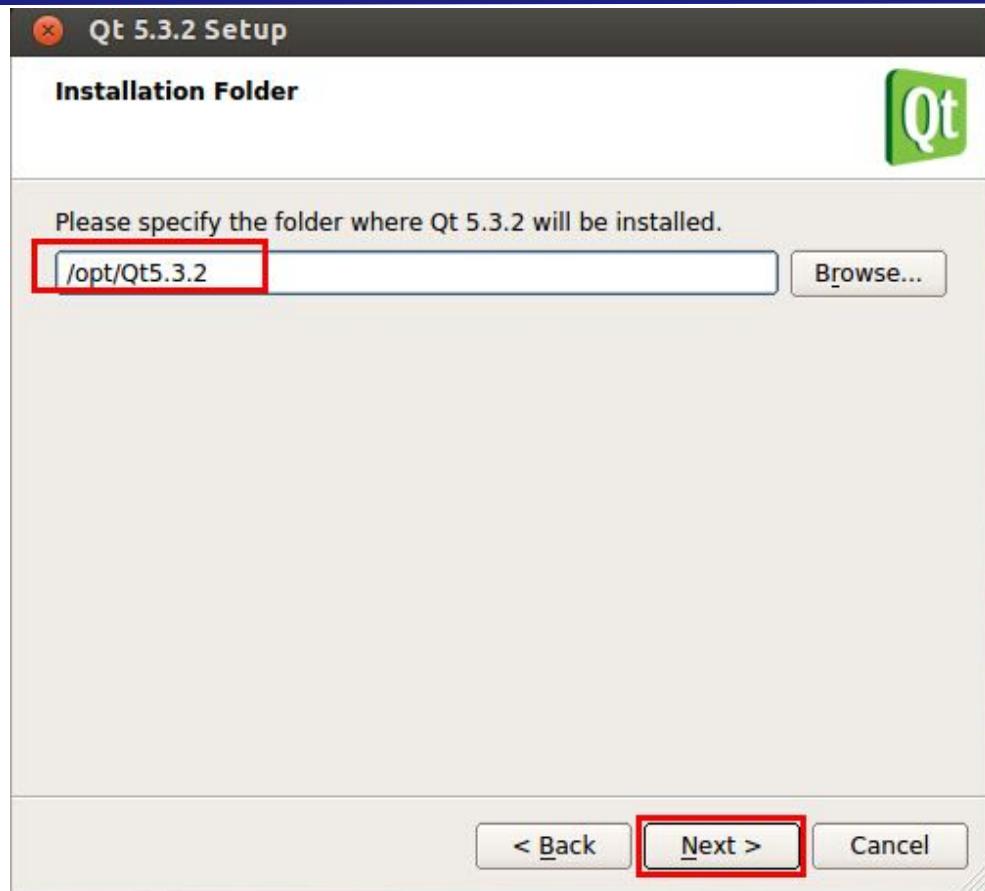
运行该程序 “./qt-opensource-linux-x64-android-5.3.2.run” 。

```
root@ubuntu:/home/topeet/Downloads# ./qt-opensource-linux-x64-  
android-5.3.2.run
```

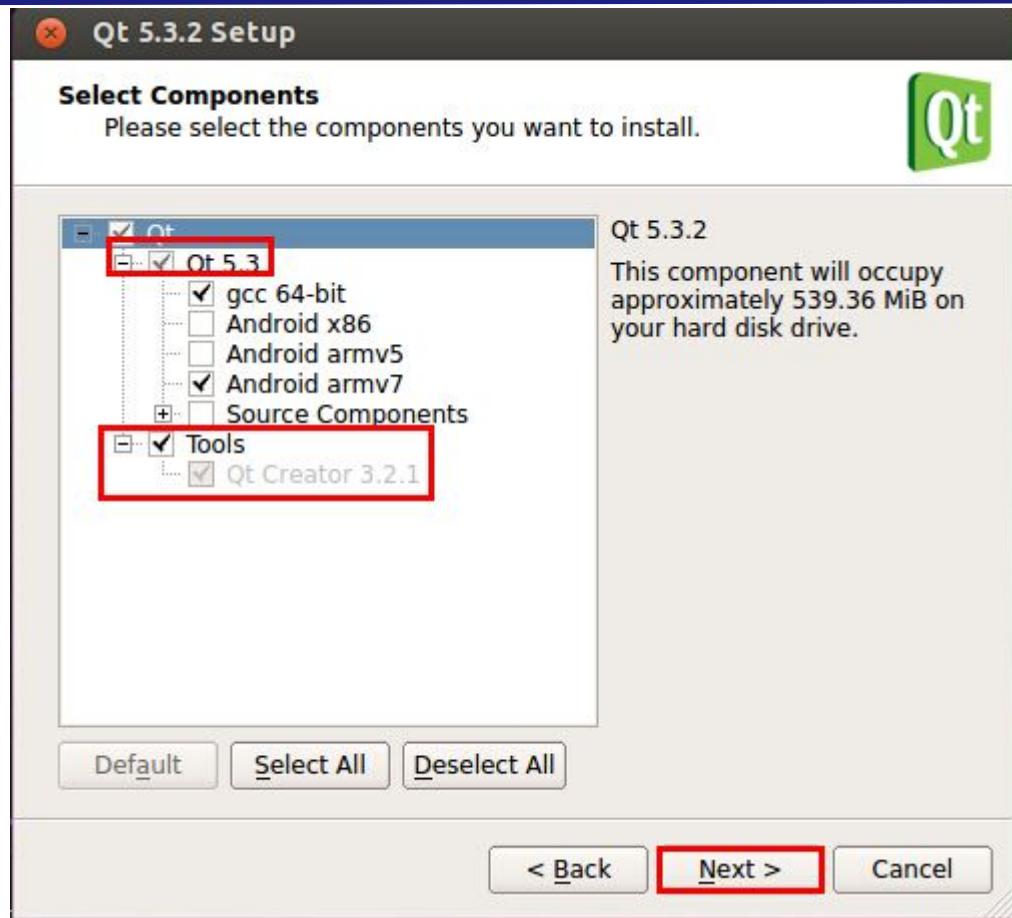
出现安装向导，单击 “Next” ，继续。



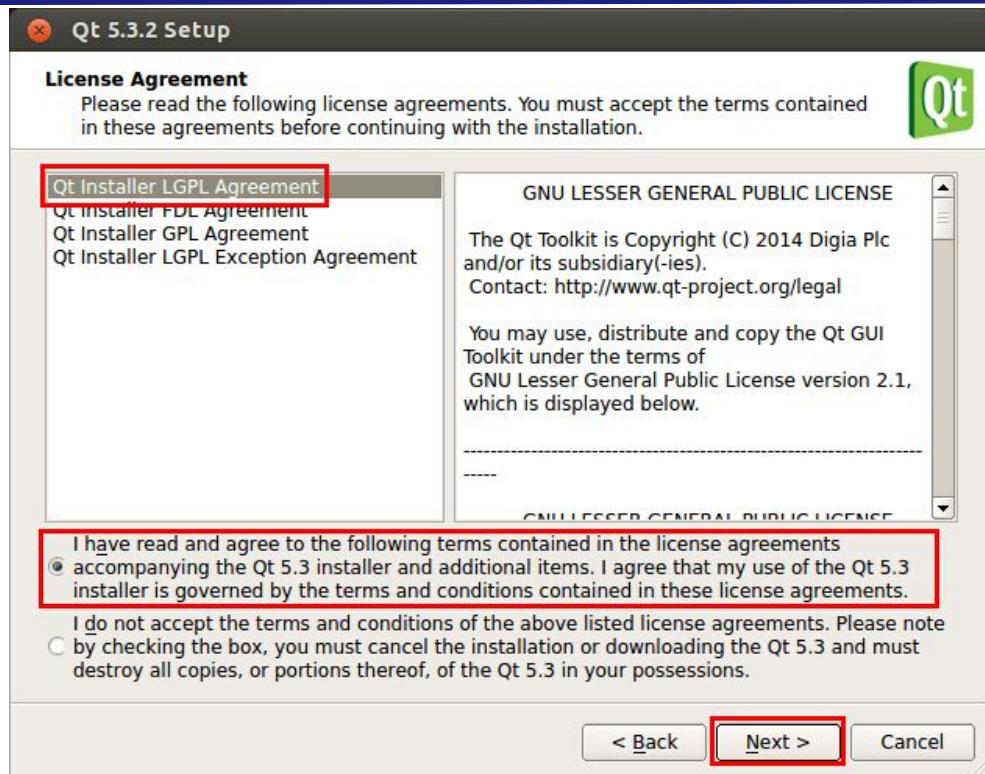
如下图，安装目录，选择默认 “/opt/Qt5.3.2” 即可。单击 “Next” ，继续。



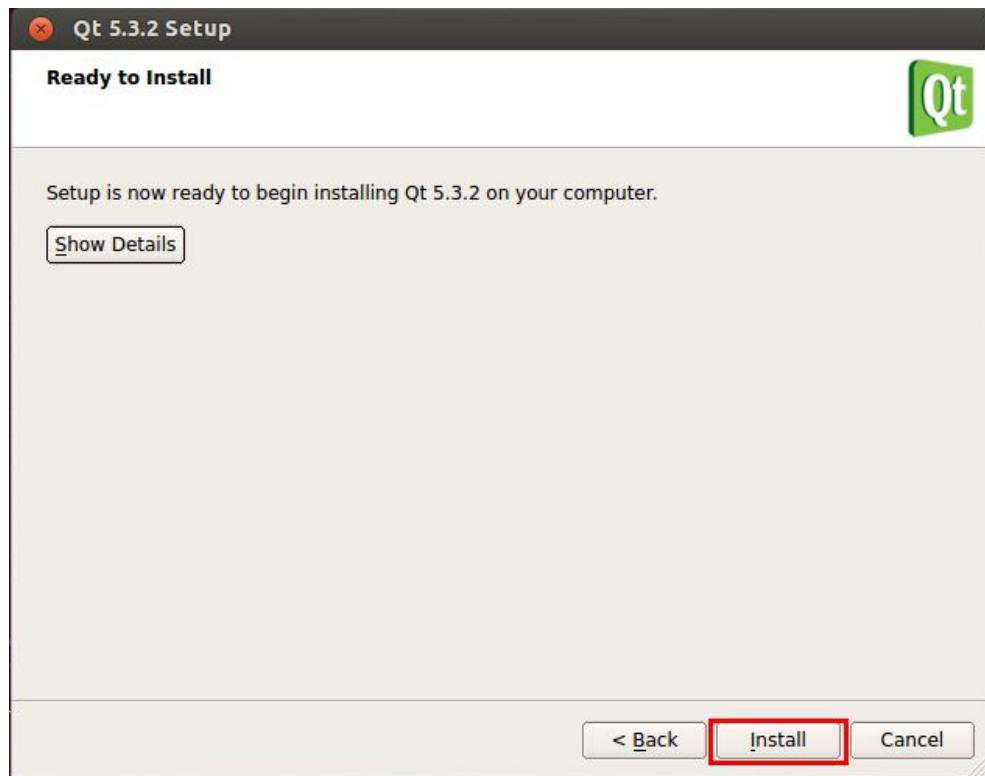
如下图，安装了 Qt5.3 的插件和 Qt Creator，单击“Next”继续。需要注意的是，我们在这里安装了 Qt5.3 的插件，仅仅是能够支持后面的应用程序在 PC 机上运行，而不是在开发板上运行。在开发板上运行的 Qt 版本是 Qt/E4.7.1，这里大家要注意区分。



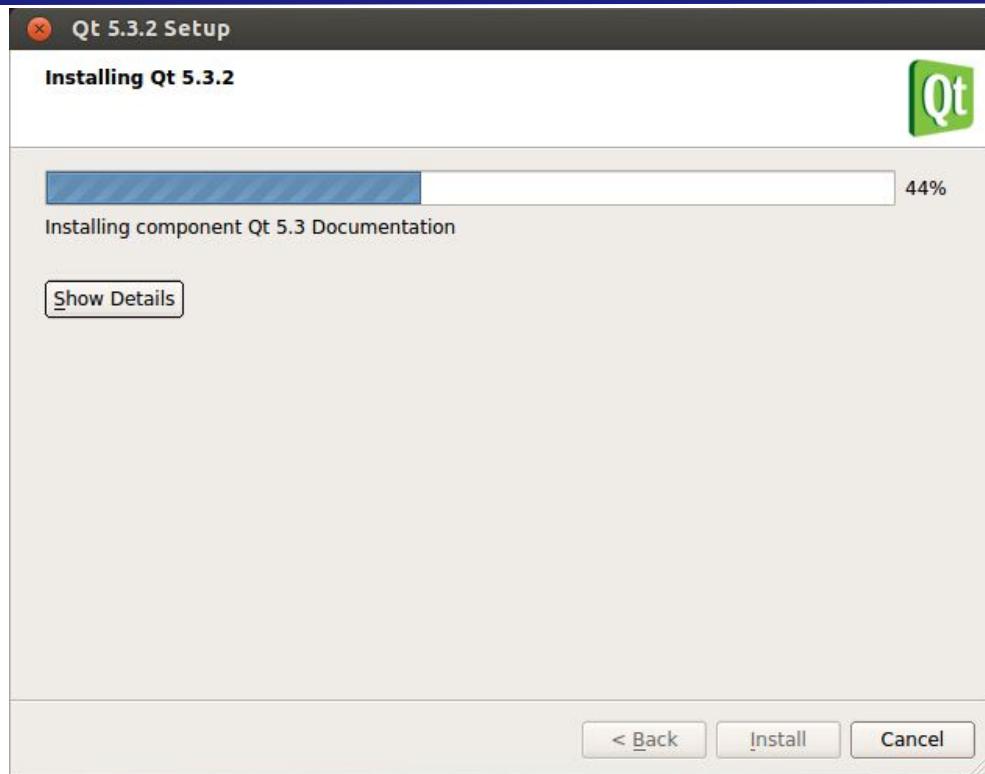
如下图，接收安装协议。单击“Next” ，继续安装。



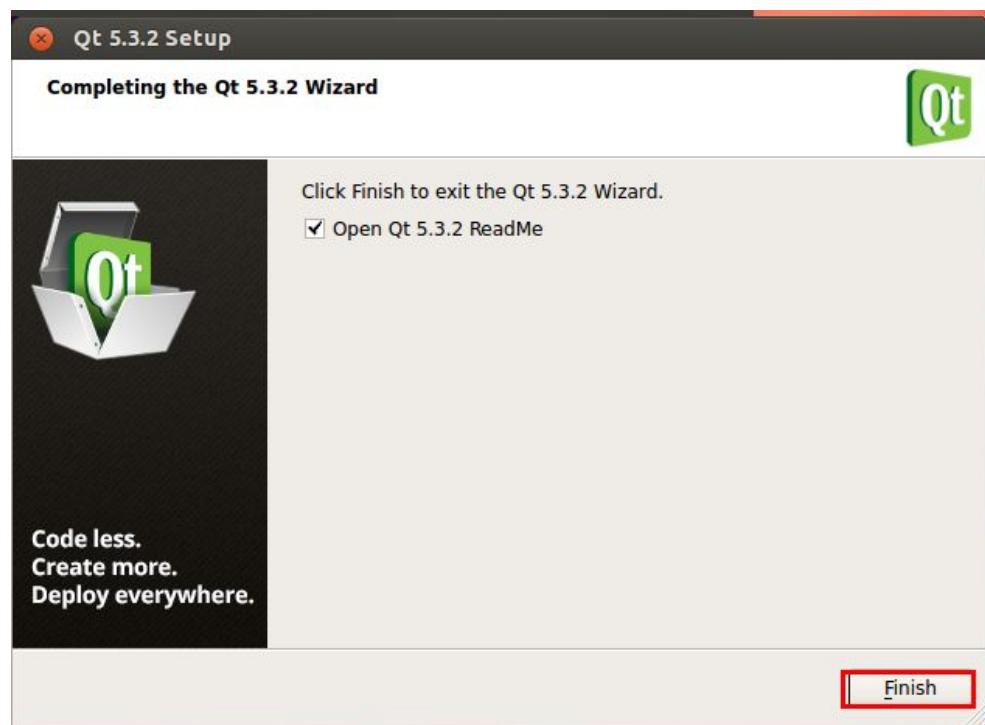
如下图，单击“Install”，继续安装。



如下图，安装中。



如下图，单击“Finish”，安装完成。



如下图，会弹出警告和报错，直接忽略。使用“Ctrl+d”退出。

```
root@ubuntu:/home/topeet/Downloads# ./qt-opensource-linux-x64-
android-5.3.2.run
Qt: Session management error: None of the authentication proto-
cols specified are supported
root@ubuntu:/home/topeet/Downloads#
** (gedit:27296): WARNING **: The connection is closed

(gedit:27296): EggSMClient-WARNING **: Failed to connect to th-
e session manager: None of the authentication protocols specif-
ied are supported

** (gedit:27296): WARNING **: Could not connect to session bus
^C
root@ubuntu:/home/topeet/Downloads#
```

然后进入 “/opt/Qt5.3.2” 目录，如下图。

```
root@ubuntu:/home/topeet/Downloads# cd /opt/Qt5.3.2/
root@ubuntu:/opt/Qt5.3.2# ls
5.3           Licenses          qt-project.org.html
components.xml  MaintenanceTool  README.txt
Docs           MaintenanceTool.dat Tools
Examples        MaintenanceTool.ini
InstallationLog.txt  network.xml
root@ubuntu:/opt/Qt5.3.2#
```

如下图，避免麻烦，直接使用命令 “#chmod 777 \*” 改权限。

```
root@ubuntu:/opt/Qt5.3.2# chmod 777 *
root@ubuntu:/opt/Qt5.3.2#
```

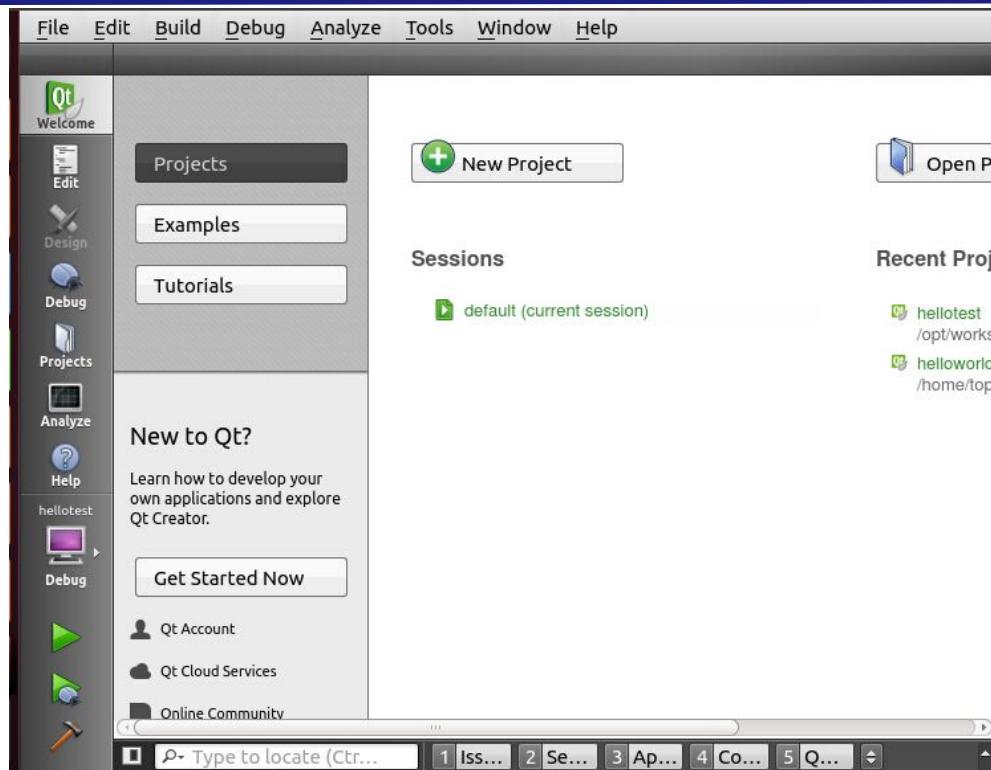
如下图，使用命令 “#Tools/QtCreator/bin/” ，可以看到 “Qt Creator” 软件。

```
root@ubuntu:/opt/Qt5.3.2# cd Tools/QtCreator/bin/
root@ubuntu:/opt/Qt5.3.2/Tools/QtCreator/bin# ls
imports      qbs-setup-qt      qtcreator
plugins      qbs-setup-toolchains  qtcreator_process_stub
qbs          qml              qtcreator.sh
qbs-config   qml2puppet       qtcreator
qbs-config-ui qmlpuppet       sdktool
qbs-qmltypes  qt.conf
root@ubuntu:/opt/Qt5.3.2/Tools/QtCreator/bin#
```

如下图，运行编译软件,使用命令 “#./qtcreator” 。

```
root@ubuntu:/opt/Qt5.3.2/Tools/QtCreator/bin# ./qtcreator
```

如下图所示，弹出编译环境。



如下图，打开终端，可以看到报错和警告。一个错误是因为，作者使用了“root”用户，从这一点看来，“诺基亚”的软件也是很严谨的，如果使用一般用户运行软件，就没有问题；还有一个错误，是某个守护进程没有打开，也是不用管的。

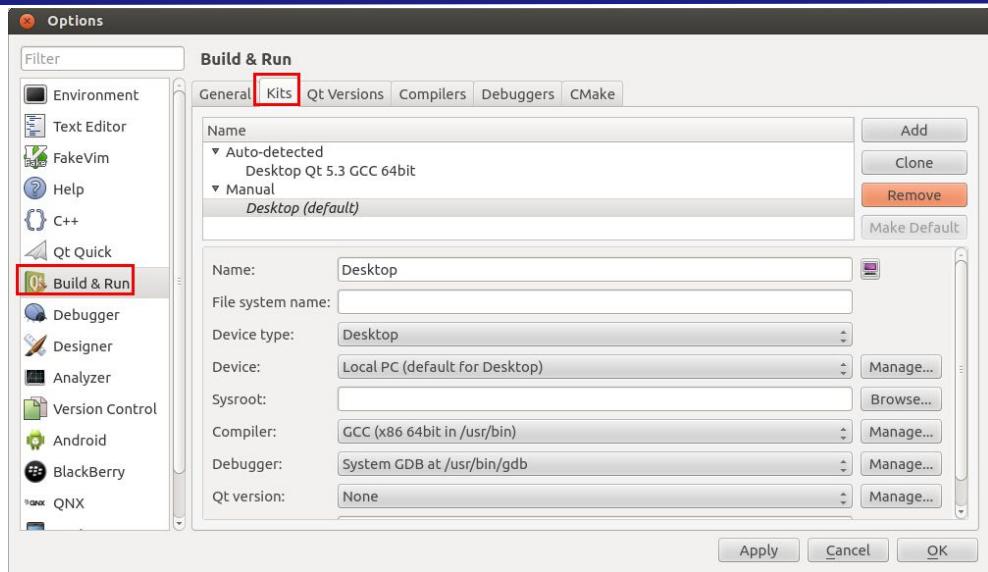
```
root@ubuntu:/opt/Qt5.3.2/Tools/QtCreator/bin# ./qtcreator

** (qtcreator:27335): WARNING **: The connection is closed
Qt: Session management error: None of the authentication protocols specified are supported

(qtcreator:27335): GConf-WARNING **: Client failed to connect
to the D-BUS daemon:
Did not receive a reply. Possible causes include: the remote application did not send a reply, the message bus security policy blocked the reply, the reply timeout expired, or the network connection was broken.

** (qtcreator:27335): WARNING **: Unable to create Ubuntu Menu
Proxy: The connection is closed
```

接着，我们回到 Qt Creator，配置编译器等，打开菜单“Tools”→“Options”，如下图所示。



如上图，迅为电子提供的安装包已经自带了 Ubuntu 的 Kits，只需要开发环境处于默认的状态，编译出来的应用程序，就可以在 Ubuntu 上运行。

## 11.2 使用集成开发环境开发 Qt 应用程序

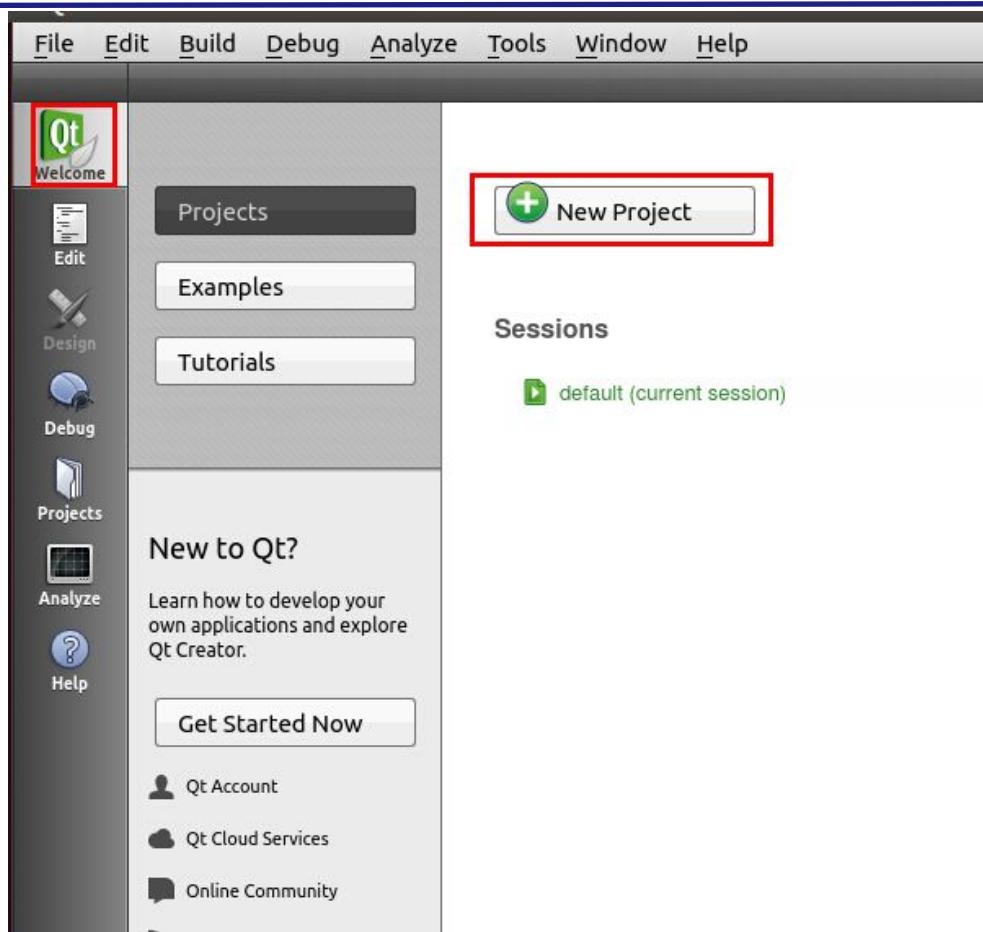
从第七章节开始准备 Qt/E4.7.1 的库，到这一章节，所有的工具都准备完毕了，万事俱备，只欠东风。

本小节就介绍一下，如何在 Ubuntu 运行 helloworld 例程以及在开发板上运行 helloworld 例程。

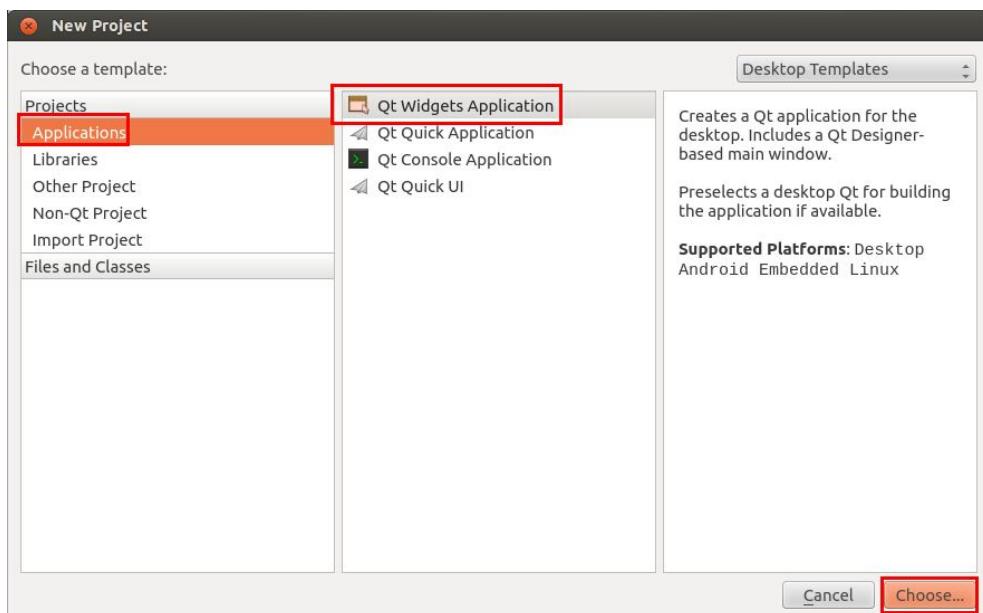
### 11.2.1 Ubuntu 上运行 helloworld

接着 11.1 小节，所有的都是默认状态，即可编译出在 Ubuntu 上运行的应用程序，因为我们安装 Qt Creator 的时候，选择了 Qt5.3 下的插件 “gcc 64-bit”。

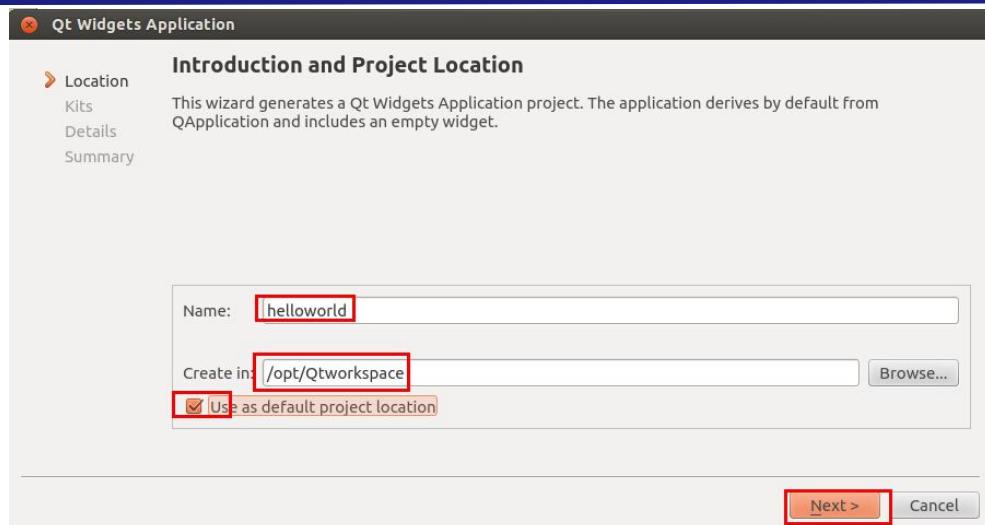
如下图，新建一个 helloworld 工程。



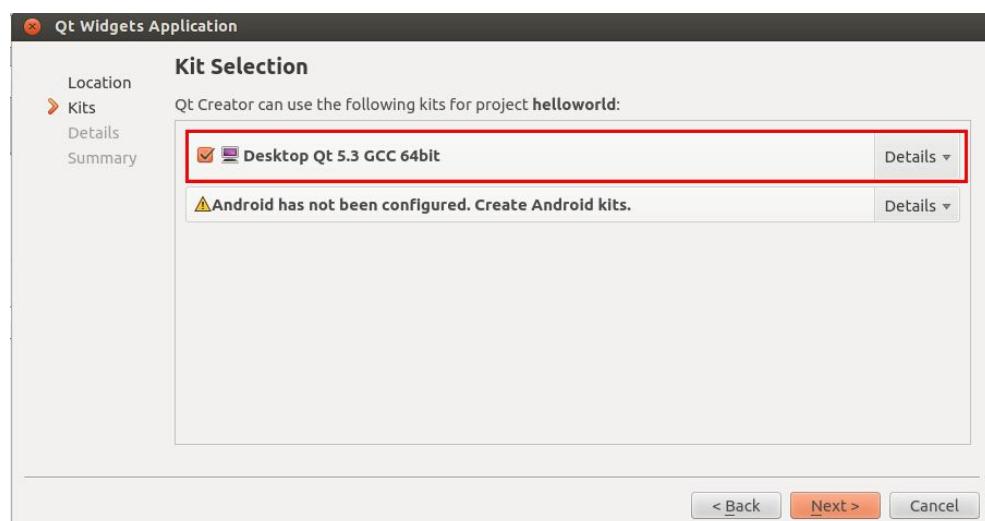
如下图所示。



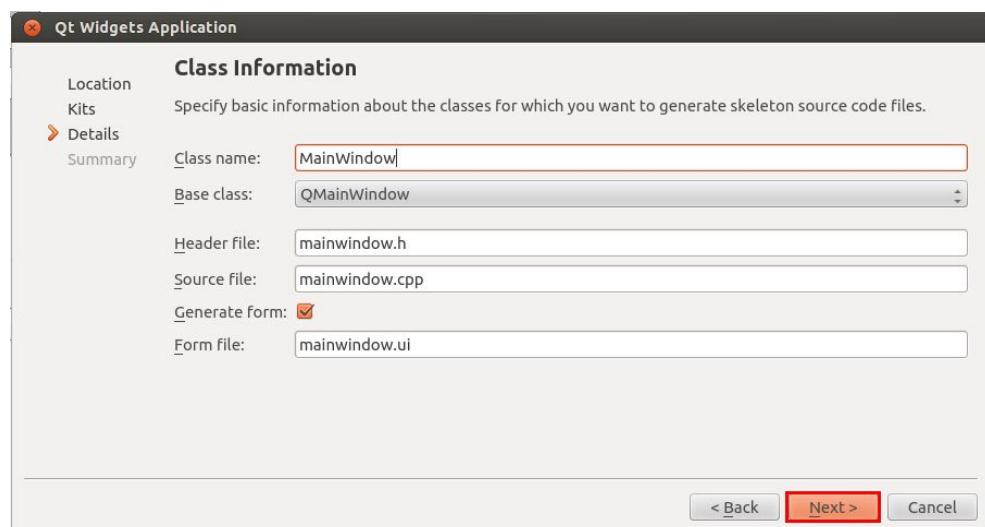
如下图所示，工程命名为“helloworld”，重新定义了工程存储目录（用户根据个人习惯设置工程目录）。然后单击“Next”，继续。



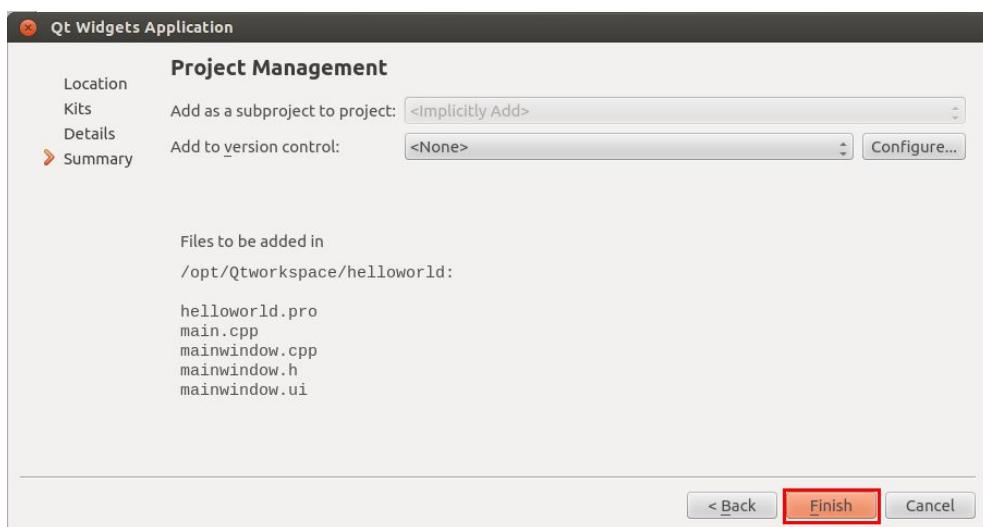
如下图，选择支持 Ubuntu 的 GCC 64 位编译器。单击“Next”继续。



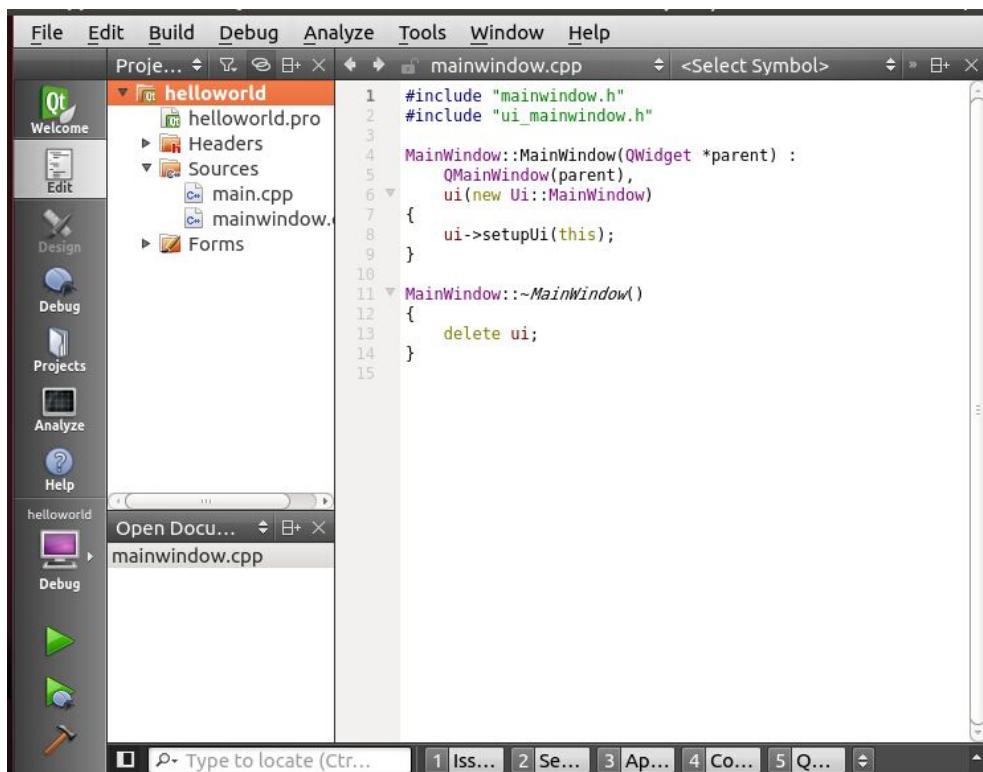
如下图，默认即可。单击“Next”，继续。



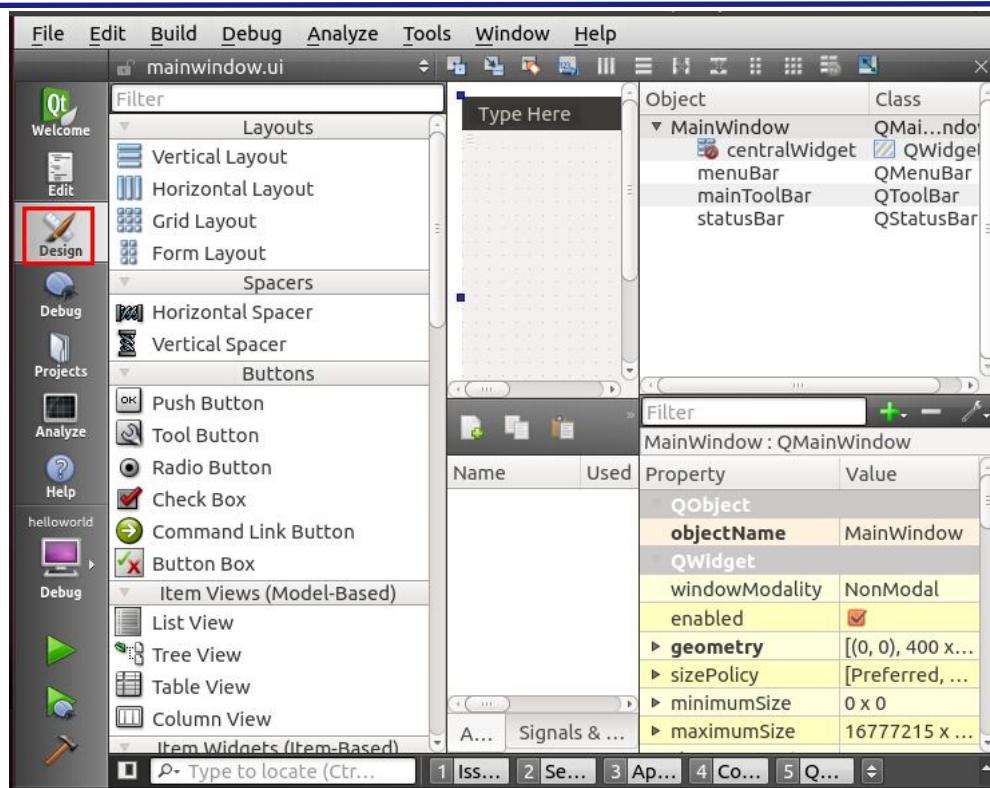
如下图，默认即可。单击“Finish”，结束。



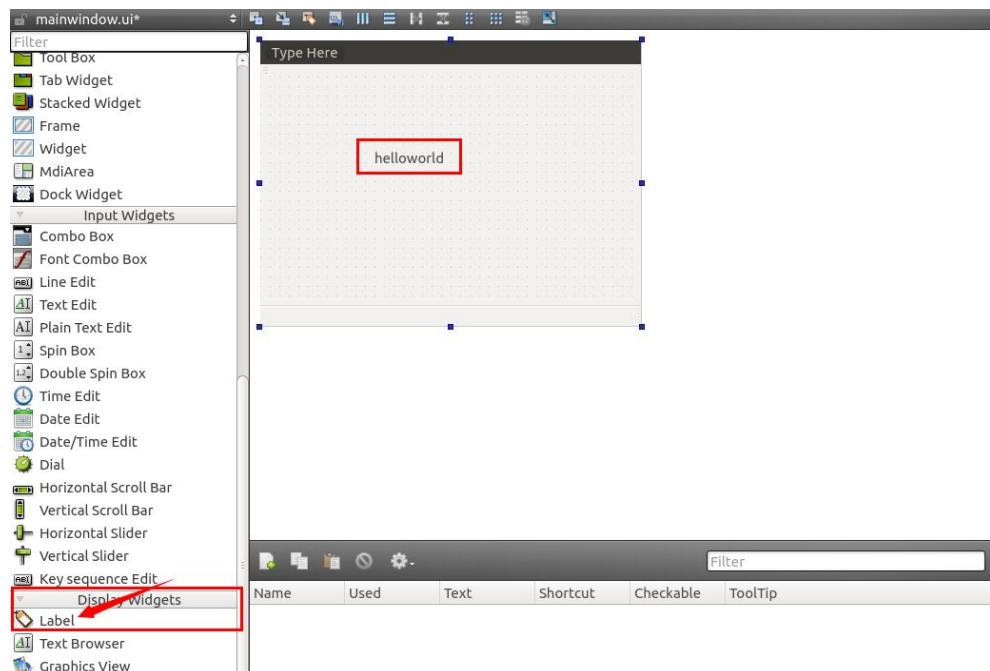
如下图，新建工程完成。



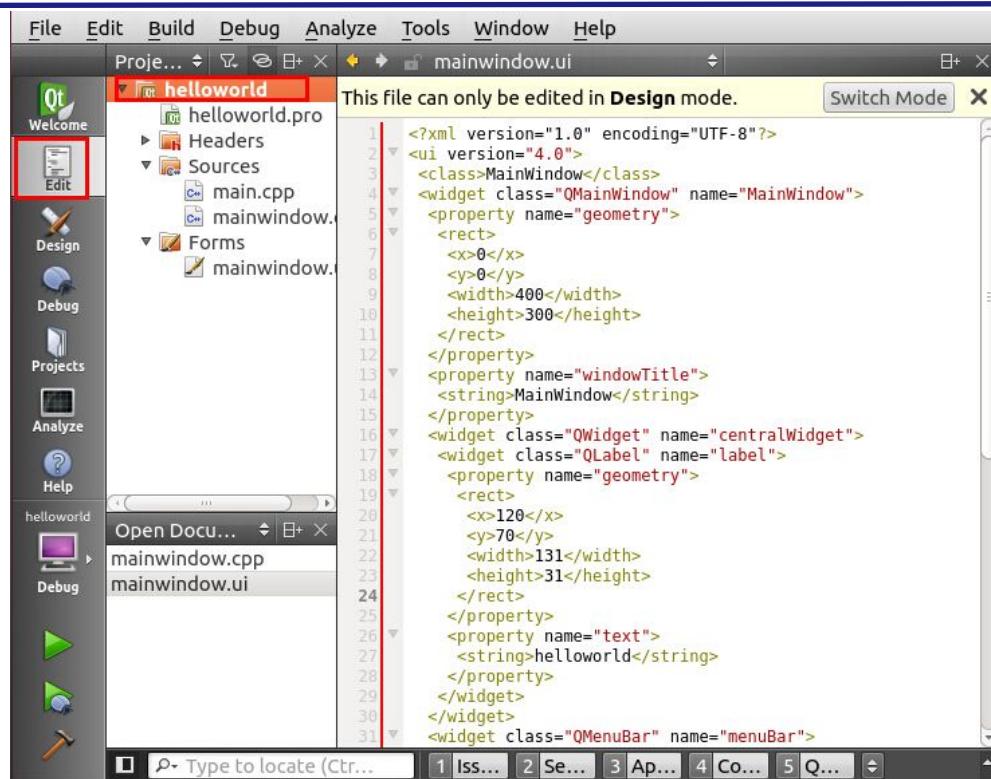
如下图，单击窗口左边的“Design”。



如下图，使用 Label,给 Qt 的应用的图形添加 helloworld 显示。然后保存工程。



选择左边菜单 “Edit” ，选择 “helloworld” ，如下图。



如下图，编译调试运行。在上方的菜单有“Build”，里面有“Build All” “Run”等，用户也可以使用左下角的“三角”按钮进行编译运行。



如下图，进入终端，查看一下在 Ubuntu 上运行的 Qt 应用程序。

```
root@ubuntu:/opt/Qtworkspace/build-helloworld/Desktop_Qt_5_3_GCC_64bit-Debug
root@ubuntu:/opt/Qtworkspace# ls
build-helloworld/Desktop_Qt_5_3_GCC_64bit-Debug helloworld
root@ubuntu /opt/Qtworkspace# cd build-helloworld/Desktop_Qt_5_3_GCC_64bit-Debug
/
root@ubuntu:/opt/Qtworkspace/build-helloworld/Desktop_Qt_5_3_GCC_64bit-Debug# ls
helloworld mainwindow.o moc_mainwindow.cpp ui_mainwindow.h
mainwindow.o Makefile moc_mainwindow.o
root@ubuntu:/opt/Qtworkspace/build-helloworld/Desktop_Qt_5_3_GCC_64bit-Debug#
```

如下图，运行 Qt 应用 helloworld。使用命令 “`#./helloworld`” 运行。

```
root@ubuntu:/opt/Qtworkspace/build-helloworld/Desktop_Qt_5_3_GCC_64bit-Debug
root@ubuntu:/opt/Qtworkspace# ls
build-helloworld/Desktop_Qt_5_3_GCC_64bit-Debug helloworld
root@ubuntu:/opt/Qtworkspace# cd build-helloworld/Desktop_Qt_5_3_GCC_64bit-Debug
/
root@ubuntu:/opt/Qtworkspace/build-helloworld/Desktop_Qt_5_3_GCC_64bit-Debug# ls
helloworld main.cpp
main.o Makefile
root@ubuntu:/opt/Qtworkspace/build-helloworld/Desktop_Qt_5_3_GCC_64bit-Debug# ./helloworld

** (helloworld:2): warning: Qt: Session management is not supported
helloworld
tocols specified are
menu Proxy: The connection is closed
```

## 11.2.2 在 iTOP-4412 开发板上运行 helloworld

### 11.2.2.1 移植

接上一小节，如下图，有两个文件夹，一个是直接在 PC 上运行的代码以及应用程序，另外一个是源码。现在进入源码文件夹。

```
root@ubuntu:/opt/Qtworkspace# ls  
build-helloworld-Desktop_Qt_5_3_GCC_64bit-Debug  helloworld  
root@ubuntu:/opt/Qtworkspace#
```

进行下面几步，大家就容易理解“为什么 Qt 跨平台非常方便”。

这个 hellworld 应用相当于已经在 PC 机上调试通过了，现在我们只需要做简单的几步就可以将其移植到开发板 4412 平台上。

```
root@ubuntu:/opt/Qtworkspace/helloworld# ls
helloworld.pro      main.cpp     mainwindow.h
helloworld.pro.user  mainwindow.cpp mainwindow.ui
root@ubuntu:/opt/Qtworkspace/helloworld#
```

如下图，在第七章中我们编译生成了“/opt/qt-4.7.1/”，这个文件夹包含了移植所需要的最重要的工具 qmake。进入“/opt/qt-4.7.1/bin”，可以看到 qmake 文件。

```
root@ubuntu:/opt/qt-4.7.1/bin# ls
lrelease  moc  qmake  rcc  uic
root@ubuntu:/opt/qt-4.7.1/bin#
```

查看了 Qt/E4.7.1 的 qmake 之后，再进入 hellworld 的源码文件夹，如下图所示。

```
root@ubuntu:/opt/Qtworkspace/helloworld# ls
helloworld.pro      main.cpp     mainwindow.h
helloworld.pro.user  mainwindow.cpp mainwindow.ui
root@ubuntu:/opt/Qtworkspace/helloworld#
```

然后，在该文件夹中运行 qmake “#/opt/qt-4.7.1/bin/qmake”，如下图所示。

```
root@ubuntu:/opt/Qtworkspace/helloworld# /opt/qt-4.7.1/bin/qmake
root@ubuntu:/opt/Qtworkspace/helloworld#
```

如下图，多了一个 Makefile 文件。

```
root@ubuntu:/opt/Qtworkspace/helloworld# ls
helloworld.pro      main.cpp     mainwindow.h  Makefile
helloworld.pro.user  mainwindow.cpp mainwindow.ui
root@ubuntu:/opt/Qtworkspace/helloworld#
```

然后，执行编译命令“#make”，如下图所示。

```
root@ubuntu:/opt/Qtworkspace/helloworld# ls
helloworld.pro      main.cpp     mainwindow.h   Makefile
helloworld.pro.user  mainwindow.cpp mainwindow.ui
root@ubuntu:/opt/Qtworkspace/helloworld# make
[...]
root@ubuntu:/opt/Qtworkspace/helloworld# ls
helloworld          mainwindow.cpp  moc_mainwindow.cpp
helloworld.pro       mainwindow.h   moc_mainwindow.o
helloworld.pro.user  mainwindow.o   ui_mainwindow.h
main.cpp            mainwindow.ui
main.o              Makefile
root@ubuntu:/opt/Qtworkspace/helloworld#
```

生成了“helloworld”。

```
root@ubuntu:/opt/Qtworkspace/helloworld# ls
helloworld          mainwindow.cpp  moc_mainwindow.cpp
helloworld.pro       mainwindow.h   moc_mainwindow.o
helloworld.pro.user  mainwindow.o   ui_mainwindow.h
main.cpp            mainwindow.ui
main.o              Makefile
root@ubuntu:/opt/Qtworkspace/helloworld#
```

然后，我们使用 file 命令测试一下。如下图，“#file helloworld”，可以看到 helloworld 应用文件的基本信息，它是属于 ARM 平台的。

```
root@ubuntu:/opt/Qtworkspace/helloworld# file helloworld
helloworld: ELF 32-bit LSB executable, ARM, version 1 (SYSV),
dynamically linked (uses shared libs), for GNU/Linux 2.6.14,
not stripped
root@ubuntu:/opt/Qtworkspace/helloworld#
```

然后，测试一下 X86 上位机平台的文件，如下图所示，可以观察到明显的不同。

```
root@ubuntu:/opt/Qtworkspace/build-helloworld-Desktop_Qt_5_3_G
CC_64bit-Debug# file helloworld
helloworld: ELF 64-bit LSB executable, x86-64, version 1 (SYSV)
, dynamically linked (uses shared libs), for GNU/Linux 2.6.24
, BuildID[sha1]=0xa999f4bd3a0f5df3805c5b96df1eb018d7038f5c, no
t stripped
root@ubuntu:/opt/Qtworkspace/build-helloworld-Desktop_Qt_5_3_G
CC_64bit-Debug#
```

### 11.2.2.2 安装运行

下面介绍一下如何在开发板上运行 hellworld。

这里我们需要用到 U 盘或者 TF 卡来上传文件。在开发板运行了 Linux-qt 系统之后（就是烧写运行了光盘 “\image\linux” 中的镜像），在超级终端中完成以下步骤。

如果使用的是 U 盘，可以新建一个目录。U 盘挂载，使用靠近 tf 卡座的 USB 接口。在 /mnt 目录中，新建目录 "/usb1"。然后挂载 "#mount /dev/udisk /mnt/usb1/"

tf 卡挂载。tf 卡插入后，可以发现 “/dev” 目录下，多了 “mmcblk1p1” ，挂载 "#mount /dev/mmcblk1p1 /mnt/disk/" 。

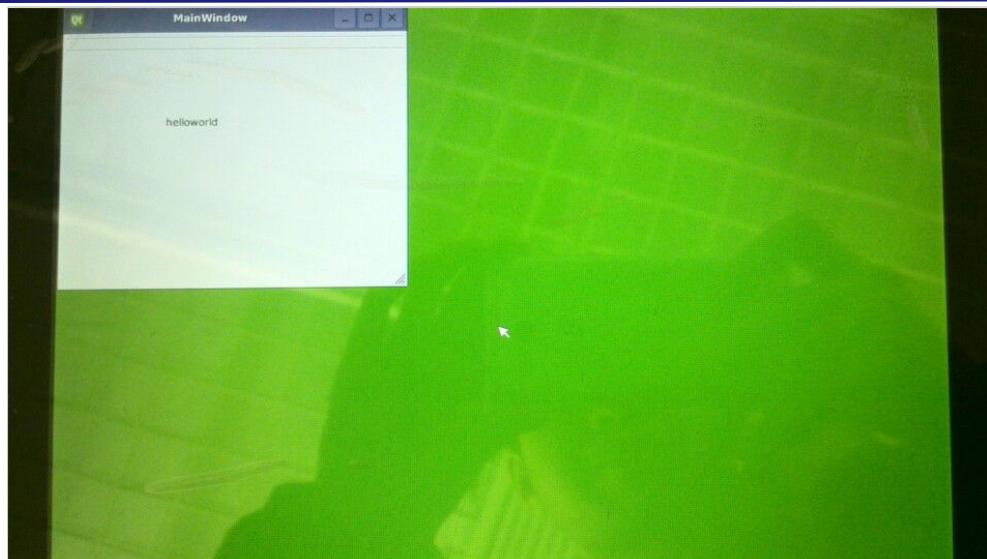
挂载完成后，以 tf 为例，在超级终端中，进入 “/mnt/disk” 目录，如下图红色框中的内容，下图中其它部分和本小节无关，是作者的 tf 卡中的其它文件，直接无视。

```
[root@iTOP-4412]# cd /mnt/disk/  
[root@iTOP-4412]# ls  
fonts                               iTOO4412_helloworld  
hellworld                           qt4  
[root@iTOP-4412]#  
[root@iTOP-4412]#
```

然后，如下图所示，在超级终端中，先修改权限 "#chmod 777 hellworld" ，然后再运行 "#hellworld -qws" 。

```
[root@iTOP-4412]# ./hellworld -qws  
^[[?1;2c[ 66.803775] s3cfb s3cfb.0: [fb2] already in  
NK_UNBLANK                                         FB_BLA  
Cannot open input device '/dev/tty0': No such file or directory
```

开发板的屏幕显示如下所示，hellworld 程序就可以开发板上运行了。



### 11.2.3 开发板上修改文件

用户只需要通过超级终端，就可以直接修改开发板上的一部分文件。如下图所示，修改系统的启动文件 “/etc/init.d/rcS” .

```
[root@iTOP-4412]# vi /etc/init.d/rcS
```

打开 rcS 文件后，进入最后一行，如下图，就可以直接修改。

```
echo "                                " > /dev/tty1
echo "Starting Qtopia, please waiting..." > /dev/tty1
echo "                                "
echo "Starting Qtopia4, please waiting..."

#/bin/qtopia &
/bin/qt4 &
~
```

修改方法，和在 Ubuntu 编译环境中修改文件一样，都是使用 vi 编辑器，修改完之后保存退出，然后重启，开发板就可以运行新的脚本文件。

## 11.2.4 开机运行 helloworld

光盘自带的 Linux-Qt 默认运行的是 qtopia 和 qt4。在嵌入式中，应用最广泛的形式是，系统带 Qt/E 的库，然后运行一个带界面的应用程序，例如前面小节编译的 helloworld。

将 helloworld 拷贝到 Linux-QT 文件夹下（在第六章第七章都有用到该文件夹）的“/bin”文件夹下，如下图所示。

```
root@ubuntu:/home/topeet/Linux+QT/root/bin#
root@ubuntu:/home/topeet/Linux+QT/root/bin# ls hell*
hellworld
root@ubuntu:/home/topeet/Linux+QT/root/bin#
```

修改启动脚本“root/etc/init.d/rcS”，如下图所示。

```
sleep 2

/etc/firmware/load_firmware

#mkdir /mnt/disk
#mount -t yaffs2 /dev/mtdblock3 /mnt/disk

/sbin/ifconfig lo 127.0.0.1
/etc/init.d/ifconfig-eth0

echo "                                " > /dev/tty1
echo "Starting Qtopia, please waiting..." > /dev/tty1
echo "                                "
echo "Starting Qtopia4, please waiting..."

#/bin/qtopia &
/bin/qt4 &
```

90,1 Bot

如下图，修改为默认启动 helloworld，注意要添加“-qws”，保存退出。

```
echo "                                " > /dev/tty1
echo "Starting Qtopia, please waiting..." > /dev/tty1
echo "                                "
echo "Starting Qtopia4, please waiting..."

#/bin/qtopia &
#/bin/qt4 &
/bin/helloworld -qws &
```

88,0-1 Bot

然后编译，新生成文件系统镜像“system.img”，如下图所示。

```
root@ubuntu:/home/topeet/Linux+QT# make_ext4fs -s -l 314572800  
-a root -L linux system.img root  
Creating filesystem with parameters:  
    Size: 314572800  
    Block size: 4096  
    Blocks per group: 32768  
    Inodes per group: 6400  
    Inode size: 256  
    Journal blocks: 1200  
    Label: linux  
    Blocks: 76800  
    Block groups: 3  
    Reserved block group size: 23  
Created filesystem with 8674/19200 inodes and 43728/76800 blocks  
root@ubuntu:/home/topeet/Linux+QT# ls  
root  root_20140912.tar.gz  system.img  
root@ubuntu:/home/topeet/Linux+QT#
```

将该文件系统镜像结合光盘 “\image\linux” 文件夹中的 zImage 以及 ramdisk-uboot.img 烧写到开发板。开发板启动之后，即可直接运行 helloworld。

运行后使用进程查找命令 “#ps” ，可以看到下图进程 “964 root 0:02 /bin/helloworld -qws” 。

```
  PID  PPID  CPUTIME COMMAND  
 884  root   0:00 [tvout resume wo]  
 885  root   0:00 [jbd2/mmcblk0p2-]  
 886  root   0:00 [ext4-dio-unwrit]  
 927  root   0:01 syslogd  
 930  root   0:00 /usr/sbin/inetd  
 964  root   0:02 /bin/helloworld -qws  
 965  root   0:00 -/bin/sh  
.0471  root   0:03 [kworker/0:0]  
.3217  root   0:01 [kworker/0:1]  
.5954  root   0:00 [kworker/0:2]  
.6765  root   0:00 [sh]  
.6766  root   0:00 [sh]
```

## 11.3 QtE 必备知识介绍

本节介绍一下用户需要了解的基础知识，包括开机启动脚本、qt4 的修改、触摸和鼠标库等

### 11.3.1 开机启动脚本

开机启动脚本文件是 “/etc/init.d/rcS” , 大部分操作系统都有类似的文件。该文件中的代码属于脚本语言 , 比较容易理解。

下面截取有代表性的分析一下。注意 , 下面的代码行数可能和用户的不能直接对应 , 这里并不影响用户学习 , 这里主要是了解整个启动流程 , 以及在启动过程中 , 都做了哪些事情。

```
01 #! /bin/sh  
02 PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/bin:  
03 runlevel=S  
04 prevlevel=N  
05 umask 022  
06 export PATH runlevel prevlevel
```

代码分析

01 #! /bin/sh 是指此脚本使用/bin/sh 来解释执行 , #!是特殊的表示符 , 后面是解释此脚本的 shell 的路径。

02 PATH=XX 是设置默认的有效执行路径 , 代表 root 根目录下的路径。

03 这里是设置用户等级 , 这里默认为单用户。 ( 用户记不得 , Linux 是多用户的 )

04 表示用户创建的新文件权限为 755

05 设置环境变量 ( 和搭建编译环境时修改 Ubuntu 的环境变量类似 )

```
12 /bin/hostname iTOP-4412  
14 #/bin/mount -n -t proc none /proc  
23 /sbin/mdev -s  
24 /bin/hotplug  
39 ln -sf /dev/ttyS2 /dev/tty2  
43 amixer cset numid=5 127  
50 /sbin/hwclock -s -f /dev/rtc
```

52 syslogd

12 设置主机名称为 iTOP-4412，在这里用户可以修改为自己的名称。

14 使用 mount 命令设置常用临时目录

23 mdev 命令自动产生驱动所需的节点文件

24 hotplug 挂载 USB 设备

39 ln 命令建立链接

43 amixer 设置音频的参数

50 hwclock 设置时钟

52 syslogd 记录系统运行的各种讯息

53 /etc/rc.d/init.d/netd start

.....

81 /sbin/ifconfig lo 127.0.0.1

.....

87 echo "Starting Qtopia4, please waiting..."

启动各种服务程序

ifconfig 配置网络

echo 命令向输出终端打印信息

89 #/bin/qtopia &

"#" 就是注释掉这一行， "&" 表示在后台运行

91 /bin/helloworld -qws &

表示运行 "helloworld" 的时候，QWS 功能启动， "QWS" 是 QT 的窗口系统，也就是需要加上 "-qws"，屏幕才能显示。

注意，脚本命令比较容易理解，通过本节的分析，相信用户已经具备了一定的阅读脚本代码的能力，在手册后面部分，脚本命令就不给大家做详细的分析了。大家只需要在遇到脚本命令的时候去学习即可，这里只是大家学习和理解脚本文件的一个引子。

### 11.3.2 qt4 文件的移植和修改（鼠标触摸以及字体）

在 7.4 小节介绍配置文件的时候，提到迅为制作的触摸库文件 “/usr/local/tslib” ，所以我们已经将触摸功能编译到 Qt/E4.7.1 的库文件中，开发板可以支持触摸和鼠标功能。

用户在前面运行 hellworld 的时候，会发现触摸和鼠标似乎无法使用，但是在 qt4 中却能使用触摸和鼠标。

这一小节就介绍如何将 qt4 的触摸功能移植到应用程序 hellworld 中。

如下图，打开 “etc/init.d/rcS” 文件，将最后一行修改为 “/bin/qt4\_shell &” ，这样在开机之后系统会运行 qt4.shell 程序。修改完成之后，保存退出。

```

echo "                                     " > /dev/tty1
echo "Starting Qtopia, please waiting..." > /dev/tty1
echo "
echo "Starting Qtopia4, please waiting..." > /dev/tty1

#/bin/qtopia &
/bin/qt4.shell & ←
/bin/helloworld qws &
-- INSERT --

```

然后使用命令 “#cp -r bin/qt4 bin/qt4.shell” 将 “bin/qt4” 文件拷贝生成 “bin/qt4.shell” 文件，如下图所示。

cpto	ruser	tpri	pscan	su	who
crontab	getopt	ls	pwd	sum	whoami
cryptpw	grep	lsattr	qt4	sv	xargs
cttihack	gunzip	lzmacat	qt4.shell	sync	yes
cut	gzip	lzop	qtopia	tac	zcat
date	hd	makemime	readahead	tail	
dc	head	md5sum	readlink	tar	

然后打开 “bin/qt4.shell” 文件，如下图所示。

代码 “export TSLIB\_ROOT=/usr/local/tslib” 等调用了触摸和鼠标驱动。

代码 “export QTDIR=/opt/qt-4.7.1/” 调用了 QtE 的库。

```
#!/bin/sh

export TSLIB_ROOT=/usr/local/tslib
export TSLIB_TSDEVICE=/dev/input/event2
export TSLIB_TSEVENTTYPE=input
export TSLIB_CONFFILE=/usr/local/tslib/etc/ts.conf
export TSLIB_PLUGINDIR=/usr/local/tslib/lib/ts
export TSLIB_CALIBFILE=/etc/pointercal
export TSLIB_PLUGINDIR=$TSLIB_ROOT/lib/ts
export TSLIB_CONSOLEDEVICE=none
export TSLIB_FBDEVICE=/dev/fb0
export QTDIR=/opt/qt-4.7.1/
```

如下图所示，这一段关于鼠标的驱动已经被注释掉了。

```
#if [ -c /dev/input/event2 ]; then
#    export QWS_MOUSE_PROTO="Tslib:${TSLIB_TSDEVICE}"
#    if [ -e /etc/pointercal -a ! -s /etc/pointercal ] ; then
#        rm /etc/pointercal
#        /usr/local/tslib/bin/ts_calibrate
#    fi
#else
#    export QWS_MOUSE_PROTO="MouseMan:/dev/input/mice"
#>/etc/pointercal
#fi

export QWS_MOUSE_PROTO="Tslib:${TSLIB_TSDEVICE}"
/usr/local/tslib/bin/ts_calibrate
```

将上图中与触摸鼠标相关代码修改为下面的代码。

```
if [ ! -c /dev/input/event2 ]; then
    export QWS_MOUSE_PROTO='Tslib:/dev/input/event2'
    #if [ -e /etc/pointercal -a ! -s /etc/pointercal ] ; then
    if [ -e /etc/pointercal ] ; then
        fsize=$(/bin/ls -al /etc/pointercal | cut -d' ' -f 23)
        echo $fsize
        #if [ -s /etc/pointercal ]; then
        if [ $fsize -ge "1" ]; then
            echo "/etc/pointercal is exit"
        else
            echo "/etc/pointercal is empty"
            rm /etc/pointercal
            /usr/local/tslib/bin/ts_calibrate
        fi
    else
        echo "/etc/pointercal not found"
```

```
/usr/local/tslib/bin/ts_calibrate
fi
else
    export QWS_MOUSE_PROTO="MouseMan:/dev/input/mice"
#>/etc/pointercal
fi
```

```
#export QWS_MOUSE_PROTO="Tslib:${TSLIB_TSDEVICE}"
```

```
#/usr/local/tslib/bin/ts_calibrate
```

上面代码第一行，有“！”则支持鼠标；触摸应该不能正常使用。

如果去掉“！”则支持触摸。需要先校准，使用上面这段代码只用校准一次即可，不用每次都校验。

另外如果用户在第一次校准的时候，没有依次点击“十字框”，则触摸文件可能有问题，那么可以参考 7.3.3 小节，删除掉“/etc/point\*”文件，重启开发板再次校准。

修改这段代码之后，最好删除一下“/etc/point\*”文件。例如，如果用户刚开始使用的是触摸，然后改成鼠标控制，因为已经存在“/etc/point\*”文件，所以会让人感觉鼠标可以使用，触摸能够使用但是有问题，这个时候删除“/etc/point\*”文件重启即可。

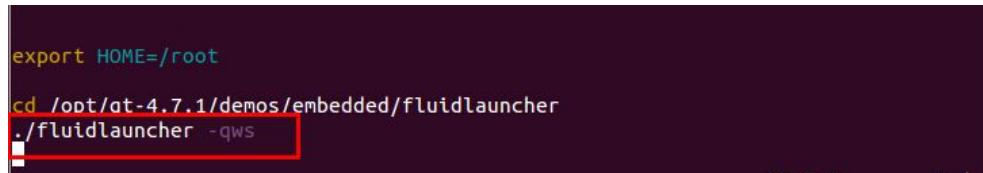
如下图所示，接着找到控制字体的参数。

```
case "$FB_SIZE" in
800,480)
export QWS_DISPLAY="LinuxFb:mmWidth91:mmHeight53:1"
;;
480,272)
export QWS_DISPLAY="LinuxFb:mmWidth76:mmHeight44:1"
;;
*)
export QWS_DISPLAY="LinuxFb:mmWidth91:mmHeight53:1"
;;
esac
#export QWS_DISPLAY=:1
```

如上图，将下面三行删除，避免字体太大导致显示不正常。

```
*)
export QWS_DISPLAY="LinuxFb:mmWidth91:mmHeight53:1"
..
"
```

如下图所示，进入最后一行，红色框默认会启动 QtE 自带的例程，这里修改为“./bin/helloworld -qws”，红色框中的代码记得注释或者删除。



```
export HOME=/root
cd /opt/at-4.7.1/demos/embedded/fluidlauncher
./fluidlauncher -qws
```

### 11.3.3 qt 挂载盘符

#### linux QT 系统下挂载 u 盘

如下图所示，qt 启动之后，在超级终端中使用命令 “mknod /dev/sda1 b 8 1” 创建 U 盘的设备节点。

```
~ #
~ # mknod /dev/sda1 b 8 1
~ # ls /dev/sda1
/dev/sda1
~ #
```

插入 U 盘之后，如下图所示插入 U 盘之后会弹出加载信息。

```
~ # [ 43.520458] usb 1-3.1: new high speed USB device number 4 using s5p-ehci
[ 43.632738] usb 1-3.1: New USB device found, idVendor=0781, idProduct=5567, bcdDevice=0126
[ 43.639719] usb 1-3.1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[ 43.646982] usb 1-3.1: New USB device Class: Class=0, SubClass=0, Protocol=0
[ 43.653952] usb 1-3.1: Product: Cruzer Blade
[ 43.658198] usb 1-3.1: Manufacturer: SanDisk
[ 43.662449] usb 1-3.1: SerialNumber: 20051536210C7A80EADE
[ 43.690501] scsi0 : usb-storage 1-3.1:1.0
[ 44.726244] scsi 0:0:0:0: Direct-Access      SanDisk Cruzer Blade      1.26 PQ: 0 ANSI: 5
[ 44.741224] sd 0:0:0:0: [sda] 7821312 512-byte logical blocks: (4.00 GB/3.72 GiB)
[ 44.750498] sd 0:0:0:0: [sda] Write Protect is off
[ 44.758283] sd 0:0:0:0: [sda] Write cache: disabled, read cache: enabled, doesn't support DPO or FUA
[ 44.766200] sd 0:0:0:0: Attached scsi generic sg0 type 0
[ 44.787387] sda: sda1
[ 44.796859] sd 0:0:0:0: [sda] Attached SCSI removable disk
```

在超级终端中使用命令 “mount /dev/sda1 /mnt/udisk/” 即可挂载 U 盘。

```
~ #
~ # mount /dev/sda1 /mnt/udisk/
~ #
```

如下图所示。使用查找命令 “ls /mnt/udisk” , 可以看到 U 盘中的内容。下图中绿色文件是作者在 U 盘中建的一个小文件 , 这里可以看到。

```
~ # mount /dev/sda1 /mnt/udisk/
~ # ls /mnt/udisk
System Volume Information test_Udisk.txt ↗
~ #
```

## linux QT 系统下挂载 tf 卡

如果用户使用的是 tf 卡 , 如下图所示 , tf 卡插入 tf 卡槽之后会弹出下面的信息。

如下图所示 , 使用加载 tf 卡的命令 “mount /dev/mmcblk1p4 /mnt/udisk1” , 然后使用查找命令 “ls /mnt/udisk1” , 可以看到 tf 卡中的文件。

```
~ # mount /dev/mmcblk1p4 /mnt/udisk1
~ # ls /mnt/udisk1
EFI          IIMT        LMT      bootm32.efi  grldr
GHO          ISO         autorun.inf  bootmgr.efi icon.ico
~ #
```

## linux QT 系统下挂载带 USB 外壳的 tf 卡

如果用户使用的将 tf 卡接入 USB 接口 ( tf 卡加一个 USB 外壳 ) , 那么插入之后 , 如下图所示。这里的是 “**sda4**” , 和 U 盘识别出来的设备节点不完全一样。

```
[ 86.229148] sd 0:0:0:0: [sda] 15564800 512-byte logical blocks: (7.96 GB/7.42
GiB)
[ 86.240993] sd 0:0:0:0: [sda] Write Protect is off
[ 86.256010] sd 0:0:0:0: Attached scsi generic sg0 type 0
[ 86.270239] sd 0:0:0:0: [sda] No Caching mode page present
[ 86.274299] sd 0:0:0:0: [sda] Assuming drive cache: write through
[ 86.343868] sd 0:0:0:0: [sda] No Caching mode page present
[ 86.365108] sd 0:0:0:0: [sda] Assuming drive cache: write through
[ 86.377695] sda: sda4 ↗
[ 86.386074] sd 0:0:0:0: [sda] No Caching mode page present
[ 86.390136] sd 0:0:0:0: [sda] Assuming drive cache: write through
[ 86.398072] sd 0:0:0:0: [sda] Attached SCSI removable disk
```

使用挂载命令 “mount /dev/**sda4** /mnt/udisk” , 然后输入查找命令 “ ls /mnt/udisk” 。可以看到 tf 卡中的内容被是被出来了。

```
~ # mount /dev/sda4 /mnt/udisk
~ # ls /mnt/udisk
EFI          ILMT        LMT      bootm32.efi  grldr
GHO          ISO         autorun.inf bootmgr.efi icon.ico
~ #
```

### 11.3.4 QT 支持 HDMI 显示

Linux-QT 系统默认不支持 HDMI 显示，如果想支持 HDMI 显示，则需要重新配置一下内核以及修改一下文件系统。

#### 配置内核

在内核目录下运行命令 “cp config\_for\_ubuntu\_hdmi .config” ，如下图：

```
iTop4412_Kernel_3.0#
iTop4412_Kernel_3.0#
iTop4412_Kernel_3.0# cp config_for_ubuntu_hdmi .config
```

然后执行 “make” ，开始编译内核，如下图：

```
iTop4412_Kernel_3.0#
iTop4412_Kernel_3.0# make
```

编译完成会生成镜像：zImage

#### 修改 linux-QT 文件系统

如下图所示，进入 linux-QT 文件系统，注意用户自己的文件夹目录可能不一样，根据实际情况进入文件系统即可。

进入到我们的 linux 文件系统，如下图：

```
root@ubuntu:/home/broswer/root#
root@ubuntu:/home/broswer/root# ls
bin dev etc home lib linuxrc mnt opt proc root sbin sys system tmp usr var
root@ubuntu:/home/broswer/root#
root@ubuntu:/home/broswer/root#
```

然后使用命令“ vi bin/qt4 ”，如下图：

```
root@ubuntu:/home/broswer/root#
root@ubuntu:/home/broswer/root# vi bin/qt4
```

然后在 qt4 文件的开始处添加：

```
mv /dev/fb0 /dev/fb0_bak
ln -s /dev/fb6 /dev/fb0
```

如下图所示：

```
#!/bin/sh

mv /dev/fb0 /dev/fb0_bak
ln -s /dev/fb6 /dev/fb0

export TSLIB_ROOT=/usr/local/tslib
export TSLIB_TSDEVICE=/dev/input/event2
export TSLIB_TSEVENTTYPE=input
export TSLIB_CONFFILE=/usr/local/tslib/etc/ts.conf
export TSLIB_PLUGINDIR=/usr/local/tslib/lib/ts
export TSLIB_CALIBFILE=/etc/pointercal
export TSLIB_PLUGINDIR=$TSLIB_ROOT/lib/ts
export TSLIB_CONSOLEDEVICE=none
export TSLIB_FBDEVICE=/dev/fb0
export QTDIR=/opt/qt-4.7.1/
```

然后退出并保存，使用“ make\_ext4fs -s -l 314572800 -a root -L linux system.img

root “命令，生成 system.img 镜像，如下图：

```
root@ubuntu:/home/broswer/root# cd ..
root@ubuntu:/home/broswer#
root@ubuntu:/home/broswer#
root@ubuntu:/home/broswer# make_ext4fs -s -l 314572800 -a root -L linux system.img root
Creating filesystem with parameters:
  Size: 314572800
  Block size: 4096
  Blocks per group: 32768
  Inodes per group: 6400
  Inode size: 256
  Journal blocks: 1200
  Label: linux
  Blocks: 76800
  Block groups: 3
  Reserved block group size: 23
Created filesystem with 10607/19200 inodes and 51616/76800 blocks
root@ubuntu:/home/broswer#
root@ubuntu:/home/broswer#
root@ubuntu:/home/broswer#
```

最后把生成的 zImage 和 system.img 烧写到 iTOP-4412 开发板即可。另外两个文件没有变化，不需要修改。

### 11.3.5 QtE 连接 WIFI

目前光盘中的 QtE 不支持 WIFI，需要到 github 上下载最新的源码，编译烧写 QtE 文件系统之后，按一下步骤即可连接 WIFI。

连接 WIFI 之前确认 WIFI 的名称和密码，启动开发板之后输入命令：

```
#wpa_passphrase XXX "YYY" >> /etc/wpa_supplicant.conf
```

其中 XXX 代表您的 WiFi 网络名称， YYY 代表 WPA-PSK 或者 WPA2-PSK 加密的密码

然后执行 /etc/init.d/mt6620 脚本即可连接到 WiFi 网络。

## 十二 Ubuntu 的应用

### 12.1 烧写 Ubuntu

**学习型用户**不建议学习这一小节，arm-Ubuntu 目前应用不是那么广泛，学习资料有一些欠缺。

#### 12.1.1 TF 卡读写速度测试

烧写 Ubuntu 对于 TF 卡的要求比较高，很多老旧的卡都无法烧写 Ubuntu，下面提供一种相对可靠的 TF 卡进行测试方法。注意：这里的 TF 卡测试只是针对 Ubuntu 的烧写，烧写其它文件系统的时候，一般的 TF 卡就可以。

在网盘文件夹"iTOP-4412 开发板烧写镜像所需要的工具及相关驱动"→"TF 卡测试工具"中可以下载测试软件 "ATTO Disk bench32.exe" 。

这里需要注意的是，TF 卡必须是格式 fat32。

下面具体讲一下测试的方法以及测试参数。

如下图所示，将 TF 卡接入 PC 机，打开测试软件，设置一下参数：

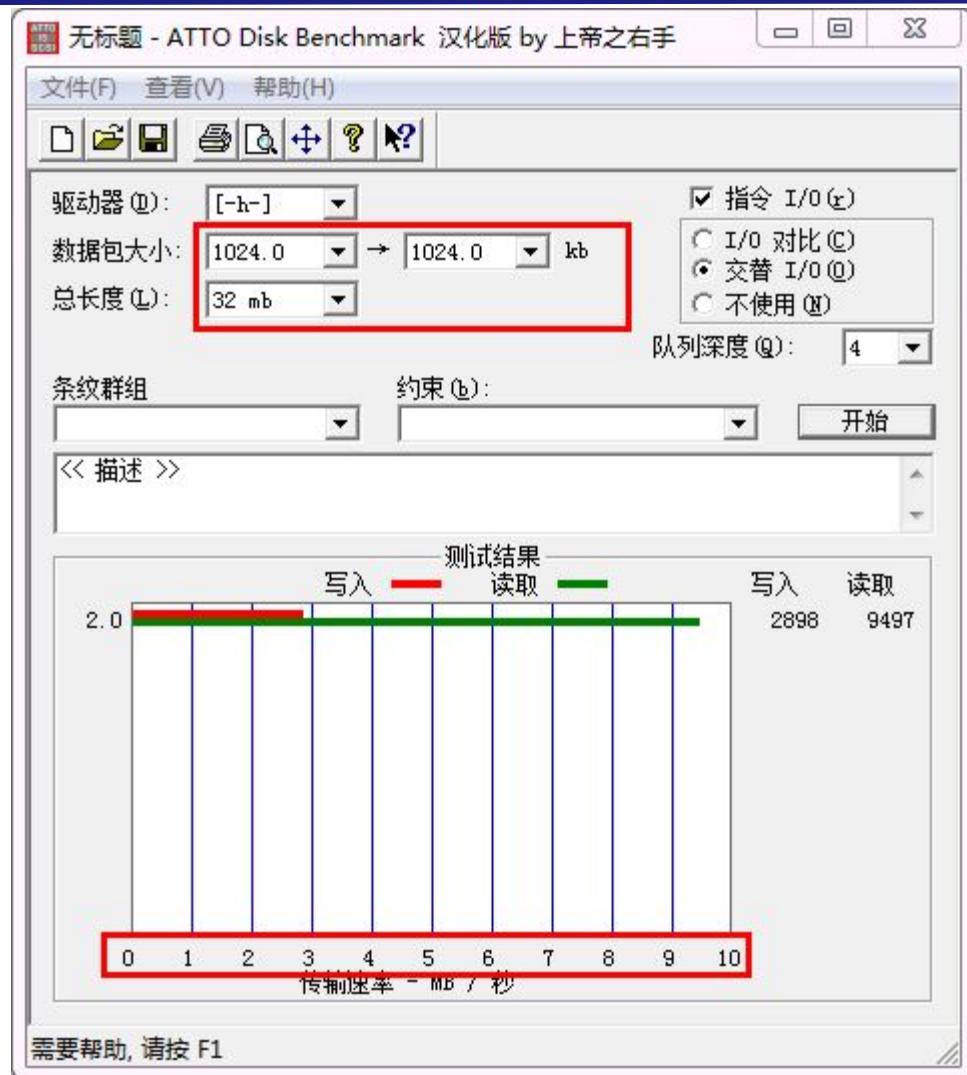
"驱动器" 选择接入 PC 的 TF 卡"

"数据包大小" 选择 "1024->1024"

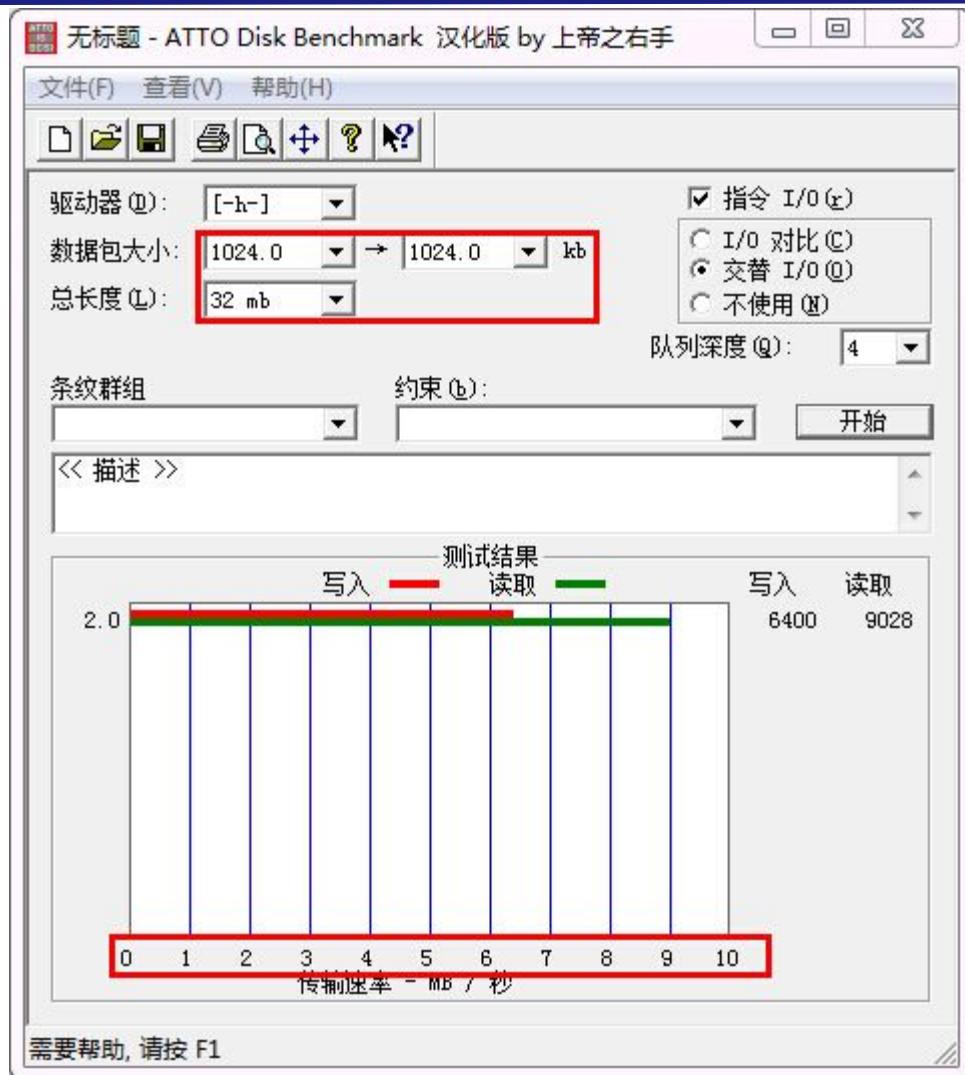
"总长度" 选择 "32mb"

单击按钮 "开始" 测试，结束后，会显示 TF 卡的测试结果。

下图测试的卡 1，在测试后，显示写入速度不够 5MB/秒，不能用于烧写 Ubuntu。



下图所示的卡 2，在测试后，显示写入速度达到了 5MB/秒，可以用于烧写 Ubuntu。



### 12.1.2 烧写 Ubuntu

Ubuntu 文件系统有两个，一个是支持 LCD 的，也就是 7 寸或者 9.7 寸屏幕；或者支持 HDMI 显示的。

LCD 显示版本：iT0P4412\_ubuntu\_12.04\_for\_LCD\_20141230.tar.gz

HDMI 显示版本：iT0P4412\_ubuntu\_12.04\_for\_HDMI\_20141230.tar.gz

想要支持对应的显示器，烧写对应的文件系统就成，烧写方法一样。

如下图所示，在光盘的“05\_镜像\_Ubuntu 文件系统” → “system” 文件夹中有支持 LCD 的 Ubuntu 文件系统。

iTOP4412\_ubuntu\_12.04\_for\_LCD\_20141230.tar.gz

支持 HDMI 的 Ubuntu 文件系统在网盘“ iTOP-4412 开发板系统源码及镜像（其他）”  
→ “支持 HDMI 的 Ubuntu 资料” 中，而且也要配套使用网盘中对应的 support HDMI 的内核。

下面的烧写方式仅仅适用于烧写 Ubuntu 系统。

注意：烧写 Ubuntu 系统对 TF 卡要求较高，建议购买 Class10 的正品 TF 卡。在视频中，演示是使用 2G 的 TF 卡，用户在使用的时候不一定要用 2G 的卡。

烧写 Ubuntu 的平台和前面烧写安卓的平台类似，需要用到 PC 机的 Ubuntu。

1 ) 给 TF 卡分区格式化。先将 TF 卡插入开发板，然后启动开发板，进入 Uboot 模式，给 TF 卡分区，具体操作如下：

如下图所示，输入命令 “fdisk -c 1 2700 50 50”

```
iTOP-4412 #  
iTOP-4412 # fdisk -c 1 2700 50 50  
.fdisk is completed  


| partition # | size(MB) | block start # | block count | partition_Id |
|-------------|----------|---------------|-------------|--------------|
| 1           | 4650     | 5790000       | 9525000     | 0x0C         |
| 2           | 2702     | 45000         | 5535000     | 0x83         |
| 3           | 51       | 5580000       | 105000      | 0x83         |
| 4           | 51       | 5685000       | 105000      | 0x83         |

  
iTOP-4412 #
```

如下图所示，输入命令 “fatformat mmc 1:1”

```
.fdisk is completed  


| partition # | size(MB) | block start # | block count | partition_Id |
|-------------|----------|---------------|-------------|--------------|
| 1           | 4650     | 5790000       | 9525000     | 0x0C         |
| 2           | 2702     | 45000         | 5535000     | 0x83         |
| 3           | 51       | 5580000       | 105000      | 0x83         |
| 4           | 51       | 5685000       | 105000      | 0x83         |

  
iTOP-4412 # fatformat mmc 1:1  
Start format MMC&d partition&d ...  
Partition1: Start Address(0x585930), Size(0x915708)  
.....size checking ...  
Under 8G  
write FAT info: 32  
Fat size : 0x2455  
..Erase FAT region.....  
.....
```

如下图所示，输入命令 “ext3format mmc 1:2”

```
.....  
.Partition1 format complete.  
iTOP-4412 # ext3format mmc 1:2  
Start format MMC1 partition2 ....  
** Partition2 is not ext2 file-system 1 **  
Partition2: Start Address(0xafc8), Size(0x355458)  
Start ext2format...  
Wirte 0/14block-group  
Reserved blocks for journaling : 8202  
Start write addr : 0xafc8  
  
.....  
b338.....  
.....d_indirect_point:0xe310  
..Wirte 1/14block-group  
Reserved blocks for journaling : 8202  
Start write addr : 0x4afc8  
....Erase inode table(1) - 0x4b338.....  
Wirte 2/14block-group
```

如下图所示，输入命令 “ext3format mmc 1:3”

```
Start write addr : 0x34a1c8  
..Erase inode table(13) - 0x34af8.....  
iTOP-4412 # ext3format mmc 1:3  
Start format MMC1 partition3 ....  
** Partition3 is not ext2 file-system 1 **  
Partition3: Start Address(0x360420), Size(0x19a28)  
Start ext2format...  
Wirte 0/1block-group  
Reserved blocks for journaling : 1025  
Start write addr : 0x360420  
.....Erase inode table(0) - 0x360458.....  
.....iTOP-4412 #
```

如下图所示，输入命令 “ext3format mmc 1:4”

```
iTOP-4412 # ext3format mmc 1:4  
Start format MMC1 partition4 ....  
** Partition4 is not ext2 file-system 1 **  
Partition4: Start Address(0x379e48), Size(0x19a28)  
Start ext2format...  
Wirte 0/1block-group  
Reserved blocks for journaling : 1025  
Start write addr : 0x379e48  
.....Erase inode table(0) - 0x379e80.....  
.....iTOP-4412 #
```

2 ) 将 TF 接入 PC 机的 Ubuntu 系统，系统识别 TF 卡后，在 Ubuntu 命令行中输入 Linux 命令 “df -l” ，该命令可以查看到 TF 卡的盘符，TF 卡盘符在前一步中已经分成四个区，这里会显示有四个新的 TF 卡盘符。需要注意的是，在这四个分区中，用户将要使用的是 **2.7G** 大小的 TF 卡盘符，这个 **2.7G** 大小的盘符名在下一步中将要用到。如下图所示。

```
root@ubuntu:~# df -l
Filesystem 1K-blocks Used Available Use% Mounted on
/dev/sda1 205391608 74326808 120631520 39% /
udev 2013744 4 2013740 1% /dev
tmpfs 809208 828 808380 1% /run
none 5120 0 5120 0% /run/lock
none 2023012 200 2022812 1% /run/shm
root@ubuntu:~# df -l
Filesystem 1K-blocks Used Available Use% Mounted on
/dev/sda1 205391608 74327016 120631312 39% /
udev 2013744 4 2013740 1% /dev
tmpfs 809208 848 808360 1% /run
none 5120 0 5120 0% /run/lock
none 2023012 200 2022812 1% /run/shm
/dev/sdb4 50840 4140 44076 9% /media/4ffffbbcb-edd9-e629-52f6-b
ec2f1d05d9d
/dev/sdb3 50840 4140 44076 9% /media/c3ddf5eb-27fa-94c6-c6d4-3
2a165afdf17b
/dev/sdb2 2723980 69664 2515944 3% /media/cc4d381a-6a28-d6f4-43c1-3
b116e1fdaeb
/dev/sdb1 4753180 4 4753176 1% /media/0000-3333
root@ubuntu:~#
```

3 ) 将用户光盘"05\_镜像\_Ubuntu 文件系统"→ "system" 文件夹中的文件

"iTOP4412\_ubuntu\_12.04\_for\_LCD\_xxxxxx.tar.gz"

拷贝到 Ubuntu 系统中 , 如下图所示 , 是将文件拷贝到 "/home/topeet/ubuntu" 中。

```
/dev/sdb4 50840 4140 44076 9% /media/4ffffbbcb-edd9-e629-52f6-b
ec2f1d05d9d
/dev/sdb3 50840 4140 44076 9% /media/c3ddf5eb-27fa-94c6-c6d4-3
2a165afdf17b
/dev/sdb2 2723980 69664 2515944 3% /media/cc4d381a-6a28-d6f4-43c1-3
b116e1fdaeb
/dev/sdb1 4753180 4 4753176 1% /media/0000-3333
root@ubuntu:~# cd /home/topeet/ubuntu/
root@ubuntu:/home/topeet/ubuntu# ls
iTOP4412_ubuntu_12.04_for_LCD_20141230.tar.gz ←
root@ubuntu:/home/topeet/ubuntu#
```

4 ) 将 "iTOP4412\_ubuntu\_12.04\_for\_LCD\_xxxxxx.tar.gz" 拷贝到 2.7G 的文件夹中 ,

如下图所示。这个过程大概一到两分钟。

```
udev 2013744 4 2013740 1% /dev
tmpfs 809208 848 808360 1% /run
none 5120 0 5120 0% /run/lock
none 2023012 200 2022812 1% /run/shm
/dev/sdb4 50840 4140 44076 9% /media/4ffffbbcb-edd9-e629-52f6-b
ec2f1d05d9d
/dev/sdb3 50840 4140 44076 9% /media/c3ddf5eb-27fa-94c6-c6d4-3
2a165afdf17b
/dev/sdb2 2723980 69664 2515944 3% /media/cc4d381a-6a28-d6f4-43c1-3
b116e1fdaeb
/dev/sdb1 4753180 4 4753176 1% /media/0000-3333
root@ubuntu:~# cd /home/topeet/ubuntu/
root@ubuntu:/home/topeet/ubuntu# ls
iTOP4412_ubuntu_12.04_for_LCD_20141230.tar.gz ←
root@ubuntu:/home/topeet/ubuntu# cp -r iTOP4412 ubuntu 12.04 for LCD 20141230.tar
.gz /media/cc4d381a-6a28-d6f4-43c1-3b116e1fdaeb/
```

5 ) 拷贝完成之后 , 进入 tf 卡 2.7G 的目录 , 如下图所示。

```

root@ubuntu: ~ cd /home/topeet/ubuntu/
root@ubuntu:/home/topeet/ubuntu# ls
iT0P4412_ubuntu_12.04_for_LCD_20141230.tar.gz
root@ubuntu:/home/topeet/ubuntu# cp -r iT0P4412_ubuntu_12.04_for_LCD_20141230.tar.gz /media/cc4d381a-6a28-d6f4-43c1-3b116e1fdaeb/
root@ubuntu:/home/topeet/ubuntu# cd /media/cc4d381a-6a28-d6f4-43c1-3b116e1fdaeb/
root@ubuntu:/media/cc4d381a-6a28-d6f4-43c1-3b116e1fdaeb#
root@ubuntu:/media/cc4d381a-6a28-d6f4-43c1-3b116e1fdaeb#
root@ubuntu:/media/cc4d381a-6a28-d6f4-43c1-3b116e1fdaeb# 

```

6 ) 使用解压命令 “tar -xvf xxxx.tar.gz” 解压压缩包文件 , 如下图所示。

```

.gz /media/cc4d381a-6a28-d6f4-43c1-3b116e1fdaeb/
root@ubuntu:/home/topeet/ubuntu# cd /media/cc4d381a-6a28-d6f4-43c1-3b116e1fdaeb/
root@ubuntu:/media/cc4d381a-6a28-d6f4-43c1-3b116e1fdaeb#
root@ubuntu:/media/cc4d381a-6a28-d6f4-43c1-3b116e1fdaeb#
root@ubuntu:/media/cc4d381a-6a28-d6f4-43c1-3b116e1fdaeb# tar -xvf iT0P4412_ubuntu_12.04_for_LCD_20141230.tar.gz 

```

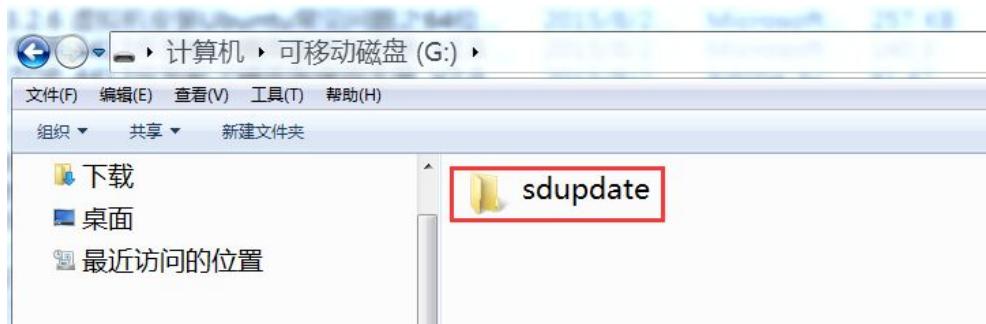
7 ) 上面的解压缩过程一般需要三十分钟左右 , 如果 PC 机的配置不高 , 耗费的时间可能会更长。如下图所示 , 解压完成。

```

var/games/ranjongg/tictactoe.scores
var/games/gnomine.Custom.scores
var/run
var/local/
var/backups/
var/mail/
var/crash/
root@ubuntu:/media/cc4d381a-6a28-d6f4-43c1-3b116e1fdaeb# ls
bin   home          media   root      srv    usr
boot  iT0P4412_ubuntu_12.04_for_LCD_20141230.tar.gz  mnt     run      sys    var
dev   lib           opt     sbin     selinux  system
etc   lost+found    proc
root@ubuntu:/media/cc4d381a-6a28-d6f4-43c1-3b116e1fdaeb# 

```

8 ) 解压缩完成后 , 如下图所示 , 将 tf 卡连接到 window 系统 , 在 TF 卡上建立文件夹 “sdupdate” 。



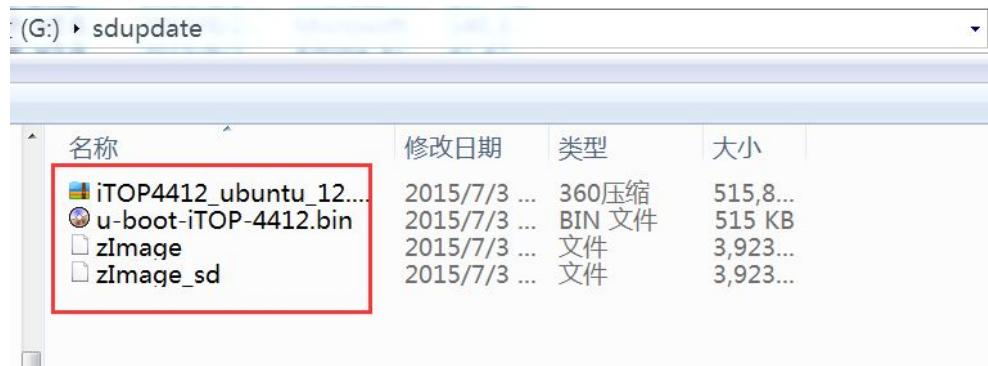
9 ) 拷贝用户光盘 “05\_镜像\_Ubuntu 文件系统” 文件夹中 “uboot” 中对应核心板的镜像 “u-boot-iTOP-4412.bin” 到 TF 卡的文件夹 “sdupdate” 中。

拷贝用户光盘 “05\_镜像\_Ubuntu 文件系统” 文件夹中 “zImage” 中对应核心板的镜像 “zImage\_sd” 和 “zImage” 到 TF 卡的文件夹 “sdupdate” 中。

如果 TF 卡空间的还有富余，可以将 “Ubuntu” 文件夹中文件 “iTOP4412\_ubuntu\_12.04\_for\_LCD\_xxxxxx.tar.gz” 拷贝到 TF 卡的文件夹 “sdupdate” 中。

如果 TF 卡的空间不够，则需要准备 U 盘，将文件 “iTOP4412\_ubuntu\_12.04\_for\_LCD\_xxxxxx.tar.gz” 拷贝到 U 盘中。

如下图所示，使用的是 8G 的 TF 卡，空间有富余，则直接拷贝到 TF 卡中。



10 ) 先将 TF 卡插入开发板（如果前一步使用了 U 盘，则需要将 U 盘插入靠近 TF 座的 USB 接口），再启动开发板，进入 uboot 模式，给 eMMC 分区和烧写镜像，具体操作如下：

在超级终端中，执行下面命令，类似前面的操作，这里就只提供操作命令，不截图了。

- fdisk -c 0 2700 300 300
- fatformat mmc 0:1
- ext3format mmc 0:2
- ext3format mmc 0:3
- ext3format mmc 0:4

11 ) 分区完成之后，在超级终端中，执行下面烧写 uboot 的命令。

“sdfuse flash bootloader u-boot-iTOP-4412.bin” 如下图所示。

```

..Erase inode table(2) - 0x4728c6.....
iT0P-4412 # sdfuse flash bootloader u-boot-iTOP-4412.bin
SD sclk_mmc is 400K HZ
SD sclk_mmc is 50000K HZ
SD sclk_mmc is 50000K HZ
[Fusing Image from SD Card.]
[Partition table on MoviNAND]
ptn 0 name='bootloader' start=0x0 len=N/A (use hard-coded info. (cm
ptn 1 name='kernel' start=N/A len=N/A (use hard-coded info. (cmd: m
ptn 2 name='ramdisk' start=N/A len=0x300000 (~3072KB) (use hard-code
ptn 3 name='Recovery' start=N/A len=0x600000 (~6144KB) (use hard-cod
ptn 4 name='system' start=0x1235400 len=0x6A4A0C00 (~1741443KB)
ptn 5 name='userdata' start=0x6B6D6000 len=0x12E40C00 (~309507KB)
ptn 6 name='cache' start=0x7E516C00 len=0x12E40C00 (~309507KB)
ptn 7 name='fat' start=0x91357800 len=0x56F17800 (~1424478KB)
Partition: bootloader, File: /sdupdate/u-boot-iTOP-4412.bin
Partition1: Start Address(0x585930), Size(0x915708)
reading /sdupdate/u-boot-iTOP-4412.bin

```

12 ) uboot 烧写完成 , 如下图所示。

```

MMC write: dev # 0, block # 0, count 1038. 1038 blocks write finish
1038 blocks verify1: OK
eMMC CLOSE Success.!!
completed
partition 'bootloader' flashed
iT0P-4412 #

```

13 ) 如下图所示 , 烧写 sd 卡启动的内核 “sdfuse flash kernel zImage\_sd”

```

partition 'bootloader' flashed
iT0P-4412 # sdfuse flash kernel zImage_sd
SD sclk_mmc is 400K HZ
SD sclk_mmc is 50000K HZ
SD sclk_mmc is 50000K HZ
[Fusing Image from SD Card.]
[Partition table on MoviNAND]
ptn 0 name='bootloader' start=0x0 len=N/A (use hard-coded i
ptn 1 name='kernel' start=N/A len=N/A (use hard-coded info.
ptn 2 name='ramdisk' start=N/A len=0x300000 (~3072KB) (use h
ptn 3 name='Recovery' start=N/A len=0x600000 (~6144KB) (use
ptn 4 name='system' start=0x1235400 len=0x6A4A0C00 (~1741443
ptn 5 name='userdata' start=0x6B6D6000 len=0x12E40C00 (~3095
ptn 6 name='cache' start=0x7E516C00 len=0x12E40C00 (~309507K
ptn 7 name='fat' start=0x91357800 len=0x56F17800 (~1424478KB
Partition: kernel, File: /sdupdate/zImage_sd
Partition1: Start Address(0x585930), Size(0x915708)
reading /sdupdate/zImage_sd
4017032 (0x003d4b88) bytes read
flashing 'kernel'
writing kernel.. 1120, 12288
MMC write: dev # 0, block # 1120, count 12288. 12288 blocks
12288 blocks written: OK
completed
partition 'kernel' flashed
iT0P-4412 #

```

需要注意的是，这里烧写的内核镜像是“zImage\_sd”文件。

14 ) 输入重启开发板命令 “reset” ，如下图所示。

```
401'032 (0x003d4b88) bytes read
flashing 'kernel'
writing kernel.. 1120, 12288
MMC write: dev # 0, block # 1120, count 12288. 12288 blocks write finish
12288 blocks written: OK
completed
partition 'kernel' flashed
iT0P-4412 # reset
```

15 ) 重启开发板后，开发板会运行 Ubuntu 系统。超级终端出现如下所示打印信息，则表明 Ubuntu 已经启动。

```
[ 22.487126] init: Failed to create pty - disabling logging for j
[ 22.501887] init: Failed to create pty - disabling logging for j
[ 22.672832] init: Failed to create pty - disabling logging for j
[ 22.691558] init: Failed to create pty - disabling logging for j
Welcome to Linaro 12.04 (GNU/Linux 3.0.15 armv7l)

* Documentation: https://wiki.linaro.org/

0 packages can be updated.
0 updates are security updates.

0 packages can be updated.
0 updates are security updates.
```



16 ) 如下图所示，超级终端中使用命令 “df -l” ，查找到有 update 文件夹的盘符。

```
root@iT0P4412-ubuntu-desktop:~# df -l
Filesystem      1K-blocks    Used Available Use% Mounted on
/dev/root        2723980  2031812     553796  79% /
none            307048       4    307044   1% /dev
none            61412     676    60736   2% /run
none            5120        0    5120    0% /run/lock
none            307048      24    307024   1% /run/shm
/dev/mmcblk0p1   398208       4    398204   1% /media/0000-3333
/dev/mmcblk1p1  4753180  524244   4228936  12% /media/0000-3333
/dev/mmcblk0p3  299780   16576   267732   6% /media/c014f322-5fef-cbbb-fec9-30d863e6cfb2
/dev/mmcblk1p3  50840    4140    44076   9% /media/c3ddf5eb-27fa-94c6-c6d4-32a165afdf17b
/dev/mmcblk1p4  50840    4140    44076   9% /media/4ffffbbcb-edd9-e629-52f6-bec2f1d05d9d
/dev/mmcblk0p2  2723396  69664   2515388  3% /media/2562922e-c43c-3009-6317-952501f26dbe
/dev/mmcblk0p4  299780   16576   267732   6% /media/3aab677-1244-7f10-b11e-1deb50f9bcc5
```

17 ) 如下图所示，使用 cd 命令进入有 update 的盘符，具体盘符名称以用户实际为准。

```

root@iTOP4412-ubuntu-desktop:~# df -l
Filesystem      1K-blocks  Used Available Use% Mounted on
/dev/root        2723980 2031812   553796  7% /
none             307048     4   307044  1% /dev
none             61412    676   60736  2% /run
none              5120     0   5120  0% /run/lock
none             307048    24   307024  1% /run/shm
/dev/mmcblk0p1   398208     4   398204  1% /media/0000-3333
/dev/mmcblk0p1   4753180 524244  4228936 12% /media/0000-3333
/dev/mmcblk0p3   299780 16576   267732  6% /media/c014f322-5fef-cbbb-fec9-30d863e6cfb2
/dev/mmcblk1p3   50840    4140   44076  9% /media/c3ddf5eb-27fa-94c6-c6d4-32a165af17b
/dev/mmcblk1p4   50840    4140   44076  9% /media/4ffffbbcb-edd9-e629-52f6-bec2f1d05d9d
/dev/mmcblk0p2   2723396 69664   2515388 3% /media/2562922e-c43c-3009-6317-952501f26dbe
/dev/mmcblk0p4   299780 16576   267732  6% /media/3aab677-1244-7f10-b1le-1deb50f9bcc5
root@iTOP4412-ubuntu-desktop:~# cd /media/0000-3333 /sdupdate/
root@iTOP4412-ubuntu-desktop:/media/0000-3333/_sdupdate# ls
iTOP4412_ubuntu_12.04_for_LCD_20141230.tar.gz  zImage
u-boot-iTOP-4412.bin                            zImage_sd
root@iTOP4412-ubuntu-desktop:/media/0000-3333/_sdupdate#

```

18 ) 如下图所示 , 可以看到 “/dev/mmcblk0p2” 大小为 2.7G。

```

root@iTOP4412-ubuntu-desktop:~# df -l
Filesystem      1K-blocks  Used Available Use% Mounted on
/dev/root        2723980 2031812   553796  7% /
none             307048     4   307044  1% /dev
none             61412    676   60736  2% /run
none              5120     0   5120  0% /run/lock
none             307048    24   307024  1% /run/shm
/dev/mmcblk0p1   398208     4   398204  1% /media/0000-3333
/dev/mmcblk0p1   4753180 524244  4228936 12% /media/0000-3333
/dev/mmcblk0p3   299780 16576   267732  6% /media/c014f322-5fef-cbbb-fec9-30d863e6cfb2
/dev/mmcblk1p3   50840    4140   44076  9% /media/c3ddf5eb-27fa-94c6-c6d4-32a165af17b
/dev/mmcblk1p4   50840    4140   44076  9% /media/4ffffbbcb-edd9-e629-52f6-bec2f1d05d9d
/dev/mmcblk0p2   2723396 69664   2515388 3% /media/2562922e-c43c-3009-6317-952501f26dbe
/dev/mmcblk0p4   299780 16576   267732  6% /media/3aab677-1244-7f10-b1le-1deb50f9bcc5

```

19 ) 使用 cp 命令将压缩包拷贝的上图对应的 eMMC 的 2.7G 盘符中 , 这个过程可能需要花费几分钟。

如下图所示 , 进入 eMMC 的 2.7G 盘符中 , 可以看到压缩包已经被拷贝进去。

```

root@iTOP4412-ubuntu-desktop:/media/2562922e-c43c-3009-6317-952501f26dbe#
root@iTOP4412-ubuntu-desktop:/media/2562922e-c43c-3009-6317-952501f26dbe#
root@iTOP4412-ubuntu-desktop:/media/2562922e-c43c-3009-6317-952501f26dbe#
root@iTOP4412-ubuntu-desktop:/media/2562922e-c43c-3009-6317-952501f26dbe#
root@iTOP4412-ubuntu-desktop:/media/2562922e-c43c-3009-6317-952501f26dbe# ls
iTOP4412_ubuntu_12.04_for_LCD_20141230.tar.gz  lost+found
root@iTOP4412-ubuntu-desktop:/media/2562922e-c43c-3009-6317-952501f26dbe#

```

20 ) 在该文件中使用解压命令

“tar -xvf iTOP4412\_ubuntu\_12.04\_for\_LCD\_20141230.tar.gz” , 将压缩包解压到当前目录中 , 这个过程大概有三十分钟。如下图所示 , 解压完成。

```

tar: etc: time stamp 2014-04-21 08:32:19 is 72689179.9405765 s in the future
tar: dev: time stamp 2012-04-26 08:02:12 is 10047372.939969291 s in the future
tar: bin: time stamp 2012-04-26 08:39:23 is 10049603.939475791 s in the future
root@iTOP4412-ubuntu-desktop:/media/2562922e-c43c-3009-6317-952501f26dbe#
root@iTOP4412-ubuntu-desktop:/media/2562922e-c43c-3009-6317-952501f26dbe#
root@iTOP4412-ubuntu-desktop:/media/2562922e-c43c-3009-6317-952501f26dbe#
root@iTOP4412-ubuntu-desktop:/media/2562922e-c43c-3009-6317-952501f26dbe# ls
bin  iTOP4412_ubuntu_12.04_for_LCD_20141230.tar.gz  opt  selinux  usr
boot  lib                           proc  srv  var
dev   lost+found                     root  sys
etc   media                         run  system
home  mnt                           sbin  tmp
root@iTOP4412-ubuntu-desktop:/media/2562922e-c43c-3009-6317-952501f26dbe#

```

21 ) 在解压缩完成后，重启开发板，进入 uboot 模式，将内核镜像 “zImage” 烧写到 eMMC 中，如下图所示 “sdfuse flash kernel zImage”

需要注意的是，这里的内核镜像是 “zImage” 。

```
iTOP-4412 #
iTOP-4412 # sdfuse flash kernel zImage
SD sclk_mmc is 400K HZ
SD sclk_mmc is 50000K HZ
SD sclk_mmc is 50000K HZ
[Fusing Image from SD Card.]
[Partition table on MoviNAND]
ptn 0 name='bootloader' start=0x0 len=N/A (use hard-coded info. (cmd: movi))
ptn 1 name='kernel' start=N/A len=N/A (use hard-coded info. (cmd: movi))
ptn 2 name='ramdisk' start=N/A len=0x300000 (~3072KB) (use hard-coded info. (cmd: movi))
ptn 3 name='Recovery' start=N/A len=0x600000 (~6144KB) (use hard-coded info. (cmd: movi))
ptn 4 name='system' start=0x1235400 len=0xA8E11800 (~2766918KB)
ptn 5 name='userdata' start=0xAA046C00 len=0x12E40C00 (~309507KB)
ptn 6 name='cache' start=0xBCE87800 len=0x12E40C00 (~309507KB)
ptn 7 name='fat' start=0xCFCC8400 len=0x185A6C00 (~399003KB)
Partition: kernel, File: /sdupdate/zImage
Partition1: Start Address(0x585930), Size(0x915708)
reading /sdupdate/zImage
4017032 (0x003d4b88) bytes read
flashing 'kernel'
writing kernel.. 1120, 12288
MMC write: dev # 0, block # 1120, count 12288. 12288 blocks write finish
12288 blocks written: OK
completed
partition 'kernel' flashed
iTOP-4412 #
```

22 ) 最后重启开发板，Ubuntu 系统重启后就可以在开发板上运行起来了，烧写 Ubuntu 系统到此结束。

## 12.2 Ubuntu 的 uboot 以及内核编译

### 12.2.1 uboot 的编译

Ubuntu 系统对应 uboot，源码，编译器和 Android4.0.3 一模一样。参考使用手册 5.3.1 小节。

#### 12.2.1.1 参数配置

编译 uboot 的脚本是源码文件夹中的 “buid\_uboot.sh” ，在编译的时候需要向脚本传参数，根据核心板的不同，脚本执行参数如下表所示。

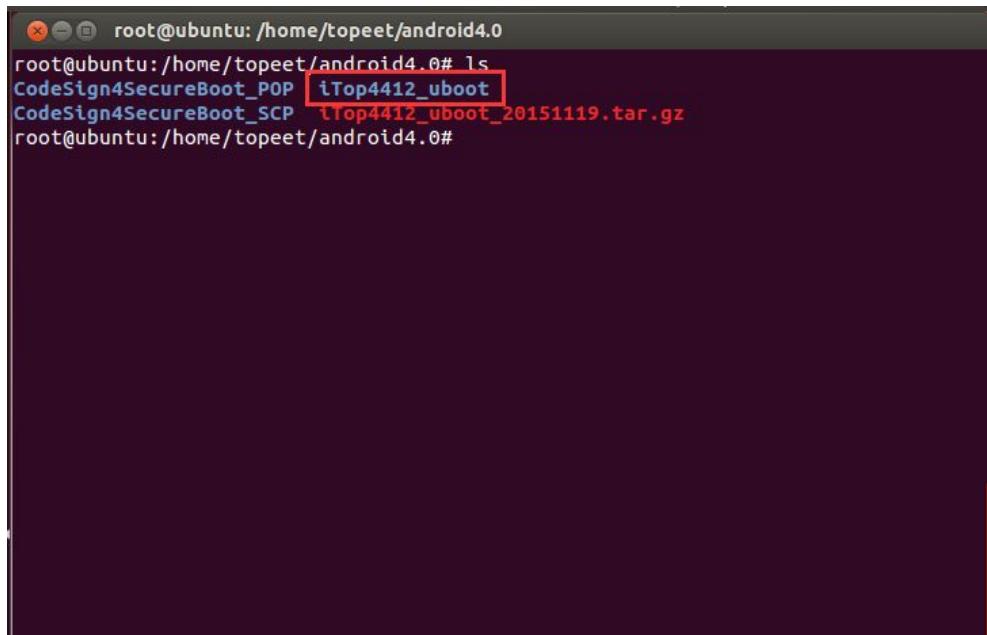
硬件分类	脚本执行参数
------	--------

核心板 SCP 1G 内存	SCP_1GDDR_Ubuntu
核心板 SCP 2G 内存	SCP_2GDDR_Ubuntu
核心板 POP 1G 内存	POP_1GDDR_Ubuntu

### 5.3.1.4 编译生成 uboot 镜像举例

这里以 SCP 1G 核心板为例编译 uboot 镜像。

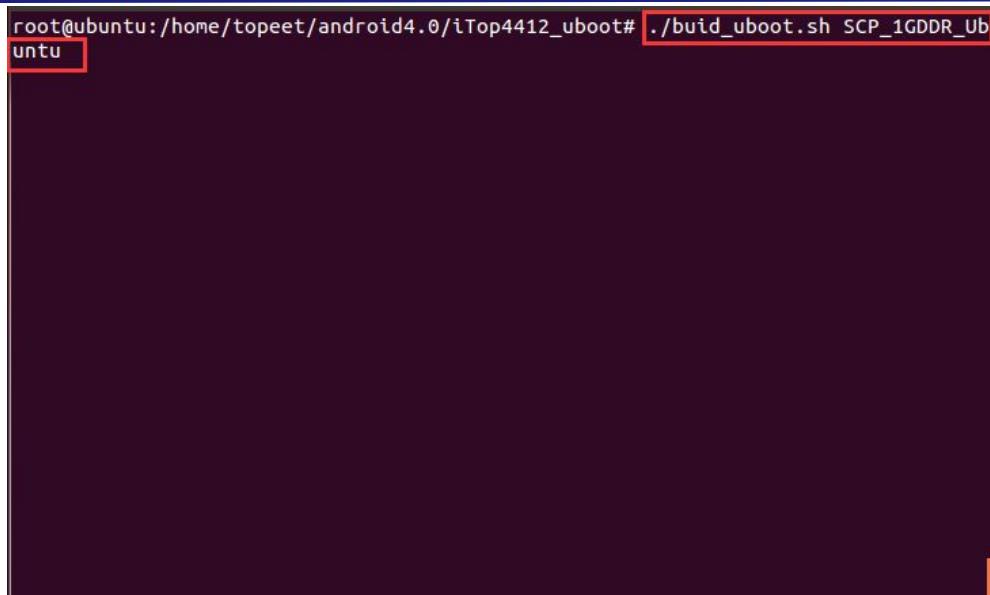
将光盘 “06\_源码\_uboot 和 kernel” 目录下 “CodeSign4SecureBoot\_POP” 、 “CodeSign4SecureBoot\_SCP” 以及 “iT0p4412\_uboot\_xxx.tar.gz” 拷贝到 Ubuntu 系统下，然后将 “iT0p4412\_uboot\_xxx.tar.gz” 解压，得到 “iT0p4412\_uboot” 文件夹，如下图所示。



进入 “iT0p4412\_uboot” 文件夹，使用编译脚本 “buid\_uboot.sh” 编译 uboot，这里需要编译的是 “SCP 1G 核心板” 的 uboot 镜像，那么编译命令是

“./build\_uboot.sh SCP\_1GDDR\_Ubuntu”

输入编译命令，如下图所示。这里一定先确定核心板是哪种类型，然后将对应的参数传到脚本。



如下图所示，编译中。

```
CoreBoard OS is Ubuntu.....  
Configuring for itop_4412_ubuntu board...  
Generating include/autoconf.mk  
Generating include/autoconf.mk.dep  
for dir in cpu/arm_cortexa9 /home/topeet/android4.0/iTop4412_uboot/cpu/arm_cortexa9/ ; do \  
    make -C $dir _depend ; done  
make[1]: Entering directory `/home/topeet/android4.0/iTop4412_uboot/cpu/arm_cortexa9'  
make[1]: Leaving directory `/home/topeet/android4.0/iTop4412_uboot/cpu/arm_cortexa9'  
make[1]: Entering directory `/home/topeet/android4.0/iTop4412_uboot/cpu/arm_cortexa9'  
make[1]: Nothing to be done for `_depend'.  
make[1]: Leaving directory `/home/topeet/android4.0/iTop4412_uboot/cpu/arm_cortexa9'  
make[1]: Entering directory `/home/topeet/android4.0/iTop4412_uboot/cpu/arm_cortexa9'  
make[1]: Nothing to be done for `_depend'.  
make[1]: Leaving directory `/home/topeet/android4.0/iTop4412_uboot/cpu/arm_cortexa9'  
make -C cpu/arm_cortexa9 start.o  
make -C lib_generic/
```

如下图所示，脚本执行完成，在“iTop4412\_uboot”文件夹下生成了“u-boot-iTOP-4412.bin”文件。生成的文件“u-boot-iTOP-4412.bin”文件就是SCP 1G 内存核心板对应的uboot镜像文件。

```
root@ubuntu:/home/topeet/android4.0/iTop4412_uboot# ls
all00_padding.bin    examples      lib_sh      rules.mk
api                  fs           lib_sparc   sdfuse
board                include      MAINTAINERS sdfuse.d
build_uboot.sh       lib_arm     MAKEALL     System.map
checksum_bl2_14k.bin lib_avr32   Makefile    tc4_cmm.cmm
common               lib_blackfin mkbbl2     tools
config.mk            lib_fdt     mkconfig   u-boot
COPYING              lib_generic mkuboot   u-boot.bin
cpu                  lib_i386    nand_spl   u-boot.lds
CREDITS              lib_m68k    net        u-boot.map
disk                 lib_microblaze onenand_ipl u-boot.readme.txt
doc                  lib_mips    paddingaa  u-boot.srec
drivers              lib_nios    post      u-boot.srec
E4212                lib_nios2   README    u-boot.txt
E4412_N.bl1.bin      lib_ppc     readme.txt
root@ubuntu:/home/topeet/android4.0/iTop4412_uboot#
```

## 12.2.2 Linux 内核的编译

Ubuntu 系统对应 uboot , 源码 , 编译器和 Android4.0.3 一模一样。参考使用手册 5.3.1 小节。

### 12.2.2.1 参数配置

内核的编译是组合式配置文件，基本的配置文件名是 “config\_for\_ubuntu\_YY” ，YY 表示用下表所示的参数替代。

硬件分类	配置文件
核心板 SCP 1G 或者 2G 内存 , LCD 显示	config_for_ubuntu_scp
核心板 SCP 1G 或者 2G 内存 , HDMI 显示	config_for_ubuntu_hdmi_scp
核心板 POP 内存 , LCD 显示	config_for_ubuntu_pop
核心板 POP 内存 , HDMI 显示	config_for_ubuntu_hdmi_pop

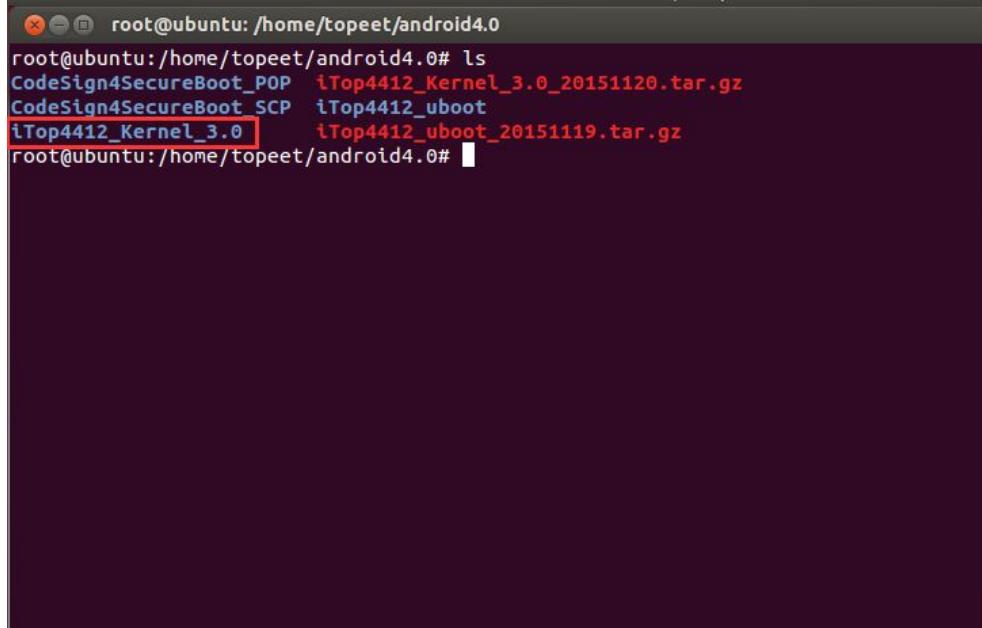
### 5.3.2.4 编译生成内核镜像举例

这里以 SCP 1G 核心板为例编译 zImage 内核镜像,想要生成 LCD 显示的镜像 , 那么配置文件为 “config\_for\_ubuntu\_scp” 。

将光盘“06\_源码\_uboot 和 kernel”目录下的压缩包

“iT0p4412\_Kernel\_3.0\_xxx.tar.gz”拷贝到 Ubuntu，然后解压，得到文件夹

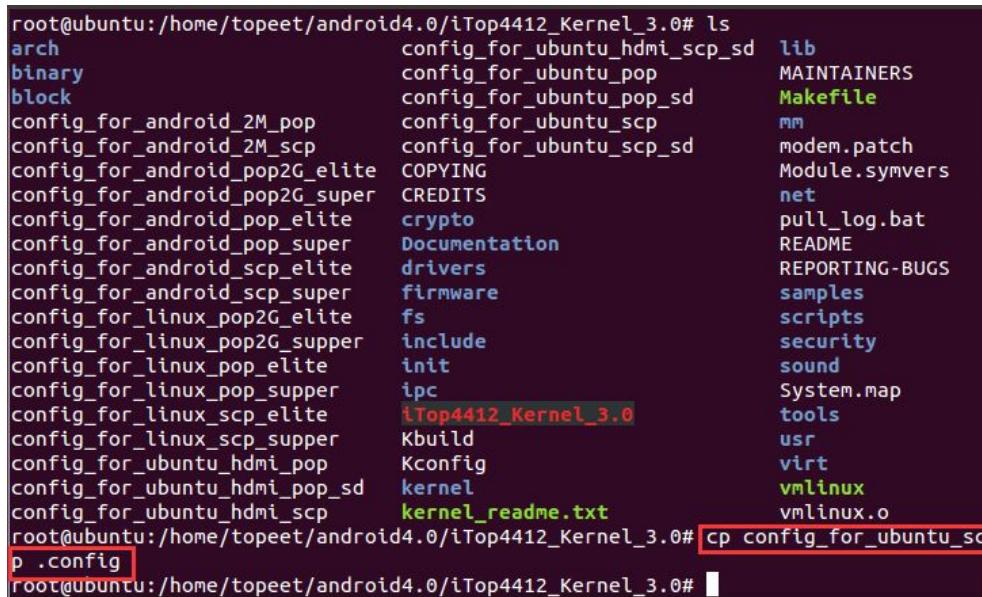
“iT0p4412\_Kernel\_3.0”，如下图所示。



```
root@ubuntu:/home/topeet/android4.0
root@ubuntu:/home/topeet/android4.0# ls
CodeSign4SecureBoot_POP  iT0p4412_Kernel_3.0_20151120.tar.gz
CodeSign4SecureBoot_SCP  iT0p4412_uboot
iT0p4412_Kernel_3.0      iT0p4412_uboot_20151119.tar.gz
root@ubuntu:/home/topeet/android4.0#
```

进入文件夹“iT0p4412\_Kernel\_3.0”，使用命令

“cp config\_for\_ubuntu\_scp .config” 覆盖自带的配置文件，如下图所示。



```
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0# ls
arch          config_for_ubuntu_hdmi_scp_sd lib
binary        config_for_ubuntu_pop      MAINTAINERS
block         config_for_ubuntu_pop_sd   Makefile
config_for_android_2M_pop config_for_ubuntu_scp mm
config_for_android_2M_scp config_for_ubuntu_scp_sd modem.patch
config_for_android_pop2G_elite COPYING Module.symvers
config_for_android_pop2G_super CREDITS net
config_for_android_pop_elite crypto pull_log.bat
config_for_android_pop_super Documentation README
config_for_android_scp_elite drivers REPORTING-BUGS
config_for_android_scp_super firmware samples
config_for_linux_pop2G_elite fs scripts
config_for_linux_pop2G_supper include security
config_for_linux_pop_elite init sound
config_for_linux_pop_supper ipc System.map
config_for_linux_scp_elite iTop4412_Kernel_3.0 tools
config_for_linux_scp_supper Kbuild usr
config_for_ubuntu_hdmi_pop Kconfig virt
config_for_ubuntu_hdmi_pop_sd kernel vmlinuz
config_for_ubuntu_hdmi_scp kernel_readme.txt vmlinuz.o
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0# cp config_for_ubuntu_scp
p .config
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0#
```

然后使用编译命令“make zImage”，如下图所示。

```
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0# ls
arch           config_for_ubuntu_hdmi_scp_sd    lib
binary         config_for_ubuntu_pop          MAINTAINERS
block          config_for_ubuntu_pop_sd        Makefile
config_for_android_2M_pop   config_for_ubuntu_scp      mm
config_for_android_2M_scp    config_for_ubuntu_scp_sd    modem.patch
config_for_android_pop2G_elite  COPYING            Module.symvers
config_for_android_pop2G_super CREDITS           net
config_for_android_pop_elite   crypto             pull_log.bat
config_for_android_pop_super  Documentation       README
config_for_android_scp_elite   drivers            REPORTING-BUGS
config_for_android_scp_super   firmware           samples
config_for_linux_pop2G_elite   fs                 scripts
config_for_linux_pop2G_super  include            security
config_for_linux_pop_elite    init               sound
config_for_linux_pop_supper   ipc                System.map
config_for_linux_scp_elite    iTop4412_Kernel_3.0  tools
config_for_linux_scp_supper   Kbuild            usr
config_for_ubuntu_hdmi_pop   Kconfig            virt
config_for_ubuntu_hdmi_pop_sd kernel             vmlinux
config_for_ubuntu_hdmi_scp   kernel_readme.txt  vmlinux.o
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0# cp config_for_ubuntu_sc
p .config
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0# make zImage
```

编译中，如下图所示。

```
config_for_ubuntu_hdmi_scp      kernel_readme.txt          vmlinux.o
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0# cp config_for_ubuntu_sc
p .config
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0# make zImage
scripts/kconfig/conf --silentoldconfig Kconfig
  CHK  include/linux/version.h
  CHK  include/generated/utsrelease.h
make[1]: `include/generated/mach-types.h' is up to date.
  CALL  scripts/checksyscalls.sh
  HOSTCC scripts/commakehash
  CHK  include/generated/compile.h
  CC   init/do_mounts.o
  LD   init/mounts.o
  LD   init/built-in.o
  CC   arch/arm/kernel/process.o
arch/arm/kernel/process.c:132: warning: function declaration isn't a prototype
  CC   arch/arm/kernel/setup.o
  LD   arch/arm/kernel/built-in.o
  CC   arch/arm/mach-exynos/pm-exynos4.o
arch/arm/mach-exynos/pm-exynos4.c: In function 'exynos4_pm_resume':
arch/arm/mach-exynos/pm-exynos4.c:341: warning: format '%08x' expects type 'unsi
gned int', but argument 2 has type 'long unsigned int'
  CC   arch/arm/mach-exynos/cpufreq.o
```

编译完成，如下图所示。

```
CC      init/version.o
LD      init/built-in.o
LD      .tmp_vmlinux1
KSYM   .tmp_kallsyms1.S
AS      .tmp_kallsyms1.o
LD      .tmp_vmlinux2
KSYM   .tmp_kallsyms2.S
AS      .tmp_kallsyms2.o
LD      vmlinux
SYSMAP System.map
SYSMAP .tmp_System.map
OBJCOPY arch/arm/boot/Image
Kernel: arch/arm/boot/Image is ready
AS      arch/arm/boot/compressed/head.o
GZIP   arch/arm/boot/compressed/piggy.gzip
AS      arch/arm/boot/compressed/piggy.gzip.o
CC      arch/arm/boot/compressed/misc.o
CC      arch/arm/boot/compressed/decompress.o
SHIPPED arch/arm/boot/compressed/lib1funcs.S
AS      arch/arm/boot/compressed/lib1funcs.o
LD      arch/arm/boot/compressed/vmlinux
OBJCOPY arch/arm/boot/zImage
Kernel: arch/arm/boot/zImage is ready
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0#
```

文件夹 “iTop4412\_Kernel\_3.0” 下的 “arch” --> “arm” --> “boot” 会生成镜像文件 “zImage” , 这个 zImage 镜像可以给 **SCP 1G 和 SCP 2G** 的核心板使用 , 如下图所示。

```
LD      .tmp_vmlinux1
KSYM   .tmp_kallsyms1.S
AS      .tmp_kallsyms1.o
LD      .tmp_vmlinux2
KSYM   .tmp_kallsyms2.S
AS      .tmp_kallsyms2.o
LD      vmlinux
SYSMAP System.map
SYSMAP .tmp_System.map
OBJCOPY arch/arm/boot/Image
Kernel: arch/arm/boot/Image is ready
AS      arch/arm/boot/compressed/head.o
GZIP   arch/arm/boot/compressed/piggy.gzip
AS      arch/arm/boot/compressed/piggy.gzip.o
CC      arch/arm/boot/compressed/misc.o
CC      arch/arm/boot/compressed/decompress.o
SHIPPED arch/arm/boot/compressed/lib1funcs.S
AS      arch/arm/boot/compressed/lib1funcs.o
LD      arch/arm/boot/compressed/vmlinux
OBJCOPY arch/arm/boot/zImage
Kernel: arch/arm/boot/zImage is ready
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0# ls arch/arm/boot/
bootp compressed Image install.sh Makefile zImage
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0#
```

## 12.3 Ubuntu 下使用 wifi

用户使用 Ubuntu 的时候 , 需要修改一下配置 , wifi 才能使用。

需要注意的是，全能版 V3.5 及其以后的版本，需要按照精英版 wifi 配置，因为 V3.5 的全能版和精英版的 wifi 都是 mt6620 了。如果是以前版本的全能版，则需要使用全能版的修改方法。

### 12.3.1 精英版 wifi 配置

开发板上电启动进入到 Ubuntu 系统后，在开发板的串口里面使用命令 “vi /etc/init.d/itop-set” ，如下图：

```
root@iTOP4412-ubuntu-desktop: #
root@iTOP4412-ubuntu-desktop:~#
root@iTOP4412-ubuntu-desktop:~# vi /etc/init.d/itop-set
```

然后打开 itop-set 文件，把里面的内容：

```
# mknod /dev/stpwmt c 190 0
# mknod /dev/stpgps c 191 0
# mknod /dev/fm c 193 0
# mknod /dev/wmtWifi c 194 0

# insmod /system/lib/modules/mtk_hif_sdio.ko
# insmod /system/lib/modules/mtk_stp_wmt.ko
# insmod /system/lib/modules/mtk_stp_uart.ko
# insmod /system/lib/modules/mtk_stp_gps.ko
# insmod /system/lib/modules/mt6620_fm_drv.ko
# insmod /system/lib/modules/mtk_fm_priv.ko
# insmod /system/lib/modules/mtk_wmt_wifi.ko WIFI_major=194
# insmod /system/lib/modules/wlan.ko

# chmod 0666 /dev/stpwmt
# chmod 0666 /dev/stpgps
# chmod 0666 /dev/fm
# chmod 0666 /dev/wmtWifi
# chmod 0660 /dev/ttySAC0
```

```
# chmod 0666 /dev/gps  
  
# /system/bin/6620_launcher -m 1 -b 921600 -n  
/system/etc/firmware/mt6620_patch_hdr.bin -d /dev/ttySAC0 &  
  
# sleep 1  
  
# echo 1 > /dev/wmtWifi
```

```
# wpa_supplicant -d -Dwext -iwlan0 -c /etc/wpa_supplicant.conf &  
# dhclient wlan0 &
```

修改成下面的内容（其他的地方不要修改，因为这个文件里也包括全功能板的 wifi 配置）：

```
mknod /dev/stpwmt c 190 0  
mknod /dev/stpgps c 191 0  
mknod /dev/fm c 193 0  
mknod /dev/wmtWifi c 194 0
```

```
insmod /system/lib/modules/mtk_hif_sdio.ko  
insmod /system/lib/modules/mtk_stp_wmt.ko  
insmod /system/lib/modules/mtk_stp_uart.ko  
insmod /system/lib/modules/mtk_stp_gps.ko  
insmod /system/lib/modules/mt6620_fm_drv.ko  
insmod /system/lib/modules/mtk_fm_priv.ko  
insmod /system/lib/modules/mtk_wmt_wifi.ko WIFI_major=194  
insmod /system/lib/modules/wlan.ko
```

```
chmod 0666 /dev/stpwmt  
chmod 0666 /dev/stpgps  
chmod 0666 /dev/fm  
chmod 0666 /dev/wmtWifi
```

```
chmod 0660 /dev/ttySAC0
```

```
chmod 0666 /dev/gps
```

```
/system/bin/6620_launcher -m 1 -b 921600 -n
```

```
/system/etc/firmware/mt6620_patch_hdr.bin -d /dev/ttySAC0 &
```

```
sleep 1
```

```
echo 1 > /dev/wmtWifi
```

```
wpa_supplicant -d -Dwext -iwlan0 -c /etc/wpa_supplicant.conf &  
dhclient wlan0 &
```

修改完成以后保存并退出，然后使用命令“vi /etc/wpa\_supplicant.conf”，如下图：

```
root@iTOP4412-ubuntu-desktop:~#  
root@iTOP4412-ubuntu-desktop:~#  
root@iTOP4412-ubuntu-desktop:~# vi /etc/wpa_supplicant.conf
```

将会打开 wpa\_supplicant.conf 文件，如下图：

```
ctrl_interface=/var/run/wpa_supplicant  
network={  
    ssid="XW-TOPEETm"  
    proto=WPA  
    key_mgmt=WPA-PSK  
    psk="aaaaaaaa"  
}
```

在这个文件里根据您的路由器来设置里面的无线网络名称，密码，网络类型，修改完成后保存并退出，然后重启开发板就可以自动连接 wifi 了。

### 12.3.2 全功能板 wifi 配置

开发板上电启动进入到 Ubuntu 系统后，在开发板的串口里面使用命令 “vi /etc/init.d/itop-set” ，如下图：

```
root@iTOP4412-ubuntu-desktop: #
root@iTOP4412-ubuntu-desktop:~#
root@iTOP4412-ubuntu-desktop:~# vi /etc/init.d/itop-set
```

然后打开 itop-set 文件，把里面的内容：

```
# insmod /lib/firmware/ath6k/AR6003/hw2.1.1/cfg80211.ko
# insmod /lib/firmware/ath6k/AR6003/hw2.1.1/ath6kl_sdio.ko
```

```
# sleep 1
```

改成下面的内容（其他的地方不要修改，因为这个文件里也包括精英板的 wifi 配置）：

```
insmod /lib/firmware/ath6k/AR6003/hw2.1.1/cfg80211.ko
insmod /lib/firmware/ath6k/AR6003/hw2.1.1/ath6kl_sdio.ko
```

```
sleep 1
```

然后保存并退出，重启开发板就可以在桌面右上角的 wifi 图标来连接无线网络了。

## 附录一 QT 第三方库文件的编译

解压光盘资料里面的 linux 目录下的 3rdpart-lib-for-Qtopia2.2.0.tar.gz 后，该目录有如下的几个文件：

arminclude : 编译 Qtopia2.2.0 需要的头文件 ( 迅为制作 )

armlib : 编译 Qtopia2.2.0 需要的库文件 ( 迅为制作 )

jpeg-6b : 一个 jpeg 图形编码解码程序库

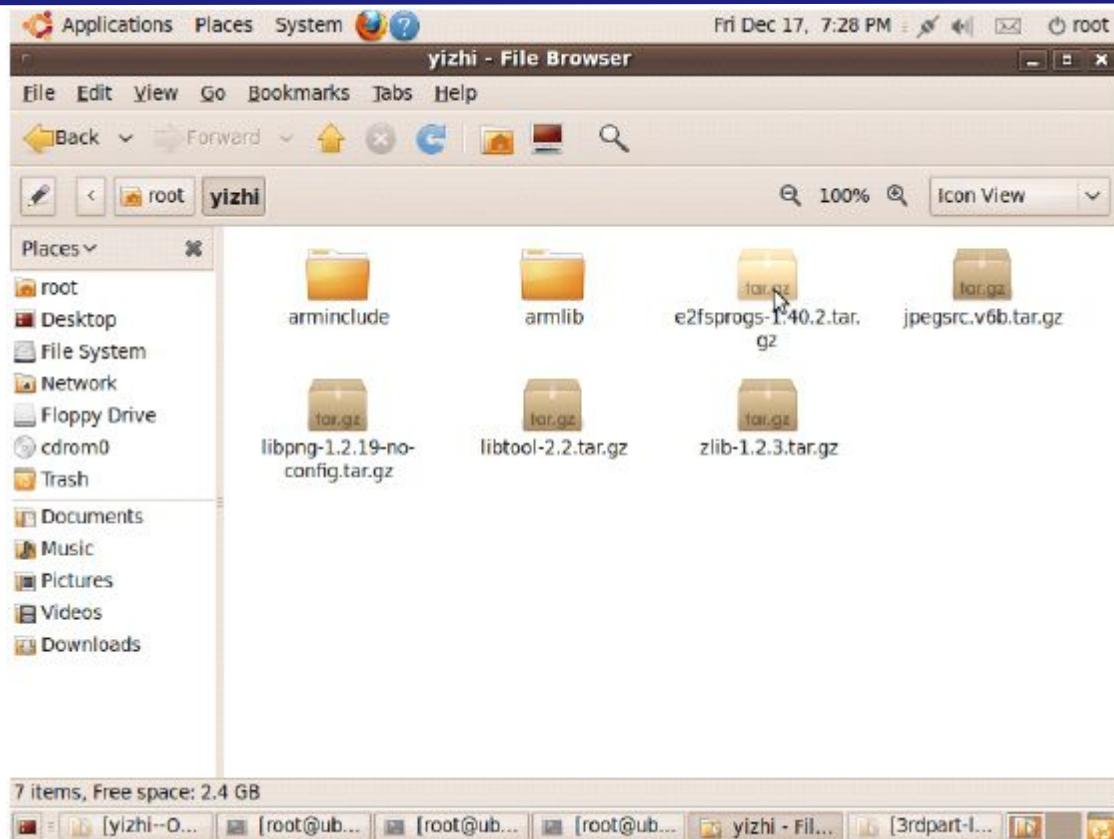
libpng-1.2.19 : 一个 png 图形编码解码程序库

zlib-1.2.3 : 一个压缩解压程序库

e2fsprogs-1.40.2 : 提供 uuid 支持

libtool-2.2.tar.gz 工具包，编译 jpeg-6b 过程中需要该工具

把 jpeg-6b , libpng-1.2.19 , zlib-1.2.3 , e2fsprogs-1.40.2 压缩包拷贝到 Ubuntu 的 root/yizhi 文件夹下面。如果您的 PC 没有 yizhi 目录，自己先建立一个，同时需要在 yizhi 文件夹下面建立 arminclude 和 armlib 文件夹，用来存放您自己制作的头文件和库文件。



## 交叉编译 jpeg 库

解压 jpegsrc.v6b.tar.gz 到/root/yizhi

```
#cd jpeg-6b
```

设置环境变量

```
#export PATH=/usr/local/arm/4.4.1/bin:$PATH
```

```
#export CC=/usr/local/arm/4.4.1/bin/arm-linux-gcc
```

```
./configure --enable-shared
```

修改生成的 Makefile 文件

```
CC=/usr/local/arm/4.4.1/bin/arm-linux-gcc
```

```
AR=/usr/local/arm/4.4.1/bin/arm-linux-ar rc
```

AR2=/usr/local/arm/4.4.1/bin/arm-linux-ranlib

保存

```
#cp jconfig.doc jconfig.h
```

```
#make
```

如果出现错误，提示找不到 libtool，如下图：

解决办法：

Ubuntu 上需要安装 libtool 工具，重新打开一个终端窗口，注意不要关闭编译 jpeg 时的窗口，在新窗口中执行以下命令：

```
#cd /root/yizhi/  
#tar zxvf libtool-2.2.tar.gz  
#cd /root/yizhi/libtool-2.2  
#./configure --prefix=/usr  
#make  
#make install
```

这样 libtool 工具即可完成。

Libtool 工具安装完成。切换到编译 jpeg-6b 时的窗口，然后执行如下的命令：

```
#cp /usr/share/libtool/config/config.guess /root/yizhi/jpeg-6b  
#cp /usr/share/libtoo;/config.sub /root.yizhi/jpeg-6b  
#cd /root/yizhi/jpeg-6b  
#./configure -enable-shared -enable-static
```

修改 Makefile 文件中的 CC , AR , AR2 变量，修改为：

```
CC=/usr/local/arm/4.4.1/bin/arm-linux-gcc  
AR=/usr/local/arm/4.4.1/bin/arm-linux-ar rc  
AR2=/usr/local/arm/4.4.1/bin/arm-linux-ranlib
```

保存，退出。

```
#make
```

编译通过。

执行以下命令即可完成 jpeg 头文件和库文件的制作工作。

```
#cp jpeglib.h jconfig.h jmorecfg.h /root/yizhi/arminclude  
#cp .libs/libjpeg.so* /root/yizhi/armlib
```

## 交叉编译 e2fsprogs-1.40.2

编译 e2fsprogs-1.40.2 是为了得到二个文件，uuid.h 及 libuuid.so，编译 Qtopia2.2.0 需要这两个文件。

首先解压 e2fsprogs-1.40.2，解压后执行下面的命令：

```
cd /root/yizhi/e2fsprogs-1.40.2
```

置环境变量

```
#export PATH=/usr/local/arm/4.4.1/bin:$PATH  
#export CC=/usr/local/arm/4.4.1/bin/arm-linux-gcc
```

以上的两个环境变量如果已经设置，就不需要重新设置了，执行下面的命令来配置和编译 e2fsprogs-1.40.2：

```
# ./configure --enable-elf-shlibs --host=arm-linux  
--with-cc=/usr/local/arm/4.4.1/bin/arm-linux-gcc  
--with-linker=/usr/local/arm/4.4.1/bin/arm-linux-ld  
# make
```

也许会提示“makeinfo 命令没有找到”，不过没关系。

将 e2fsprogs-1.40.2/lib/录下 uuid 的文件夹复制到 yizhi/arminclude 下

```
# cp -r lib/uuid /root/yizhi/arminclude
```

将 e2fsprogs-1.40.2/lib/录下面的库文件复制到/root/yizhi/armlib 下

```
# cp lib/libuuid.so* /root/yizhi/armlib/
```

e2fsprogs-1.40.2 相关工作完成。

## 交叉编译 libpng 库

将 libpng-1.2.19.tar.bz2 解压到/root/yizhi 录下

```
# cd /root/yizhi/libpng-1.2.1  
# cp scripts/makefile.linux./Makefile
```

注意，这里的 makefile 不是用./configure 生成的，而是直接从 scripts 文件夹里面复制的（见上面的 cp 命令），修改 Makefile 文件：

```
# gedit Makefile
```

修改：

```
AR_RC=/usr/local/arm/4.4.1/bin/arm-linux-ar rc  
CC=/usr/local/arm/4.4.1/bin/arm-linux-gcc  
RANLIB=/usr/local/arm/4.4.1/bin/arm-linux-ranlib
```

保存

```
# make
```

```
# cp libpng12.so* /root/yizhi/armlib  
# cp *.h /root/yizhi/arminclude  
完成 libpng 头文件和库文件的制作。
```

## 交叉编译 zlib

编译前解压 zlib-1.2.3 , 解压后

```
# cd /root/yizhi/zlib-1.2.3
```

设置环境变量

```
# export PATH=/usr/local/arm/4.4.1/bin:$PATH
```

```
# export LD_LIBRARY_PATH=/usr/local/arm/4.4.1/arm-  
linux/lib:$LD_LIBRARY_PATH
```

```
# export CC= /usr/local/arm/4.4.1/bin/arm-linux-gcc
```

如果上面已经设置了 PATH 和 LD\_LIBRARY\_PATH 环境变量 , 这里不需要重新设置 ,  
执行以下命令生成 Makefile 文件 :

```
# ./configure –shared
```

修改一下 :

```
# gedit Makefile
```

修改为 :

```
CC=/usr/local/arm/4.4.1/bin/arm-linux-gcc
```

```
LDSHARED=/usr/local/arm/4.4.1/bin/arm-linux-gcc -shared-Wl , -soname ,  
libz.so.1
```

```
CPP=/usr/local/arm/4.4.1/bin/arm-linux-gcc-E
```

```
AR=/usr/local/arm/4.4.1/bin/arm-linux-ar rc
```

```
RANLIB=/usr/local/arm/4.4.1/bin/arm-linux-ranlib
```

保存

```
#make
```

将编译生成的动态库及相关的头文件分别复制到 armlib 和 arminclude 录下

```
# cp libz.so* /root/yizhi/armlib
```

```
# cp *.h /root/yizhi/arminclude
```

经过以上步骤的操作，编译 Qtopia2.2.0 时所需的头文件和库文件制作完成。

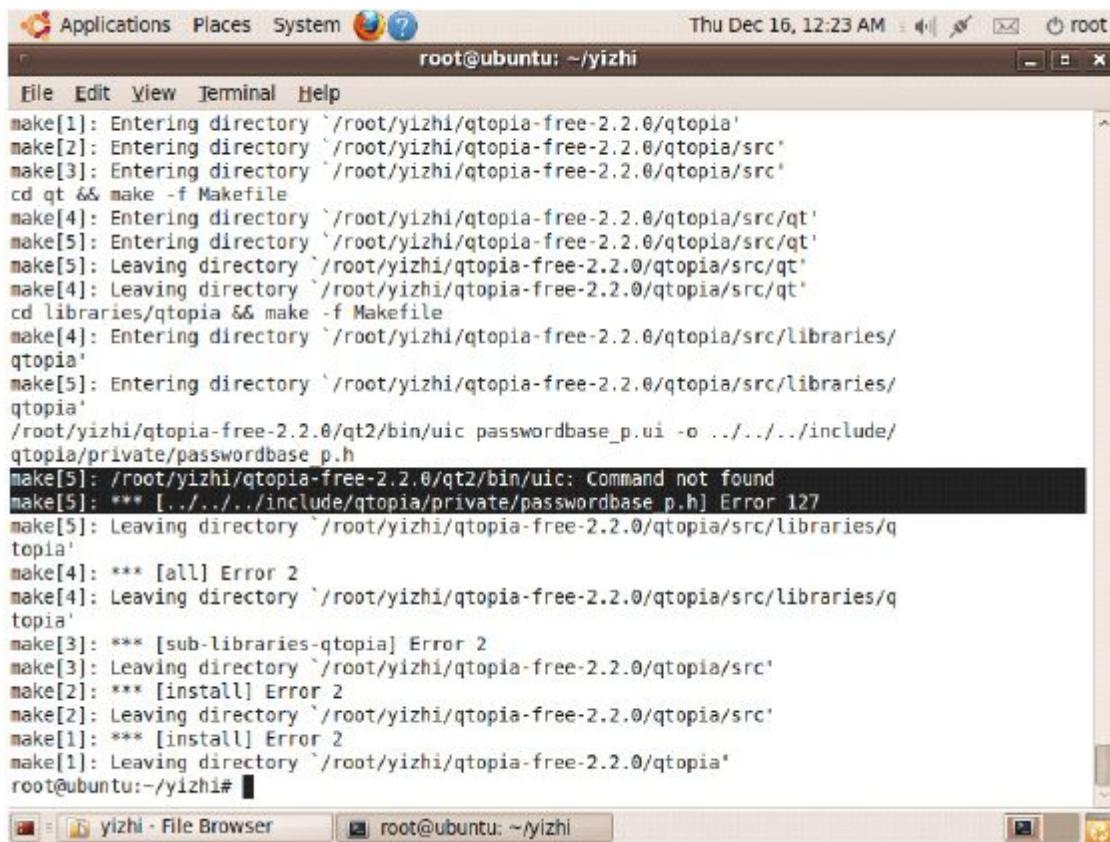
## 附录二 编译 ARM-qtopia-free-src-2.2.0 常见错误的处理

关于一些编译错误的处理

编译过程中如果产生错误信息，需要解决掉才能继续编译下去，错误信息的产生主要是因为

为编译器版本造成的，共有以下几类错误信息：

### 1 缺少工具错误



The screenshot shows a terminal window titled 'root@ubuntu: ~yizhi' running on an Ubuntu system. The window displays a command-line session where the user is attempting to build the qtopia-free-src-2.2.0 package. The output shows several 'make' commands entering and exiting directories, including 'qtopia', 'src/qt', 'src/libraries/qtopia', and 'src/libraries/qtopia'. At one point, the command 'uic passwordbase\_p.ui -o ../../include/qtopia/private/passwordbase\_p.h' is run, but it fails with the error 'Command not found'. This indicates that the 'uic' tool required for UI compilation is missing or not in the system's PATH. The session ends with the user exiting the terminal.

```
File Edit View Terminal Help
Thu Dec 16, 12:23 AM root@ubuntu: ~yizhi
make[1]: Entering directory `/root/yizhi/qtopia-free-2.2.0/qtopia'
make[2]: Entering directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src'
make[3]: Entering directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src'
cd qt && make -f Makefile
make[4]: Entering directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src/qt'
make[5]: Entering directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src/qt'
make[5]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src/qt'
make[4]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src/qt'
cd libraries/qtopia && make -f Makefile
make[4]: Entering directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src/libraries/qtopia'
make[5]: Entering directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src/libraries/qtopia'
/root/yizhi/qtopia-free-2.2.0/qt2/bin/uic passwordbase_p.ui -o ../../include/qtopia/private/passwordbase_p.h
make[5]: /root/yizhi/qtopia-free-2.2.0/qt2/bin/uic: Command not found
make[5]: *** [../../include/qtopia/private/passwordbase_p.h] Error 127
make[5]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src/libraries/qtopia'
make[4]: *** [all] Error 2
make[4]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src/libraries/qtopia'
make[3]: *** [sub-libraries-qtopia] Error 2
make[3]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src'
make[2]: *** [install] Error 2
make[2]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src'
make[1]: *** [install] Error 2
make[1]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia'
root@ubuntu:~/yizhi#
```

这是编译时遇到的第一个问题，编译 qt2 时需要使用 uic 工具，uic 是 PC 上运行的工具，

用来把 Qt 的界面文件，也就是.ui 文件转换成.h 文件和.cpp 文件，转换完成后才能交给交叉编译器编译。

**解决办法**：把 qtopia-free-2.2.0/dqt/bin 里面的 uic 文件拷贝到 qtopia-free-2.2.0/qt2/bin 下面即可。

注意：uic 拷贝到 qtopia-free-2.2.0/qt2/bin 后，需要修改一下 build 脚本文件，因为如果您重新执行 build 命令，该脚本会把刚才编译中的 qtopia-free-2.2.0 文件夹删除，重新解

压，形成新的 qtopia-free-2.2.0，开始编译，这样刚才拷贝的 uic 就没意义了。

修改方法如下图所示：

```

#!/bin/bash

#rm -fr qtopia-free-2.2.0
#tar xfvz qtopia-free-src-2.2.0.tar.gz

cd qtopia-free-2.2.0

#set compile environment variable
export QTDIR=/root/yizhi/qtopia-free-2.2.0/qt2
export QPEDIR=/root/yizhi/qtopia-free-2.2.0/qtopia
export LD_LIBRARY_PATH=$QTDIR/lib:$QPEDIR/lib:$LD_LIBRARY_PATH
export TMAKEDEDIR=/root/yizhi/qtopia-free-2.2.0/tmake
export TMAKEPATH=$TMAKEDEDIR/lib/qws/linux-arm-g++
export PATH=/usr/local/arm/arm-linux-gcc-3.4.5/bin:$PATH
#mkdir /root/yizhi/qtopia

echo yes | ./configure -qte "-embedded -no-xft -qconfig qpe -depths 16,32 -system-jpeg -qt-zlib -qt-libpng -gif -no-g++-exceptions -no-qvfb -xplatform linux-arm-g++ -tslib -I/usr/local/tslib/include -I/usr/local/tslib/lib" -qpe 'edition pda -displaysize 480x272 -fontfamilies "helvetica fixed micro smallsmooth smoothtimes unifont" -xplatform linux-arm-g++ -luuid' -qt2 '-no-opengl -no-xft' -dqt '-no-xft -thread'

make && make install

```

就是在 rm -fr qtopia-free-2.2.0 和 tar xfvz qtopia-free-src-2.2.0.tar.gz 语句前加上 #注释掉，修改后保存退出。

然后执行`#./build`，重新编译。

## 2 缺少类声明

error: ‘QWSInputMethod’ has not been declared

修改方法如下：

```
#cd /root/yizhi/qtopia-free-2.2.0/qt2/src/kernel
```

```
# gedit qwindowsystem_qws.h
```

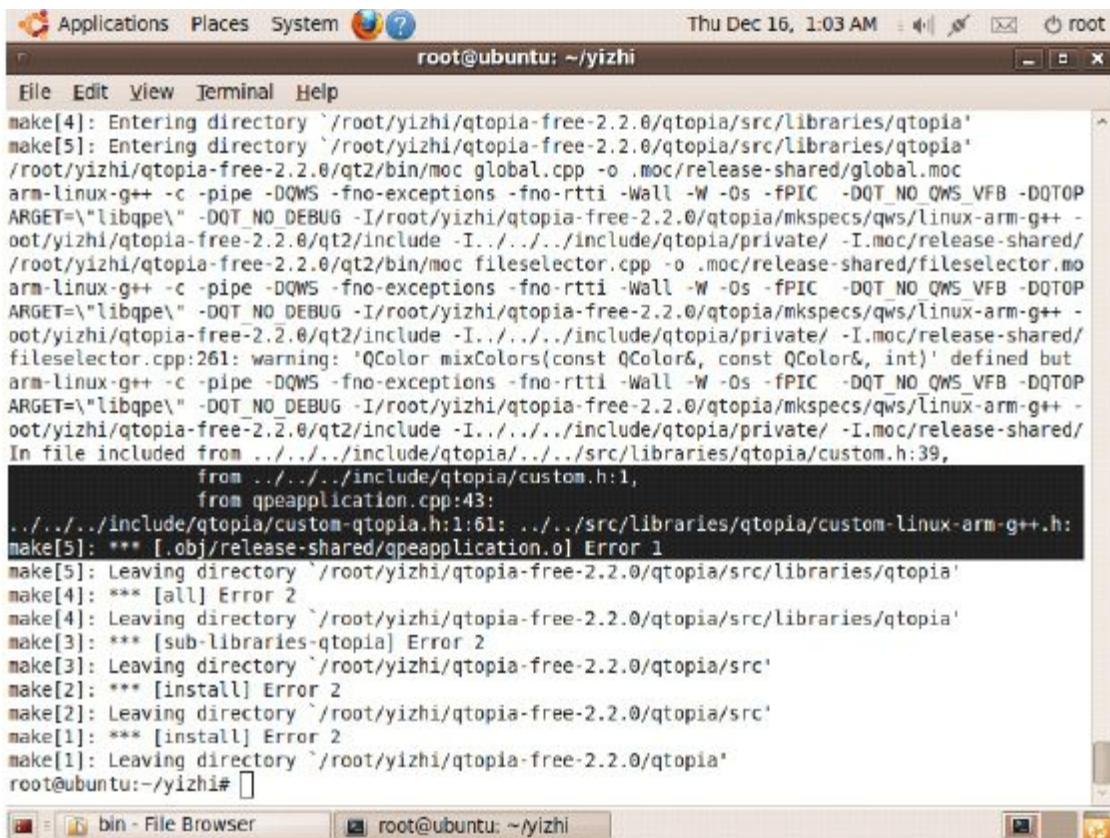
在前面增加以下两行：

```
class QWSInputMethod;
```

```
class QWSGestureMethod;
```

保存退出，继续编译。

### 3 缺少文件错误



```
Thu Dec 16, 1:03 AM root@ubuntu: ~yizhi
File Edit View Terminal Help
make[4]: Entering directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src/libraries/qtopia'
make[5]: Entering directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src/libraries/qtopia'
/root/yizhi/qtopia-free-2.2.0/qt2/bin/moc global.cpp -o .moc/release-shared/global.moc
arm-linux-g++ -c -pipe -DQWS -fno-exceptions -fno-rtti -Wall -W -Os -fPIC -DQT_NO_QWS_VFB -DQTOP
ARGET=\"libqpe\" -DQT_NO_DEBUG -I/root/yizhi/qtopia-free-2.2.0/qtopia/mkspecs/qws/linux-arm-g++ -
oot/yizhi/qtopia-free-2.2.0/qt2/include -I../../include/qtopia/private/ -I.moc/release-shared/
/root/yizhi/qtopia-free-2.2.0/qt2/bin/moc fileselector.cpp -o .moc/release-shared/fileselector.mo
arm-linux-g++ -c -pipe -DQWS -fno-exceptions -fno-rtti -Wall -W -Os -fPIC -DQT_NO_QWS_VFB -DQTOP
ARGET=\"libqpe\" -DQT_NO_DEBUG -I/root/yizhi/qtopia-free-2.2.0/qtopia/mkspecs/qws/linux-arm-g++ -
oot/yizhi/qtopia-free-2.2.0/qt2/include -I../../include/qtopia/private/ -I.moc/release-shared/
fileselector.cpp:261: warning: 'QColor mixColors(const QColor&, const QColor&, int)' defined but
arm-linux-g++ -c -pipe -DQWS -fno-exceptions -fno-rtti -Wall -W -Os -fPIC -DQT_NO_QWS_VFB -DQTOP
ARGET=\"libqpe\" -DQT_NO_DEBUG -I/root/yizhi/qtopia-free-2.2.0/qtopia/mkspecs/qws/linux-arm-g++ -
oot/yizhi/qtopia-free-2.2.0/qt2/include -I../../include/qtopia/private/ -I.moc/release-shared/
In file included from ../../../../../../include/qtopia/custom.h:1,
                 from qpeapplication.cpp:43:
../../../../include/qtopia/custom-qtopia.h:1:61: ../../src/libraries/qtopia/custom-linux-arm-g++.h:
make[5]: *** [.obj/release-shared/qpeapplication.o] Error 1
make[5]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src/libraries/qtopia'
make[4]: *** [all] Error 2
make[4]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src/libraries/qtopia'
make[3]: *** [sub-libraries-qtopia] Error 2
make[3]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src'
make[2]: *** [install] Error 2
make[2]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src'
make[1]: *** [install] Error 2
make[1]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia'
root@ubuntu:~yizhi#
```

源代码需要 custom-linux-arm-g++.h/cpp 文件，但是没找到。

解决办法：

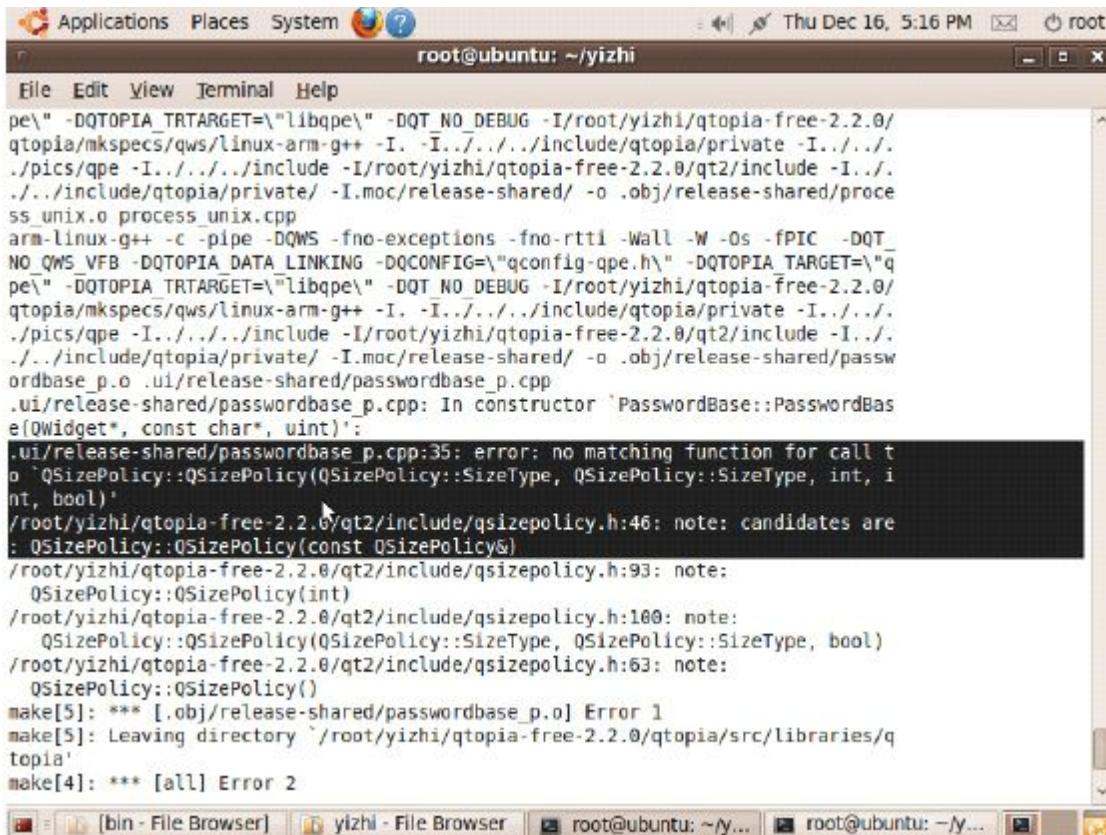
```
#cd ./qtopia-free-2.2.0/qtopia/src/libraries/qtopia
#cp custom-linux-ipaq-g++.h custom-linux-arm-g++.h
#cp custom-linux-ipaq-g++.cpp custom-linux-arm-g++.cpp
```

继续编译

```
#cd /root/yizhi
./build
```

## 4 无法匹配到 QSizePolicy 类的构造函数

即 QSizePolicy 类没有提供特定参数的构造函数

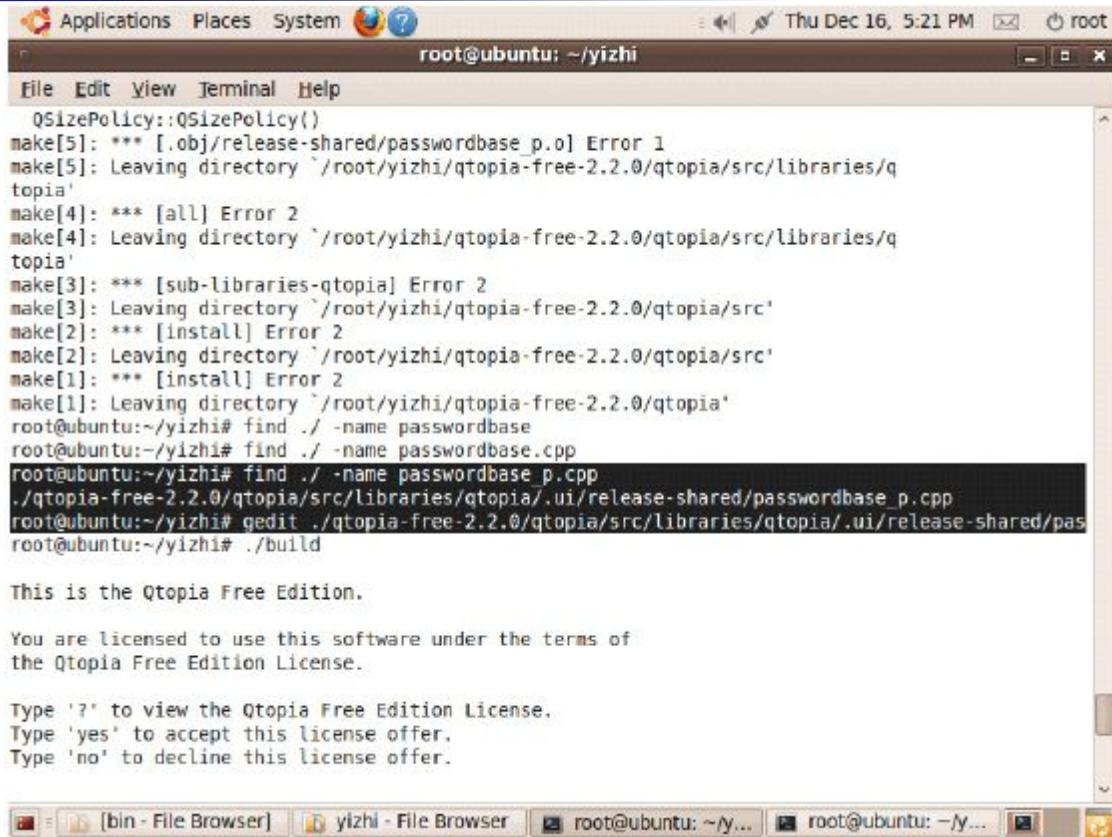


```
pe\" -DQTQPIA_TRTARGET=\"libqpe\" -DQT_NO_DEBUG -I/root/yizhi/qtopia-free-2.2.0/qtopia/mkspecs/qws/linux-arm-g++ -I. -I../../include/qtopia/private -I../../pics/qpe -I../../include -I/root/yizhi/qtopia-free-2.2.0/qt2/include -I../../include/qtopia/private/ -I.moc/release-shared/ -o .obj/release-shared/process_unix.o process_unix.cpp  
arm-linux-g++ -c -pipe -DQWS -fno-exceptions -fno-rtti -Wall -Os -fPIC -DQT_NO_QWS_VFB -DQTQPIA_DATA_LINKING -DQCONFIG=\"qconfig-qpe.h\" -DQTQPIA_TARGET=\"qpe\" -DQTQPIA_TRTARGET=\"libqpe\" -DQT_NO_DEBUG -I/root/yizhi/qtopia-free-2.2.0/qtopia/mkspecs/qws/linux-arm-g++ -I. -I../../include/qtopia/private -I../../pics/qpe -I../../include -I/root/yizhi/qtopia-free-2.2.0/qt2/include -I../../include/qtopia/private/ -I.moc/release-shared/ -o .obj/release-shared/passwordbase_p.o  
.ui/release-shared/passwordbase_p.cpp: In constructor 'PasswordBase::PasswordBase(QWidget*, const char*, uint)':  
.ui/release-shared/passwordbase_p.cpp:35: error: no matching function for call to 'QSizePolicy::QSizePolicy(QSizePolicy::SizeType, QSizePolicy::SizeType, int, int, bool)'  
/root/yizhi/qtopia-free-2.2.0/qt2/include/qsizepolicy.h:46: note: candidates are  
: QSizePolicy::QSizePolicy(const QSizePolicy&)  
/root/yizhi/qtopia-free-2.2.0/qt2/include/qsizepolicy.h:93: note:  
 QSizePolicy::QSizePolicy(int)  
/root/yizhi/qtopia-free-2.2.0/qt2/include/qsizepolicy.h:100: note:  
 QSizePolicy::QSizePolicy(QSizePolicy::SizeType, QSizePolicy::SizeType, bool)  
/root/yizhi/qtopia-free-2.2.0/qt2/include/qsizepolicy.h:63: note:  
 QSizePolicy::QSizePolicy()  
make[5]: *** [.obj/release-shared/passwordbase_p.o] Error 1  
make[5]: Leaving directory '/root/yizhi/qtopia-free-2.2.0/qtopia/src/libraries/qtopia'  
make[4]: *** [all] Error 2
```

解决办法：

查看出错信息的文件，这里是 passwordbase\_p.cpp，且出错的地方是第 35 行，使用 find 命令找到该文件在 Ubuntu 中的具体位置：

```
#find ./ -name passwordbase_p.cpp
```



```
root@ubuntu:~/yizhi$ make
QSizePolicy::QSizePolicy()
make[5]: *** [.obj/release-shared/passwordbase_p.o] Error 1
make[5]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src/libraries/qtopia'
make[4]: *** [all] Error 2
make[4]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src/libraries/qtopia'
make[3]: *** [sub-libraries-qtopia] Error 2
make[3]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src'
make[2]: *** [install] Error 2
make[2]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src'
make[1]: *** [install] Error 2
make[1]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia'
root@ubuntu:~/yizhi# find ./ -name passwordbase
root@ubuntu:~/yizhi# find ./ -name passwordbase.cpp
root@ubuntu:~/yizhi# find ./ -name passwordbase_p.cpp
./qtopia-free-2.2.0/qtopia/src/libraries/qtopia/.ui/release-shared/passwordbase_p.cpp
root@ubuntu:~/yizhi# gedit ./qtopia-free-2.2.0/qtopia/src/libraries/qtopia/.ui/release-shared/passwordbase_p.cpp
root@ubuntu:~/yizhi# ./build

This is the Qtopia Free Edition.

You are licensed to use this software under the terms of
the Qtopia Free Edition License.

Type '?' to view the Qtopia Free Edition License.
Type 'yes' to accept this license offer.
Type 'no' to decline this license offer.
```

找到后在./qtopia-free-2.2.0/qtopia/src/libraries/qtopia/.ui/release-shared 目录下面，执行以下命令，修改该文件：

```
# gedit ./qtopia-free-2.2.0/qtopia/src/libraries/qtopia/.ui/release-shared/passwordbase_p.cpp
```

移动光标到第 35 行，如下图所示：

```
if ( !name )
    setName( "PasswordBase" );
QFont f( font() );
f.setPointSize( 24 );
setFont( f );
PasswordBaseLayout = new QVBoxLayout( this, 6, 6, "PasswordBaseLayout" );

prompt = new QLabel( this, "prompt" );
prompt->setSizePolicy( QSizePolicy( (QSizePolicy::SizeType)7, (QSizePolicy::SizeType)1, 0, prompt->sizePolicy().hasHeightForWidth() ) );
QFont prompt_font( prompt->font() );
prompt_font.setPointSize( 18 );
prompt->setFont( prompt_font );
PasswordBaseLayout->addWidget( prompt );

display = new QLineEdit( this, "display" );
display->setEnabled( FALSE );
display->setMaxLength( 8 );
display->setEchoMode( QLineEdit::Password );
PasswordBaseLayout->addWidget( display );

Layout3 = new QGridLayout( 0, 1, 1, 0, 6, "Layout3" );

button_R = new QPushButton( this, "button_R" );
```

把

prompt->setSizePolicy(QSizePolicy((QSizePolicy::SizeType)7 ,

(QSizePolicy::SizeType) 7 , 0 , 0 , prompt->sizePolicy().hasHeightForWidth()))

函数中的 0 , 0 去掉 , 即 :

prompt->setSizePolicy(QSizePolicy((QSizePolicy::SizeType)7 ,

(QSizePolicy::SizeType) 7 , prompt->sizePolicy().hasHeightForWidth()))

去掉后如下图 :

```
if ( !name )
    setName( "PasswordBase" );
QFont f{ font() };
f.setPointSize( 24 );
setFont( f );
PasswordBaseLayout = new QVBoxLayout( this, 6, 6, "PasswordBaseLayout" );

prompt = new QLabel( this, "prompt" );
prompt->setSizePolicy( QSizePolicy( QSizePolicy::SizeType)7, (QSizePolicy::SizeType)
1.prompt->sizePolicy().hasHeightForWidth() ) );
QFont prompt_font( prompt->font() );
prompt_font.setPointSize( 18 );
prompt->setFont( prompt_font );
PasswordBaseLayout->addWidget( prompt );

display = new QLineEdit( this, "display" );
display->setEnabled( FALSE );
display->setMaxLength( 8 );
display->setEchoMode( QLineEdit::Password );
PasswordBaseLayout->addWidget( display );

Layout3 = new QGridLayout( 0, 1, 1, 0, 6, "Layout3" );

button_R = new QPushButton( this, "button_R" );
```

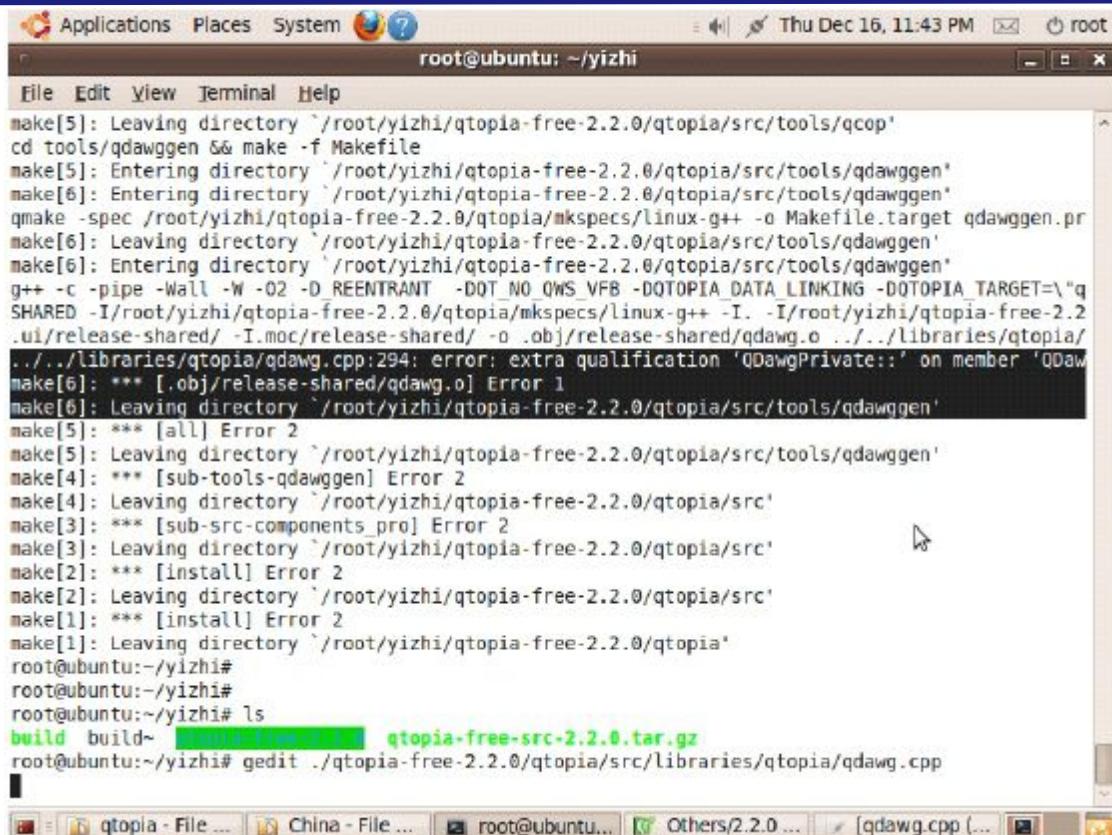
保存退出，执行如下命令重新编译：

```
#./build
```

重新编译后，会出现若干次这个类型的错误，解决办法非常相似，这里就不一一介绍了。

## 5 类的成员函数前有额外的类名字

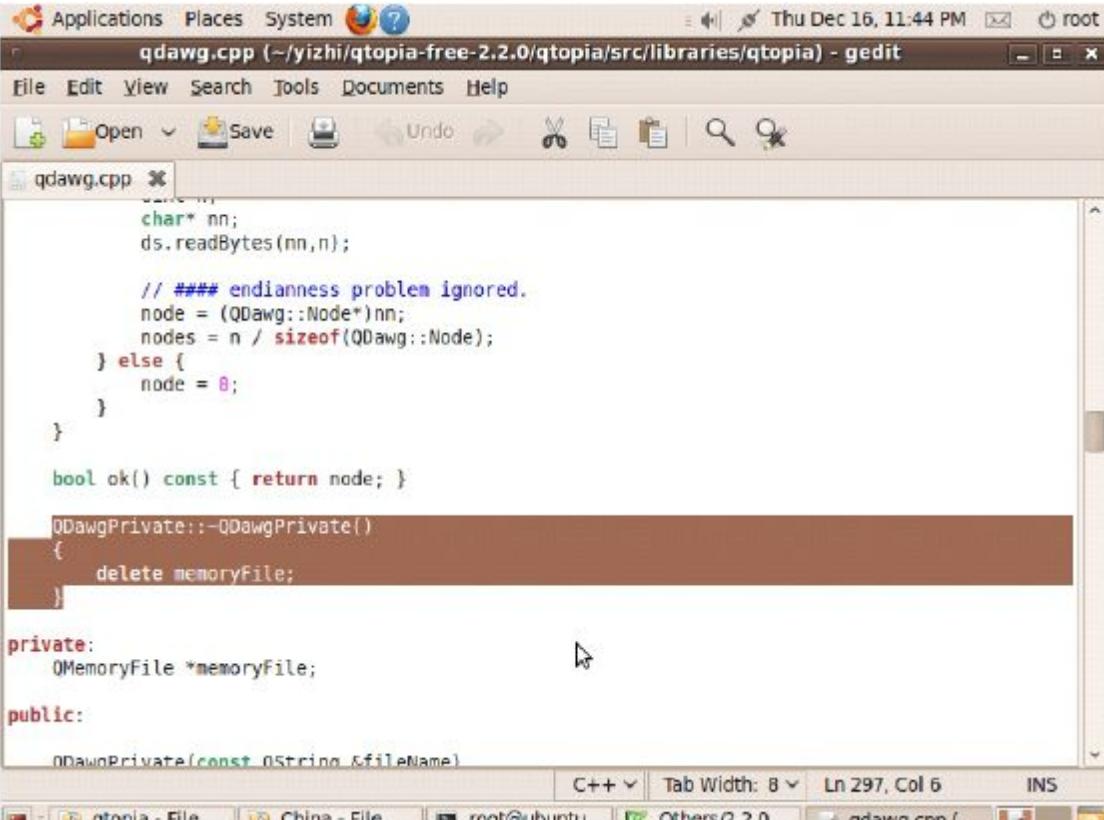
如下图：



```
root@ubuntu: ~/yizhi
File Edit View Terminal Help
make[5]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src/tools/qcopp'
cd tools/qdawggen && make -f Makefile
make[5]: Entering directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src/tools/qdawggen'
make[6]: Entering directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src/tools/qdawggen'
qmake -spec /root/yizhi/qtopia-free-2.2.0/qtopia/mkspecs/linux-g++ -o Makefile.target qdawggen.pr
make[6]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src/tools/qdawggen'
make[6]: Entering directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src/tools/qdawggen'
g++ -c -pipe -Wall -O2 -D REENTRANT -D QT_NO_QWS_VFB -DQTPIA DATA LINKING -DOTPIA_TARGET=\\\"q
SHARED -I/root/yizhi/qtopia-free-2.2.0/qtopia/mkspecs/linux-g++ -I. -I/root/yizhi/qtopia-free-2.2
.ui/release-shared/ -I.moc/release-shared/ -o .obj/release-shared/qdawg.o ../../libraries/qtopia/
../../libraries/qtopia/qdawg.cpp:294: error: extra qualification 'QDawgPrivate::' on member 'QDaw
make[6]: *** [.obj/release-shared/qdawg.o] Error 1
make[6]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src/tools/qdawggen'
make[5]: *** [all] Error 2
make[5]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src/tools/qdawggen'
make[4]: *** [sub-tools-qdawggen] Error 2
make[4]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src'
make[3]: *** [sub-src-components_pro] Error 2
make[3]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src'
make[2]: *** [install] Error 2
make[2]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src'
make[1]: *** [install] Error 2
make[1]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia'
root@ubuntu:~/yizhi#
root@ubuntu:~/yizhi#
root@ubuntu:~/yizhi# ls
build build~ 中国 - File ... qtopia-free-src-2.2.0.tar.gz
root@ubuntu:~/yizhi# gedit ./qtopia-free-2.2.0/qtopia/src/libraries/qtopia/qdawg.cpp
```

解决办法：

根据错误提示信息找到 qdawg.cpp 文件，gedit 打开该文件，移动光标到 294 行。



```

Applications Places System Thu Dec 16, 11:44 PM root
qdawg.cpp (~/yizhi/qtopia-free-2.2.0/qtopia/src/libraries/qtopia) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
qdawg.cpp x
char* nn;
ds.readBytes(nn,n);

// ##### endianness problem ignored.
node = (QDawg::Node*)nn;
nodes = n / sizeof(QDawg::Node);
} else {
    node = 0;
}

bool ok() const { return node; }

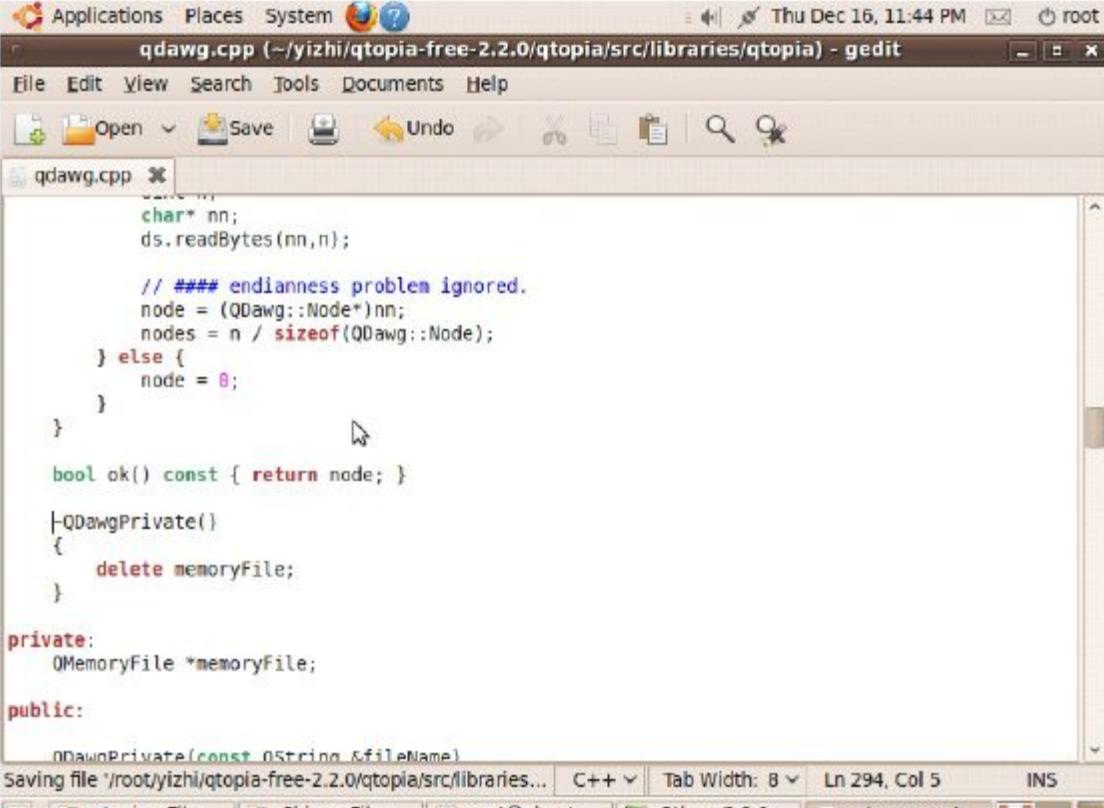
~QDawgPrivate::~QDawgPrivate()
{
    delete memoryFile;
}

private:
    QMemoryFile *memoryFile;

public:
    QDawgPrivate(const QString &fileName)
    C++ Tab Width: 8 Ln 297, Col 6 INS

```

修改后如下图：



```

Applications Places System Thu Dec 16, 11:44 PM root
qdawg.cpp (~/yizhi/qtopia-free-2.2.0/qtopia/src/libraries/qtopia) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
qdawg.cpp x
char* nn;
ds.readBytes(nn,n);

// ##### endianness problem ignored.
node = (QDawg::Node*)nn;
nodes = n / sizeof(QDawg::Node);
} else {
    node = 0;
}

bool ok() const { return node; }

-QDawgPrivate()
{
    delete memoryFile;
}

private:
    QMemoryFile *memoryFile;

public:
    QDawgPrivate(const QString &fileName)
Saving file '/root/yizhi/qtopia-free-2.2.0/qtopia/src/libraries/qtopia/qdawg.cpp'
C++ Tab Width: 8 Ln 294, Col 5 INS

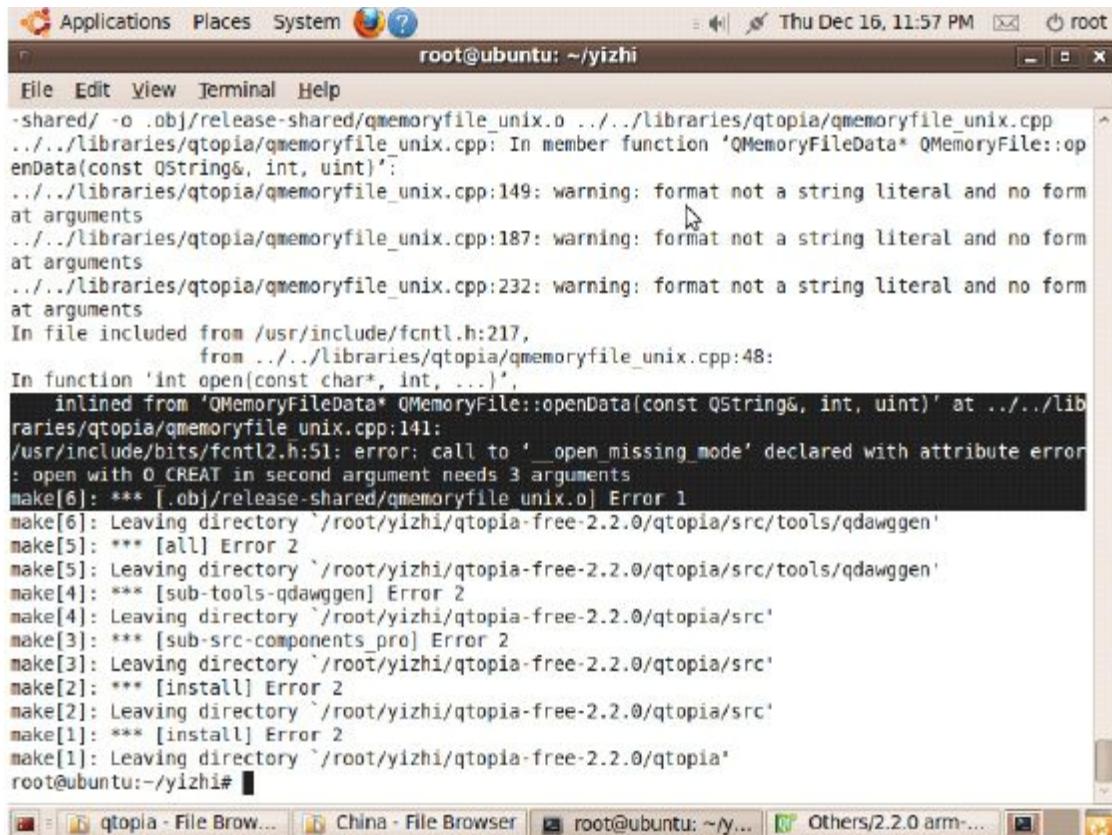
```

保存退出，继续编译

#./build

## 6. open 函数调用缺少必要的参数

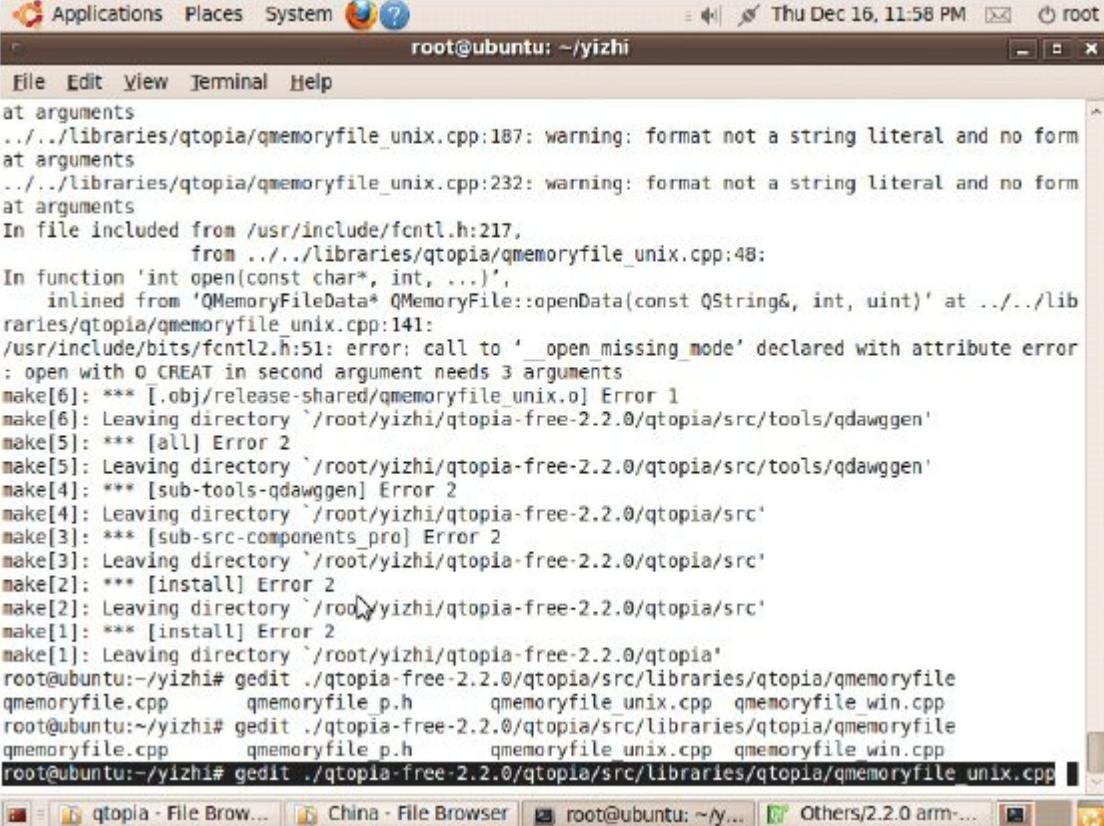
如下图提示的错误：



The screenshot shows a terminal window titled 'root@ubuntu: ~yizhi' running on an Ubuntu system. The terminal displays a compilation error for the file 'qtopia-qmemoryfile\_unix.cpp'. The error message indicates that the 'open' function call lacks a necessary argument ('mode'). The terminal also shows the build process starting from 'make[6]' and ending with 'root@ubuntu:~/yizhi#'. The error line is highlighted in black.

```
root@ubuntu: ~yizhi
File Edit View Terminal Help
-shared/ -o .obj/release-shared/qmemoryfile_unix.o ../../libraries/qtopia/qmemoryfile_unix.cpp
../../libraries/qtopia/qmemoryfile_unix.cpp: In member function 'QMemoryFileDialog* QMemoryFile::openData(const QString&, int, uint)':
../../libraries/qtopia/qmemoryfile_unix.cpp:149: warning: format not a string literal and no form
at arguments
../../libraries/qtopia/qmemoryfile_unix.cpp:187: warning: format not a string literal and no form
at arguments
../../libraries/qtopia/qmemoryfile_unix.cpp:232: warning: format not a string literal and no form
at arguments
In file included from /usr/include/fcntl.h:217,
                 from ../../libraries/qtopia/qmemoryfile_unix.cpp:48:
In function 'int open(const char*, int, ...)',
         inlined from 'QMemoryFileDialog* QMemoryFile::openData(const QString&, int, uint)' at ../../lib
raries/qtopia/qmemoryfile_unix.cpp:141:
/usr/include/bits/fcntl2.h:51: error: call to '__open_missing_mode' declared with attribute error
: open with O_CREAT in second argument needs 3 arguments
make[6]: *** [obj/release-shared/qmemoryfile_unix.o] Error 1
make[6]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src/tools/qdawggen'
make[5]: *** [all] Error 2
make[5]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src/tools/qdawggen'
make[4]: *** [sub-tools-qdawggen] Error 2
make[4]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src'
make[3]: *** [sub-src-components_pro] Error 2
make[3]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src'
make[2]: *** [install] Error 2
make[2]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src'
make[1]: *** [install] Error 2
make[1]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia'
root@ubuntu:~/yizhi#
```

解决办法参考下图：

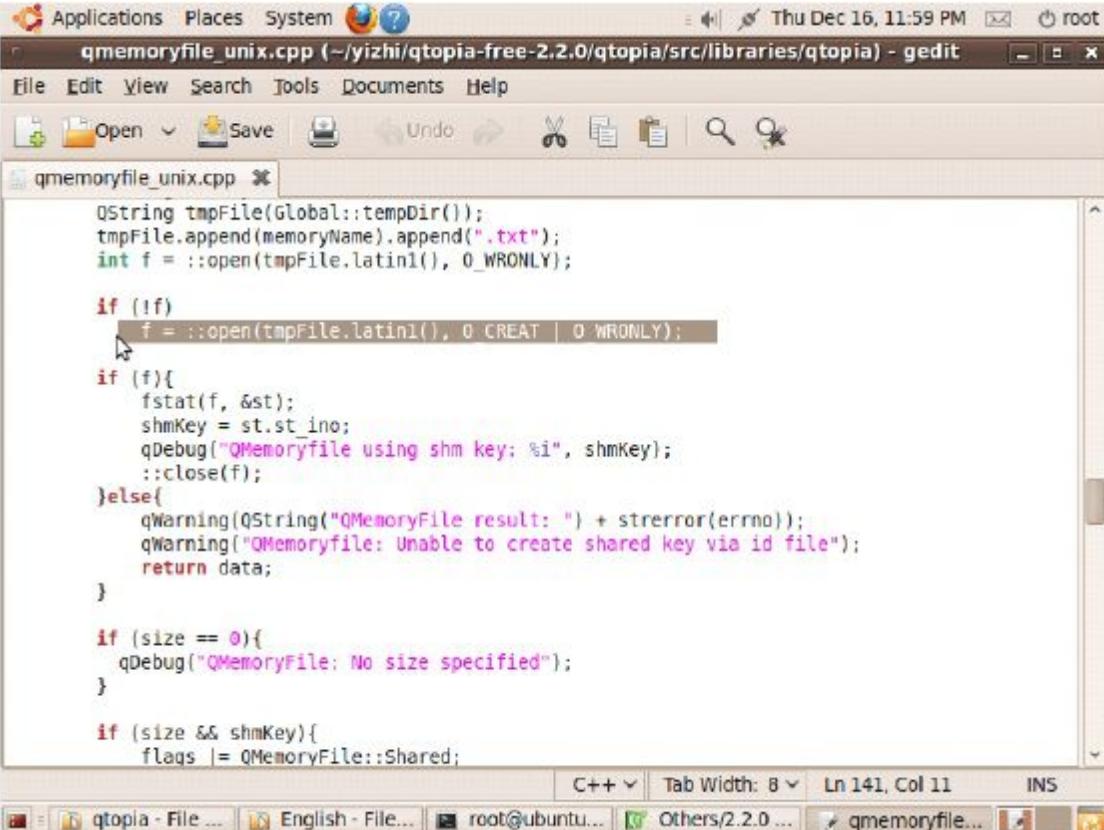


```

File Edit View Terminal Help
at arguments
../../libraries/qtopia/qmemoryfile_unix.cpp:187: warning: format not a string literal and no form
at arguments
../../libraries/qtopia/qmemoryfile_unix.cpp:232: warning: format not a string literal and no form
at arguments
In file included from /usr/include/fcntl.h:217,
                 from ../../libraries/qtopia/qmemoryfile_unix.cpp:48:
In function 'int open(const char*, int, ...)',
  inlined from 'QMemoryFileData* QMemoryFile::openData(const QString&, int, uint)' at ../../lib
raries/qtopia/qmemoryfile_unix.cpp:141:
/usr/include/bits/fcntl2.h:51: error: call to '_open_missing_mode' declared with attribute error
: open with O_CREAT in second argument needs 3 arguments
make[6]: *** [.obj/release-shared/qmemoryfile_unix.o] Error 1
make[6]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src/tools/qdawggen'
make[5]: *** [all] Error 2
make[5]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src/tools/qdawggen'
make[4]: *** [sub-tools-qdawggen] Error 2
make[4]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src'
make[3]: *** [sub-src-components_pro] Error 2
make[3]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src'
make[2]: *** [install] Error 2
make[2]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia/src'
make[1]: *** [install] Error 2
make[1]: Leaving directory `/root/yizhi/qtopia-free-2.2.0/qtopia'
root@ubuntu:~/yizhi# gedit ./qtopia-free-2.2.0/qtopia/src/libraries/qtopia/qmemoryfile
qmemoryfile.cpp      qmemoryfile.p.h      qmemoryfile_unix.cpp qmemoryfile win.cpp
root@ubuntu:~/yizhi# gedit ./qtopia-free-2.2.0/qtopia/src/libraries/qtopia/qmemoryfile
qmemoryfile.cpp      qmemoryfile.p.h      qmemoryfile_unix.cpp qmemoryfile win.cpp
root@ubuntu:~/yizhi# gedit ./qtopia-free-2.2.0/qtopia/src/libraries/qtopia/qmemoryfile_unix.cpp

```

打开 qmemoryfile\_unix.cpp 文件，修改第 51 行



```

qmemoryfile_unix.cpp (~/yizhi/qtopia-free-2.2.0/qtopia/src/libraries/qtopia) - gedit
File Edit View Search Tools Documents Help
Open Save Undo Redo Cut Copy Paste Find Replace
qmemoryfile_unix.cpp
OString tmpFile(Global::tempDir());
tmpFile.append(memoryName).append(".txt");
int f = ::open(tmpFile.latin1(), O_WRONLY);

if (!f)
    f = ::open(tmpFile.latin1(), O_CREAT | O_WRONLY);
if (f){
    fstat(f, &st);
    shmKey = st.st_ino;
    qDebug("QMemoryfile using shm key: %i", shmKey);
    ::close(f);
} else{
    qWarning(QString("QMemoryFile result: ") + strerror(errno));
    qWarning("QMemoryfile: Unable to create shared key via id file");
    return data;
}

if (size == 0){
    qDebug("QMemoryFile: No size specified");
}

if (size && shmKey){
    flags |= QMemoryFile::Shared;
}

```

f=::open(tempFile , latin1() , 0\_CREAT | 0\_WRONLY); 修改为  
f=::open(tempFile , latin1() , 0\_CREAT | 0\_WRONLY , 0666);即增加 0666 参数，  
保存，退出。

对于 open 函数来说，仅当创建新文件时，第三个参数使用，用于指定文件的访问权限位

注意：共有两处这样的修改，另外一处：

qtopia-free-2.2.0\qt2\src\tools\qmemoryfile\_unix.cpp 文件的第 143 行位置，修改方  
法与 qmemoryfile\_unix.cpp 文件相似。

#./build 重新编译

## 7 缺少 this 指针

如下图：

The screenshot shows a terminal window titled "qtopiamake.log (~/yizhi) - gedit". The window contains a log of compilation errors from the qtopia project. The errors are related to missing 'this' pointers in various functions, particularly in QValueStack and QDomSimpleReader. The errors are as follows:

```
(const QString&, bool) const :  
xml/qxml.cpp:1963: warning: format not a string literal and no format  
arguments  
xml/qxml.cpp: In member function 'virtual void QDomSimpleReader::setFeature  
(const QString&, bool)':  
xml/qxml.cpp:2019: warning: format not a string literal and no format  
arguments  
In file included from xml/qxml.h:54,  
      from xml/qxml.cpp:39:  
/root/yizhi/qtopia-free-2.2.0/qt2/include/qvaluestack.h: In member function  
'T QValueStack<T>::pop() [with T = QMap<QString, QString>]':  
xml/qxml.cpp:513: instantiated from here  
/root/yizhi/qtopia-free-2.2.0/qt2/include/qvaluestack.h:57: error: cannot  
convert 'QValueListIterator<QMap<QString, QString> >' to 'const char*' for  
argument '1' to 'int remove(const char*)'  
/root/yizhi/qtopia-free-2.2.0/qt2/include/qvaluestack.h: In member function  
'T QValueStack<T>::pop() [with T = QString]':  
xml/qxml.cpp:2532: instantiated from here  
/root/yizhi/qtopia-free-2.2.0/qt2/include/qvaluestack.h:57: error: cannot  
convert 'QValueListIterator<QString>' to 'const char*' for argument '1' to  
'int remove(const char*)'  
make[1]: *** [tmn/release-shared-linux-g++/xml/qxml.o] Error 1
```

解决办法：执行下面的命令

```
# gedit ./qtopia-free-2.2.0/qt2/include/qvaluestack.h
```

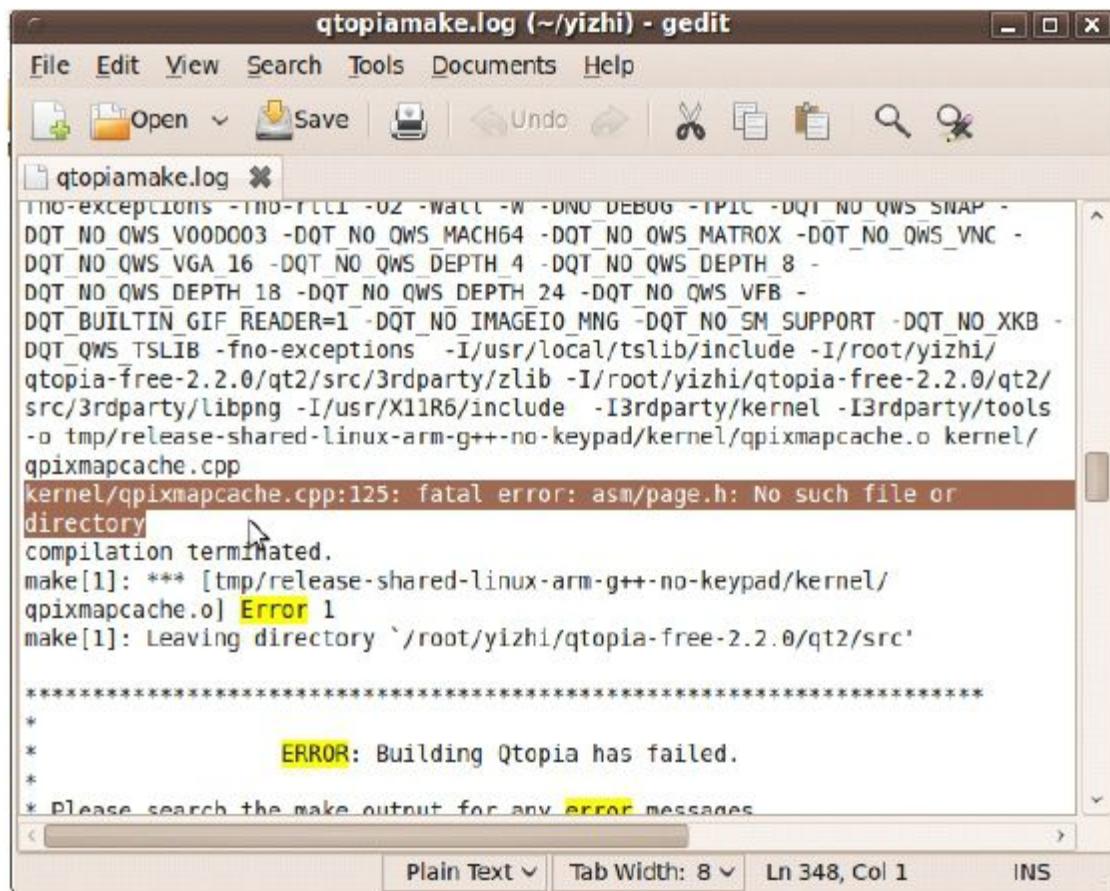
移动光标到 57 行

将 `remove( this->fromLast());` 改为 `this->remove( this->fromLast());`

保存退出。

8 缺少系统头文件

如下图所示：



解决办法如下：

```
# gedit .\qtopia-free-2.2.0\qt2\src\kernel\qpixmapcache.cpp
```

## 修改第 125 行为

```

#include "qapplication.h"
#include "qgffx_qws.h"
#include "qlock_qws.h"
#include "qwsdisplay_qws.h"
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <stdlib.h>
#include <time.h>
#include <stdio.h>
#include <signal.h>

#ifndef THROW_AWAY_UNUSED_PAGES
# include <sys/mman.h> // madvise

#define PAGE_SHIFT 12
#define PAGE_SIZE (1UL << PAGE_SHIFT)
#define PAGE_MASK (~(PAGE_SIZE-1))
//# include <asm/page.h> // PAGE_SIZE, PAGE_MASK, PAGE_ALIGN

#ifndef PAGE_ALIGN
# define PAGE_ALIGN(addr) ((addr)+PAGE_SIZE-1)&PAGE_MASK
# endif // PAGE_ALIGN

```

即：

```

#define PAGE_SHIFT 12

#define PAGE_SIZE (1UL << PAGE_SHIFT)

#define PAGE_MASK (~(PAGE_SIZE-1))

//# include<asm/page.h> //PAGE_SIZE , PAGE_MASK , PAGE_ALIGN

```

这样修改是因为新的 Linux 内核源码中已经不包含这个头文件了。

另外其他的地方也用到了该头文件，同样需要去掉，执行以下命令：

```
# gedit ./qtopia-free-2.2.0/qt2/tools/qvfb/qvfbview.cpp
```

移动到第 39 行，修改为下图所示：

```

#include <qwmatrix.h>
#include "qanimationwriter.h"

#include <stdlib.h>
#include <unistd.h>
#include <sys/ipc.h>
#include <sys/types.h>
#include <sys/shm.h>
#include <sys/mman.h>

//#include <asm/page.h>
#define PAGE_SHIFT 12
#define PAGE_SIZE (1UL << PAGE_SHIFT)
#define PAGE_MASK (~(PAGE_SIZE-1))

#include <sys/stat.h>
#include <sys/sem.h>
#include <fcntl.h>
#include <errno.h>
#include <math.h>

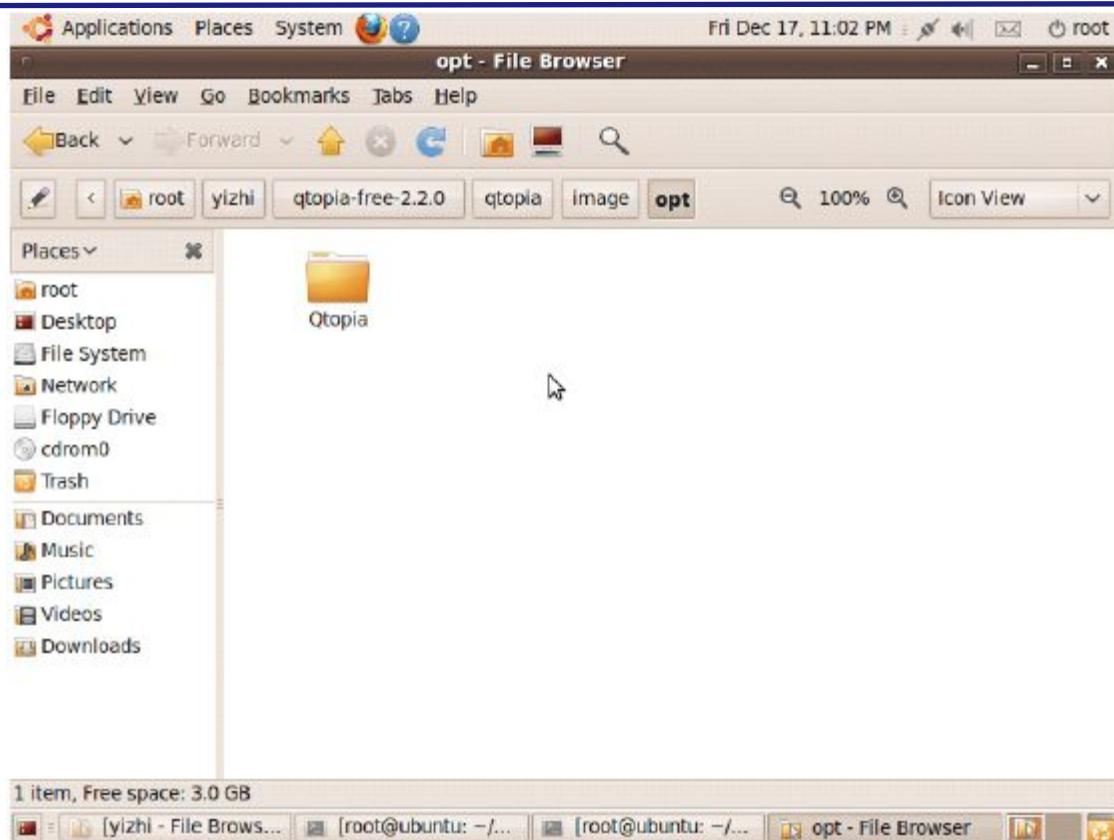
QVFBView::QVFBView( int display_id, int _w, int _h, int d, Rotation r, bool use_mmap, QWidget *parent,
                    const char *name, uint flags )
    : QScrollArea( parent, name, flags ), locked( 1 ), emulateTouchscreen( false )
{
    Saving file '/root/yizhi/qtopia-free-2.2.0/qt2/tools/qvfb...' C++ Tab Width: 8 Ln 40, Col 16 INS

```

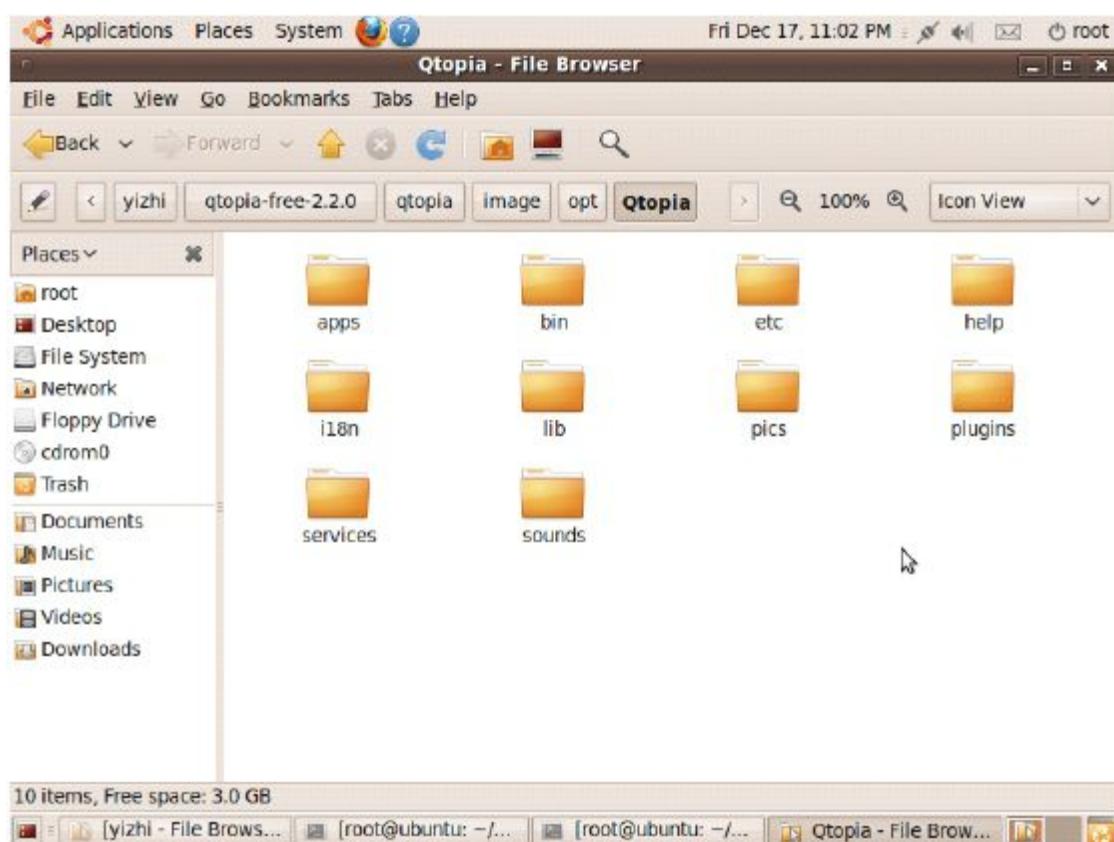
修改原因是因为 `asm/page.h` 文件里面定义了一些宏，咱们在这里重新定义了必须使用的宏，所以这里就不需要 `asm/page.h` 文件了。

说明：继续编译后会出现几次以上类型的错误，出现的错误类型就是以上总结的几种（出错的信息通过 google 能找到解决方法），修改方法按以上步骤即可，主要就是发现哪个文件出错，找到该文件，用 gedit 打开该文件，移动到出错的行上，修改，保存退出。出错的文件和行号是根据编译时的错误提示信息来找到的，一定要查找第一个 Error 的地方，这里是需要重点关注的。通过这几种错误类型的修改，您会对 Qt 移植有了新的认识，也提高了您的软件移植能力。

编译成功完成后，Qtopia2.2.0 会自动的安装到 `qtopia-free-2.2.0/qtopia/zImage/opt` 下面，如下图



进入到 Qtopia 目录下面，如图：



注意：编译好的Qtopia下面没有字体，自己需要把字体放到Qtopia/lib/fonts下面，qtopia-free-2.2.0/qt2/lib/fonts文件夹下有各种可以用的字体，但是我们只需要helvetica类型，该类型支持中文。执行以下命令：

```
#cp/root/yizhi/qtopia-free-2.2.0/qt2/lib/fonts/Helvetica* /root/yizhi/qtopia-free-2.2.0/qtopia/image/opt/Qtopia/lib/fonts/
```

另外需要把/root/yizhi/armlib下面的文件全部拷贝到/root/yizhi/qtopia-free-2.2.0/qtopia/image/opt/Qtopia/lib/下面  
字体文件和库文件拷贝完成后，把Qtopia整个录拷贝到文件系统的/opt录下面，如果该目录下面已经有迅为电子提供的Qtopia文件夹，您可以把迅为电子提供的Qtopia文件夹删除掉，来运行您自己编译的版本，运行方法参考下一章节的Qtopia2.2.0的使用。

## 附录三 内核配置详细说明

对 linux 内核进行编译之前，首先要使用命令 ‘make menuconfig’ 对 linux 的编译选项进行配置。

下面详细讲解使用这个命令时所遇到的重要配置参数：

### 第一部分

Code maturity level options ---> 代码成熟等级选项

[ ] Prompt for development and/or incomplete code/drivers

默认情况下是选择的，这将会在设置界面中显示还在开发或者还没有完成的代码与驱动。你应该选择它，因为有许多设备可能必需选择这个选项才能进行配置，实际上它是安全的。

[ ] Select only drivers expected to compile cleanly

选择这个选项你将不会看到一些已知的存在问题的驱动程序选项，默认的情况下也是选择的。如果你有设备没有找到驱动选项，你可以将这一项去掉，或许就可以找到相关驱动了，不过它可能是有 BUG 的。

### 第二部分

General setup ---> 常规安装选项

Local version - append to kernel release

[\*] Support for paging of anonymous memory (swap)

这个选项将使你的内核支持虚拟内存，也就是让你的计算机好象拥有比实际内存更多 多的内存空间用来执行很大的程序。默认是选择的。

#### System V IPC

是组系统调用及函数库，它能让程序彼此间同步进行交换信息。某些程序以及 DOS 模拟环境都需要它。为进程提供通信机制，这将使系统中各进程间有交换信息与保持同步的能力。有些程序只有在选 Y 的情况下才能运行，所以不用考虑，这里一定要选。

#### POSIX Message Queues

POSIX 消息队列，这是 POSIX IPC 中的一部分

#### BSD Process Accounting

一般用户所执行的程序，可通知内核将程序统计数据写成文件，详细记录相关信息。

#### Sysctl support

此项功能可在不重新编译内核或是重新开机的前提下，动态变更某些特定的内核参数和变量，赋予内核较大的弹性。除非内存太少或是编译出来的内核是给安装、救援磁盘所使用，否则这个选项一定要选上。这将提供一个接口让你可以动态的更改一些核心参数与变量，而不需要重新启动系统。打开这个选项将会增加内核的体积至少 8KB。如果你的内核仅用制作安装与恢复系统系统盘那么可以不选，以减少对内存的占用。

#### Auditing support

审计支持，用于和内核的某些子模块同时工作，例如 SELinux。只有选择此项及它的子项，才能调用有关审计的系统调用。

#### --- Support for hot-pluggable devices

是否支持热插拔的选项，肯定要选上。不然 USB、PCMCIA 等这些设备都用不了。

#### Kernel Userspace Events

内核中分为系统区和用户区，这里系统区和用户区进行通讯的一种方式，选上。

#### Kernel .config support

这将会把内核的配置信息与相关的文档说明编译进内核中，以后可以使用一些工具来提取它用来重新构建内核，一般不用选它。

[ ] Configure standard kernel features (for small systems) --->

这是为了编译某些特殊用途的内核使用的，例如引导盘系统。通常你可以不选择这一选项，你也不用对它下面的子项操心了。

## 第三部分

Loadable module support ---> 可引导模块支持 建议作为模块加入内核

[\*] Enable loadable module support

这个选项可以让你的内核支持模块，模块是什么呢？模块是一小段代码，编译后可在系统内核运行时动态的加入内核，从而为内核增加一些特性或是对某种硬件进行支持。一般一些不常用到的驱动或特性可以编译为模块以减少内核的体积。在运行时可以使用 modprobe 命令来加载它到内核中去(在不需要时还可以移除它)。一些特性是否编译为模块的原则是，不常使用的，特别是在系统启动时不需要的驱动可以将其编译为模块，如果是一些在系统启动时就要用到的驱动比如说文件系统，系统总线的支持就不要编为模块了，否在无法启动系统。

[\*] Module unloading

这个选项可以让你卸载不再使用的模块，如果不选的话你将不能卸载任何模块(有些模块一旦加载就不能卸载，不管是否选择了这个选项)。不选择这个选项会让你的内核体积减小一点。

[\*] Forced module unloading

允许强制卸载正在使用中的模块(比较危险)

[ ] Module versioning support (EXPERIMENTAL)

允许使用其他内核版本的模块(可能会出问题)

[ ] Source checksum for all modules

这个功能是为了防止更改了内核模块的代码但忘记更改版本号而造成版本冲突。我估计现在没有哪家公司在开发中还没使用版本控制工具，所以不需要这项了。如果你不是自己写内核模块，那就更不需要这一选项了。

#### [\*] Automatic kernel module loading

一般情况下，如果用户的内核在某些任务中要使用一些被编译为模块的驱动或特性时，我们要先使用 modprobe 命令来加载它，内核才能使用。不过，如果你选择了这个选项，在内核需要一些模块时它可以自动调用 modprobe 命令来加载需要的模块，这是个很棒的特性，当然要选 Y 哟。

## 第四部分

System Type ---> 系统类型

ARM system type (Samsung S3C2410) ---> ARM 系统，基于 S3C2410

S3C24XX Implementations ---> 基于 S3C24xx 架构的实现

--- S3C2410 Boot

--- S3C2410 Setup

[\*] S3C2410 DMA support 支持 DM 功能

[ ] S3C2410 DMA support debug 支持 DMA 调试功能

(0) S3C2410 UART to use for low-level messages

--- Processor Type 处理器类型

--- Processor Features 处理器特性

[\*] Support Thumb user binaries 支持 Thumb 指令集

[ ] Disable I-Cache 禁止指令缓存 ( Instruction Cache , I-Cache )

[ ] Disable D-Cache 禁止数据缓存 ( Data Cache , D-Cache )

[ ] Force write through D-cache 强制回写数据缓存

## 第五部分

Bus support --->总线支持

PCCARD (PCMCIA/CardBus) support --->

< > PCCard (PCMCIA/CardBus) support

PCMCIA 卡支持

## 第六部分

Kernel Features

[ ] Symmetric Multi-Processing (EXPERIMENTAL)

[ ] Preemptible Kernel (EXPERIMENTAL)

## 第七部分

Boot options --->

(0x0) Compressed ROM boot loader base address

(0x0) Compressed ROM boot loader BSS address

(console=ttyS0 , 115200) Default kernel command string

[ ] Kernel Execute-In-Place from ROM

## 第八部分

Floating point emulation --->

--- At least one emulation must be selected

- [\*] NWFPE math emulation
- [ ] Support extended precision
- [ ] FastFPE math emulation (EXPERIMENTAL)

## 第九部分

Userspace binary formats ---> 支持的可执行文件格式

- [\*] Kernel support for ELF binaries

ELF 是开放平台下最常用的二进制文件格式，支持动态连接，支持不同的硬件平台.除非你知道自己在做什么，否则必选

- < > Kernel support for a.out and ECOFF binaries

这是早期 UNIX 系统的可执行文件格式，目前已经被 ELF 格式取代。

- < > Kernel support for MISC binaries

允许插入二进制的封装层到内核中，使用 Java , .NET , Python , Lisp 等语言编写的程序时需要它

- < > RISC OS personality

## 第十部分

Power management options ---> 电源管理选项

- [ ] Power Management support

电源管理有 APM 和 ACPI 两种标准且不能同时使用.即使关闭该选项，X86 上运行的 Linux 也会在空闲时发出 HLT 指令将 CPU 进入睡眠状态

## 第十一部分

Device Drivers ---> 设备驱动程序

Generic Driver Options --->

驱动程序通用选项

[\*] Select only drivers that don't need compile-time external firmware

只显示那些不需要内核对外部设备的固件作 map 支持的驱动程序，除非你有某些怪异硬件，否则请选上

[\*] Prevent firmware from being built

不编译固件. 固件一般是随硬件的驱动程序提供的，仅在更新固件的时候才需要重新编译.

建议选上

<\*> Hotplug firmware loading support

加载热插拔固件支持， 在内核树之外编译的模块可能需要它

[] Driver Core verbose debug messages

让驱动程序核心在系统日志中产生冗长的调试信息，仅供调试

Memory Technology Devices (MTD) --->

特殊的存储技术装置，如常用于数码相机或嵌入式系统的闪存卡

Parallel port support --->

并口支持(传统的打印机接口)

Plug and Play support --->

即插即用支持，若未选则应当在 BIOS 中关闭"PNP OS". 这里的选项与 PCI 设备无关

Block devices --->

ATA/ATAPI/MFM/RLL support --->

这个是有关各种接口的硬盘/光驱/磁带/软盘支持的，内容太多了，使用缺省的选项吧，

如果你使用了比较特殊的设备，比如 PCMCIA 等，就到里面自己找相应的选项吧

SCSI device support --->

SCSI 设备

Multi-device support (RAID and LVM) --->

多设备支持(RAID 和 LVM).RAID 和 LVM 的功能是使多个物理设备组建成一个单独的逻辑磁盘

Fusion MPT device support --->

Fusion MPT 设备支持

IEEE 1394 (FireWire) support --->

IEEE 1394(火线)

I2O device support --->

I2O(智能 IO)设备使用专门的 I/O 处理器负责中断处理/缓冲存取/数据传输等烦琐任务以减少 CPU 占用，一般的主板上没这种东西

Networking support --->

网络支持

ISDN subsystem --->

综合业务数字网(Integrated Service Digital Network)

Input device support --->

输入设备

Character devices --->

字符设备

I2C support --->

I2C 是 Philips 极力推动的微控制应用中使用的低速串行总线协议，可用于监控电压/风扇转速/温度等.SMBus(系统管理总线)是 I2C 的子集.除硬件传感器外"Video For Linux"也需要该模块的支持

Misc devices --->

Multimedia devices --->

多媒体设备

Graphics support --->

图形设备/显卡支持

Sound --->

声卡

USB support --->

USB 支持

MMC/SD Card support --->

MMC/SD 卡支持

## 第十二部分

File systems ---> 文件系统

<\*> Second extended fs support

Ext2 文件系统是 Linux 的标准文件系统，擅长处理稀疏文件

[\*] Ext2 extended attributes

Ext2 文件系统扩展属性(与 inode 关联的 name:value 对)支持

[\*] Ext2 POSIX Access Control Lists

POSIX ACL(访问控制列表)支持，可以更精细的针对每个用户进行访问控制，需要外部库  
和程序的支持

[ ] Ext2 Security Labels

安全标签允许选择使用不同的安全模型实现(如 SELinux)的访问控制模型，如果你没有使  
用需要扩展属性的安全模型就别选

<\*> Ext3 journalling file system support

Ext3 性能平庸，使用 journal 日志模式时数据完整性非常好(但怪异的是此时多线程并发读写速度却最快)

[\*] Ext3 extended attributes

Ext3 文件系统扩展属性(与 inode 关联的 name:value 对)支持

[] Ext3 POSIX Access Control Lists

POSIX ACL(访问控制列表)支持，可以更精细的针对每个用户进行访问控制，需要外部库和程序的支持

[] Ext3 Security Labels

安全标签允许选择使用不同的安全模型实现(如 SELinux)的访问控制模型，如果你没有使用需要扩展属性的安全模型就别选

[ ] JBD (ext3) debugging support 仅供开发者使用

< > Reiserfs support

性能几乎全面超越 Ext2(处理稀疏文件比 Ext2 慢)，小文件(小于 4k)性能非常突出，创建和删除文件速度最快，处理大量目录和文件(5k-20k)时仍然非常迅速.日志模式建议使用 Ordered，追求极速可使用 Writeback 模式，追求安全可使用 Journal 模式.建议使用 noatime , notail 选项挂载分区以提高速度和避免 bug.用于 NFS 和磁盘限额时需要额外的补丁

< > JFS filesystem support

IBM 的 JFS 文件系统

XFS support --->

碎片最少，多线程并发读写最佳，大文件(>64k)性能最佳，创建和删除文件速度较慢.由于 XFS 在内存中缓存尽可能多的数据且仅当内存不足时才会将数据刷到磁盘，所以应当仅在确保电力供应不会中断的情况下才使用 XFS

< > Minix fs support

古老董文件系统

< > ROM file system support

用于嵌入式系统的内存文件系统的支持

[ ] Quota support

磁盘配额支持，限制某个用户或者某组用户的磁盘占用空间，Ext2/Ext3/Reiserfs 都支持

它

<\*> Kernel automounter support

内核自动加载远程文件系统(v3，就算选也不选这个旧的)

< > Kernel automounter version 4 support (also supports v3)

新的(v4)的内核自动加载远程文件系统的支持，也支持 v3

CD-ROM/DVD Filesystems --->

CD-ROM/DVD 文件系统

DOS/FAT/NT Filesystems --->

DOS/Windows 的文件系统

Pseudo filesystems --->

伪文件系统

Miscellaneous filesystems --->

非主流的杂项文件系统

Network File Systems --->

网络文件系统

Partition Types --->

高级磁盘分区类型，不确定可以全不选

## Native Language Support --->

本地语言支持.如果你仅仅使用几种主流的 Linux 文件系统(ext2/3/4 , Reiserfs , JFS , XFS) , 就不需要这个东西.但是如果你需要使用 FAT/NTFS 分区的话 , 就需要这个东西了.

## 第十三部分

### Profiling support --->

#### [ ] Profiling support (EXPERIMENTAL)

对系统的活动进行分析 , 仅供内核开发者使用

## 第十四部分

### Kernel hacking ---> 内核 hack 选项 , 普通用户是用不着这个功能的

#### [ ] Show timing information on printks

在 printk 的输出中包含时间信息 , 可以用来分析内核启动过程各步骤所用时间

#### [\*] Kernel debugging

不是内核开发者的别选

#### [ ] Magic SysRq key

不懂的千万别选

#### (14) Kernel log buffer size (16 => 64KB , 17 => 128KB)

#### [ ] Collect scheduler statistics

#### [ ] Debug memory allocations

#### [ ] Spinlock debugging

#### [ ] Sleep-inside-spinlock checking

#### [ ] kobject debugging

- [ ] Compile the kernel with debug info
  - [ ] Debug Filesystem
  - [\*] Verbose user fault messages
  - [ ] Wait queue debugging
  - [ ] Verbose kernel error messages
  - [\*] Kernel low-level debugging functions
  - [ ] Kernel low-level debugging via EmbeddedICE DCC channel
  - [\*] Kernel low-level debugging messages via S3C2410 UART
- (0) S3C2410 UART to use for low-level debug

## 第十五部分

Security options ---> 安全选项，这里的选项不明白的建议不要选，否则有可能弄巧成拙。

- [ ] Enable access key retention support

在内核中保留 authentication token 和 access key

- [ ] Enable different security models

允许内核选择不同的安全模型，如果未选中则内核将使用默认的安全模型

## 第十六部分

Cryptographic options ---> 加密选项

- [\*[\*] Cryptographic API

提供核心的加密 API 支持。这里的加密算法被广泛的应用于驱动程序通信协议等机制中。子选项可以全不选，内核中若有其他部分依赖它，会自动选上

- [ ] HMAC support

为 IPSec 所必须，可为 PPPoE 提供压缩支持

< > Null algorithms

NULL 加密算法(什么也不做)，用于 IPsec 协议的封装安全载荷模块(ESP)

< > MD4 digest algorithm

老旧的摘要算法，已经过时

<\*> MD5 digest algorithm

主流摘要算法，128 位(已被中国山东大学王小云攻破，可以快速找到碰撞)

< > SHA1 digest algorithm

主流摘要算法，160 位(已被中国山东大学王小云攻破，可以快速找到碰撞)，速度与 MD5 相当

< > SHA256 digest algorithm

更好的摘要算法，256 位，速度较 SHA1 稍慢

< > SHA384 and SHA512 digest algorithms

更好的摘要算法，384/512 位，速度大约只有 SHA1 的 40-50%

< > Whirlpool digest algorithms

最安全的摘要算法，512 位，已被列入 ISO 标准，目前最新版本为 3.0(2003 年发布)

< > Tiger digest algorithms

号称最快的摘要算法，192 位，专门为 64 位 CPU 进行了优化

<\*> DES and Triple DES EDE cipher algorithms

老迈的(DES)和尚佳的(Triple DES)对称加密算法

< > Blowfish cipher algorithm

又老又慢的对称加密算法

< > Twofish cipher algorithm

很强的对称加密算法，使用较广

< > Serpent cipher algorithm

很强的对称加密算法

< > AES cipher algorithms

最佳的对称加密算法(Rijndael) , 128/192/256 位 , 强度最高 , 快速且节省内存

< > CAST5 (CAST-128) cipher algorithm

对称加密算法

< > CAST6 (CAST-256) cipher algorithm

对称加密算法

< > TEA and XTEA cipher algorithms

较弱的对称加密算法

< > ARC4 cipher algorithm

脆弱的流对称加密算法

< > Khazad cipher algorithm

对称加密算法

< > Anubis cipher algorithm

对称加密算法

< > Deflate compression algorithm

压缩算法 , 当在 IPSec 中使用 IP COMP 协议时才需要

< > Michael MIC keyed digest algorithm

摘要算法 , 仅仅用于校验 iSCSI 设备传输的数据 , 因为算法本身比较脆弱

< > CRC32c CRC algorithm

摘要算法 , 可用于校验 iSCSI 设备传输的数据

< > Testing module

快速且丑陋的测试模块

Hardware crypto devices --->

仅有 VIA C7 系列处理器支持硬件加密(VIA PadLock 高级加密引擎)

## 第十七部分

Library routines ---> 库子程序

--- CRC-CCITT functions

传送 8-bit 字符，欧洲标准

--- CRC32 functions

用于点对点的同步数据传输中，传输网络数据包所必须的

<\*> CRC32c (Castagnoli , et al) Cyclic Redundancy-Check

用于点对点的同步数据传输中，比如 iSCSI 设备

## 第十八部分

Load an Alternate Configuration File

读入一个外部配置文件

Save Configuration to an Alternate File

将配置保存到一个外部文件

## 附录四 Linux 下多核处理器相关知识

多核处理器是指在一枚处理器中集成两个或多个完整的计算引擎（内核）。多核技术的开发源于工程师们认识到，仅仅提高单核芯片的速度会产生过多热量且无法带来相应的性能改善，先前的处理器产品就是如此。他们认识到，在先前产品中以那种速率，处理器产生的热量很快会超过太阳表面。即便是没有热量问题，其性价比也令人难以接受，速度稍快的处理器价格要高很多。

基于以上事实，工程师们开发了多核芯片，使之满足‘横向扩展’（而非‘纵向扩充’）的方法，从而提高性能。

1. 在 Linux 下，如何确认是多核或多 CPU：

```
#cat /proc/cpuinfo
```

如果有多个类似以下的项目，则为多核或多 CPU：

```
processor : 0
```

.....

```
processor : 1
```

2. Linux 下，如何看每个 CPU 的使用率：

```
#top -d 1
```

之后按下 1. 则显示多个 CPU

```
Cpu0 : 1.0%us, 3.0%sy, 0.0%ni, 96.0%id, 0.0%wa, 0.0%hi,  
0.0%si, 0.0%st  
Cpu1 : 0.0%us, 0.0%sy, 0.0%ni, 100.0%id, 0.0%wa, 0.0%hi, 0.0%si,  
0.0%st
```

3. 如何察看某个进程在哪个 CPU 上运行：

```
#top -d 1
```

之后按下 f.进入 top Current Fields 设置页面：

选中 : j: P = Last used cpu (SMP)

则多了一项 : P 显示此进程使用哪个 CPU。

Sam 经过试验发现：同一个进程，在不同时刻，会使用不同 CPU Core.这应该是 Linux Kernel SMP 处理的。

#### 4. 配置 Linux Kernel 使之支持多 Core :

内核配置期间必须启用 CONFIG\_SMP 选项，以使内核感知 SMP。

Processor type and features ---> Symmetric multi-processing support

察看当前 Linux Kernel 是否支持（或者使用）SMP

```
#uname -a
```

#### 5. Kernel 2.6 的 SMP 负载平衡：

在 SMP 系统中创建任务时，这些任务都被放到一个给定的 CPU 运行队列中。通常来说，我们无法知道一个任务何时是短期存在的，何时需要长期运行。因此，最初任务到 CPU 的分配可能并不理想。

为了在 CPU 之间维护任务负载的均衡，任务可以重新进行分发：将任务从负载重的 CPU 上移动到负载轻的 CPU 上。Linux 2.6 版本的调度器使用负载均衡（load balancing）提供了这种功能。每隔 200ms，处理器都会检查 CPU 的负载是否不均衡；如果不均衡，处理器就会在 CPU 之间进行一次任务均衡操作。

这个过程的一点负面影响是新 CPU 的缓存对于迁移过来的任务来说是冷的（需要将数据读入缓存中）。

记住 CPU 缓存是一个本地（片上）内存，提供了比系统内存更快的访问能力。如果一个任务是在某个 CPU 上执行的，与这个任务有关的数据都会被放到这个 CPU 的本地缓存中，这就称为热的。如果对于某个任务来说，CPU 的本地缓存中没有任何数据，那么这个缓存就称为冷的。

不幸的是，保持 CPU 繁忙会出现 CPU 缓存对于迁移过来的任务为冷的情况。

## 6. 应用程序如何利用多 Core :

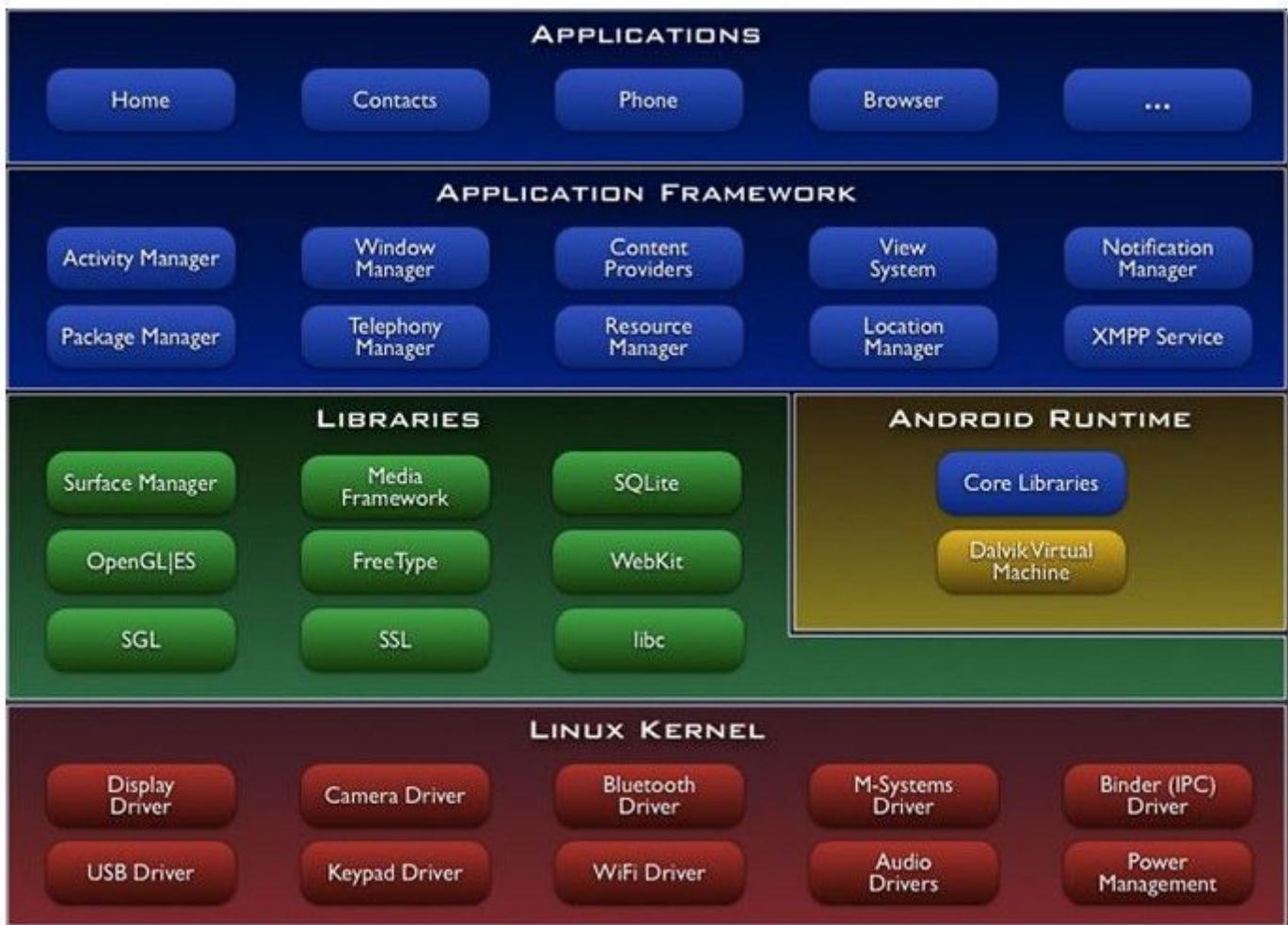
开发人员可将可并行的代码写入线程，而这些线程会被 SMP 操作系统安排并发运行。

另外，Sam 设想，对于必须顺序执行的代码。可以将其分为多个节点，每个节点为一个 thread，并在节点间放置 channel. 节点间形如流水线。这样也可以大大增强 CPU 利用率。

## 附录五 Android 系统架构

### 1、架构图直观

下面这张图展示了 Android 系统的主要组成部分：



可以很明显看出，Android 系统架构由 5 部 分组成，分别是：Linux Kernel、Android Runtime、Libraries、Application Framework、Applications。第二部分将详细介绍这 5 个部分。

### 2、架构详解

现在拿起手术刀来剖析各个部分。其实这部分 SDK 文档已经帮我们做得很好了，我们要做的就是拿来主义，然后再加上自己理解。下面自底向上分析各层。

## 2.1、Linux Kernel

Android 基于 Linux 2.6 提供核心系统服务，例如：安全、内存管理、进程管理、网络堆栈、驱动模型。Linux Kernel 也作为硬件和软件之间的抽象层，它隐藏具体硬件细节而为上层提供统一的服务。

如果你学过计算机网络知道 OSI/RM，就会 知道分层的好处就是使用下层提供的服务而为上层提供统一的服务，屏蔽本层及以下层的差异，当本层及以下层发生了变化不会影响到上层。也就是说各层各司其职，各层提供固定的 SAP ( Service Access Point )，专业点可以说是高内聚、低耦合。

如果你只是做应用开发，就不需要深入了解 Linux Kernel 层。

## 2.2、Android Runtime

Android 包含一个核心库的集合，提供大部分在 Java 编程语言核心类库中可用的功能。每一个 Android 应用程序是 Dalvik 虚拟机中的实例，运行在他们自己的进程中。Dalvik 虚拟机设计成，在一个设备可以高效地运行多个虚拟机。 Dalvik 虚拟机可执行文件格式是.dex，dex 格式是专为 Dalvik 设计的一种压缩格式，适合内存和处理器速度有限的系统。

大多数虚拟机包括 JVM 都是基于栈的，而 Dalvik 虚拟机则是基于寄存器的。两种架构各有优劣，一般而言，基于栈的机器需要更多指令，而基于寄存器的机器 指令更大。dx 是一套工具，可以将 Java .class 转换成 .dex 格式。一个 dex 文件通常会有多个.class。由于 dex 有时必须进行最佳化，会使文件大小增加 1-4 倍，以 ODEX 结尾。

Dalvik 虚拟机依赖于 Linux 内核提供基本功能，如线程和底层内存管理。

## 2.3、Libraries(本地库)

Android 包含一个 C/C++ 库的集合，供 Android 系统的各个组件使用。这些功能通过 Android 的应用程序框架（ application framework ）暴露给开发者。下面列出一些核心库：

系统 C 库 —— 标准 C 系统库 ( libc ) 的 BSD 衍生，调整为基于嵌入式 Linux 设备

媒体库 —— 基于 PacketVideo 的 OpenCORE。这些库支持播放和录制许多流行的音频和视频格式，以及静态图像文件，包括 MPEG4、H.264、MP3、AAC、AMR、JPG、PNG

界面管理 —— 管理访问显示子系统和无缝组合多个应用程序的二维和三维图形层

LibWebCore —— 新式的 Web 浏览器引擎，驱动 Android 浏览器和内嵌的 web 视图

SGL —— 基本的 2D 图形引擎

3D 库 —— 基于 OpenGL ES 1.0 APIs 的实现。库使用硬件 3D 加速或包含高度优化的 3D 软件光栅

FreeType —— 位图和矢量字体渲染

SQLite —— 所有应用程序都可以使用的强大而轻量级的关系数据库引擎

## 2.4、 Application Framework

通过提供开放的开发平台，Android 使开发者能够编制极其丰富和新颖的应用程序。开发者可以自由地利用设备硬件优势、访问位置信息、运行后台服务、设置闹钟、向状态栏添加通知等等，很多很多。

开发者 可以完全使用核心应用程序所使用的框架 APIs。应用程序的体系结构旨在简化组件的重用，任何应用程序都能发布他的功能且任何其他应用程序可以使用这些功能（需要服从框架执行的安全限制）。这一机制允许用户替换组件。

所有的应用程序其实是一组服务和系统，包括：

视图 ( View ) —— 丰富的、可扩展的视图集合，可用于构建一个应用程序。包括包括列表、网格、文本框、按钮，甚至是内嵌的网页浏览器

内容提供者 ( Content Providers ) —— 使应用程序能访问其他应用程序（如通讯录）的数据，或共享自己的数据

资源管理器 ( Resource Manager ) —— 提供访问非代码资源，如本地化字符串、图形和布局文件

通知管理器 ( Notification Manager ) ——使所有的应用程序能够在状态栏显示自定义警告

活动管理器 ( Activity Manager ) ——管理应用程序生命周期，提供通用的导航回退功能

## 2.5、Applications

Android 装配一个核心应用程序集合，包括电子邮件客户端、SMS 程序、日历、地图、浏览器、联系人和其他设置。所有应用程序都是用 Java 编程语言写的。更加丰富 的应用程序有待我们去开发！

### 源码结构

Google 提供的 Android 包含了原始 Android 的目标机代码，主机编译工具、仿真环境，代码包经过解压缩后，第一级别的目录和文件如下所示：

```
.  
|-- Makefile ( 全局的 Makefile )  
|-- bionic ( Bionic 含义为仿生，这里面是一些基础的库的源代码 )  
|-- bootloader ( 引导加载器 )  
|-- build ( build 目录中的内容不是目标所用的代码，而是编译和配置所需要的脚本和工具 )  
|-- dalvik ( JAVA 虚拟机 )  
|-- development ( 程序开发所需要的模板和工具 )  
|-- external ( 目标机器使用的一些库 )  
|-- frameworks ( 应用程序的框架层 )  
|-- hardware ( 与硬件相关的库 )  
|-- kernel ( Linux 的源代码 )  
|-- packages ( Android 的各种应用程序 )  
|-- prebuilt ( Android 在各种平台下编译的预置脚本 )
```

|-- recovery ( 与目标的恢复功能相关 )

`-- system ( Android 的底层的一些库 )

# 附录六 iTOP-4412 源码的开发版本下载和使用

## 6.1 Uboot 的下载和编译

代码下载分为 uboot,kernel ,android4.0 三大部分。iTOP-4412 所有的 uboot 集成到一起，下载地址一样，只是编译参数稍微有点不同。

如下图表所示，Ubuntu 系统联网之后，通过 git 命令可以下载源码和编译对应的 uboot 镜像。

uboot 源码下载方法	# git clone https://github.com/TOPEET-Develop/iTop4412_uboot_public_merge.git #cd iTop4412_uboot_public_merge
<b>Android4.0 文件系统 uboot 的编译</b>	
SCP 核心板 1G 内存	# ./build_uboot.sh SCP_1GDDR # make
SCP 核心板 2G 内存	# ./build_uboot.sh SCP_2GDDR # make
POP 核心板 1G 内存	# ./build_uboot.sh POP_1GDDR # make
POP 核心板 2G 内存	# ./build_uboot.sh POP_2GDDR # make
<b>Android4.4 文件系统 uboot 的编译</b>	
SCP 核心板 1G 内存	# ./build_uboot.sh SCP_1GDDR # make
SCP 核心板 2G 内存	# ./build_uboot.sh SCP_2GDDR # make
POP 核心板 1G 内存	# ./build_uboot.sh POP_1GDDR # make

<b>Linux Qt 文件系统 uboot 的编译</b>	
SCP 核心板 1G 内存	# ./build_u-boot.sh SCP_1GDDR # make
SCP 核心板 2G 内存	# ./build_u-boot.sh SCP_2GDDR # make
POP 核心板 1G 内存	# ./build_u-boot.sh POP_1GDDR # make
POP 核心板 2G 内存	# ./build_u-boot.sh POP_2GDDR # make
<b>Ubuntu 文件系统 uboot 的编译</b>	
SCP 核心板 1G 内存	# ./build_u-boot.sh SCP_1GDDR_Ubuntu # make
SCP 核心板 2G 内存	# ./build_u-boot.sh SCP_2GDDR_Ubuntu # make
POP 核心板 1G 内存	# ./build_u-boot.sh POP_1GDDR_Ubuntu # make

## 6.2 Kernel 源码下载及编译

Kernel 源码分为两套，一套是针对 Android4.4 文件系统，一套是针对其它所有文件系统。

Kernel 源码适用于 SCP 核心板，POP 核心板, POP2G DDR 核心板；

Kernel 源码适用于精英底板,英文名称 elite 及全能底板,英文名称 supper

注意：kernel 配置时 config\_for\_XX YY ZZ 文件很重要，其中：

XX 是操作系统，包括 android,linux,Ubuntu

YY 是 核心板类型，包括 scp, pop ,pop2G

ZZ 是底板类型，包括精英版 elite, 全能版 supper

Android4.4 对应的 Kernel 源码下载及编译方法如下表所示。

Android4.4 的 kernel 目前不区分底板是精英版还是全能版，仅仅区分核心板类型。

<b>kernel 源码下载方法</b>	#git clone https://github.com/TOPEET-Develop/iTop4412_kernel_public_merge.git
SCP 核心板	常规配置： #cp config_for_android_scp .config #make zImage WiFi 支持配置： #cp config_for_android_scp_wifi&Bluetooth .config
POP 核心板	常规配置： #cp config_for_android_pop .config #make zImage WiFi 支持配置： #cp config_for_android_pop_wifi&Bluetooth .config #make zImage

其它所有文件系统使用的 Kernel 代码下载和编译方法如下表所示：

<b>kernel 源码下载方法</b>	#git clone https://github.com/TOPEET-Develop/iTop4412_kernel_public_merge.git
<b>Android4.0 文件系统 kernel 的编译</b>	
SCP 核心板+精英底板	#cp config_for_android_scp_elite .config #make zImage
SCP 核心板+全能底板	#cp config_for_android_scp_supper .config #make zImage
POP 核心板+精英底板	#cp config_for_android_pop_elite .config #make zImage
POP 核心板+全能底板	#cp config_for_android_pop_supper .config #make zImage
POP2G 核心板+精英底板	#cp config_for_android_pop2G_elite .config #make zImage

POP2G 核心板+全能底板	#cp config_for_android_pop2G_supper .config #make zImage
<b>Linux Qt 文件系统 Kernel 的编译</b>	
SCP 核心板+精英底板	#cp config_for_linux_scp_elite .config #make zImage
SCP 核心板+全能底板	#cp config_for_ubuntu_scp_supper .config #make zImage
POP 核心板+精英底板	#cp config_for_linux_pop_elite .config #make zImage
POP 核心板+全能底板	#cp config_for_linux_pop_supper .config #make zImage
POP2G 核心板+精英底板	#cp config_for_linux_pop2G_elite .config #make zImage
POP2G 核心板+全能底板	#cp config_for_linux_pop2G_supper .config #make zImage
<b>Ubuntu 文件系统 Kernel 的编译</b>	
SCP 核心板	LCD 显示版本：  #cp config_for_ubuntu_scp .config #make zImage  HDMI 显示版本：  #cp config_for_ubuntu_hdmi_scp .config #make zImage
POP 核心板	LCD 显示版本：  #cp config_for_ubuntu_pop .config #make zImage  HDMI 显示版本：  #cp config_for_ubuntu_hdmi_pop .config #make zImage

## 6.3 文件系统的下载

### 6.3.1 android4.0 代码下载和编译

Android4.0 源码可以从光盘，网盘获取稳定版本，也可以从 GitHub 下载我们的开发版本。GitHub 仅提供源码下载，不提供二进制下载，二进制文件存放在光盘和网盘中。

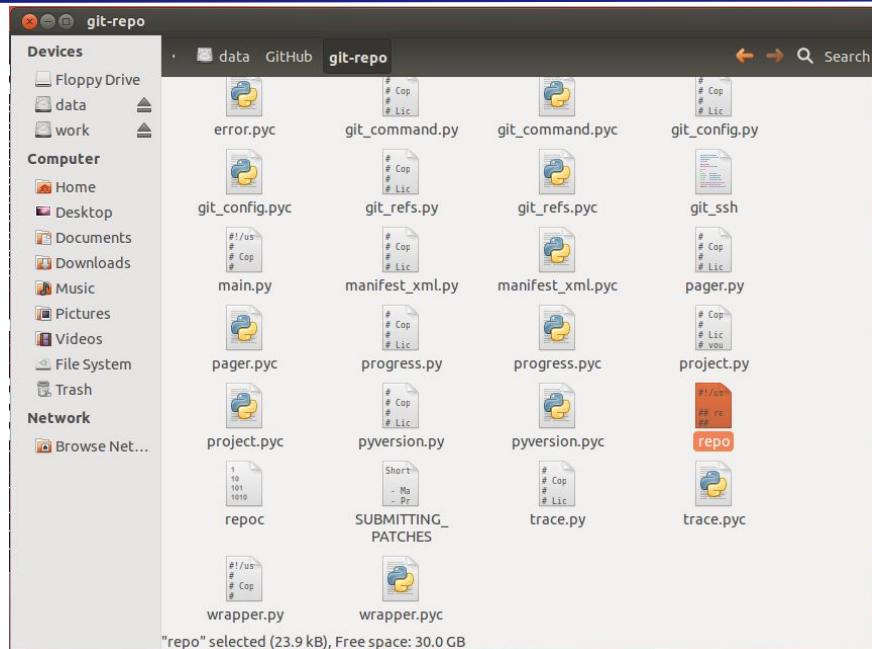
#### 6.3.1.1 repo 下载

android 代码管理不同于 uboot,kernel，由于 Android 代码比较庞大，我们把 Android 项目按照文件夹进行项目拆分，android 源码根目录下面的每个子目录都会划分为一个仓库或者多个仓库进行版本管理，Android 的代码下载需要使用 repo 工具，repo 工具为 Git 的封装，底层是使用 Git 命令进行下载的。

假设将要下载的 repo 工具存放于 /media/data/gitHub 目录：

```
#cd /media/data/GitHub/  
#git clone git://aosp.tuna.tsinghua.edu.cn/android/git-repo.git/
```

git-repo 仓库下载完成后会看到该文件夹内有 repo 脚本文件：



### 6.3.1.2 Android4.0 代码下载

假设我们的 Android4.0 代码存放在 /media/data/GitHub/iTop4412\_ICS\_git 目录：

```
# cd /media/data/GitHub/iTop4412_ICS_git
```

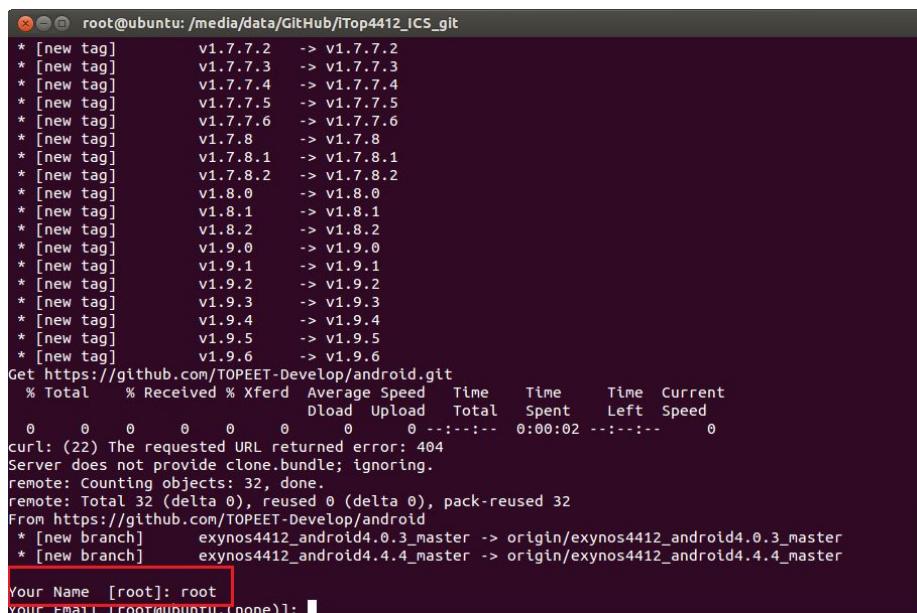
```
#..../git-repo/repo init -u https://github.com/TOPEET-Develop/android.git -b
```

## exynos4412\_android4.0.3\_master

注意上一条命令中，换行字符 “-b” 之后有空格。

注： repo init 命令中-u 参数指定 android 仓库下载地址， -b 参数指定仓库中的下载分支。

这里我下载的是 exynos4412\_android4.0.3\_master 分支，该命令执行过程中需要输入相关的信息，如下图所示：

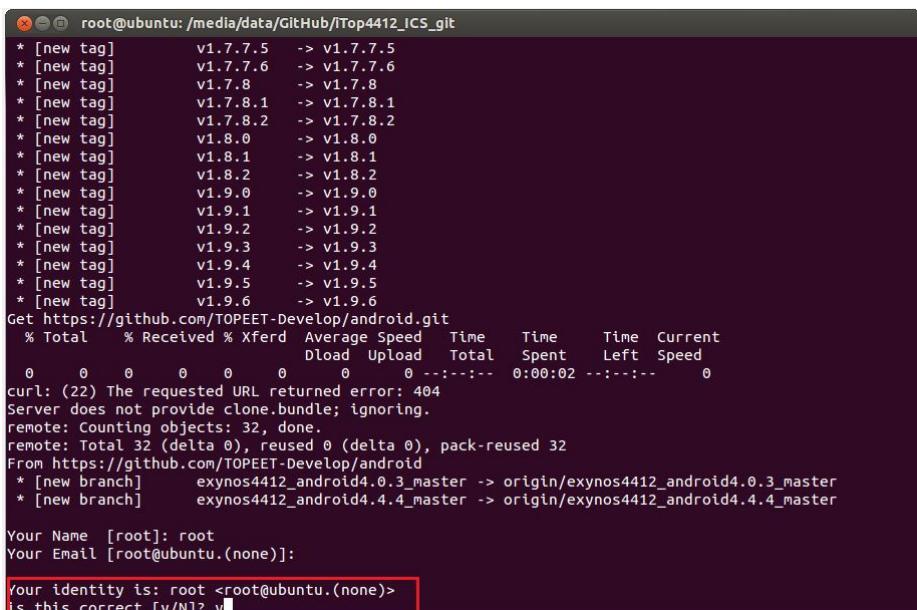


```
* [new tag]      v1.7.7.2    -> v1.7.7.2
* [new tag]      v1.7.7.3    -> v1.7.7.3
* [new tag]      v1.7.7.4    -> v1.7.7.4
* [new tag]      v1.7.7.5    -> v1.7.7.5
* [new tag]      v1.7.7.6    -> v1.7.7.6
* [new tag]      v1.7.8      -> v1.7.8
* [new tag]      v1.7.8.1    -> v1.7.8.1
* [new tag]      v1.7.8.2    -> v1.7.8.2
* [new tag]      v1.8.0      -> v1.8.0
* [new tag]      v1.8.1      -> v1.8.1
* [new tag]      v1.8.2      -> v1.8.2
* [new tag]      v1.9.0      -> v1.9.0
* [new tag]      v1.9.1      -> v1.9.1
* [new tag]      v1.9.2      -> v1.9.2
* [new tag]      v1.9.3      -> v1.9.3
* [new tag]      v1.9.4      -> v1.9.4
* [new tag]      v1.9.5      -> v1.9.5
* [new tag]      v1.9.6      -> v1.9.6
Get https://github.com/TOPEET-Develop/android.git
% Total % Received % Xferd Average Speed Time Time Current
          Dload Upload Total Spent Left Speed
0 0 0 0 0 0 0 0 --:--:-- 0:00:02 --:--:-- 0
curl: (22) The requested URL returned error: 404
Server does not provide clone.bundle; ignoring.
remote: Counting objects: 32, done.
remote: Total 32 (delta 0), reused 0 (delta 0), pack-reused 32
From https://github.com/TOPEET-Develop/android
 * [new branch]   exynos4412_android4.0.3_master -> origin/exynos4412_android4.0.3_master
 * [new branch]   exynos4412_android4.4.4_master -> origin/exynos4412_android4.4.4_master

Your Name [root]: root
Your Email [root@ubuntu.(none)]:
```

Your Name 输入 root

Your Email: 直接回车即可，然后在输入 y, 回车继续：



```
* [new tag]      v1.7.7.5    -> v1.7.7.5
* [new tag]      v1.7.7.6    -> v1.7.7.6
* [new tag]      v1.7.8      -> v1.7.8
* [new tag]      v1.7.8.1    -> v1.7.8.1
* [new tag]      v1.7.8.2    -> v1.7.8.2
* [new tag]      v1.8.0      -> v1.8.0
* [new tag]      v1.8.1      -> v1.8.1
* [new tag]      v1.8.2      -> v1.8.2
* [new tag]      v1.9.0      -> v1.9.0
* [new tag]      v1.9.1      -> v1.9.1
* [new tag]      v1.9.2      -> v1.9.2
* [new tag]      v1.9.3      -> v1.9.3
* [new tag]      v1.9.4      -> v1.9.4
* [new tag]      v1.9.5      -> v1.9.5
* [new tag]      v1.9.6      -> v1.9.6
Get https://github.com/TOPEET-Develop/android.git
% Total % Received % Xferd Average Speed Time Time Current
          Dload Upload Total Spent Left Speed
0 0 0 0 0 0 0 0 --:--:-- 0:00:02 --:--:-- 0
curl: (22) The requested URL returned error: 404
Server does not provide clone.bundle; ignoring.
remote: Counting objects: 32, done.
remote: Total 32 (delta 0), reused 0 (delta 0), pack-reused 32
From https://github.com/TOPEET-Develop/android
 * [new branch]   exynos4412_android4.0.3_master -> origin/exynos4412_android4.0.3_master
 * [new branch]   exynos4412_android4.4.4_master -> origin/exynos4412_android4.4.4_master

Your Name [root]: root
Your Email [root@ubuntu.(none)]:
```

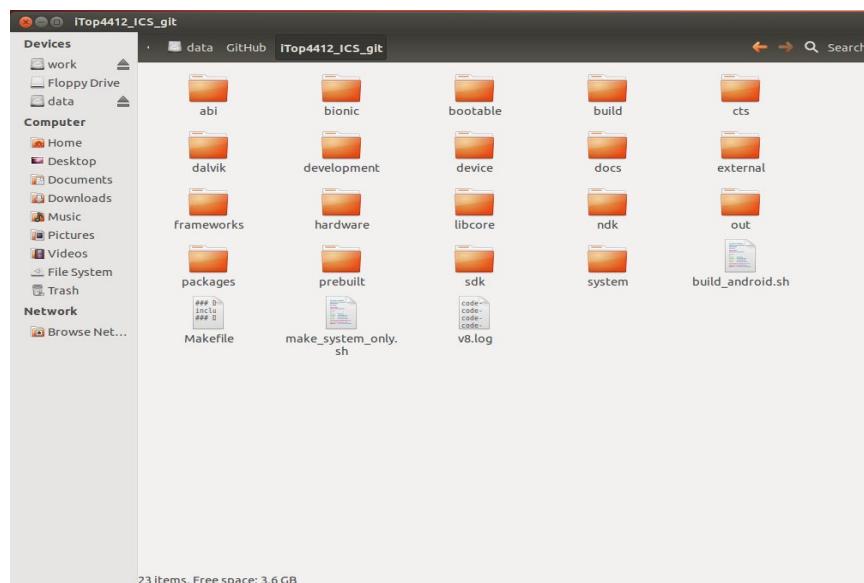
Your identity is: root <root@ubuntu.(none)>  
Is this correct [y/N]? y

#./git-repo/repo sync

注： repo sync 开始下载 Android 代码，下载过程与网络环境有一定的关系，如果下载过程中长时间没有进度显示，可以 `ctrl+c` 终止下载，然后执行 `repo sync` 命令重新开始，下图为源码下载过程中的进度显示：

```
root@ubuntu: /media/data/GitHub/iTop4412_ICS_git
* [new tag]      android-cts-4.0.3_r1 -> android-cts-4.0.3_r1
* [new tag]      android-cts-4.0.3_r2 -> android-cts-4.0.3_r1
* [new tag]      android-cts-4.0_r1 -> android-cts-4.0_r1
* [new tag]      android-cts-4.1_r1 -> android-cts-4.1_r1
* [new tag]      android-cts-4.1_r2 -> android-cts-4.1_r2
* [new tag]      android-cts-4.1_r4 -> android-cts-4.1_r4
* [new tag]      android-cts-4.2_r1 -> android-cts-4.2_r1
* [new tag]      android-cts-4.2_r2 -> android-cts-4.2_r2
* [new tag]      android-cts-4.4_r1 -> android-cts-4.4_r1
* [new tag]      android-cts-5.0_r3 -> android-cts-5.0_r3
* [new tag]      android-cts-5.1_r1 -> android-cts-5.1_r1
* [new tag]      android-cts-5.1_r2 -> android-cts-5.1_r2
* [new tag]      android-cts-verifier-4.0.3_r1 -> android-cts-verifier-4.0.3_r1
* [new tag]      android-cts-verifier-4.0_r1 -> android-cts-verifier-4.0_r1
* [new tag]      android-l-preview_r2 -> android-l-preview_r2
* [new tag]      android-m-preview -> android-m-preview
* [new tag]      android-m-preview-1 -> android-m-preview-1
* [new tag]      android-sdk-4.0.3-tools_r1 -> android-sdk-4.0.3-tools_r1
* [new tag]      android-sdk-4.0.3_r1 -> android-sdk-4.0.3_r1
* [new tag]      android-sdk-4.4.2_r1 -> android-sdk-4.4.2_r1
* [new tag]      android-sdk-4.4.2_r1.0.1 -> android-sdk-4.4.2_r1.0.1
* [new tag]      android-sdk-adt_r16.0.1 -> android-sdk-adt_r16.0.1
* [new tag]      android-sdk-adt_r20 -> android-sdk-adt_r20
* [new tag]      android-sdk-support_r11 -> android-sdk-support_r11
* [new tag]      android-wear-5.0.0_r1 -> android-wear-5.0.0_r1
* [new tag]      android-wear-5.1.0_r1 -> android-wear-5.1.0_r1
* [new tag]      android-wear-5.1.1_r1 -> android-wear-5.1.1_r1
^CReceiving objects: 20% (50/247), 38.22 MiB | 12 KiB/s    31/247), 22.39 MiB | 23 KiB/s
```

代码下载完成后会在 `/media/data/GitHub/iTop4412_ICS_git` 目录下面显示 Android 系统源码目录：



每个文件夹目录下面都会有一个或者多个 Git 仓库，默认这些仓库是没有分支的，我们需要使用 `repo star` 命令创建分支，这里我们创建 `master` 分支，您也可以使用别的名字来定义分支名称，`repo branch` 命令可以查看创建的分支：

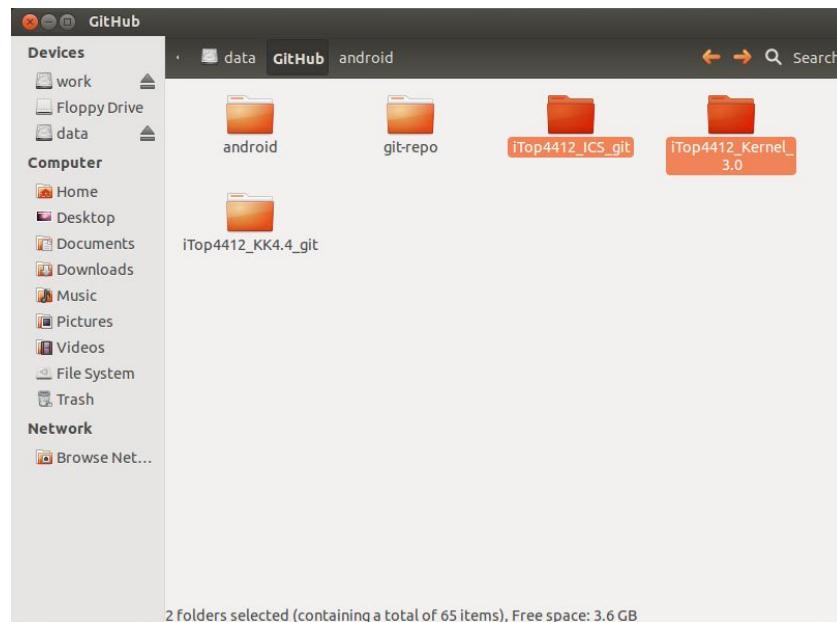
```
#..../git-repo/repo start master --all
```

```
#../git-repo/repo branch
```

```
root@ubuntu:/media/data/GitHub/iTop4412_ICS_git#  
root@ubuntu:/media/data/GitHub/iTop4412_ICS_git# .. /git-repo/repo branch  
* master           | in all projects  
root@ubuntu:/media/data/GitHub/iTop4412_ICS_git# █
```

### 6.3.1.3 Android4.0 源码编译

Android 代码下载完成后就可以进行编译了，Android 源码编译需要 kernel 支持，所以我们需要把 Android 代码与 kernel 代码放到同一目录下面：



注：上图中 iTOP4412\_Kernel\_3.0 为 Android 的内核代码，与 Android4.0 源码目录 iTOP4412\_ICS\_git 在同一级，另外编译 Android4.0 之前需要编译内核代码，在内核代码目录执行：

```
# cp config_for_android_XX_YY .config  
# make zImage  
# make modules
```

config\_for\_android\_XX\_YY 中的 XX 指核心板类型：scp, pop, pop2G, YY 指的是底板类型：精英版 elite，或者全能版 supper.

make modules 会编译驱动库文件\*.ko，Android4.0 会拷贝驱动库文件到 Android 的目录结构，否则 Android 的某些功能无法使用。

Android4.0 的执行编译命令：

```
#cd iTop4412_ICS_git  
# ./build_android.sh
```

另外 Android 代码下载、编译完成后，可以定期使用 repo sync 命令同步我们的 Git 仓库到本地，这样您的本地代码就跟我们的仓库保持同步更新了，编译后就会形成最新的系统镜像。

我们的 Android4.0 代码区分精英版和全能版，默认我们下载的代码编译形成的镜像可以在精英板运行，要编译全能板版本，只需要修改一项配置即可：

文件路径：iTop4412\_ICS\_git/device/samsung/smdk4x12/BoardConfig.mk

配置项：BOARD\_HAVE\_MPU6050，该值配置成 false 代表编译版本是精英板版本，如果设置成 true，则会编译全能板版本。

精英版配置：

```

# For Olca3.2 driver
#BOARD_WPA_SUPPLICANT_DRIVER := NL80211
#BOARD_HOSTAPD_DRIVER := NL80211
#WPA_SUPPLICANT_VERSION := VER_0_8_ATHEROS
#HOSTAPD_VERSION := VER_0_8_ATHEROS
#WIFI_DRIVER_MODULE_PATH := "/system/lib/modules/ath6kl_sdio.ko"
#WIFI_DRIVER_MODULE_NAME := "ath6kl_sdio"

#WIFI_SDIO_IF_DRIVER_MODULE_PATH := "/system/lib/modules/cfg80211.ko"
#WIFI_SDIO_IF_DRIVER_MODULE_NAME := "cfg80211"
#WIFI_SDIO_IF_DRIVER_MODULE_ARG := ""

BOARD_HAVE_MTK_MT6620 := true
BOARD_GPS_LIBRARIES := true
BOARD_WPA_SUPPLICANT_DRIVER := WEXT
WIFI_DRIVER_MODULE_PATH := /system/lib/modules/wlan.ko
WIFI_DRIVER_MODULE_NAME := wlan0
WPA_BUILD_SUPPLICANT := true
#CONFIG_CTRL_INTERFACE := y
WPA_SUPPLICANT_VERSION := VER_0_6_X
P2P_SUPPLICANT_VERSION := VER_0_8_X
BOARD_P2P_SUPPLICANT_DRIVER := NL80211

#add by cym 20130623
BOARD_HAVE_MPU6050 := false
#end add
#add by cym 20150305
BOARD_HAVE_GNS57560 := false
#end add

```

全能版配置：

```

# For Olca3.2 driver
#BOARD_WPA_SUPPLICANT_DRIVER := NL80211
#BOARD_HOSTAPD_DRIVER := NL80211
#WPA_SUPPLICANT_VERSION := VER_0_8_ATHEROS
#HOSTAPD_VERSION := VER_0_8_ATHEROS
#WIFI_DRIVER_MODULE_PATH := "/system/lib/modules/ath6kl_sdio.ko"
#WIFI_DRIVER_MODULE_NAME := "ath6kl_sdio"

#WIFI_SDIO_IF_DRIVER_MODULE_PATH := "/system/lib/modules/cfg80211.ko"
#WIFI_SDIO_IF_DRIVER_MODULE_NAME := "cfg80211"
#WIFI_SDIO_IF_DRIVER_MODULE_ARG := ""

BOARD_HAVE_MTK_MT6620 := true
BOARD_GPS_LIBRARIES := true
BOARD_WPA_SUPPLICANT_DRIVER := WEXT
WIFI_DRIVER_MODULE_PATH := /system/lib/modules/wlan.ko
WIFI_DRIVER_MODULE_NAME := wlan0
WPA_BUILD_SUPPLICANT := true
#CONFIG_CTRL_INTERFACE := y
WPA_SUPPLICANT_VERSION := VER_0_6_X
P2P_SUPPLICANT_VERSION := VER_0_8_X
BOARD_P2P_SUPPLICANT_DRIVER := NL80211

#add by cym 20130623
BOARD_HAVE_MPU6050 := true
#end add
#add by cym 20150305
BOARD_HAVE_GNS57560 := false
#end add

```

### 6.3.2 android4.4 代码下载和编译

Android 源码可以从光盘，网盘获取稳定版本，也可以从 GitHub 下载我们的开发版本。

GitHub 仅提供源码下载，不提供二进制下载，二进制文件存放在光盘和网盘中。

#### 6.3.2.1 repo 下载

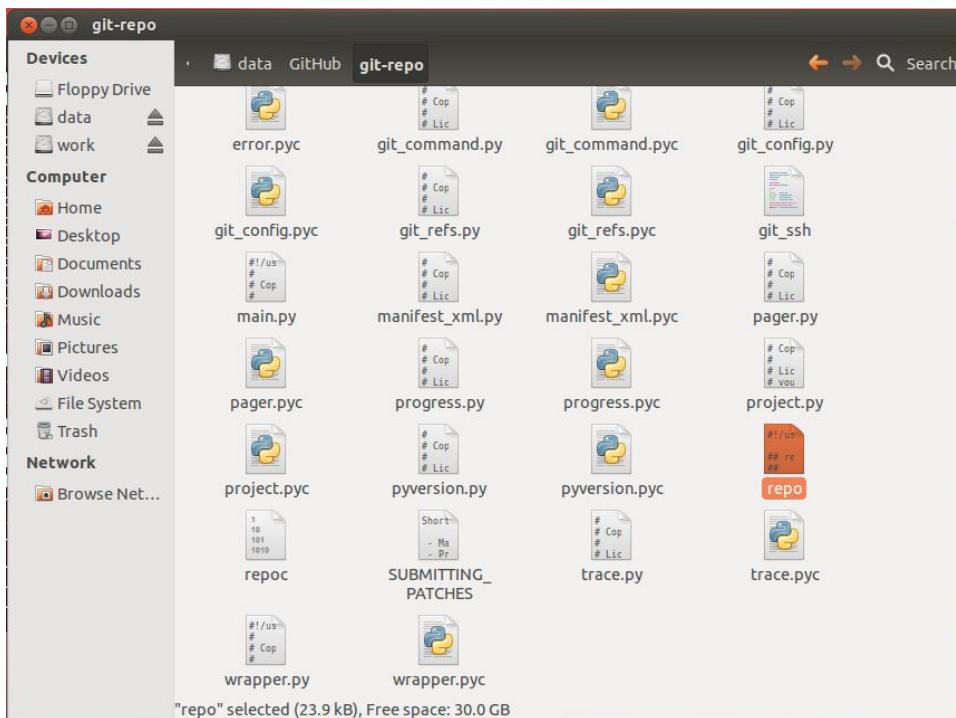
android 代码管理不同于 uboot,kernel , 由于 Android 代码比较庞大 , 我们把 Android 项目按照文件夹进行项目拆分 , android 源码目录下面的每个子目录都会划分为一个仓库或者多个仓库进行版本管理 , Android 的代码下载需要使用 repo 工具 , repo 工具为 Git 的封装 , 底层是使用 Git 命令进行下载的。

假设 repo 工具存放于 /media/data/gitHub 目录:

```
cd /media/data/GitHub/
```

```
git clone git://aosp.tuna.tsinghua.edu.cn/android/git-repo.git/
```

git-repo 仓库下载完成后会看到该文件夹内有 repo 脚本文件:



### 6.3.2.2 Android4.4 代码下载

假设我们的 Android4.4 代码存放在 /media/data/GitHub/iTop4412\_KK4.4\_git 目录：

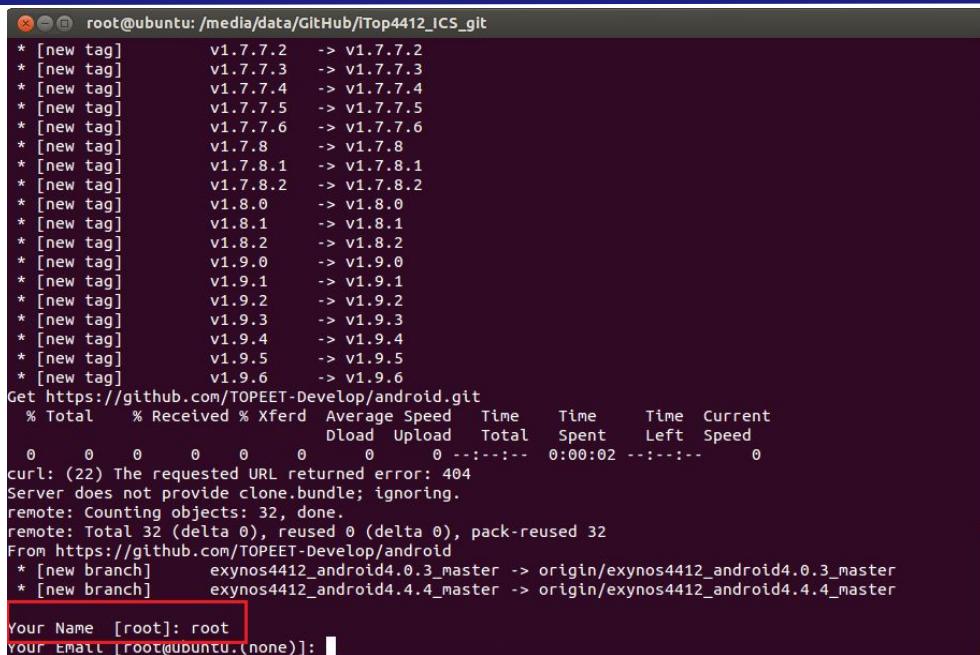
```
# cd /media/data/GitHub/iTop4412_KK4.4_giti
```

```
#./git-repo/repo init -u https://github.com/TOPEET-Develop/android.git -b  
exynos4412_android4.4.4_master
```

上一条命令字符 “-b” 后有空格

注：repo init 命令中-u 参数指定 android 仓库下载地址，-b 参数指定仓库中的下载分支。

这里我下载的是 exynos4412\_android4.4.4\_master 分支，该命令执行过程中需要输入相关的信息，如下图所示：

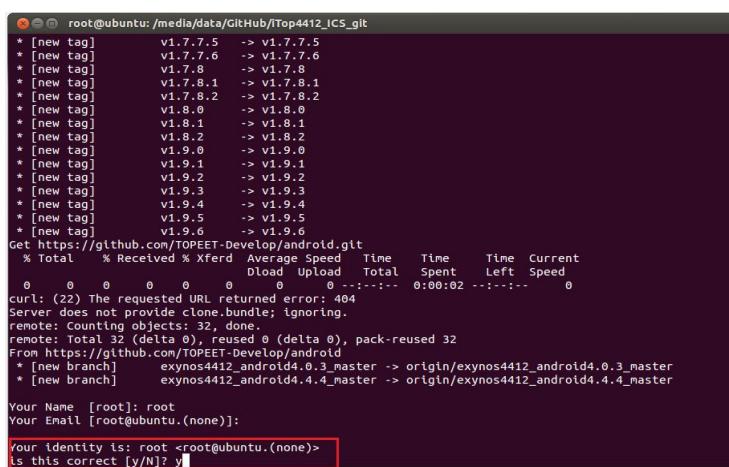


```
* [new tag]      v1.7.7.2    -> v1.7.7.2
* [new tag]      v1.7.7.3    -> v1.7.7.3
* [new tag]      v1.7.7.4    -> v1.7.7.4
* [new tag]      v1.7.7.5    -> v1.7.7.5
* [new tag]      v1.7.7.6    -> v1.7.7.6
* [new tag]      v1.7.8       -> v1.7.8
* [new tag]      v1.7.8.1    -> v1.7.8.1
* [new tag]      v1.7.8.2    -> v1.7.8.2
* [new tag]      v1.8.0       -> v1.8.0
* [new tag]      v1.8.1       -> v1.8.1
* [new tag]      v1.8.2       -> v1.8.2
* [new tag]      v1.9.0       -> v1.9.0
* [new tag]      v1.9.1       -> v1.9.1
* [new tag]      v1.9.2       -> v1.9.2
* [new tag]      v1.9.3       -> v1.9.3
* [new tag]      v1.9.4       -> v1.9.4
* [new tag]      v1.9.5       -> v1.9.5
* [new tag]      v1.9.6       -> v1.9.6
Get https://github.com/TOPEET-Develop/android.git
  % Total    % Received % Xferd  Average Speed   Time     Time   Current
     0       0       0      0      0      0      0      0 --:--:--  0:00:02 --:--:--  0
curl: (22) The requested URL returned error: 404
Server does not provide clone.bundle; ignoring.
remote: Counting objects: 32, done.
remote: Total 32 (delta 0), reused 0 (delta 0), pack-reused 32
From https://github.com/TOPEET-Develop/android
 * [new branch]  exynos4412_android4.0.3_master -> origin/exynos4412_android4.0.3_master
 * [new branch]  exynos4412_android4.4.4_master -> origin/exynos4412_android4.4.4_master

Your Name [root]: root
Your Email [root@ubuntu.(none)]:
```

Your Name 输入 root

Your Email: 直接回车即可，然后在输入 y, 回车继续：



```
* [new tag]      v1.7.7.5    -> v1.7.7.5
* [new tag]      v1.7.7.6    -> v1.7.7.6
* [new tag]      v1.7.8       -> v1.7.8
* [new tag]      v1.7.8.1    -> v1.7.8.1
* [new tag]      v1.7.8.2    -> v1.7.8.2
* [new tag]      v1.8.0       -> v1.8.0
* [new tag]      v1.8.1       -> v1.8.1
* [new tag]      v1.8.2       -> v1.8.2
* [new tag]      v1.9.0       -> v1.9.0
* [new tag]      v1.9.1       -> v1.9.1
* [new tag]      v1.9.2       -> v1.9.2
* [new tag]      v1.9.3       -> v1.9.3
* [new tag]      v1.9.4       -> v1.9.4
* [new tag]      v1.9.5       -> v1.9.5
* [new tag]      v1.9.6       -> v1.9.6
Get https://github.com/TOPEET-Develop/android.git
  % Total    % Received % Xferd  Average Speed   Time     Time   Current
     0       0       0      0      0      0      0      0 --:--:--  0:00:02 --:--:--  0
curl: (22) The requested URL returned error: 404
Server does not provide clone.bundle; ignoring.
remote: Counting objects: 32, done.
remote: Total 32 (delta 0), reused 0 (delta 0), pack-reused 32
From https://github.com/TOPEET-Develop/android
 * [new branch]  exynos4412_android4.0.3_master -> origin/exynos4412_android4.0.3_master
 * [new branch]  exynos4412_android4.4.4_master -> origin/exynos4412_android4.4.4_master

Your Name [root]: root
Your Email [root@ubuntu.(none)]:

Your identity is: root <root@ubuntu.(none)>
Is this correct [y/N]? y
```

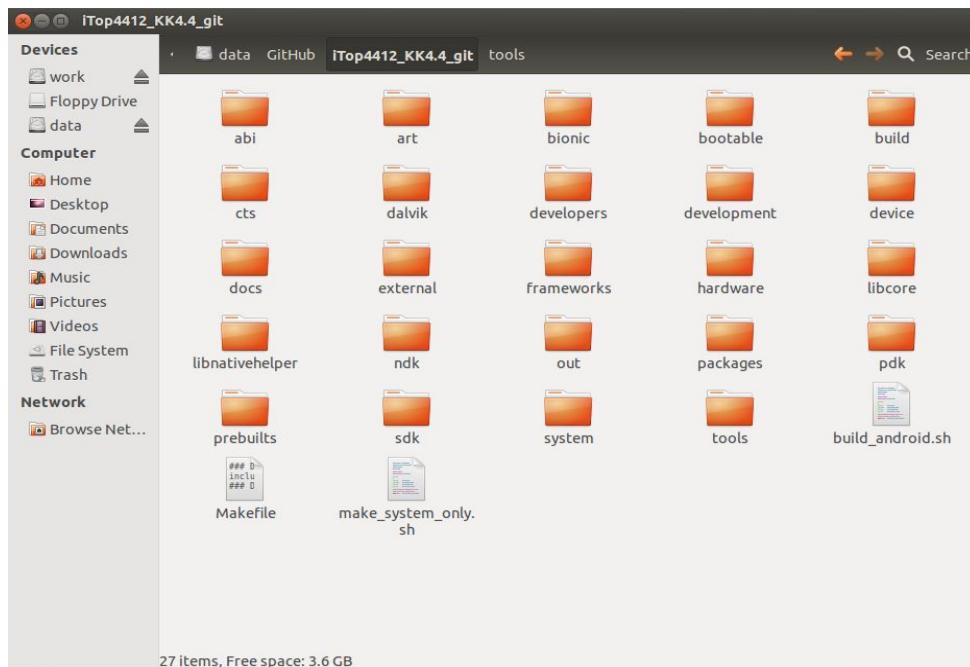
#./git-repo/repo sync

注： repo sync 开始下载 Android 代码，下载过程与网络环境有一定的关系，如果下载过程中长时间没有进度显示，可以 **ctrl+c** 终止下载，然后执行 **repo sync** 命令重新开始，下图为源码下载过程中的进度显示：

```
* [new tag]      android-cts-4.0.3_r1 -> android-cts-4.0.3_r1
* [new tag]      android-cts-4.0.3_r2 -> android-cts-4.0.3_r2
* [new tag]      android-cts-4.0_r1 -> android-cts-4.0_r1
* [new tag]      android-cts-4.1_r1 -> android-cts-4.1_r1
* [new tag]      android-cts-4.1_r2 -> android-cts-4.1_r2
* [new tag]      android-cts-4.1_r4 -> android-cts-4.1_r4
* [new tag]      android-cts-4.2_r1 -> android-cts-4.2_r1
* [new tag]      android-cts-4.2_r2 -> android-cts-4.2_r2
* [new tag]      android-cts-4.4_r1 -> android-cts-4.4_r1
* [new tag]      android-cts-5.0_r3 -> android-cts-5.0_r3
* [new tag]      android-cts-5.1_r1 -> android-cts-5.1_r1
* [new tag]      android-cts-5.1_r2 -> android-cts-5.1_r2
* [new tag]      android-cts-verifier-4.0.3_r1 -> android-cts-verifier-4.0.3_r1
* [new tag]      android-cts-verifier-4.0_r1 -> android-cts-verifier-4.0_r1
* [new tag]      android-l-preview_r2 -> android-l-preview_r2
* [new tag]      android-m-preview -> android-m-preview
* [new tag]      android-m-preview-1 -> android-m-preview-1
* [new tag]      android-sdk-4.0.3-tools_r1 -> android-sdk-4.0.3-tools_r1
* [new tag]      android-sdk-4.0.3_r1 -> android-sdk-4.0.3_r1
* [new tag]      android-sdk-4.4.2_r1 -> android-sdk-4.4.2_r1
* [new tag]      android-sdk-4.4.2_r1.0.1 -> android-sdk-4.4.2_r1.0.1
* [new tag]      android-sdk-adt_r16.0.1 -> android-sdk-adt_r16.0.1
* [new tag]      android-sdk-adt_r20 -> android-sdk-adt_r20
* [new tag]      android-sdk-support_r11 -> android-sdk-support_r11
* [new tag]      android-wear-5.0.0_r1 -> android-wear-5.0.0_r1
* [new tag]      android-wear-5.1.0_r1 -> android-wear-5.1.0_r1
* [new tag]      android-wear-5.1.1_r1 -> android-wear-5.1.1_r1
^CReceiving objects: 20% (50/247), 38.22 MiB | 12 KiB/s    31/247), 22.39 MiB | 23 KiB/s
```

代码下载完成后会在 /media/data/GitHub/ iTop4412\_KK4.4\_git 目录下面显示

Android 系统源码目录。Android4.4 的源码大概 20G , 第一次下载过程比较漫长 , 请耐心等待。代码下载完成后 , 再次使用 repo sync 命令时 , 只更新升级代码 , 速度就比较快了。



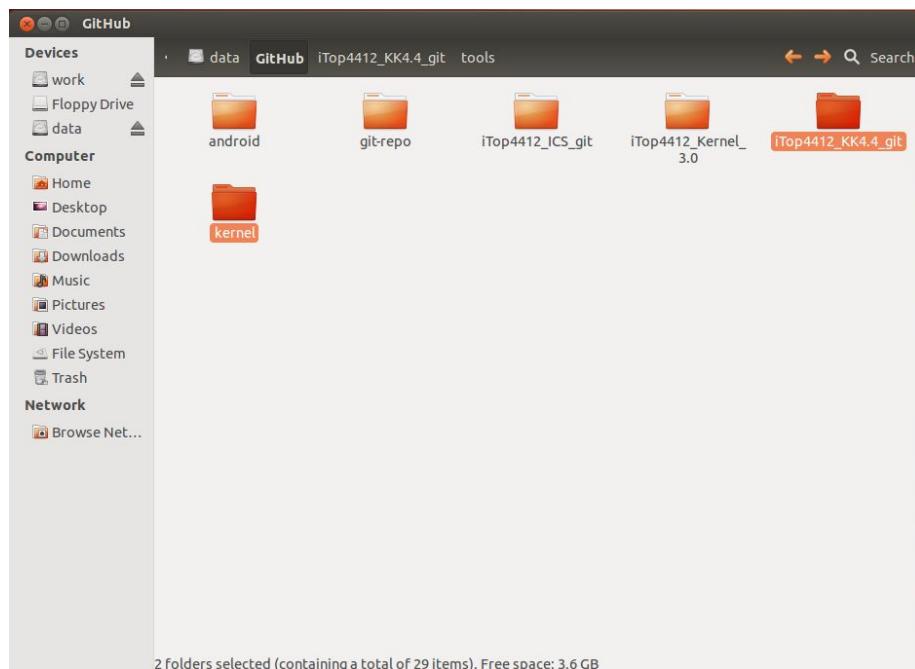
每个文件夹目录下面都会有一个或者多个 Git 仓库 , 默认这些仓库是没有分支的 , 我们需要使用 repo start 命令创建分支 , 这里我们创建 master 分支 , 您也可以使用别的名字来定义分支名称 , repo branch 命令可以查看创建的分支 :

```
#./git-repo/repo start master --all
```

```
#../git-repo/repo branch
```

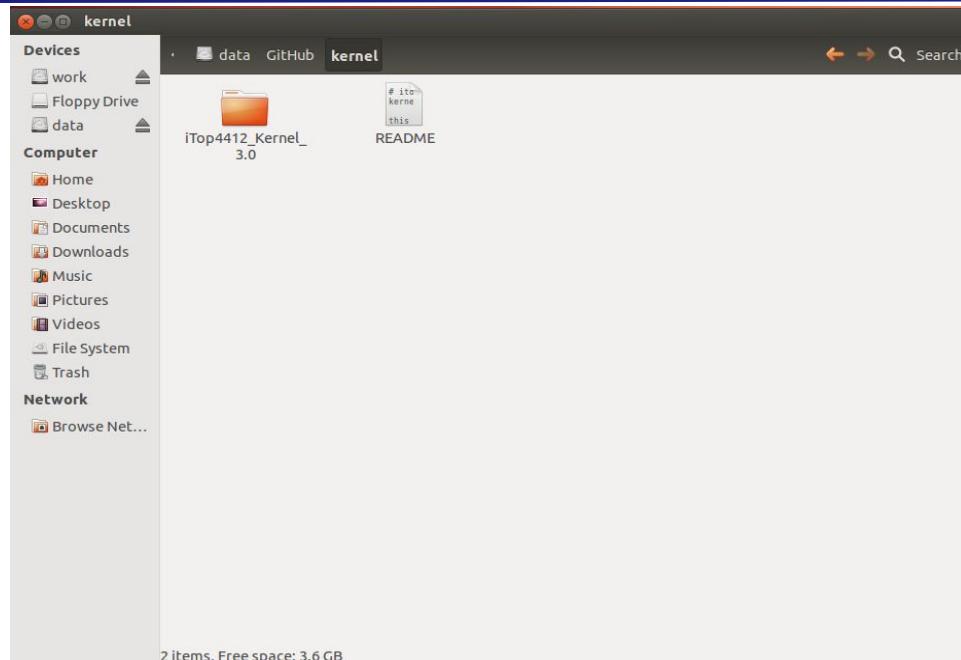
### 6.3.2.3 Android4.4 源码编译

Android 代码下载完成后就可以进行编译了，Android 源码需要 kernel 头文件支持，所以我们需要把 Android 代码与 kernel 代码放到同一级目录下面：



上图中 kernel 为 Android4.4 的内核代码，与 Android4.4 源码目录 iTOP4412\_KK4.4\_git 在同一级。

kernel 目录为我们下载的 Android4.4 的内核代码：



执行 Android4.4 的编译命令：

```
#cd iT0p4412_KK4.4_git  
# ./build_android.sh
```

注意: Android 代码下载，编译完成后，可以定期使用 `repo sync` 命令同步我们的 Git 仓库到本地，这样您的本地代码就跟我们的仓库保持同步更新了，编译后就会形成最新的系统镜像。另外 Android4.4 代码目前不分区精英底板与全能底板，代码编译形成的镜像既可以在精英版运行也可以在全能版运行，与 Android4.0 有所不同。

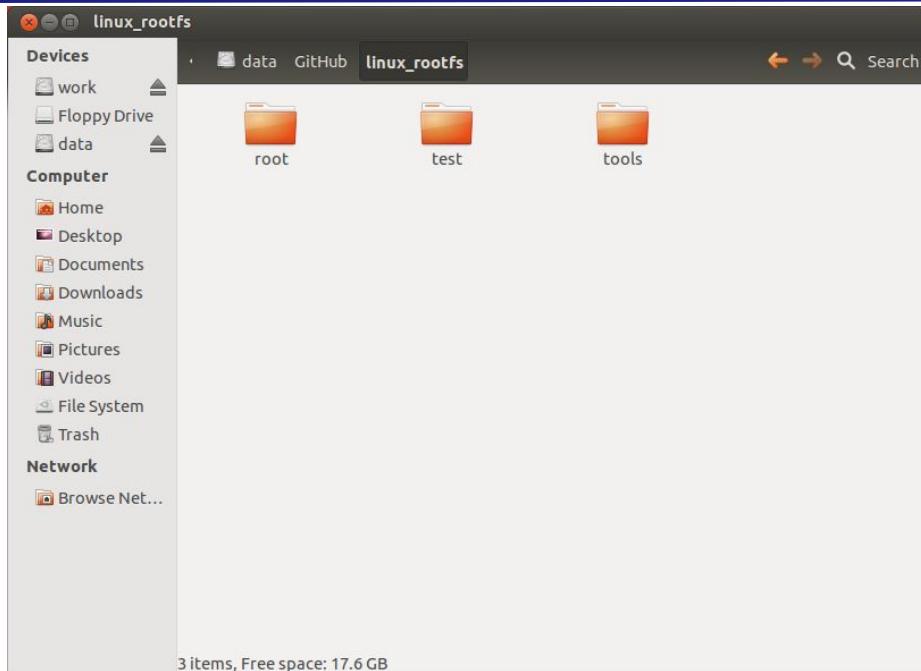
### 6.3.3 Linux Qt 文件系统下载及制作

Git 命令下载：

```
# git clone https://github.com/TOPEET-Develop/linux_rootfs.git -b  
rootfs_qt_master
```

上一条命令字符 “-b” 后有空格

下载完成后可以看到如下的目录结构：



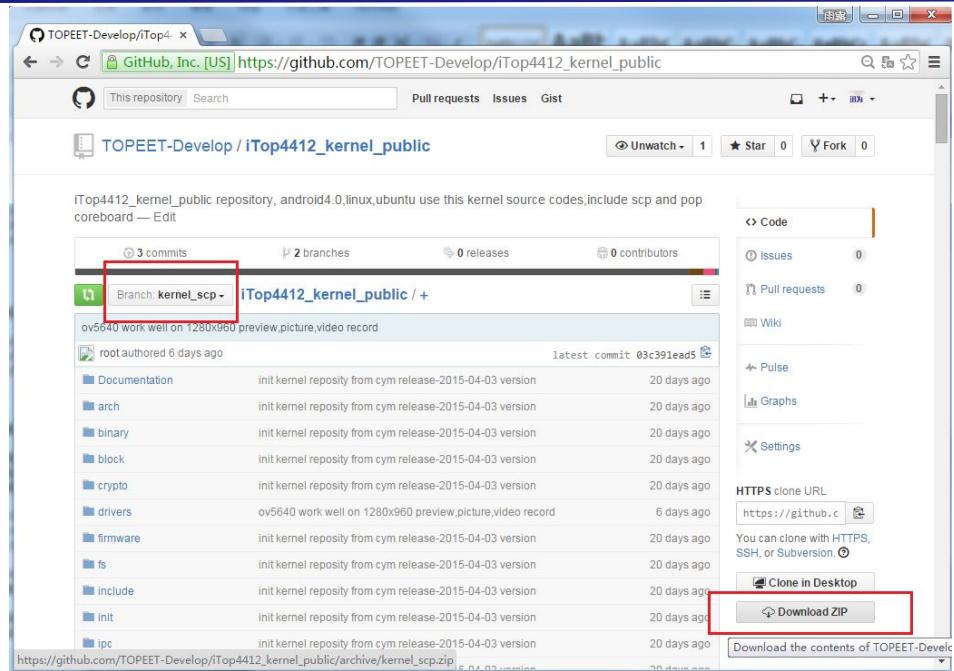
root: Linux 的根文件系统；

test: Linux 的测试程序；

tools: 用于制作文件系统的工具及 ramdisk.img 二进制文件，该文件专用于 Linux 系统。

Tools 目录下面的 mkfs 脚本用于制作文件系统镜像，把该脚本拷贝到 Linux\_rootfs 目录下面，命令行执行即可生成 system.img.

注：可以使用 git clone 命令下载 uboot, kernel , Linux 文件系统，也可以通过网页下载源码压缩包。进入到 GitHub 主页，选择要下载的仓库：



单击左上角红框部分，选择好该仓库的分支，然后单击页面右下角的“Download ZIP”即可通过浏览器下载压缩包。

### 6.3.4 Ubuntu 文件系统

Ubuntu 文件系统分为：

LCD 显示版本：iTOP4412\_ubuntu\_12.04\_for\_LCD\_20141230.tar.gz

HDMI 显示版本：iTOP4412\_ubuntu\_12.04\_for\_HDMI\_20141230.tar.gz

这两个版本未放到 GitHub 平台，目前通过光盘和网盘发布给客户使用。

## 联系方式

北京迅为电子有限公司致力于嵌入式软硬件设计，是高端开发平台以及移动设备方案提供商；基于多年的技术积累，在工控、仪表、教育、医疗、车载等领域通过 OEM/ODM 方式为客户创造价值。

iTOP-4412 开发板是迅为电子基于三星最新四核处理器 Exynos 4412 研制的一款实验开发平台，可以通过该产品评估 Exynos 4412 处理器相关性能，并以此为基础开发出用户需要的特定产品。

本手册主要介绍 iTOP-4412 开发板的使用方法，旨在帮助用户快速掌握该产品的应用特点，通过对开发板进行后续软硬件开发，衍生出符合特定需求的应用系统。

如需‘平板电脑’产品级方案支持，请访问迅为平板方案网‘[www.topeet.com](http://www.topeet.com)’。我司将有能力为您提供全方位的技术服务，保证您产品设计无忧！

本手册将持续更新，并通过多种方式发布给新老用户，希望迅为电子的努力能给您的学习和开发带来帮助。

迅为电子

2015 年 11 月