第5章 嵌入式 Linux 在 EBD9200 上的开发

内容点击

本章我们首先了解嵌入式 linux，介绍在 EBD9200 上进行 Linux 开发，我们需要什么，如何去做？如何下载？

1．应该对嵌入式 Linux 的开发有个大致的了解

当然，采用 Linux 进行 EBD9200 的开发，你应该首先对 Linux 有个大致的了解，如果你具备嵌入式 Linux 的开发经验，可以跳过本节，否则，你需要了解下面几个内容：

- 了解 Linux 操作系统的一些基本内容，比如：linux 的启动过程，Linux 的源码和开放性、Linux 的命令，Linux 的开发工具和环境；
- 你尝试在 Linux 下进行 C 语言程序的简单开发，比如“Hello World”程序，熟悉 Linux 的编译工具 gcc，Makefile 文件等等，如果可以进行 gdb 调试更好；
- 你了解了 Linux 后，应该了解嵌入式 Linux 开发是宿主机—目标机（HOST-TARGET）交叉开发，系统编译工具要换成交叉编译工具，建立交叉编译环境；
- 了解嵌入式 Linux 系统下的 BootLoader；
- 目标板各种资源在 Linux 下的驱动程序；
- Linux 的内核和文件系统的裁剪和编译；
- 当然最后一点就是如何下载内核了。

我们在此主要介绍交叉编译环境，BootLoader、Linux 内核、文件系统与下载。

1．1 了解 BootLoader

在 EBD9200 上，我们采用的是 U-BOOT，这部分内容我们在第 3 章已经进行了介绍。U-BOOT 在嵌入式系统中相当于 PC 机的 BIOS 加上操作系统引导头部的内容 ，并且引导操作系统进行装载和运行，U-BOOT 启动后有一系列的命令，使得我们能够方便地对 FLASH、RAM 进行操作，U-BOOT 已经对系统的频率、定时器进行了设置，初始化了一个调试串口，我们可以通过串口或者以太网进行数据的下载。

```
go          - 在地址 'addr' 处开始程序执行
run         - 运行命令
bootm       - 从内存中进行应用程序影象运行
bootp       - 通过网络用 BootP/TFTP 协议来启动影象
tftpboot    - 通过网络用 TFTP 协议、设置服务器和客户机
                的 IP 地址进行影象文件传送
loadb       - 通过串口线(kermit mode) 来装载二进制文件
printenv    - 打印环境变量
setenv      - 设置环境变量
saveenv     - 保存环境变量到内存
```

下面是 U-BOOT 中的简单环境变量。

baudrate 波特率

bootdelay　boot 延迟
bootcmd　Boot 命令
bootargs　Boot 参数
bootfile
ipaddr　客户机 IP 地址
serverip　服务器地址
loadaddr　装载地址
ethaddr　网卡 MAC 地址

如果想了解更为详细的情况，请参照第 3 章的内容和详细的英文文档。

1．2 常用的命令

- go - start application at address 'addr'
- run - run commands in an environment variable
- bootm - boot application image from memory
- bootp - boot image via network using BootP/TFTP protocol
- tftpboot- boot image via network using TFTP protocol
- and env variables "ipaddr" and "serverip"
- (and eventually "gatewayip")
- rarpboot- boot image via network using RARP/TFTP protocol
- diskboot- boot from IDE devicebootd - boot default, i.e., run 'bootcmd'
- loads - load S-Record file over serial line
- loadb - load binary file over serial line (kermit mode)
- md - memory display
- mm - memory modify (auto-incrementing)
- nm - memory modify (constant address)
- mw - memory write (fill)
- cp - memory copy
- cmp - memory compare
- crc32 - checksum calculation
- imd - i2c memory display
- imm - i2c memory modify (auto-incrementing)
- inm - i2c memory modify (constant address)
- imw - i2c memory write (fill)
- icrc32 - i2c checksum calculation
- iprobe - probe to discover valid I2C chip addresses
- iloop - infinite loop on address range
- isdram - print SDRAM configuration information
- sspi - SPI utility commands
- base- print or set address offset
- printenv- print environment variables
- setenv - set environment variables
- saveenv - save environment variables to persistent storage
- protect - enable or disable FLASH write protection
- erase - erase FLASH memory

- flinfo - print FLASH memory information
- bdinfo - print Board Info structure
- iminfo - print header information for application image
- coninfo - print console devices and informations
- ide - IDE sub-system
- loop- infinite loop on address range
- mtest   - simple RAM test
- icache - enable or disable instruction cache
- dcache - enable or disable data cache
- reset   - Perform RESET of the CPU
- echo- echo args to console
- version - print monitor version
- help- print online help
- ?   - alias for 'help'

1．3 交叉开发环境的建立

你可以有两种选择，一种是采用已经编译好的开发环境，你只需要解开到安装的位置，另一种就是自己下载原码、编译和建立开发环境，对于注重于应用层的工程师，建议采用第一种方法即可，因为一般开发商都帮你提供好了。

1．3．1 安装预先编译好的开发环境

你可以在 EBD9200 的光盘的目录(LINUX 开发工具)中得到预先编译的交叉开发环境 cross-2.95.3.tar.bz2，当然你也可以从网站 ftp://ftp.arm.linux.org.uk/pub/armlinux/toolchain/上下载，不过需要编译了，网站上一般提供原码 ，主要包含下面内容：

- Bin utilities
- ARM linux C compiler and linker
- glibc Library
- ARM linux C++ compiler

首先要在 PC 机的 LINUX 上安装好 cross-2.95.3.tar.bz2，英贝德建议的安装位置在/usr/local/下，你可以安装到任何地方，但是要记住位置。在/usr/local 下建个 arm 的文件夹，把 cross-2.95.3.tar.bz2 复制到 arm 文件下，解压就成了。（建议以 root 用户登录）

```
$ cd /usr/local
$ cd   arm                                  进入 ARM 目录
$ tar   -jxvf   cross-2.95.3.tar.bz2          解开
```

当然也可以简单用图形界面完成这一切，只要把原码压缩包解开即可。

这样你的开发环境已经建立在/usr/local/arm/2.95.3/bin 下面，当你使用的时候，应该指出编译器的位置。

1．3．2 自己制作交叉开发环境

你需要自己下载原码，并且阅读其中的 readme 和 makefile，按照过程自己进行编译，

最后安装到具体位置。这样的文章较多，此处不作具体介绍。

1．4 内核与文件系统

当你完成上面两步时，你所要做的就是对内核和文件系统进行配置了，这两部分是嵌入式 Linux 的核心内容，我们放到后面章节做详细介绍，本章后面部分，我们将对编译好的内核与文件系统映像下载到 9200 板的操作步骤做详细介绍。

2．嵌入式 Linux 在 EBD9200 上的装载

本章我们介绍两种装载和启动模式，网络服务器的启动模式和 FLASH 启动模式，前者应用于网络模式，内核与文件系统的映象文件存放在 PC 服务器上，系统加电或复位后，通过 TFTP 协议把操作系统的内核与文件系统直接传输到系统的 SDRAM 中执行；后者的内核与文件系统放在 FLASH 中，加电与复位后，从 FLASH 中执行，属于单机模式。

2．1 网络服务器的启动模式

首先你的 U-BOOT 已经正常启动，这时我们应该从光盘上找到 Linux 的内核与文件系统的映象(ulmage 和 ramdisk)，（**路径**：EBD9200 企业板（刻录 I 型板新）\EBD9200-I 企业板光盘\linux 编译好的文件\tftpboot）然后再向下进行。

2.1.1 准备工作

设置硬件

- 连接串口：将串口线连接到 PC 机的串口和 EBD9200 的 P1 口上。
- 连接网线，采用双机相连网线直接相连，或者采用 HUB 相连。

设置 PC 机(在 Windows 环境下)

- 打开超级终端。并设置串口（115200、8、无、1、无），
- 建立 tftpboot 目录、把 tftpd.exe 拷贝到该目录下。
- 把内核（ulmage）和文件系统(ramdisk)的映象拷贝到 tftpboot 的目录下。
- 一定要关掉你的 PC 主机的防火墙（如果你的机子装了防火墙）。

2.1.2 进行下载

有两种方式可以进行下载，串口模式和以太网模式。

2.1.2.1 串口模式(不建议使用，速度太慢)

Uboot> loadb 21100000

通过 kermit 模式下载文件系统（ramdisk）

Uboot> loadb 21000000

通过 kermit 模式下载内核（ulmage）

Uboot> bootm 21000000 ;从 21000000 处开始运行

2.1.2.2 以太网模式

STEP 1：进行网络参数设置

U-Boot > setenv ethaddr 12:34:56:78:99:aa ；MAC 地址设置

U-Boot > setenv ipaddr IP 地址（缺省 192.168.0.2） ；目标板 IP 地址

U-Boot > setenv serverip 服务器地址（缺省 192.168.0.111）；一般是 PC 主机 IP 地址

U-Boot > setenv bootdelay 3 ；延时

U-Boot >setenv bootargs root=/dev/ram rw initrd=0x21100000,8000000 ramdisk_size=20000 console=ttyS0,115200,mem=32M

U-Boot > saveenv ；可以进行保存，如果不保存，掉电后需要重新设置

STEP 2：下载

U-Boot >tftp 21000000 ulmage ；下载内核
U-Boot >tftp 21100000 ramdisk ；下载文件系统
U-Boot >bootm 21000000 ；开始运行

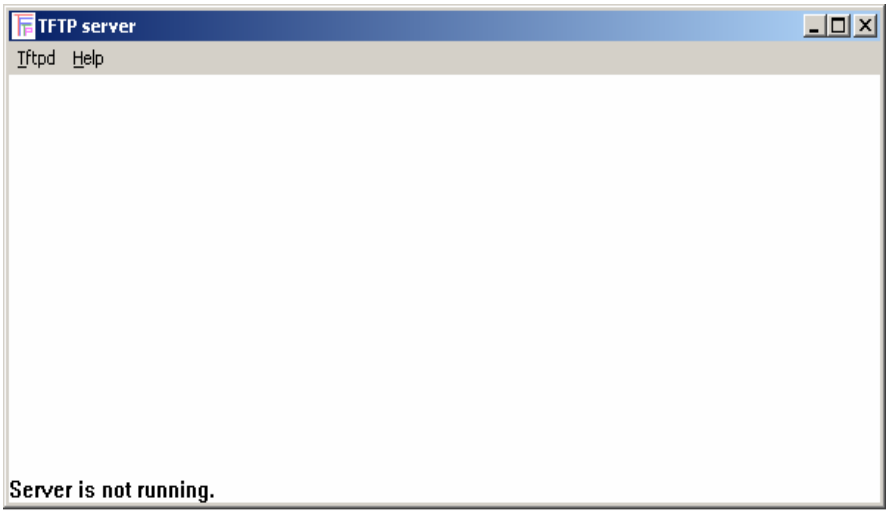
2.1.3 设置系统自动运行参数

U-Boot >setenv bootcmd tftp 21000000 ulmage\; tftp 21100000 ramdisk \;bootm 21000000
U-Boot > saveenv

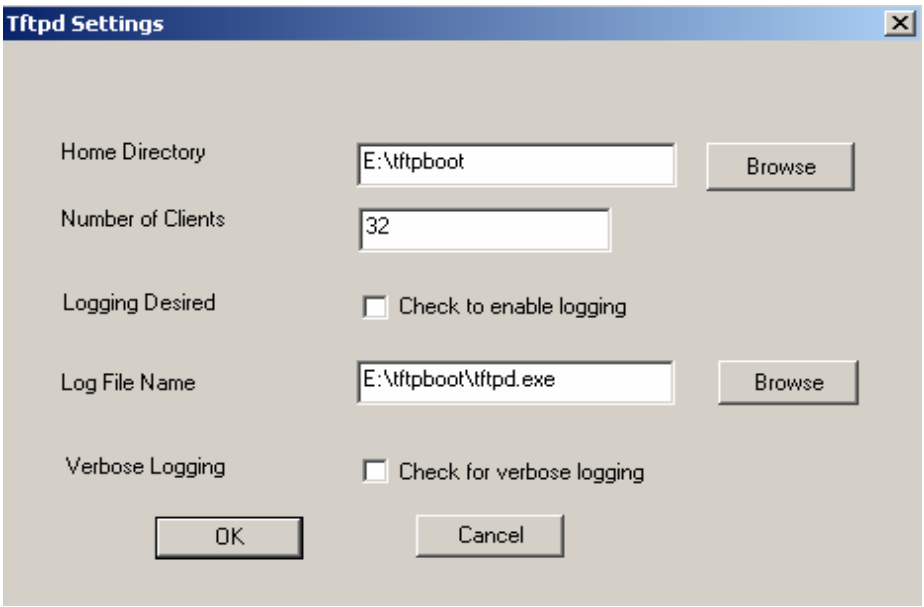
2.1.4 系统下载和运行过程

STEP 1：

到 tftpboo 目录，运行 tftpd.exe，出现。

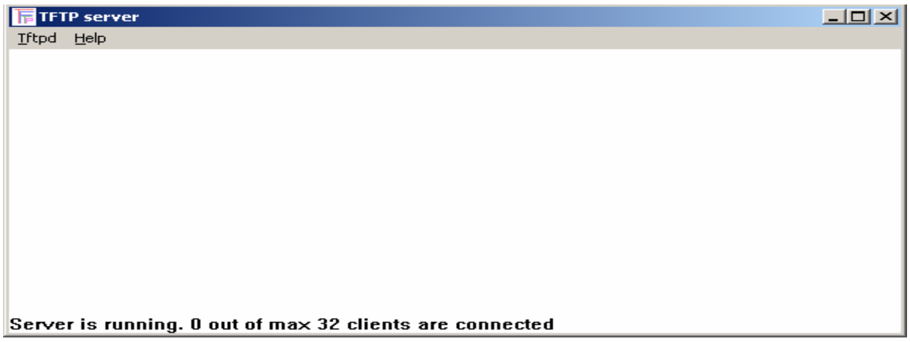


接着 选择 Tftpd->Configure,设置内核映射文件所在的路径。



点击 OK，接着选择 Tftpd->Start。

tftp 已经启动：



STEP 2：

下载内核，U-Boot > tftp 21000000 ulmage ：

```
U-Boot 1.1.1 (Nov 16 2004 - 18:01:43)

U-Boot code: 21F00000 -> 21F16F2C  BSS: -> 21F1B368
RAM Configuration:
Bank #0: 20000000 32 MB
Flash: 16 MB
In:    serial
Out:   serial
Err:   serial
Hit any key to stop autoboot:  0
Uboot>
Uboot> tftp 21000000 ulmage
TFTP from server 192.168.0.111; our IP address is 192.168.0.2
Filename 'ulmage'.
Load address: 0x21000000
Loading: #################################################################
         #################################################################
         ####################################
done
Bytes transferred = 848268 (cf18c hex)
Uboot> _
```

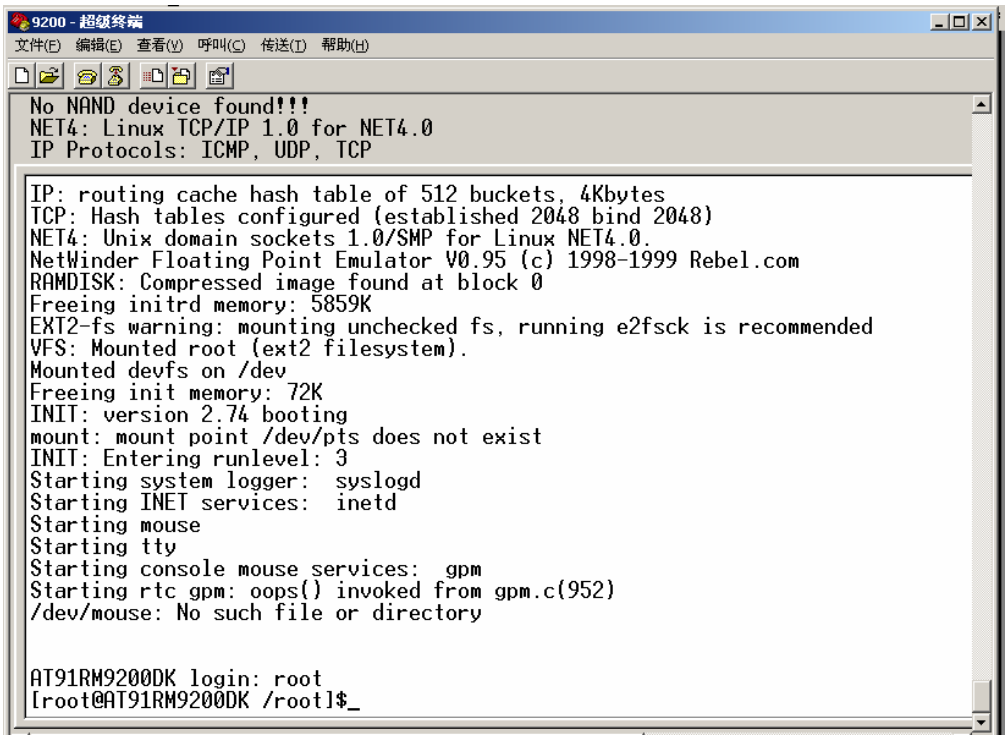
STEP 3：

下载文件系统 ramdisk，U-Boot > tftp 21100000 ramdisk ：

```
Uboot> tftp 21100000 ramdisk
TFTP from server 192.168.0.111; our IP address is 192.168.0.2
Filename 'ramdisk'.
Load address: 0x21100000
Loading: #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         ##############################################################
done
Bytes transferred = 6969703 (6a5967 hex)
Uboot>
```

STEP 4：

运行，U-Boot > bootm 21000000 ：



Linux 系统已经正常启动，可以对板子上的目录结构进行操作了。

2．2 FLASH 启动模式

当然对于一个 FLASH 启动的系统我们必须做两件事情，第一，我们如何下载操作系统到 FLASH 中；第二，操作系统如何能够正确执行。在此，我们根据系统已经烧写好的系统步骤来进行。

STEP 1：

移植烧写好 boot.bin（0x10000000）和 u-boot.gz (10010000) 。（前面已讲过）

STEP 2：设置环境变量（TFTP 协议和网络部分）

UBoot > setenv ethaddr 12:34:56:78:99:aa ； MAC 地址设置

UBoot > setenv ipaddr IP 地址（缺省 192.168.0.2） ； 目标板 IP 地址

UBoot > setenv serverip 服务器地址（缺省 192.168.0.111） ； 服务器 IP 地址

UBoot > setenv bootdelay 3 ； 延时

UBoot > saveenv ；保存网络设置变量

STEP 3：传输内核并烧写入 FLASH（0x10020000）

UBoot > tftp 21000000 ulmage

UBoot > cp.b 21000000 10020000 ulmage _size（TFTP 传输完成后出现的文件大小）

STEP 4：传输文件系统并烧写入 FLASH（0x10100000）

UBoot > tftp 21100000 ramdisk

UBoot > cp.b 21100000 10100000 ramdisk_size（TFTP 传输完成后出现的文件大小）

STEP 4：设置文件系统的启动参数

UBoot >setenv bootargs root=/dev/ram rw initrd=0x21100000,8000000 ramdisk_size=20000 console=ttyS0,115200,mem=32M

UBoot >saveenv ；保存文件系统调用

STEP 5：设置自动启动命令

UBoot >setenv bootcmd cp.b 10020000 21000000 ulmage_size \;cp.b 10100000 21100000 ramdisk_size \;bootm 21000000

UBoot >saveenv ；保存自动启动命令

STEP 6：系统复位和自动运行

重新复位，系统自动运行

3．到此时，你所想和需要的是什么？

你的 Linux 系统已经启动了，系统的硬件部分你也比较清楚了，同时你也可以随心所欲地运行其中的各种程序，但是此时你可能还有很多的疑惑之处，可能你会提出下面具体的问题。

我的内核如何来配置和编译？如何裁剪？

我的文件系统如何建立？

我的顶层程序如何编写和编译，如何加入到我的文件系统之中？

图形和网络浏览器如何使用？

……

其实，你不需要着急，你所想的也是大多数工程师想的，在后面的章节中，我们会一一解决这些问题。