



Hi3510 Linux 开发环境

## 用户指南

文档版本	01
发布日期	2006-08-31
BOM编码	N/A

深圳市海思半导体有限公司为客户提供全方位的技术支持，用户可与就近的海思办事处联系，也可直接与公司总部联系。

## 深圳市海思半导体有限公司

地址：深圳市龙岗区坂田华为基地华为电气生产中心 邮编：518129

网址：<http://www.hisilicon.com>

客户服务电话：0755-28788858

客户服务传真：0755-28788838

客户服务邮箱：[support@hisilicon.com](mailto:support@hisilicon.com)

**版权所有 © 深圳市海思半导体有限公司 2006。 保留一切权利。**

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

### 商标声明



、Hisilicon、海思，均为深圳市海思半导体有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

### 注意

由于产品版本升级或其它原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。



## 目 录

前 言.....	1
1 开发环境.....	1-1
关于本章.....	1-1
1.1 嵌入式开发环境.....	1-2
1.2 Hi3510 Linux 开发环境.....	1-2
1.3 搭建 Linux 开发环境.....	1-3
1.3.1 安装 Linux 服务器.....	1-3
1.3.2 安装交叉编译工具.....	1-4
1.3.3 安装 Hi3510 SDK.....	1-4
2 HiBoot .....	2-1
关于本章.....	2-1
2.1 HiBoot 简介.....	2-2
2.2 启动 HiBoot.....	2-2
2.3 编译 HiBoot.....	2-2
2.4 烧写 HiBoot.....	2-3
2.5 HiBoot 常用命令.....	2-5
2.6 HiBoot 环境变量.....	2-10
3 Linux 内核.....	3-1
关于本章.....	3-1
3.1 内核源代码.....	3-2
3.2 配置内核.....	3-2
3.3 编译内核.....	3-2
3.4 使用 mkimage 工具.....	3-3
4 根文件系统.....	4-1
关于本章.....	4-1
4.1 根文件系统简介.....	4-2
4.2 利用 busybox 制作根文件系统.....	4-3
4.2.1 获取 busybox 源代码.....	4-3
4.2.2 配置 busybox.....	4-3



4.2.3 编译和安装 busybox .....	4-3
4.2.4 制作根文件系统 .....	4-3
4.3 文件系统简介 .....	4-4
4.3.1 Cramfs .....	4-4
4.3.2 JFFS2 .....	4-5
4.3.3 NFS .....	4-6
<b>5 烧写内核和根文件系统 .....</b>	<b>5-1</b>
关于本章 .....	5-1
5.1 存储器地址空间 .....	5-2
5.2 通过网口烧写 .....	5-2
5.2.1 参数设置和建立 tftp 服务 .....	5-2
5.2.2 下载内核 .....	5-3
5.2.3 下载根文件系统 .....	5-3
5.3 通过串口烧写 .....	5-3
5.3.1 连接设备 .....	5-4
5.3.2 下载内核 .....	5-4
5.3.3 下载根文件系统 .....	5-5
<b>6 启动 Linux .....</b>	<b>6-1</b>
关于本章 .....	6-1
6.1 设置启动参数 .....	6-1
6.2 启动 Linux .....	6-2
6.3 设置 HiBoot 自动启动 Linux .....	6-2
<b>7 应用程序开发简介 .....</b>	<b>7-1</b>
关于本章 .....	7-1
7.1 编写代码 .....	7-1
7.2 运行应用程序 .....	7-1
7.3 使用 gdb server 调试应用程序 .....	7-2
<b>A 建立 Linux 开发环境 .....</b>	<b>A-1</b>
A.1 安装 Linux 系统的配置选项 .....	A-1
A.2 配置必要的系统服务 .....	A-1
<b>B 缩略语 .....</b>	<b>B-1</b>



## 插图目录

图 1-1 嵌入式开发图例.....	1-2
图 1-2 Hi3510 开发环境.....	1-2
图 1-3 安装 Hi3510 SDK 成功后 VSSDKV100R001BXX 的文件目录 .....	1-5
图 2-1 Multi ICE Server.....	2-3
图 2-2 project properties 设置.....	2-4
图 2-3 Connection properties 设置 .....	2-5
图 4-1 根文件系统顶层目录结构图.....	4-2
图 5-1 FLASH（32MB）地址空间分配示意图（仅供参考） .....	5-2
图 5-2 串口设置.....	5-4
图 5-3 发送文件窗口.....	5-5



## 表格目录

表 1-1 Hi3510 开发环境的各部分软件描述 .....	1-3
表 2-1 HiBoot 常用命令说明 .....	2-6
表 2-2 HiBoot 常用环境变量说明 .....	2-10
表 3-1 mkimage 参数表 .....	3-3
表 4-1 嵌入式系统中可忽略的目录说明 .....	4-2
表 4-2 jffs2 参数表 .....	4-5
表 A-1 安装 Linux 系统的配置选项说明 .....	A-1



# 前 言

## 概述

本节介绍本文档的内容、对应的产品版本、适用的读者对象、行文表达约定、历史修订记录等。

## 产品版本

与本文档相对应的产品版本如下所示。

产品名称	产品版本
Hi3510 通信媒体处理器 芯片（简称 Hi3510）	Hi3510 V100
	Hi3510 V101
	Hi3510 V110
Hi3510 DVS 方案	Hi3510 DMS V100R001
Hi3510 VPhone 方案	Hi3510 DMS V200R001

## 读者对象

本指南主要适用于以下工程师：

- 基于 Hi3510 视频评估板的软件开发工程师
- Hi3510 技术支持工程师

## 内容简介

本文档介绍了 Hi3510 视频评估板（VSEVB）的 Linux 开发环境和各种系统软件的使用。全书共分为 7 章和 2 个附录。






章节	内容
1 开发环境	本章介绍搭建 Hi3510 视频评估板需要的 Linux 开发环境。首先介绍了嵌入式开发环境的组成，然后详细介绍了 Linux 主机中交叉编译工具和 Hi3510 视频评估板 SDK 的安装。通过阅读本章，可以指导读者完成开发环境的搭建。
2 HiBoot	本章详细介绍海思半导体有限公司开发的 BootLoader 即 HiBoot。本章包括如何使用 HiBoot 以及如何编译 HiBoot，并使用调试器将 HiBoot 烧写入 Hi3510VSEVB 板中。
3 Linux 内核	介绍 HiLinux 内核的配置和编译。通过阅读本章，按照操作步骤，读者能够重新编译出可运行于 Hi3510VSEVB 板的内核。
4 根文件系统	本章详细介绍 HiLinux 根文件系统的制作。内容包括嵌入式根文件系统概述、利用 busybox 制作根文件系统以及文件系统（Cramfs、Jffs2 和 NFS）的使用。
5 烧写内核和根文件系统	本章介绍通过 HiBoot 使用网口或者串口将内核和根文件系统烧写入 Flash。
6 启动 Linux	本章介绍通过 HiBoot 启动 Linux 内核。内容包括了引导参数的配置和自动启动的设置方法。
7 应用程序开发简介	简要介绍在 Linux 下的应用程序的开发，如何添加到 Hi3510 视频评估板上运行以及使用 gdb Server 进行调试。
A 建立 Linux 开发环境	简要介绍了如何安装 Linux 开发环境。
B 缩略语	介绍本书中出现的缩略语。

## 约定

### 符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

符号	说明
 危险	以本标志开始的文本表示有高度潜在危险，如果不能避免，会导致人员死亡或严重伤害。
 警告	以本标志开始的文本表示有中度或低度潜在危险，如果不能避免，可能导致人员轻微或中等伤害。
 注意	以本标志开始的文本表示有潜在风险，如果忽视这些文本，可能导致设备损坏、数据丢失、设备性能降低或不可预知的结果。





符号	说明
🔑 窍门	以本标志开始的文本能帮助您解决某个问题或节省您的时间。
📖 说明	以本标志开始的文本是正文的附加信息，是对正文的强调和补充。

## 通用格式约定

格式	说明
宋体	正文采用宋体表示。
黑体	一级、二级、三级标题采用黑体。
楷体	警告、提示等内容一律用楷体，并且在内容前后增加线条与正文隔离。
“Terminal Display” 格式	“Terminal Display” 格式表示屏幕输出信息。此外，屏幕输出信息中夹杂的用户从终端输入的信息采用加粗字体表示。

## 修改记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

修改日期	版本	修改说明
2006-11-01	01（第一次版本发布）	



# 1 开发环境

## 关于本章

本章描述了嵌入式开发环境、Hi3510 Linux 开发环境和如何搭建 Linux 开发环境，内容如下表所示。

标题	内容
<a href="#">1.1 嵌入式开发环境</a>	介绍了通用的嵌入式开发环境。
<a href="#">1.2 Hi3510 Linux 开发环境</a>	介绍了 Hi3510 在 Linux 下的开发环境。
<a href="#">1.3 搭建 Linux 开发环境</a>	介绍了如何安装 Linux 服务器、安装交叉编译工具和安装 Hi3510SDK。



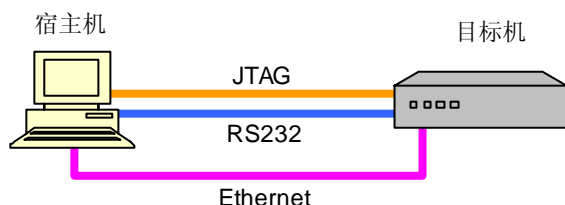
## 1.1 嵌入式开发环境

由于嵌入式单板的资源有限，不可能在单板上运行开发、调试工具。通常需要交叉编译调试的方式进行，即“宿主机+目标机（评估板）”的形式。目标机和宿主机一般采用串口连接，亦可同时通过网口或者 JTAG 连接，如图 1-1 所示。

宿主机和目标机的处理器通常情况下不相同。宿主机需要建立适合于目标机的交叉编译环境。程序在宿主机上“编译—连接—定位”，得到可执行文件，通过一定的方法烧写到目标机中，然后在目标机上运行。

目标机上的 Bootloader 启动后，输入/输出重新定位到串口或者网口，在主机上的控制台中输入命令，可以控制目标机。

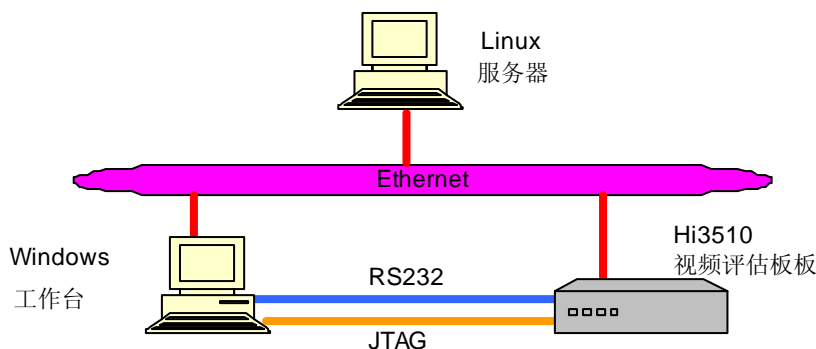
图1-1 嵌入式开发图例



## 1.2 Hi3510 Linux 开发环境

Hi3510 Linux 开发环境通常包括 Linux 服务器、Windows 工作台及 Hi3510 视频评估板，三者同处于一个网络中，如图 1-2 所示。

图1-2 Hi3510 开发环境



在 Linux 服务器上建立交叉编译环境，Windows 工作台通过串口和 JTAG 与 Hi3510 单板连接（JTAG 用于 ADS/RealView Debugger 等软件），开发人员可以在 Windows 工作台进行程序开发或者远程登陆到 Linux 服务器进行开发。各部分具体软件介绍如表 1-1 所示。



说明：

开发环境中使用了 Windows 工作台，实际上很多工作也可以在 Linux 服务器上完成，如使用 minicom 代替超级终端等，用户可自行选择。

表1-1 Hi3510 开发环境的各部分软件描述

软件		描述
Windows 工作台	操作系统	Windows 98/me/2000/XP。
	应用软件	putty、超级终端、tftp 服务器、ADS/RealView Debugger 等软件。
Linux 服务器	操作系统	无特别要求，可为 Redhat、Debian 等。内核版本支持 2.6.x 或者 2.4.x。安装时建议选择完全安装。
	应用软件	NFS、telnetd、samba、VIM、arm 交叉编译环境（Binutils 版本 2.16.91，Gcc 版本 3.4.3）等。其他应用软件根据具体开发需要而定，通常系统都已默认安装，只要适当配置即可。
Hi3510 视频评估板	引导程序	HiBoot。基于 U-Boot 1.1.4 开发而成。
	操作系统	Hisilicon Linux（简称 HiLinux）。HiLinux 内核基于 Linux 标准内核 2.6.14 版本移植开发，根文件系统基于 busybox 1.00 版本制作而成。
	应用软件	包含 telnetd、gdb server 等 Linux 常用命令。
	程序开发库	glibc 2.3.4 版本。

## 1.3 搭建 Linux 开发环境

本节介绍了如何安装 Linux 服务器、安装交叉编译工具和安装 Hi3510SDK。

### 1.3.1 安装 Linux 服务器

建议选择常用的 Linux 发行版，便于寻找各类技术资源。

RedHat 较新的发行版都可以使用，如 RedHat Fedora Core 系列和 Redhat Enterprise Linux，较老的如 RedHat 9.0 等也可以使用。建议使用较新版本，以方便获取各类资源，推荐使用 Fedora Core 系列，以便获得技术支持。

Debian 的各类发行版也是常用的，使用 Debian 的好处是各类安装包都可以随时在线更新，各类软件包资源也很丰富。



## 1.3.2 安装交叉编译工具

可以使用从其它渠道得到的 ARM 交叉编译工具（如：网络下载），这需要用户熟悉交叉编译环境的安装及使用过程。建议使用与 Hi3510 SDK 配套的交叉编译环境，具体请参见《Hi3510 媒体处理软件开发指南 附录》。



### 注意

使用从网络等渠道得到的交叉编译工具可能与我们使用的内核并不配套，由此可能造成开发过程中一些不可预料的问题。

## 1.3.3 安装 Hi3510 SDK

Hi3510 SDK 是基于 Hi3510 视频评估板的软件开发工具，包含了在 Linux 相关应用开发时使用的各种工具及其源代码，是用户开发中最基本的平台软件。将 Hi3510 SDK 安装到 Linux 服务器中的安装步骤如下所示。

### 1. 拷贝

将 VSSDKV100R001BXX.tar.gz（XX 是版本号）拷贝到 Linux 服务器上。

### 2. 解压

解压文件，使用命令：`tar -zxf VSSDKV100R001BXX.tar.gz`，过程中没有提示信息，请等待命令执行完毕。

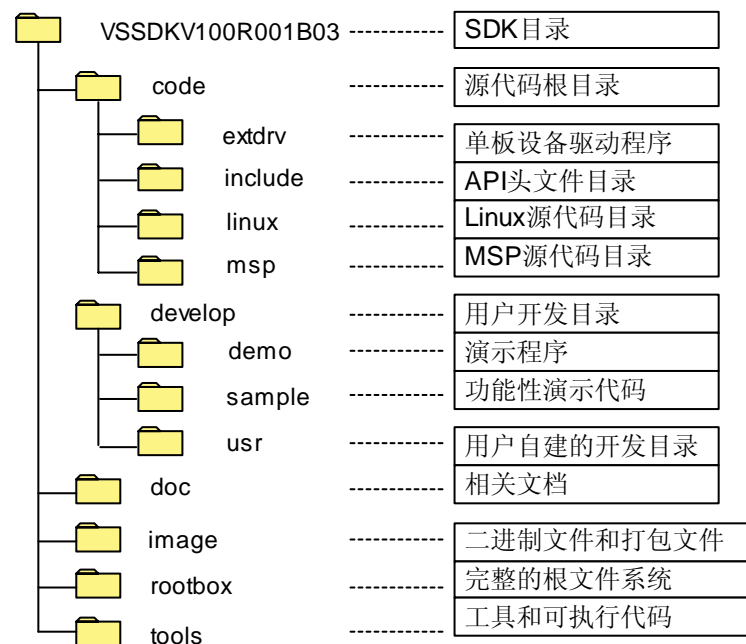
### 3. 安装

解压完成后进入 VSSDKV100R001BXX 目录，执行 `./vssdk.install`，执行完毕，即安装成功。如果用户不是 root 权限，安装过程中必要的时候会提示输入 root 密码或 sudo 密码。

安装成功后，在目录 VSSDKV100R001BXX 下包含目录如图 1-3 所示。



图1-3 安装 Hi3510 SDK 成功后 VSSDKV100R001BXX 的文件目录





# 2 HiBoot

## 关于本章

本章描述了 HiBoot 的启动、编译、烧写的具体操作和 HiBoot 的常用命令、环境变量，内容如下表所示。

标题	内容
<a href="#">2.1 HiBoot 简介</a>	简单介绍了 HiBoot。
<a href="#">2.2 启动 HiBoot</a>	介绍了启动 HiBoot 的信息。
<a href="#">2.3 编译 HiBoot</a>	介绍了编译 HiBoot 的操作。
<a href="#">2.4 烧写 HiBoot</a>	介绍了烧写 HiBoot 的操作。
<a href="#">2.5 HiBoot 常用命令</a>	介绍了 HiBoot 的常用命令定义。
<a href="#">2.6 HiBoot 环境变量</a>	介绍了 HiBoot 的环境变量及设置。



## 2.1 HiBoot 简介

HiBoot 是在 U-Boot 1.1.4（或以上版本）基础上进行开发的。

简单地说，Bootloader 就是在操作系统内核运行之前运行的一段小程序。通过这段小程序，可以初始化硬件设备、建立内存空间的映射图，使系统的软硬件环境处于一个合适的状态，为最终调用操作系统内核准备好正确的环境。

HiBoot 除了作为一个 Bootloader 外，还是一个烧写器。在 HiBoot 下，可以通过串口、网口下载 Linux 内核或者应用程序到 RAM 或 Flash 中。

## 2.2 启动 HiBoot

给 Hi3510 视频评估板上电后，在控制台上出现命令提示符。Hi3510 视频评估板的标准输入、标准输出重定位到 UART0，UART0 通过 57600 波特率连接到调试主机上，调试主机采用 Windows 超级终端做控制台（如果调试主机是 Linux，采用 MiniCOM）。

系统上电后，控制台上如有如下类似的信息显示，表示 HiBoot 已经启动：

```
HiBoot 1.01 (Apr 20 2005 - 14:20:00)

HiBoot code: F8000000 -> F801D724 BSS: -> F80616F4
RAM Configuration:
Bank #0: 00000000 0 kB
Flash: 32 MB
In: serial
Out: serial
Err: serial
hisilicon#
```

## 2.3 编译 HiBoot

HiBoot 可以通过修改配置文件 include/configs/hidvs3510.h 实现功能的裁减。如想了解相关参数的具体含义及功能请认真阅读 U-Boot 1.1.4/README 文件。

编译 HiBoot 操作如下：

```
hisilicon$cd ~/hilinux/hiboot/
hisilicon$make mrproper
hisilicon$make hidvs3510_config
hisilicon$make all
```

如果编译过程中编译 example 目录时出现错误，进入 example 目录，执行一下 touch \* 命令：

```
hisilicon$cd example
hisilicon$touch *
```

编译生成的目标文件：





- elf 文件: hiboot
- 二进制文件: hiboot.bin

## 2.4 烧写 HiBoot

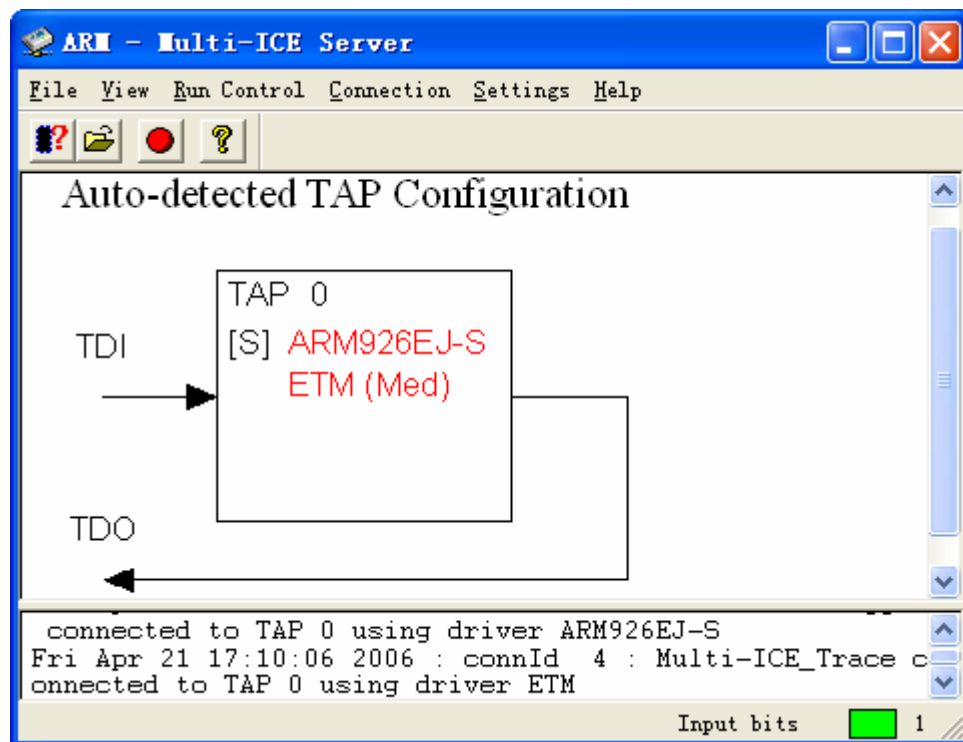
把 HiBoot 写入 Flash 需要 ARM 公司的调试器 Multi ICE 或者 RealView ICE 配合 RealView Debugger 完成。其中 RealView Debugger 也可以用 AXD Debugger 替代, 但 AXD Debugger 只支持 Multi ICE, RealView Debugger 则支持 Multi ICE 和 RealView ICE 两者。

烧写 HiBoot 之前, 首先需要执行 SDRAM 初始化脚本 (tools\flash\_easy\_Hi3510DVS\下的扩展名为 inc 的文件, 该文件只适用于 Hi3510 视频评估板, 并且 AXD Debugger 不能使用该脚本); 初始化 SDRAM 后, 再将 HiBoot 烧写入 Flash。关于 ARM 公司相关工具的安装和使用请参考 ARM 公司提供的文档, 以下是已经安装成功 ARM 公司相关工具后的操作。

下面介绍 Multi ICE 配合 RealView Debugger 工具完成烧写工作。操作步骤如下所示。

- 步骤 1 使用调试器将 Windows 工作台和 Hi3510 视频评估板连接。此时需确认 Windows 主机与调试器相连的并口已经开启。
- 步骤 2 打开 Multi-ICE Server 应用程序, 单击 “#?” 按钮 (或选择菜单项 “file>auto-configure”) 可以查找到当前硬件连接的 ARM 芯片。如图 2-1 所示。

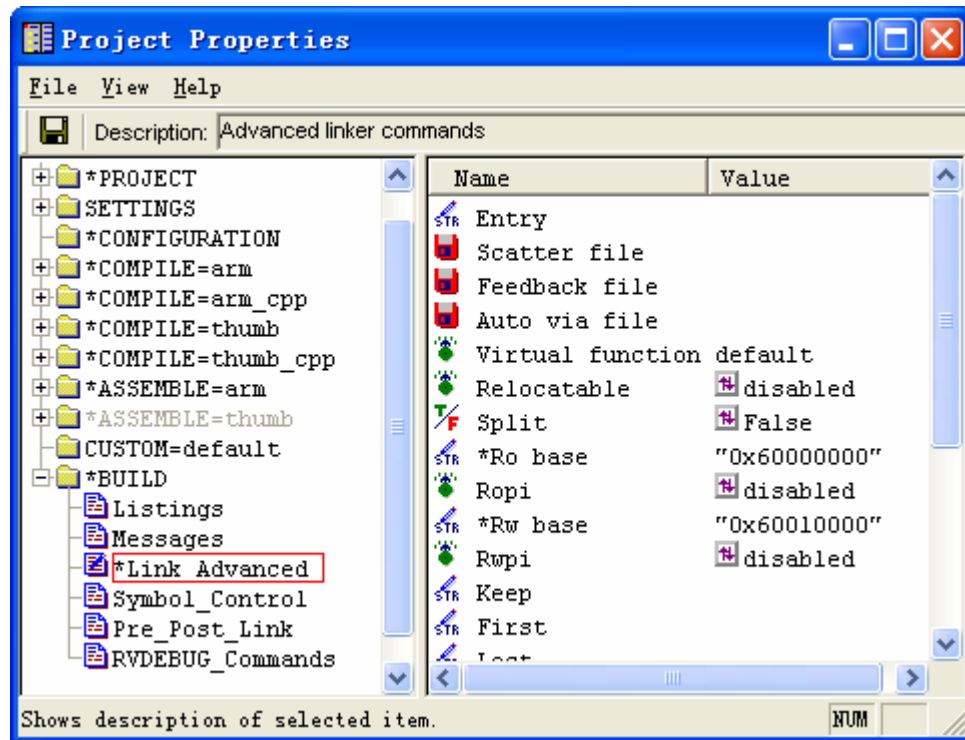
图2-1 Multi ICE Server





- 步骤 3 启动 RealView Debugger，在“Connction Control”窗口中选 ARM926EJ-S，与 Multi-ICE Server 建立连接。
- 步骤 4 选择“Project>Open Project”菜单项，打开 H3510 SDK 光盘中的 tools\flash\_easy\_Hi3510DVS\目录下的“flash\_easy.prj”文件。
- 步骤 5 选择“project>project properties”菜单项。设置 RO base 为 0x60000000，RW base 为 0x60010000，如图 2-2 所示。

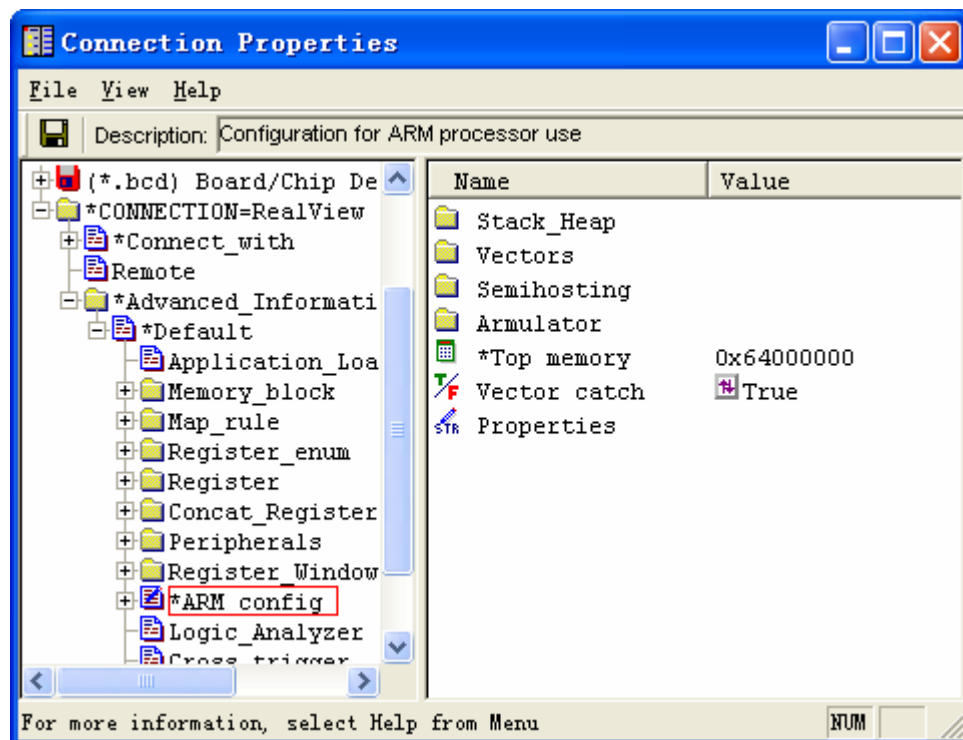
图2-2 project properties 设置



- 步骤 6 选择“File>Connection>Connection properties”菜单项，在此菜单里修改 Top Memory 的值为 0x64000000，如图 2-3 所示。



图2-3 Connection properties 设置



步骤 7 选择“Tools>Build”菜单项，编译程序，如果没有错误，就可以载入程序。

步骤 8 选择“File/Load image”菜单项，载入 H3510 SDK 光盘中  
tools\flash\_easy\_Hi3510DVS\Debug 目录下的 flash\_easy.axf。

步骤 9 单击“运行”按钮，然后在“CMD”窗口下按照提示先输入要下载的文件名（如：  
c:\hiboot.bin），然后输入要写入的 Flash 地址：0x34000000。

----结束

## 2.5 HiBoot 常用命令

Hi3510 常用命令有：

bootm	cmp	cp	erase	flinfo
go	help	?	loadb	md
mm	mw	nm	ping	printenv
protect	saveenv	setenv	tftp	

HiBoot 常用命令的描述如表 2-1 所示。



说明

HiBoot 支持命令自动补齐，当输入命令的部分字母时，按下 Tab 键，系统将自动补齐或者列出可能的命令列表。

表2-1 HiBoot 常用命令说明

命令	描述
?	得到所有命令列表或者列出某个命令的帮助。 用法：? [command ...]。 说明：列出命令的帮助信息，当不带参数时，列出所有命令及简要说明。需要查看具体命令的详细帮助，在? 后输入命令的名称。
help	help 同? 。
printenv	打印环境变量。 用法：printenv [name ...]。 说明：打印环境变量，当不带参数时，打印所有变量。
setenv	设置或者删除变量。 用法：setenv name [ value ]。 说明：name 一般是 Hiboot 环境变量的名字（请参见 <a href="#">2.6 HiBoot 环境变量</a> ），也可以是用户自定义的变量；当 value 为空时，删除变量“name”，否则设置变量“name”，且值为“value”。
saveenv	保存变量。 用法：saveenv。 说明：保存变量及其值至 Flash。
ping	用于简单判断目标机网络状态或本机网络工作状态。 用法：ping <ipaddr>。 说明：ipaddr 表示目的主机的 IP。 当网络工作正常时，结果显示“host <ipaddr> is alive”， 否则显示“ping failed;host <ipaddr> is not alive”。
loadb	通过串口 Kermit 协议下载二进制文件。 用法：loadb [ addr ] [ baud ]。 说明：addr 参数为存储文件的地址，baud 为串口下载速率。输入命令后，在超级终端的菜单中选择[传送/发送文件]，在弹出的窗口中，协议必须选择“Kermit”。 例子：hisilicon > loadb 0x63000000 57600 注意：使用 loadb，只能下载到 RAM 中，不能直接下载到 Flash。



命令	描述
tftp	<p>从 tftp 服务器中下载文件至 RAM 或者 Flash 中。</p> <p>用法: tftp addr file。</p> <p>说明: 将 file 文件下载到地址为 addr 的 RAM 者 Flash 中。</p> <p>注意: 使用 tftp 时, 必须先设置好网络配置, 使用 setenv 配置 ipaddr、netmask、serverip 参数。</p> <p>例:</p> <pre>hisilicon &gt; setenv ipaddr 192.168.1.1 hisilicon &gt; setenv netmask 255.255.255.0 hisilicon &gt; setenv serverip 192.168.1.254 hisilicon &gt; tftp 0x61000000 vmlinux</pre>
cp	<p>拷贝内存。</p> <p>用法: cp [.b,.w,.l] source target count。</p> <p>说明: 把地址为 source 的内存区域的值, 拷贝到地址 target 的内存区域, 区域的大小为 count。source 和 target 可以是 SDRAM 的地址范围, 也可以是 Flash 的地址范围。实际拷贝的大小, 因命令的不同而不同。</p> <ul style="list-style-type: none"><li>• cp.b 拷贝 1×count 个字节</li><li>• cp.w 拷贝 2×count 个字节</li><li>• cp.l 拷贝 4×count 个字节</li><li>• 简单使用 cp 时, 默认为 4*count 个字节</li></ul>
protect	<p>Flash 写保护操作。</p> <p>用法一: protect on off start end。</p> <p>说明: 对 Flash 从地址 start 到地址 end 区域进行写保护操作。</p> <p>注意: Flash 的写保护操作必须以块为最小单位, 因此地址 start 必须为某块的起始地址, end 地址则必须为某块的结束地址, 如 Flash 的基地址为 0x34000000, 块大小为 0x20000, 则操作 protect on 0x34000000 0x3401FFFF 为可操作的。而 protect on 0x34000003 0x3401FFFF 或者 protect off 0x34000000 0x3401FF00 均不可操作。</p> <p>用法二: protect on off N:SF [-SL]。</p> <p>说明: 对第 N 块 Flash 的 SF 扇区到 SL 扇区进行写保护操作。</p> <p>用法三: protect on off bank N。</p> <p>说明: 对第 N 块 Flash 进行写保护操作。</p> <p>用法四: protect on off all。</p> <p>说明: 对所有 Flash 进行写保护操作。</p>
go	<p>跳转到指定地址, 执行代码。</p> <p>用法: go addr [ arg ... ]。</p> <p>说明: 执行地址 addr 处的二进制代码, 可传递 arg 参数。</p>



命令	描述
bootm	设置运行环境，并开始执行二进制代码。 用法: bootm [ addr [ arg ... ] ]。 说明: 执行 addr 地址处的代码，要求二进制代码为 mkimage 处理过的二进制文件。
flinfo	列出 Flash 信息。 用法: flinfo [ N ]。 说明: 不带参数时列出所有 Flash 的信息，否则列出第 N 块 Flash 的信息。
md	显示内存区的内容。 用法: md [.b, .w, .l] address。 说明: 显示地址 address 内存区内容。 <ul style="list-style-type: none"><li>• 使用 md.b，显示单位为 1 字节</li><li>• 使用 md.w，显示单位为 2 字节</li><li>• 使用 md.l，显示单位为 4 字节</li><li>• 简单使用 md 时，等价于 md.l</li></ul>
mm	修改内存区的内容。地址自动增加。 用法: mm [.b, .w, .l] address。 说明: 修改地址 address 内存区内容。 <ul style="list-style-type: none"><li>• 使用 mm.b，每次修改 1 字节</li><li>• 使用 mm.w，每次修改 2 字节</li><li>• 使用 mm.，每次修改 4 字节</li><li>• 简单使用 mm 时，等价于 mm.l</li></ul>
nm	修改内存区的内容，地址不自动增加。 用法: nm [.b, .w, .l] address。 说明: 修改地址 address 内存区内容。 <ul style="list-style-type: none"><li>• 使用 nm.b，每次修改 1 字节</li><li>• 使用 nm.w，每次修改 2 字节</li><li>• 使用 nm.，每次修改 4 字节</li><li>• 简单使用 nm 时，等价于 nm.l</li></ul>



命令	描述
mw	<p>填充内存。</p> <p>用法: <code>mw [.b, .w, .l] address value [ count ]</code>。</p> <p>说明: 把地址 <code>address</code> 开始的 <code>count</code> 字节的连续内存区域的值设为 <code>value</code>。</p> <ul style="list-style-type: none"><li>• 使用 <code>mw.b</code> 时设置大小为 <math>1 \times \text{count}</math> 字节</li><li>• 使用 <code>mw.w</code> 时设置大小为 <math>2 \times \text{count}</math> 字节</li><li>• 使用 <code>mw.l</code> 时设置的大小为 <math>4 \times \text{count}</math> 字节</li><li>• 简单使用 <code>mw</code> 时, 等价于 <code>mw.l</code></li></ul> <p>例:</p> <pre>hisilicon &gt; mw 0x32000000 FF 10000</pre>
cmp	<p>比较两块内存区。</p> <p>用法: <code>cmp [.b, .w, .l] addr1 addr2 count</code>。</p> <p>说明: 比较地址 <code>addr1</code> 和地址 <code>addr2</code>, 大小 <code>count</code> 的内存内容进行比较。</p> <ul style="list-style-type: none"><li>• 使用 <code>cmp.b</code> 时比较大小为 <math>1 \times \text{count}</math> 字节</li><li>• 使用 <code>cmp.w</code> 时比较大小为 <math>2 \times \text{count}</math> 字节</li><li>• 使用 <code>cmp.l</code> 时比较的大小为 <math>4 \times \text{count}</math> 字节</li><li>• 简单使用 <code>cmp</code> 时, 等价于 <code>cmp.l</code></li></ul>
erase	<p>擦除 Flash 内容。</p> <p>用法一: <code>erase start end</code>。</p> <p>说明: 擦除地址从“start”到地址“end”区域的内容。</p> <p>注意: Flash 的擦除操作必须以块为最小单位, 因此地址 <code>start</code> 必须为某块的起始地址, <code>end</code> 地址则必须为某块的结束地址, 如 Flash 的基地址为 <code>0x34000000</code>, 块大小为 <code>0x20000</code>, 则操作 <code>erase 0x34000000 0x3401FFFF</code> 为可操作的。而 <code>erase 0x34000003 0x3401FFFF</code> 或者 <code>erase 0x34000000 0x3401FF00</code> 均不可操作。</p> <p>用法二: <code>erase N:SF [-SL]</code>。</p> <p>说明: 擦除第 <code>N</code> 块 Flash 的从扇区 <code>SF</code> 到 <code>SL</code> 扇区的内容。</p> <p>用法三: <code>erase bank N</code>。</p> <p>说明: 擦除第 <code>N</code> 块 Flash 的内容。</p> <p>用法四: <code>erase all</code>。</p> <p>说明: 擦除所有 Flash 的内容。</p>

## 2.6 HiBoot 环境变量

使用 HiBoot 常用命令“setenv”可以设置 HiBoot 环境变量, 表 2-2 列出常用环境变量及其设置格式等信息。



表2-2 HiBoot 常用环境变量说明

环境变量	描述
ipaddr	设置 Hi3510 视频评估板得 ip 地址。 格式: xxx.xxx.xxx.xxx 例子: <code>setenv ipaddr 192.168.0.100</code> 说明: 设置 ip 地址为 192.168.0.100。
serverip	设置服务端 ip 地址, 在 tftp 中被使用。 格式: xxx.xxx.xxx.xxx 例子: <code>setenv serverip 192.168.0.10</code> 说明: 设置 tftp 服务器 ip 地址为 192.168.0.10。
netmask	设置子网掩码。 格式: xxx.xxx.xxx.xxx 例子: <code>setenv netmask 255.255.255.0</code> 说明: 设置子网掩码为 255.255.255.0。
gatewayip	设置网关。 格式: xxx.xxx.xxx.xxx 例子: <code>setenv gatewayip 192.168.0.1</code> 说明: 设置网关 ip 地址为 192.168.0.1。
bootargs	启动 OS 时的启动参数。 格式: <code>arg1=value1 arg2=value2 ... argn=valuen</code> 例子: <code>setenv bootargs mem=32M console=ttyAMA0 root=/dev/mtdblock/2</code> 说明: 传递参数, 包括内存大小, 根文件系统设备等。
bootcmd	设置 HiBoot 自动启动及执行命令。启动延时依据 bootdelay 变量值 (详见 bootdelay 参数描述), 若 bootdelay 未被设置, 则默认延时时间为 2 秒。 格式: <code>cmd1;cmd2;...;cmdn</code> 例子 1: <code>setenv bootcmd bootm 0x34500000</code> 说明: 设置启动后自动执行 0x34500000 处的代码。 例子 2: <code>setenv bootcmd 'printenv;bootm 0x34500000'</code> 说明: 设置启动后自动依次执行打印参数和执行 0x34500000 处的代码。 注意: 多个参数时, 参数之间使用分号相隔。将整个参数字串用单引号包含起来。





环境变量	描述
bootdelay	<p>设置自启动延时时间。单位为秒。只有当 bootcmd 变量被设置后，该变量才有效。该变量值范围为大于等于-1 的整数。当设置为-1 时，关闭自启动的功能。</p> <p>格式：value</p> <p>例子 1：setenv bootdelay 4</p> <p>说明：设置自启动延时 4 秒。</p> <p>例子 2：setenv bootdelay -1</p> <p>说明：关闭自启动功能。</p> <p>提示：在延迟时间内可按任意键切换到命令行模式。</p> <p>注意：在产品开发调试阶段请勿设置延迟时间为 0。若设置，可以在启动瞬间使用 CTRL+C 中断程序而进入命令行模式。</p>



# 3 Linux 内核

## 关于本章

本章描述了在 Linux 服务器上进行 Linux 内核的配置、编译，以及 mkimage 工具的使用，内容如下表所示。

标题	内容
<a href="#">3.1 内核源代码</a>	简单介绍了内核源代码的路径信息。
<a href="#">3.2 配置内核</a>	介绍了配置内核的操作。
<a href="#">3.3 编译内核</a>	介绍了编译内核的操作。
<a href="#">3.4 使用 mkimage 工具</a>	介绍了如何使用 mkimage 工具。



## 3.1 内核源代码

成功安装 Hi3510 SDK 后，内核代码已存放于 SDK 目录下的 code/linux/kernel 目录中，用户可直接进入目录进行相关操作。

## 3.2 配置内核



### 注意

如果对内核和 Hi3510 平台没有足够了解，请勿修改默认配置。但可增加需要的模块。

配置内核操作步骤如下：

```
hisilicon$cd code/linux/kernel/linux-2.6.14
hisilicon$make mrproper
hisilicon$make menuconfig
```

其中“make mrproper”为可选步骤，用户可直接通过“make menuconfig”进行内核配置。如果执行了 make mrproper，必须重新加载.config 文件，方法为：

1. 执行 make menuconfig
2. 选择“Load an Alternate Configuration File”菜单项
3. 输入 arch/arm/configs/hi3510\_dvs\_defconfig
4. 选择需要的模块
5. 选择完毕后，保存并退出

也可以手动拷贝.config 文件，方法为：

```
cp arch/arm/configs/hi3510_xxx_defconfig .config
```



### 说明

配置操作中可以使用 make config 和 make xconfig 替代 make menuconfig，但 make config 界面不直观、操作繁琐；make xconfig 需要 XWindow 支持。所以建议使用 make menuconfig，便于远程操作，而且界面比较直观。

## 3.3 编译内核

配置保存后，可直接输入“make”命令编译内核，此时需要等待几分钟。



#### 说明

如果编译过程中出现错误，可执行“make clean”或者“make mrproper”，然后重新运行“make menuconfig”，加载 hi3510\_xxx\_defconfig 文件，选择完毕后执行“make”。

## 3.4 使用 mkimage 工具

内核编译成功后，在 arch/arm/boot 目录下生成内核文件，其中包括压缩文件 zImage 和没有压缩文件 Image。

在 HiBoot 中使用 bootm 命令引导内核，必须使用 mkimage 工具对 zImage 文件进行处理，增加相应的入口信息等。

#### 说明

mkimage 存放在 SDK 目录下的 tools/bin 中。为了方便地访问这下面的命令，可能需要设置 PATH 环境变量，也可以将 mkimage 拷贝到 /usr/local/bin 目录中。

具体操作如下：

```
hisilicon$ mkimage -A arm -T kernel -C none -a 0x60a00000 -e 0x60a00000 -n  
hilinux -d arch/arm/boot/zImage hikernel
```

参数说明如表 3-1 所示。

表3-1 mkimage 参数表

参数	说明
A	指定体系结构类型 ARM
T	指定 image 类型为 kernel
C	设置压缩类型 none
a	设置加载地址
e	设置入口地址
n	设置 image 名字
d	需要处理的文件

执行上面的命令后将在当前目录下生成名为 hikernel 的文件，这就是内核映像文件，该内核映像文件可以被下载到单板的任何地址（除了覆盖 Hiboot 和解压目的地址等特殊位置）运行，如烧写到 Flash 或者放在 RAM 中。

#### 说明

最好将“加载地址”和“入口地址”设置成相同的，并且都是在 RAM 中的地址。“加载地址”用于 Hiboot 将内核 image 文件拷贝到该地址，“入口地址”用于 Hiboot 加载内核 image 之后跳转到该地址。



# 4 根文件系统

## 关于本章

本章描述了 Linux 的根文件的目录结构及如何利用 busybox 制作根文件系统，简单介绍了 Cramfs、JFFS2 和 NFS3 种文件系统，内容如下表所示。

标题	内容
<a href="#">4.1 根文件系统简介</a>	简单介绍了根文件系统。
<a href="#">4.2 利用 busybox 制作根文件系统</a>	介绍了如何利用 busybox 制作根文件系统。
<a href="#">4.3 文件系统简介</a>	简单介绍了 Cramfs、JFFS2 和 NFS3 种文件系统。



# 4.1 根文件系统简介

Linux 的目录结构的最顶层是一个被称为“/”的根目录。系统加载 Linux 内核之后，就会挂载一个设备到根目录上。存在于这个设备中的文件系统被称为根文件系统。所有的系统命令，系统配置以及其他文件系统的挂载点都位于这个根文件系统中。

根文件系统常常存放于 RAM 和 Flash 中，或是基于网络的文件系统。根文件系统中存放了嵌入式系统使用的所有应用程序、库以及其他需要用到的服务。图 4-1 列出了根文件系统的顶层目录。

图4-1 根文件系统顶层目录结构图



一个通用的 Linux 系统的根文件系统中会包括上表中所有的目录，不过在嵌入式系统中，需要精简根文件系统。在嵌入式系统中可以被忽略的目录如表 4-1 所示。

表4-1 嵌入式系统中可忽略的目录说明

目录名称	描述
/home、/mnt、/opt 和 /root	所有适合提供给多用户扩展的目录，都可以被忽略。
/var 和/tmp	/var 是存放系统日志或一些服务程序的临时文件； tmp 是存放用户的一些临时文件，也可以被忽略。
/boot	/boot 目录一般用于存放内核映像，PC 机启动时一般会从该位置加载内核，但在嵌入式系统中，为了节省空间，内核映像存在于 Flash 或网络服务器中，而不是在根文件系统中。因此也可以忽略这个目录。

注：空目录并不会增大文件系统的体积，如果没有特殊情况，建议保留这些目录。



## 4.2 利用 busybox 制作根文件系统

利用 busybox 制作根文件系统需要先获取 busybox 源代码，然后配置、编译和安装 busybox，操作成功后开始制作根文件系统。

### 4.2.1 获取 busybox 源代码

成功安装 Hi3510 SDK 后，busybox 完整源代码存放在 VSSDKV100R001BXX/code/linux/fs 目录中。也可以从网站 <http://www.busybox.net> 下载。

### 4.2.2 配置 busybox

进入 busybox 所在目录，输入如下命令进行配置操作。

```
hisilicon$ make menuconfig
```

busybox 的配置界面和内核配置相似，其功能选项容易理解，可以根据自己的需求选择配置。在 Build Options 中注意下面两个选项：

```
[*]Build BusyBox as a static binary (no shared libs)
[*] Do you want to build BusyBox with a Cross Compiler? (arm-linux-) Cross Compiler
prefix
```

第一个选项选择是否把 busybox 编译成静态链接的可执行文件。如果选择该选项，编译出来的 busybox 就是静态链接的，运行时不依赖于其他动态库，但体积较大；清除该选项将得到动态链接的 busybox，体积较小，但需要其他运行时库的支持。

第二个选项是选择交叉编译器，并配置交叉编译器为 arm-linux。配置好后保存并退出。

请参考 busybox 相关文档了解 busybox 配置及各选项。

### 4.2.3 编译和安装 busybox

编译和安装 busybox 的具体操作步骤：

```
hisilicon$ make
hisilicon$ make install
```

编译并安装成功后，在 busybox 目录下的 \_install 目录下生成以下目录及文件：

```
drwxr-xr-x  2 linux  linux  4096 2005-04-22 11:01 bin
lrwxrwxrwx  1 linux  linux  11 2005-04-22 11:01 linuxrc->bin/busybox
drwxr-xr-x  2 linux  linux  4096 2005-04-22 11:01 sbin
drwxr-xr-x  4 linux  linux  4096 2005-04-22 11:01 usr
```

### 4.2.4 制作根文件系统

成功安装 Hi3510 SDK 后，在 VSSDKV100R001BXX/rootbox/ 目录中存放已制作好的根文件系统。



用户如有需要可在 busybox 的基础上制作根文件系统，busybox 源代码存放在 SDK 目录中的 code/linux/fs/ 目录下。

具体操作步骤：

```
hisilicon$mkdir rootbox
hisilicon$cd rootbox
hisilicon$cp -R code/linux/fs/busybox-1.1.2/_install/* .
hisilicon$mkdir etc dev lib tmp var mnt home proc
```

配置 etc、lib、dev 目录的必需文件。

- etc 目录可参考系统/etc 下的文件，最主要的几个文件包括 inittab、fstab、init.d/rcS 文件等，这些文件最好从 busybox 的 examples 目录下拷贝过来，再修改。因为 busybox 使用的 inittab 和 fstab 的格式与一般的 Linux 系统不同。
- dev 目录下的设备文件，可以直接从系统中拷贝过来或者使用 mknod 命令生成需要的设备文件。拷贝文件时请使用 cp -R file。
- lib 目录是存放应用程序所需要的库文件，请根据应用程序需要拷贝相应的库文件。

这样一个完整的根文件系统就生成了。

提示：

Hi3510 VSSDK 软件包已经包括配置好的完整的根文件系统，如果无特别需求，可直接使用。要添加自己开发的应用程序，只需将应用程序和相应的库文件拷贝到根文件系统的对应目录即可。

## 4.3 文件系统简介

常用文件系统包括有 Cramfs、JFF32 和 NFS。

### 4.3.1 Cramfs

Cramfs 是针对 Linux 内核 2.4 之后的版本所设计的一种新型文件系统，它是压缩和只读格式。由于它属于只读的文件系统，使用简单，加载容易，速度快。

Cramfs 的优缺点：

- 优点：将文件数据以压缩形式存储，在需要运行时进行解压缩，能节省 Flash 存储空间。
- 缺点：由于它存储的文件是压缩的格式，所以文件系统不能直接在 Flash 上运行。同时，文件系统运行时需要解压数据并拷贝至 RAM 中，会一定程度上降低读取效率。

如果想要在系统中提供 Cramfs 的能力，必须要在编译内核时把 Cramfs 的选项加入。在 make menuconfig 后，进入[File/systems]菜单项，选择“miscellaneous filesystems”，最后选中其中的“Compressed ROM file system support”选项。

mkfs.cramfs 是一个用来制作 Cramfs 文件系统映象的工具。通过这个工具处理已经制作好的根文件系统，就可以生成 Cramfs 文件系统的映象（这类似于我们把光盘制作成 ISO 文件映像）。具体操作步骤如下所示：

```
hisilicon$mkfs.cramfs ./rootbox ./cramfs-root.img
```





其中，rootbox 是之前已经制作好的根文件系统，ramfs-root.img 是生成的 Cramfs 文件系统映像文件。

## 4.3.2 JFFS2

JFFS2 是 Red Hat 的 David Woodhouse 在 JFFS 基础上改进的文件系统，用于微型嵌入式设备的原始闪存芯片的实际文件系统。JFFS2 文件系统是日志结构化的可读写的文件系统，这意味着它基本上是一长列节点，在每个节点包含相关文件的部分信息。JFFS2 具有一种称为“损耗平衡”的特点，即 Flash 所有被擦写的单元都保持相同的擦写次数。利用这种特有的保护措施，Flash 使用周期得到大的提升。

JFFS2 的优缺点：

- 优点：使用了压缩的文件格式，为 Flash 节省了大量的存储空间，它更优于 JFFS 格式在系统中使用。最重要的特性是可读写操作。
- 缺点：JFFS2 文件系统挂载时需要扫描整个 JFFS2 文件系统，因此当 JFFS2 文件系统分区增大时，挂载时间也会相应的变长。使用 JFFS2 格式可能带来少量的 Flash 空间的浪费。这主要是由于日志文件的过度开销和用于回收系统的无用存储单元，浪费的空间大小大致是若干个数据段。JFFS2 的另一缺点是当文件系统已满或接近满时，JFFS2 会大大放慢运行速度。这是因为垃圾收集的问题。

加载 JFFS2 文件系统的时候分四步进行。

- 步骤 1 扫描整个芯片，对日志节点进行校验，并且将日志节点全部装入内存缓存；
- 步骤 2 对所有日志节点进行整理，抽取有效的节点并整理出文件目录信息；
- 步骤 3 找出文件系统中无效节点并且将它们删除；
- 步骤 4 最后整理内存中的信息，将加载到缓存中的无效节点释放。

----结束

由此可以看出虽然这样能有效地提高系统的可靠性，但是在一定程度上降低了系统的速度。尤其对于较大的闪存芯片，加载过程会更慢。

为了使内核支持 JFFS2 文件系统，必须在编译内核时把 JFFS2 的选项加入（我们发布的内核默认已经加入了支持）。在 make menuconfig 后，进入“File>Systems”菜单项，选择“Miscellaneous Filesystems”，最后选中其中的“Journalling Flash File System V2 (JFFS2) Support”选项。

JFFS2 的制作方法为：

```
hisilicon$mkfs.jffs2 -d ./rootbox -l -e 0x20000 -o jffs2-root.img
```

其中，mkfs.jffs2 工具可以从网上下载，也可以在~/hilinux/tools 中找到。rootbox 为之前已经制作好的根文件系统。参数说明如表 4-2 所示。

表4-2 jffs2 参数表

参数	说明
d	指定根文件系统



参数	说明
l	little-endian 小端模式
e	Flash 的块大小
o	输出映像文件

### 4.3.3 NFS

使用 Cramfs 和 JFFS2 时，需要先将根文件系统映像烧入 Flash，系统启动时会从 Flash 中加载。但是在系统开发或移植的初期，需要经常修改或者添加应用程序。每修改一次就需要重新烧入一次，这样做不仅耗费时间，而且对 Flash 的寿命会有影响。

NFS 是一种分布式的文件系统，用于共享文件和打印机。它允许用户调用挂载远端的文件系统或设备来实现共享，使用方式与挂载本机的文件系统一样。NFS 使用“客户—服务器”模型。在这种模型中，服务器输出需要共享的目录，客户可通过网络挂载这些目录并访问其中的文件。

使用 NFS 作为根文件系统，内核会根据预先设置好的内核命令参数挂载一个 NFS sever 中输出的目录作为其根目录。这个过程不需要任何对 Flash 的操作，修改内核完全在主机中进行，非常适于开发初期的调试阶段。

在 Linux 服务器配置 NFS 根文件系统的方法为：

```
hisilicon$ vi /etc/exports
```

添加路径及参数，如：/home/VSSDKV100R001BXX/rootbox/ \*(rw,no\_root\_squash,async)

```
hisilicon$ /etc/init.d/ nfs start //启动 NFS 服务
```

以上操作必须超级用户完成，且导出的目录必须是绝对路径；如果 NFS 服务已经开启，在配置文件后只需重新启动 NFS 服务，即/etc/init.d/ nfs restart。

#### 建议：

以上所提到的 3 种文件系统各有利弊。

- Cramfs 和 JFFS2 具有好的空间特性，很适合嵌入式产品应用。
- Cramfs 为只读文件系统，
- JFFS2 为可读写文件系统，可根据需要在产品中使用其中之一或者结合两者使用。
- NFS 文件系统适用于开发初期的调试阶段。

因此，建议根据产品的特点和不同开发阶段，来选择适合自己的实现方法。



# 5 烧写内核和根文件系统

## 关于本章

本章描述了 Hi3510 的存储空间，以及如何通过网口和串口烧写内核和根文件系统，内容如下表所示。

标题	内容
<a href="#">5.1 存储器地址空间</a>	介绍了在 VSEVB 板上存储器地址空间。
<a href="#">5.2 通过网口烧写</a>	介绍了如何通过网口进行烧写操作。
<a href="#">5.3 通过串口烧写</a>	介绍了如何通过串口进行烧写操作。



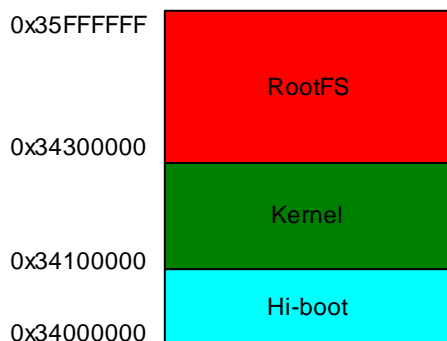
## 5.1 存储器地址空间

Hi3510 视频评估板包含 SDRAM 存储器和 Flash 存储器。SDRAM 的地址空间从 0x60000000 开始；Flash 的地址空间从 0x34000000 开始。具体大小随单板不同，可从单板硬件手册中获取。

Flash 的使用有特殊要求：

- 0x34000000~0x34100000 保留空间，供存放 Hiboot。
- 0x34100000~0x34300000 保留空间，供存放内核。
- 其余空间可自行分配。
- 随单板发布的软件在 Flash 的存放位置如图 5-1 所示。
- 其余暂保留或有其他用途。

图5-1 Flash（32MB）地址空间分配示意图（仅供参考）



具体内容请参见《Hi3510 视频评估板 用户指南》。

## 5.2 通过网口烧写

通过网口烧写内核和根文件系统，首先需要进行参数设置和建立 tftp 服务，然后才能下载内核和根文件系统。

### 5.2.1 参数设置和建立 tftp 服务

用普通网线连接 Hi3510 单板的 SF 上行口，然后在 HiBoot 中设置相关参数。HiBoot 只支持 tftp 协议。设置参数的具体操作如下所示。

```
hisilicon#setenv serverip 10.70.153.114    //设置服务器端的 IP 地址，可根据需要具体设定
hisilicon#setenv ipaddr 10.70.153.100      //设置 Hi3510 单板的 IP 地址
hisilicon#setenv netmask 255.255.255.0     //设置 netmask
hisilicon#setenv gatewayip 10.70.153.1     //设置网关
hisilicon#saveenv
hisilicon# ping 10.70.153.114              //用来判断网络是否正常。
```



目前 HiBoot 不支持广播包的接收，不能响应 ping 包，无法通过 ping 单板的方式判断网络是否畅通。但是支持向外发送 ping 包，并能接收 ping 包的响应包。

上述最后一个操作中，返回 host 10.70.153.114 is alive 表示网络工作正常；显示 ping failed; host 10.71.138.81 is not alive，说明网络不正常，需要重新检查网络设置。

另外还需要在 Windows 工作台或者 Linux 服务器中建立 tftp 服务，建议在 Windows 工作台上建立 tftp 服务器，简单方便。

## 5.2.2 下载内核

下载内核的操作步骤如下所示。

```
hisilicon#tftp 0x34100000 hikernel //将tftp服务器上的hikernel文件下载到0x34100000的位置
```

正常的下载过程超级终端中显示的信息如下所示。

```
TFTP from server 10.70.153.114; our IP address is 10.70.153.100
Filename 'hilinux'.
Load address: 0x34100000
Loading:
%#####
done
```

## 5.2.3 下载根文件系统

下载根件系统的操作步骤如下所示。

```
hisilicon#tftp 0x34300000 cramfs-root.img //将cramfs-root.img文件下载到0x34300000
```

正常的下载过程超级终端中显示的信息如下：

```
TFTP from server 10.70.153.114; our IP address is 10.70.153.100
Filename 'hilinux'.
Load address: 0x34300000
Loading:
%#####
#####
#####
#####
#####
#####
#####
#####
done
```

由于 Flash 的写操作速度较慢，如果下载的文件较大，则需要花费一定的时间，等到重新回到“hisilicon#”的提示符，表示下载完成。

## 5.3 通过串口烧写

通过串口烧写内核和根文件系统，首先需要在 PC 机和 Hi3510 视频评估板之间通过串口连接，然后才能下载内核和根文件系统。

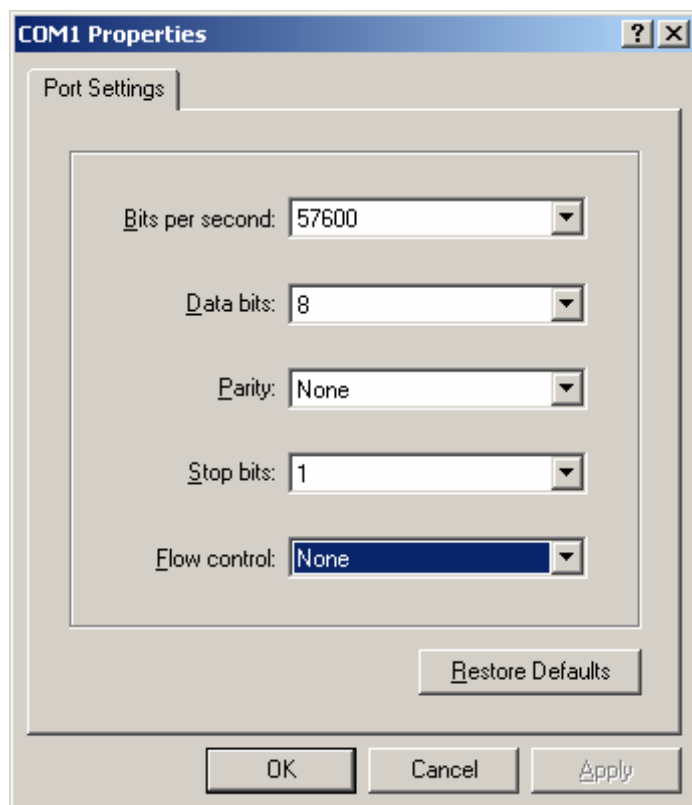


### 5.3.1 连接设备

连接设备的步骤如下。

- 步骤 1 使用串口线（DB9 接口）连接计算机的 COM1（也可以其他串口，这里假设使用 COM1）和 Hi3510 视频评估板的 COM1。
- 步骤 2 启动 Windows 工作台的超级终端软件，设置 COM1 的参数如图 5-2 所示。

图5-2 串口设置



- 步骤 3 启动 Hi3510 视频评估板，系统进入 HiBoot 命令行操作界面，表示系统工作正常，可进行下载或其他操作。

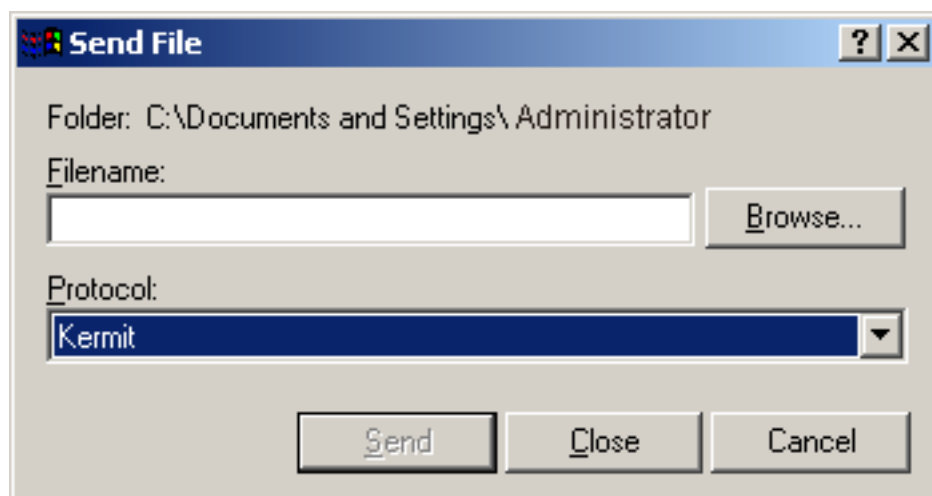
----结束

### 5.3.2 下载内核

在超级终端的 HiBoot 命令行中输入 loadb 0x61000000（存放内核的 RAM 地址），打开菜单“传输”下的“发送文件”，弹出对话框，如图 5-3 所示，选择“Protocol>Kermit”菜单项，在“Filename”中选择内核文件。



图5-3 发送文件窗口



等待下载完成后，使用 HiBoot 的 cp 命令将内核从 RAM 拷贝到 Flash 中，操作方法如下所示。

```
hisilicon$cp 0x61000000 0x34100000 0x60000 //拷贝内核到 0x34100000 位置，大小为 1.57M
```

如果提示 Flash 写保护无法写入，则先将 Flash 写保护关闭，然后再进行拷贝，操作如下所示。

```
hisilicon#protect off all  
hisilicon#cp 0x61000000 0x34100000 0x60000 //拷贝内核到 0x34100000 位置，大小为 1.57M
```

### 5.3.3 下载根文件系统

下载根文件系统和下载内核的操作方法相同。

- 步骤 1 在超级终端的 HiBoot 命令行中输入 loadb 0x61000000（存放根文件系统映像文件的 RAM 地址）。
- 步骤 2 选择“传输>发送文件”菜单项，弹出对话框，在“Protocol”的下拉列表中选择“Kermit”选项，在“Filename”中选择根文件系统映像文件（在“4 根文件系统”中已经制作好的“Cram-rootfs.img”或者“jffs2-rootfs.img”）。
- 步骤 3 等待下载完成后，使用 HiBoot 的 cp 命令将文件从 RAM 拷贝到 Flash 中。

具体操作方法如下所示。

```
hisilicon#cp 0x61000000 0x34300000 0x180000 //拷贝文件到 0x34300000 位置，大小为 6M
```

----结束

#### 建议：

使用串口下载速度很慢但操作简单，适合下载小文件，而通过网口则可以大大提高工作效率。建议使用网口下载。



# 6 启动 Linux

## 关于本章

本章描述了 Hi3510 视频评估板如何启动 Linux，具体内容包括设置启动参数、启动 Linux 和通过设置 HiBoot 自动启动 Linux，内容如下表所示。

标题	内容
<a href="#">6.1 设置启动参数</a>	介绍了在 3 种文件系统时如何设置启动参数。
<a href="#">6.2 启动 Linux</a>	介绍了如何启动 Linux。
<a href="#">6.3 设置 HiBoot 自动启动 Linux</a>	介绍了如何设置 HiBoot 实现自动启动 Linux。

## 6.1 设置启动参数

从 HiBoot 引导内核，需要给内核传递参数，包括内存大小，根文件系统挂载设备等。根据根文件系统类型不同，设置也相应不同。具体设置如下所示。

- Cramfs

```
hisilicon# setenv bootargs mmz=sdram,1,0x60cM,20M mem=32M console=ttyAMA0 , 57600  
root=/dev/mtdblock/1 rootfstype=cramfs  
hisilicon#saveenv
```

- Jffs2

```
hisilicon# setenv bootargs mmz=sdram,1,0x60cM,20M mem=32M console=ttyAMA0 , 57600  
root=/dev/mtdblock/1 rootfstype=jffs2  
hisilicon#saveenv
```

- NFS

```
hisilicon#setenv bootargs mmz=sdram,1,0x60cM,20M mem=32M console=ttyAMA0 , 57600  
root=/dev/nfs nfsroot=10.70.153.114:/share/rootbox/  
ip=10.70.153.100:10.70.153.114:10.70.153.1:255.255.255.0:hi35xx:eth0:off  
hisilicon#saveenv
```





其中，mmz 用来定义 media-mem 的分配池，语法为：

```
mmz=<name>,<gfp>,<phys_start_addr>,<size>[[;][<name>,<gfp>,<phys_start_addr>]]
```

mmz 的各个参数的说明如下：

- <name>: 字符串，分配池的名字，比如 SDRAM。
- <gfp>: 数字，表示分配池的属性。
  - 1: SDRAM
  - 2: DDR
- <phys\_start\_addr>: 16 进制数，分配池的物理起始位置，如 0x62000000。
- <size>: 16 进制数，分配池的长度，如 0x1000000。

以上每个参数都是必需的，参数之间用 “,” 号分隔。可以指定多个分配池，之间用 “;” 号分隔。实例如下：

- mmz=sdram,1,0x61000000,0x1000000;ddram,2,0xF0000000,0x2000000
- mmz=sdram,1,0x610M,16M;ddram,2,0xF00M,32M

## 6.2 启动 Linux

在 HiBoot 命令行中输入 bootm 0x34100000 即可。

```
hisilicon#bootm 0x34100000 //从 0x34100000 处启动 linux
```

## 6.3 设置 HiBoot 自动启动 Linux

设置 HiBoot 自动启动 Linux 的操作步骤如下所示。

```
hisilicon#setenv bootcmd bootm 0x34100000 //设置自启动命令参数
hisilicon#setenv bootdelay 2 //设置启动延时为 2 秒
hisilicon#saveenv
```



# 7 应用程序开发简介

本章简单介绍了如何进行应用程序开发。

## 关于本章

本章描述内容如下表所示。

标题	内容
7.1 编写代码	简单介绍在不同环境下选择代码编写工具。
7.2 运行应用程序	简单介绍运行已编译好的程序。
7.3 使用 gdb server 调试应用程序	简单介绍使用 gdb server 调试应用程序。

## 7.1 编写代码

用户可根据个人习惯选择代码编写工具。通常在 Windows 环境使用 Source Insight，而在 Linux 环境下使用 Vim+ctags+cscope，功能也相当强大。

## 7.2 运行应用程序

要运行编译好的应用程序，首先需要添加到目标系统中，必须完成以下工作：

- 将应用程序和需要的库文件（如果有）等添加到根文件系统相应的目录（VSSDKV100R001BXX /rootbox/）中。通常将应用程序放到/bin 目录里，库文件放到/lib 目录里，配置文件则放到/etc 目录里。
- 制作包含新应用程序的根文件系统。

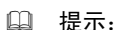


#### 说明

如果执行应用程序，需要读写文件系统操作。请选择 Jffs2 文件系统，或者使用 Cramfs 和 Jffs2 两者结合。

在调试阶段推荐使用 NFS 文件系统，可以省去重新制作根文件系统和烧写工作。设置和启动 NFS 服务（请参见“4.3.3 NFS”），然后设置启动参数即可（请参见“6.1 设置启动参数”）。以后只需要简单的拷贝应用程序到根文件系统目录中，就可以在目标系统里运行。

如果需要制作 Cramfs 或 Jffs2 文件系统，制作相应的文件系统（请参见“4.3 文件系统简介”），然后烧写根文件系统到 Flash 指定位置（0x34300000）（请参见“5 烧写内核和根文件系统”），并设置相应的启动参数。同样，启动 Linux 后便可运行新的应用程序。



#### 提示：

如果新添加的应用程序需要系统启动后自动运行，请编辑/etc/init.d/rcs 文件，添加需要启动的应用程序路径。

## 7.3 使用 gdb server 调试应用程序

在很多情况下，需要对一个应用程序进行调试。在 Linux 下调试程序，常用的工具是 gdb，但由于嵌入式系统资源有限，一般不直接在目标系统上运行 gdb 进行调试，而是采取 gdb+gdb server 的方式。gdb server 在目标系统中运行，gdb 则在宿主机上运行。Hi3510 SDK 提供的根文件系统已经包含了 gdb server。

1. Hi3510 单板启动 Linux 并登陆进入 shell。

如要进行 gdb 调试，首先要启动 gdb server。方法是先进入需要调试的程序所在目录，如：被调试的程序文件名是 hello，则输入命令：

```
hisilicon$ gdbserver :2000 hello &
```

上述命令表示在目标系统的 2000 端口开启了一个调试进程，hello 就是要调试的程序。

2. 在 Linux 服务器上启动 gdb 程序，因为目标系统为 ARM，所以启动 arm-elf-gdb。

```
arm-linux-gdb
```

3. 启动 arm-elf-gdb 后，在命令提示符状态下输入命令，与目标系统进行连接。

```
(gdb) target remote 10.70.153.100:2000
```



#### 注意

端口号和目标系统开启的端口号要一致。

4. 连接成功后，会输出提示信息，如下所示。

```
remote debugging using 10.70.153.100:2000
0x40000a70 in ?? ()
```

5. 可以使用两种方式进行符号文件加载：



- **(gdb) add-symbol-file hello 40000a70**
- **(gdb) file hello**

6. 输入各种 gdb 命令如 list、run、next、step、break 即可进行程序调试。



# A 建立 Linux 开发环境

服务器的 Linux 版本没有限制，但建议使用较新的 Linux 发行版，如 RedHat 9.0、Fedora Core、Debian 和 Mandrake 等。这里以 Fedora Core 2.0 为例，介绍如何建立 Linux 开发环境。

## A.1 安装 Linux 系统的配置选项

因为在市场销售的发行版一般都提供 60~90 天的电话技术支持，因此，建议购买在市场销售的发行版 Linux，不建议从网上下载或从其它渠道获得。

请参考随发行版附带的《Hi3510 Linux 开发环境 用户指南》进行 Linux 系统的安装，安装中应注意的配置选项如表 A-1 所示。

表A-1 安装 Linux 系统的配置选项说明

配置选项	建议	目的
默认语言	英文	避免远程登录的时候，由于终端不支持中文产生乱码。
磁盘分区	自动分区	留出足够的磁盘空间供安装SDK使用
防火墙	禁用	使后续启动的系统服务能正常工作
安装类型	完全安装	避免因缺少必要的组件导致后续安装失败

## A.2 配置必要的系统服务

按以下步骤来配置必要的系统服务：

步骤 1 Linux 安装完成后启动进入窗口界面，以 root 用户登陆。

步骤 2 配置 samba 服务，使 Linux 和 Windows 之间能方便地交换文件。在 FC2 的菜单中可以找到配置 samba 服务的菜单项。打开配置窗口后，首先要建立 samba 用户，然后再添加共享文件夹，就可以在 Windows 下测试是否能正常地访问 samba 服务了。



**步骤 3** 输入 `/etc/init.d/ssh start` 启动 ssh 服务(如果是 FC2,默认已经启动了该服务),在 Windows 下就可以使用 `putty` 等工具登陆服务器。

**步骤 4** 编辑 `/etc/exports` 文件,添加 NFS 目录,输入`/etc/init.d/nfs start` 启动 NFS 服务,就可以在让单板访问服务器的 NFS 文件夹或是直接将服务器的 NFS 文件夹作为单板的根目录启动(调试过程中常用的 NFS 方式)。

**----结束**

至此,一个基本的 Linux 环境已经搭建成功,接下来就可以安装 SDK,其安装过程请参见“《Hi3510 Linux 开发环境用户指南 正文》中 1.3.3 安装 Hi3510 SDK”。



# B 缩略语

## A

ADS	ARM Development Suite	ARM 开发工具套件
ARM	Advanced RISC Machine	ARM 公司指令集

## C

Cramfs	Compressed RAM file system	压缩 RAM 文件系统
--------	----------------------------	-------------

## D

DMS	Digital Media Solution	媒体解决方案平台
-----	------------------------	----------

## E

ELF	Executable and Linkable Format	可执行连接格式文件
-----	--------------------------------	-----------

## G

GCC	GNU Compiler Collection	GNU 编译器集合
gdb	GNU Debugger	GNU 调试器
GNU	GNU's Not UNIX	GNU

## I

IP	Internet Porotocol	Internet 协议
----	--------------------	-------------

## J

JFFS2	Journalling Flash File System v2	一种 Flash 文件系统
-------	----------------------------------	---------------

**B 缩略语**

---

JTAG	Joint Test Action Group	联合测试行动组
<b>N</b>		
NFS	Network File System	网络文件系统
<b>P</b>		
PC	Personal Computer	个人计算机
<b>S</b>		
SDRAM	Synchronous Dynamic random access memory	同步动态随机存储器
SDK	Software Development Kit	软件开发工具集
<b>U</b>		
U-Boot	Universal Boot Loader	操作系统内核运行之前需要运行的引导程序
<b>V</b>		
VSEVB	Video Solution Evaluation Board	视频评估板