Linux 开发环境

用户指南

文档版本　03

发布日期　2015-12-16

商标声明

 、HISILICON、海思和其他海思商标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

您购买的产品、服务或特性等应受海思公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，海思公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，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。

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总部　邮编：518129

网址：　http://www.hisilicon.com

客户服务邮箱：　support@hisilicon.com

前 言

概述

本文档介绍高清 Linux 开发环境。Linux 开发环境的搭建、HiBoot、Linux 内核、根文件系统以及内核和根文件系统的烧写，以及创建网络开发环境和如何启动 Linux 开发应用程序。

本文档主要提供让客户更快地了解 Linux 开发环境指导。

产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本
Hi3798C 芯片	V1XX
Hi3798C	V2XX
Hi3796C	V1XX
Hi3798M	V1XX
Hi3796M	V1XX

读者对象

本文档（本指南）主要适用于以下工程师：

- 技术支持工程师
- 软件开发工程师

作者信息

章节号	章节名称	作者信息
全文	全文	L00227819

修订记录

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。

修订日期	版本	修订说明
2014-06-09	00B01	第 1 次版本发布。
2014-10-30	01	新增支持 Hi3796MV100 芯片。
2015-04-25	02	新增支持 Hi3798CV200 芯片。
2015-12-16	03	刷新兼容 Hi3798CV200 方案。插入第 2 章 Fastboot 配置。

目 录

插图目录

表格目录

1 开发环境

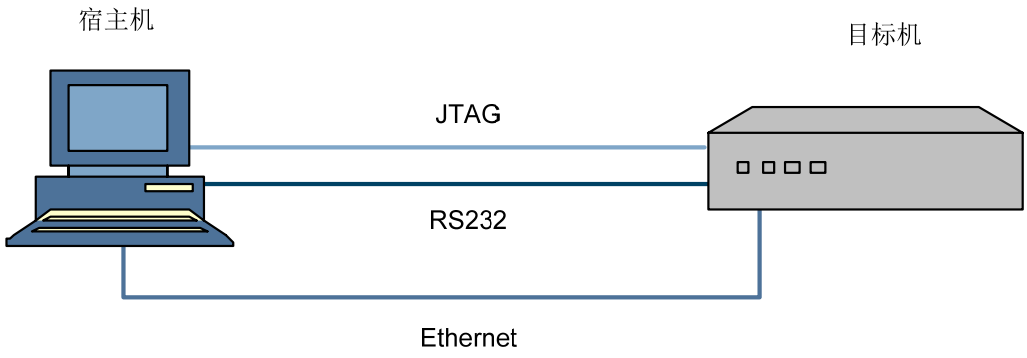
1.1 嵌入式开发环境

嵌入式单板开发，通常采用交叉编译调试的方式进行，即“宿主机＋目标机（评估板）”的形式。宿主机和目标机一般通过串口连接，同时也可以通过网口或者 JTAG 连接，如图 1-1 所示。

宿主机和目标机的处理器一般不相同。宿主机需要建立适合于目标机的交叉编译环境。程序在宿主机上经过“编译－连接－定位”得到可执行文件。把可执行文件烧写到目标机中，在目标机上运行。

目标机上的 Bootloader 启动后，目标机中的操作信息通过串口或者网口输出到宿主机上显示。在宿主机上的控制台中输入命令，可以控制目标机。

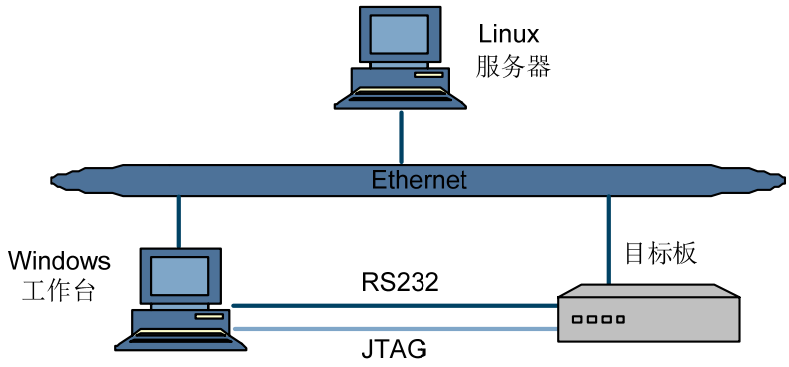
图1-1 嵌入式开发图例



1.2 高清 Linux 开发环境

高清 Linux 开发环境通常包括 Linux 服务器、Windows 工作台和高清 REFB（目标板），三者同处于一个网络中，如图 1-2 所示。

图1-2 高清 Linux 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 与高清 REFB 连接（JTAG 应用于 ADS/RealView Debugger 等软件），开发人员可以在 Windows 工作台中进行程序开发或者远程登录到 Linux 服务器进行程序开发。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 高清 Linux 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/Windows 7。
	应用软件	putty、超级终端（Windows 7 没有自带超级终端，可以使用 AbsoluteTelnet、SecureCRT 等串口管理软件）、tftp 服务器、ADS/RealView Debugger 等软件。
Linux 服务器	操作系统	发行版无特别要求，可为 Redhat、Debian 等。内核版本需为 2.6.9 以上。推荐 ubuntu 10 及以上版本。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（Gcc 版本 4.4）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。
高清 REFB	引导程序	Fastboot3.3.0

软件		描述
	操作系统	Hisilicon Linux（简称 HiLinux）。HiLinux 内核基于 Linux 标准内核 3.18.24 版本移植开发，根文件系统基于 busybox1.24.1 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	高清提供 2 套工具链：32 位工具链 Glibc2.22 对应编译器 gcc 4.9.2 ; 64 位工具链 Glibc2.22 对应编译器 gcc 5.1
	GNU make	请安装 GNU Make 3.81 版本。（选择其它版本的 Make 工具，可能会出现无法预料的编译问题）

1.3 搭建 Linux 开发环境

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。例如：

- RedHat 较新的发行版如 RedHat Fedora Core 系列和 Redhat Enterprise Linux、Red Hat 3.4.4-2
- RedHat 较老的发行版如 RedHat 9.0 等。

推荐使用较新版本，以方便获取各类资源，如 Fedora Core 系列，SUSE 系列，Ubuntu 系列等。

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

建议安装 SDK 配套的工具链，安装步骤请参见 sdk 包中的 install_notes.txt（英文版）或者 install_notes(chs).txt（中文版）文件。

注意

如果使用从网络等渠道得到的交叉编译工具可能与使用的内核并不配套，可能造成开发过程中一些不必要的问题。

1.3.3 安装高清 SDK

高清 SDK 是基于高清 REFB 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。关于安装步骤请参见 sdk 包中的 install_notes.txt（英文版）或者 install_notes(chs).txt（中文版）文件。

2 Boot 配置

说明

本章说明如何配置 SDK，使 Boot 支持多种单板类型。目前只有 Hi3798CV200 非高安芯片支持该功能。

Boot 同时支持多种不同类型的单板，不同类型的单板通过 ADC 电压来区分。目前只有 Hi3798CV200 平台支持该特性，最多支持 6 种不同类型的单板。

SDK 默认发布的单板类型与 ADC 电压的对应关系如表 2-1 所示。

表2-1 SDK 默认单板类型与 ADC 电压的对应关系

类型	Demo 板名称	ADC 电压值
0	Hi3798CV2DMB	3.3V
1	Hi3798CV2DMC	2.475V
2	Hi3798CV2DMD	1.925V
3	TBD	1.375V
4	TBD	0.825V
5	TBD	0V

SDK 中配置 Boot 支持多种类型单板的方法如下：

```
make menuconfig
    ->Board  --->
        ->Boot Regfile Config List  --->
(hi3798cv2dmb_xxx0.reg) Boot Reg File 0 (3.3V) //ADC电压为3.3V，单板类型为0
(hi3798cv2dmc_xxx1.reg) Boot Reg File 1 (2.475V) //ADC电压为2.475V，单板类型为1
(hi3798cv2dmd_xxx2.reg) Boot Reg File 2 (1.925V) //ADC电压为1.925V，单板类型为2
(hi3798cv2dme_xxx3.reg) Boot Reg File 3 (1.375V) //ADC电压为1.375V，单板类型为3
```

```
(hi3798cv2dmf_xxx4.reg) Boot Reg File 4 (0.825V) //ADC电压为0.825V，单板类型为4
(hi3798cv2dmg_xxx5.reg) Boot Reg File 5 (0V) //ADC电压为0，单板类型为5
```

注意

- 单板类型编号从 0~5，与 ADC 电压对应，不能随意更改；
- 最多可以支持 6 种不同的单板类型；
- 如果支持的单板类型少于 6 种，只需要把对应的表格配置项留空不配置即可；
- 高安芯片不支持此功能

3 Linux 内核

3.1 内核源代码

内核源代码已存放于 SDK 目录下的 source/kernel 目录下，以压缩包加 patch 的形式存放（以 HiSTBLinuxV100R005 为例，以下如不作特殊说明，均如此处理），用户可直接进入目录进行相关操作。

3.2 配置内核

SDK 针对每个高清平台的芯片都有默认的内核配置，建议使用默认的内核配置，可以在 SDK 的 make menuconfig->Kernel 菜单下选择不同芯片的配置。

用户也可以根据自身需求修改默认的配置。

注意

SDK 默认提供的内核配置针对高清平台做了优化和适配，修改默认配置，有可能导致一些组件无法运行，如果不确定修改是否可行，请咨询海思 FAE！

修改内核配置的方法（以配置 hi3798cv200_defconfig，编译器 arm-histbv310-linux-gcc 为例）：

```
Hisilicon#cd source/kernel/linux-3.18.y
Hisilicon#make ARCH=arm CROSS_COMPILE=arm-histbv310-linux-
hi3798cv200_defconfig
Hisilicon# make ARCH=arm CROSS_COMPILE=arm-histbv310-linux- menuconfig
Hisilicon#cp .config arch/arm/configs/hi3798cv200_defconfig
```

然后就可以在 SDK 的 make menuconfig->Kernel 下选择此配置。

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

保存配置后，在 SDK 根目录分别执行以下命令来编译内核：

```
make linux;
make linux_install
```

编译完成后，生成的内核放在 SDK 下的 pub/image/hi_kernel.bin。

如果要清除编译生成的中间文件：

```
make linux_clean;
```

如果用户希望在内核源码目录编译内核，可以执行以下命令来编译内核：

```
cd source/kernel/linux-3.18.y
make ARCH=arm CROSS_COMPILE=arm-histbv310-linux- uImage
```

生成的镜像名为 uImage，位于 linux-3.18.y/arch/arm/boot 目录下

说明

如果编译过程中出现错误，可执行“make ARCH=arm CROSS_COMPILE=arm-histbv310-linux-clean”，然后重新运行“make ARCH=arm CROSS_COMPILE=arm-histbv310-linux-menuconfig”，加载配置文件，最后执行“make ARCH=arm CROSS_COMPILE=arm-histbv310-linux- uImage。

4 根文件系统

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。

根文件系统通常存放于内存和 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

通用的 Linux 系统的根文件系统中会包括根文件系统顶层目录结构图中所有的目录，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，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，如果没有特殊原因，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 SDK 后，busybox 完整源代码就存放在 source/rootfs/busybox/busybox-1.24.1 目录中了。要获取 busybox 源代码也可以从网站 http://www.busybox.net 下载。

4.2.2 配置 busybox

进入 busybox 所在目录，进行配置操作需要输入如下命令（假设默认的 busybox 配置文件为 xxx.config）：

```
hisilicon$cd source/rootfs/busybox/
hisilicon$tar -xf busybox-1.24.1.tar.bz2
hisilicon$cd busybox-1.24.1
hisilicon$cp ../busybox-1.24.1.config/xxx.config .config
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 Busybox Settings ---> Build Options 中注意下面两个选项：

```
[]Build BusyBox as a static binary (no shared libs)
(arm-XXXX-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，但需要动态库的支持。默认编译成动态链接的 busybox。

第二个选项是选择交叉编译器，并配置交叉编译器为 arm-XXXX-linux- (编译器名称以发布版本中名称为准)。配置好后保存并退出。

欲了解 busybox 各选项含义请参考 busybox 配置帮助。

4.2.3 编译和安装 busybox

编译和安装 busybox 的具体操作如下：

```
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
```

注意

海思发布的 busybox **支持在** GNU Make 3.81 **版本上编译，若不是** GNU Make 3.81 **版本，无法保证能正确编译。**

4.2.4 制作根文件系统

成功安装 SDK 后，在 pub/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 source/rootfs/busybox/ 目录下。

制作根文件系统的具体操作步骤如下：

步骤 1 hisilicon$mkdir rootbox

```
hisilicon$cd rootbox
hisilicon$cp –R source/rootfs/busybox/busybox-1.24.1/_intsall/* .
hisilicon$mkdir etc dev lib lib64 tmp var mnt home proc
```

步骤 2 配置 etc、lib、dev 目录的必需文件。

1. etc 目录可参考系统/etc 下的文件。其中最主要的文件包括 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，根据需要自行修改。
2. 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。
3. lib 目录是存放 32 位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。

4. lib64 目录是存放 64 位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。同时，lib 目录要添加 64 位动态加载器软链接 ld-linux-aarch64.so.1 -> ../lib64/ld-2.22.so

----结束

完成以上两个步骤，一个完整的根文件系统就生成了。

说明

SDK 软件包中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

嵌入式系统中常用文件系统包括有 squashfs、cramfs、JFFS2、NFS、yaffs2 以及 ext4。它们的特点如下：

- cramfs 和 JFFS2 具有好的空间特性，很适合嵌入式产品应用。
- cramfs 和 squashfs 为只读文件系统。
- JFFS2 为可读写文件系统。
- NFS 文件系统适用于开发初期的调试阶段，为网络文件系统。
- yaffs2 文件系统只用于 NAND Flash。

说明

JFFS2 **文件系统**、cramfs **文件系统和** squashfs **文件系统主要用于** spi flash； yaffs2 **只能用于** nand，ext4 **一般用于** eMMC。

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，使用简单，加载容易，速度快。

cramfs 的优缺点如下：

- 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内存中，在一定程度上降低读取效率。另外 cramfs 文件系统是只读的。

如果想要在单板运行的 Linux 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”，选择“miscellaneous filesystems”，最后选中其中的“Compressed ROM file system support”。

mkfs.cramfs 是用来制作 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，cramfs-root.img 是生成的 cramfs 文件系统映像文件。

4.3.2 squashfs

squashfs 文件系统也是一种压缩的只读文件系统。与 cramfs 相比，能提供更大的压缩比，支持更大的镜像和文件。

squashfs 文件系统的支持需要配置内核（配置方法请参见 2.2 配置内核）：

```
File systems  --->
  [*] Miscellaneous filesystems  --->
    <*>  SquashFS 4.0 - Squashed file system support  //支持squashfs文件系统
      [*]    Include support for XZ compressed file systems //如果是XZ算法压缩的squashfs文件系统，则需要选中。
```

mksquashfs 工具是用来制作 squashfs 文件系统镜像的工具，命令如下）：

```
hisilicon$mksquashfs ./rootbox rootbox.squashfs -no-fragments -noappend -comp xz
```

其中，rootbox 是要文件系统目录，rootbox.squashfs 是要制作的文件系统镜像名称。

4.3.3 JFFS2

JFFS2 是 RedHat 的 David Woodhouse 在 JFFS 基础上改进的文件系统，是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。

JFFS2 的优缺点如下：

- 优点：使用了压缩的文件格式。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 运行速度会迅速降低。这是因为垃圾收集的问题。

加载 JFFS2 文件系统时的步骤如下：

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”，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 工具可以从互联网中下载，也可以在 SDK 包中找到。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
l	little-endian 小端模式
e	Flash 的块大小
o	输出映像文件

4.3.4 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 initramfs 作为根文件系统，系统启动之后再挂在 NFS 文件系统，这个过程不需要任何对 Flash 的操作，修改应用程序完全在 Linux 服务器中进行，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：编辑/etc/exports 配置文件，添加路径及参数，然后执行/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。

在 Linux 服务器上配置好 NFS 根文件系统后，在单板侧挂载 NFS 文件系统，具体操作如下所示：

步骤 1 制作 initramfs 镜像并启动系统

initramfs 文件系统需要内核的支持，需要配置内核（配置方法请参见 2.2 配置内核）：

```
General setup  --->
```

```
    [ ] Initial RAM filesystem and RAM disk (initramfs/initrd) support
```

在上面的方括号中输入文件系统目录的绝对路径。

然后编译内核，编译方法请参见 2.3 编译内核。这个内核镜像就是包含了文件系统的镜像，将这个镜像下载到 ddr 或者烧写到 flash 中就可以启动系统，不需要烧写额外的文件系统镜像。

步骤 2 挂载 NFS 文件系统

系统启动之后，使用命令挂载 NFS 文件系统：

```
mount -t nfs -o nolock,tcp, intr,soft,timeo=10,retrans=3
xx:xx:xx:xx:/rootfs_dir /mnt
```

然后就可以在/mnt 目录下进行开发了，上述 mount 选项的意思是以 tcp 方式挂载，允许 CTRL+C 键中断访问/mnt 目录的操作，如果操作阻塞超过 3s，则返回出错，在网络状态较差的情况下，这些选项可以避免在/mnt 目录下执行 ls 等操作时 shell 挂死的问题。

----结束

4.3.5 yaffs2

yaffs2 是专门为 NANDFlash 设计的嵌入式文件系统。它是日志结构的文件系统，提供了损耗平衡和掉电保护，可以有效地避免意外掉电对文件系统一致性和完整性的影响。

yaffs2 的优缺点如下：

- 优点
  - 专门针对 NANDFlash，软件结构得到优化，速度快。
  - 使用硬件的 spare area 区域存储文件组织信息，启动时只需扫描组织信息，启动比较快。
  - 采用多策略垃圾回收算法，能够提高垃圾回收的效率和公平性，达到损耗平衡的目的。
- 缺点

  没有采用压缩的文件格式。包含同样内容，yaffs2 镜像文件要比 jffs2 镜像文件大。

yaffs2 文件系统在 SDK 中是作为一个模块提供的。只需在 yaffs2 代码中的 Makefile 中加入所依赖的内核代码路径，进行编译，即可生成 yaffs2 文件系统模块。

yaffs2 镜像文件的制作和 cramfs 是一样的，也是通过工具来制作，可以直接输入命令 mkfs.yaffs2，会出现参数的相关说明，具体如下：

```
hisilicon $ mkfs.yaffs2
mkyaffs2image: image building tool for YAFFS2 built Dec 25 2009
usage: mkyaffs2image dir image_file [convert]
       dir      the directory tree to be converted
       image_file       the output file to hold the image
       pagesize         the nand flash's pagesize.value in [2048,4096,8192]
```

```
is allowed.
      ecc_type        the ecc_type you will use in hisilicon nand flash
driver: hinand.
                     0:no ecc,1:1bit ecc, 2:4bit ecc, 3:8bit ecc, 4:24bit
ecc for 1k, 5:24bit ecc for 512.
      'convert'       produce a big-endian image from a little-endian
machine
```

举例：

```
hisilicon$ mkfs.yaffs2 ./rootfs/ ./images/rootfs_2k_1bit.yaffs2 2048 1
```

其中，rootbox 是之前已经制作好的根文件系统，2k_1bit.yaffs2 是生成的 yaffs2 文件系统镜像文件。2048 表示页大小，1 表示 ECC 类型。

5 烧写内核和根文件系统

高清 REFB 包含 DDR 存储器和 Flash 存储器。DDR 的地址空间从 0x00000000 开始；具体大小随单板不同，可从单板硬件手册中获取。

具体内容请参见《Hi379X V100 样机 用户指南》。

Boot、内核与根文件系统的烧写方法请参照《HiBurn 工具快速入门视频 V1.0.exe》。

6 应用程序开发简介

6.1 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 环境下使用 Source Insight，在 Linux 环境下使用 Vim+ctags+cscope，功能也相当强大。

6.2 编译选项

6.2.1 使用 VFP

使用下面的编译选项，可以生成硬浮点指令，通过 FPU（Floating Point Unit）模块完成浮点运算。

```
-mfloat-abi=softfp -mfpu=vfpv3-d16
```

示例：

```
CFLAGS+=-march=armv7-a -mcpu=cortex-a9 -mfloat-abi=softfp -mfpu=vfpv3-d16
```

说明

推荐使用 VFP（Vector Floating-point），兼容性好。

6.2.2 使用 NEON

使用下面的编译选项，可以生成 NEON 指令，通过 NEON 模块完成浮点运算。

```
-mfloat-abi=softfp -mfpu=neon
```

示例：

```
CFLAGS+=-march=armv7-a -mcpu=cortex-a9 -mfloat-abi=softfp -mfpu=neon
```

说明

只有在 CPU **带** NEON **模块，且内核支持** NEON **的情况下，才可以使用** NEON **编译选项，否则会报指令异常错误。**

- 如何查看内核是否支持 NEON

在单板上使用命令：

```
cat /proc/cpuinfo
```

如果其中的“Features”行含有“neon”，则表示支持 NEON。

```
Features        : swp half thumb fastmult vfp edsp neon vfpv3 tls
```

- 如何查看镜像是否使用了 NEON

  在 linux 服务器上使用命令：

  ```
  arm-histbv310-linux-readelf -A a.out
  ```

  如果其中含有“Tag_Advanced_SIMD_arch: NEONv1”，则表示使用了 NEON。

  ```
  Tag_FP_arch: VFPv3
  Tag_Advanced_SIMD_arch: NEONv1
  Tag_ABI_PCS_wchar_t: 4
  ```

6.3 运行应用程序

要运行编译好的应用程序，首先需要将其添加到目标机中，必须完成以下工作：

- 将应用程序和需要的库文件（如果有）等添加到目标机的根文件系统相应的目录中。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“4.3.4 NFS”），然后将 NFS 目录挂载到 JFFS2 文件系统目录中，操作方法如下：

```
mount -t nfs -o nolock serverip:path /mnt
```

其中 serverip 表示 NFS 目录所在服务器的 ip，path 表示 NFS 目录在服务器上的路径，以后只需要简单的拷贝应用程序到 NFS 系统目录中，就可以在目标机里运行。

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（0x34200000）（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。

说明

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，添加需要启动的用程序路径。

6.4 使用 gdbserver 调试应用程序

在很多情况下，需要对应用程序进行调试。在 Linux 下调试程序，常用的工具是 gdb。由于嵌入式单板的资源有限，一般不直接在目标机上运行 gdb 进行调试，而是采取 gdb+gdbserver 的方式。gdbserver 在目标机中运行，gdb 则在宿主机上运行。根文件系统中已经包含 gdbserver。使用 gdbserver 调试应用程序的步骤如下所示：

步骤 1 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机的 2000 端口开启了一个调试进程，**hello** 就是要调试的程序。

步骤 2 在 Linux 服务器上启动 gdb 程序，因为目标机为 ARM，所以启动 arm-xxxx-gdb(以发布版本真实名称为准)。

步骤 3 启动 arm-xxx-gdb 后，在命令提示符状态下输入命令，与目标机进行连接。

```
(gdb) target remote 192.168.0.5:2000  /*192.168.0.5为单板IP*/
```

注意

端口号和目标机开启的端口号要一致。

步骤 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 ?? ()
```

步骤 5 进行符号文件加载：

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

或者使用：

```
(gdb) file hello
```

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

----结束

7 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。

7.1 安装 Linux 系统的配置选项

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 1.3.1 安装 Linux 服务器进行 Linux 系统的安装，安装中应注意的配置选项如表 7-1 所示。

表7-1 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而产生乱码。
磁盘分区	自动分区	留出足够的磁盘空间供安装 SDK。
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	避免因为缺少必要的组件导致后续安装失败。

7.2 配置必要的系统服务

配置必要的系统服务的步骤如下：

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 输入**/etc/init.d/nfs start** 启动 NFS 服务，编辑/etc/exports 文件，添加 NFS 目录，就可以将单板访问服务器的 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“1.3.3 安装高清 SDK”。

7.3 工具对 Linux PC 机内核版本要求参考

表 7-2 中，最低内核版本和最高内核版本是指测试过的，工具能在其上正常编译运行的内核版本环境，Linux 内核版本比较多，内核版本是否支持相关工具运行，以实际编译运行结果为准，此表格只提供一个参考。

表7-2 工具和内核版本兼容表

工具	最低内核版本	最高内核版本
UBI 工具 mkfs.ubifs 编译运行环境	Linux-2.6.22,	Linux-2.6.32

A 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 Complier 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
IPTV	Internet Prototocol Television	网络电视

J

JFFS2	Journalling FLASH File System v2	一种 Flash 文件系统

JTAG	Joint Test Action Group	联合测试行动组
N		
NFS	Network File System	网络文件系统
P		
PC	Personal Computer	个人计算机
S		
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
V		
VFP	Vector Floating-point	ARM 浮点架构