# Linux 系统在 S3C2440 上移植的分析与实现①

李佳佳, 马永杰

(西北师范大学 物理与电子工程学院, 兰州 730070)

**摘 要**: 在分析 Linux 系统结构的基础上, 描述了将完整的 Linux 系统向 S3C2440 平台上移植的关键步骤和过程, 并详细介绍了如何建立属于自己的交叉编译环境和制作一个纯 yaffs2 文件系统的方法. 经过对 Bootloader、内核以及根文件系统的编译和烧写, 实验结果表明, 本文采取的方法, 成功地实现了一个 Linux 系统向 S3C2440 平台上的移植.

**关键词**: S3C2440; 嵌入式 Linux; 系统移植; 交叉编译环境; yaffs2 文件系统

## Analyzing and Porting of Linux System Based on S3C2440

LI Jia-Jia, MA Yong-Jie

(College of Physics and electronic Engineering, Northwest Normal University, Lanzhou 730070, China)

**Abstract**: Based on analyzing the structure of linux system, the paper describes the key steps of transplanting a complete linux system to S3C2440 platform and also detailed introduces how to establish your own cross-compiling environment and the method of making a pure yaffs2 file system. Through compiling and delivering bootloader, kernel and root file system, the experimental results show that it implements the transplantation of linux system to S3C2440 platform successfully.

**Key words**: S3C2440; embeded Linux; porting system; cross-compiling environment; yaffs2 file system

在信息技术和网络技术日新月异的今天, 嵌入式系统因为结构小巧、稳定性高、灵活方便等诸多优点, 在工业、国防科研以及日常生活领域得到了广泛的应用, 并对各个行业的技术升级、产品性能提升、生产效率提高起到了重要的推动作用. 而 Linux 系统因为开源、可裁剪和修改、稳定性高、强大的网络支持等优点使得在嵌入式设备上移植 Linux 系统成为最佳选择[6]. 本文给出了以三星 S3C2440 开发板(ARM920T 内核)作为硬件平台移植一个完整的 Linux 系统(内核版本为 2.6.30.4)的方法及其实现过程.

## 1 Linux的系统结构分析

Linux 的系统软件由三部分组成: Bootloader、操作系统和应用程序, 如图 1 所示[7].

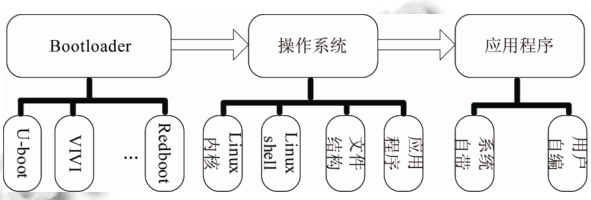


图 1 Linux 系统的软件组成

## 2 搭建自己的交叉编译环境

本文选择通过 crosstool 工具来实现一次编译生成交叉编译工具链, 该方法相对于通过分布编译, 或者安装编译工具所需的库和源代码来生成交叉编译工具链的方法, 其实现过程更简单, 耗时更少. 然而直接使用网上制作好的工具链, 虽然简单省事但其灵活性远远不及自己定制的工具链, 甚至有时出现编译器版

① 基金项目:甘肃省省属高校 2011 年度基本科研业务费专项基金项目
收稿时间:2012-03-01;收到修改稿时间:2012-03-31

本与编译程序不符的情况.

### 2.1 准备源文件

首先从网站：http://kegel.com/crosstool, ftp.kernel.org，ftp.gnu.org 下载所需资源文件 crosstool-0.43.tar.gz、linux-2.6.30.4.tar.gz、binutils-2.22.tar.bz2、gcc-4.3.3.tar. gz、glibc-2.8.tar.gz 、 glibc-linuxthreads-2.3.tar.gz 和 linux-libc-headers-2.6.12.0.tar.bz2. 然后将这些文件放在新建目录/home/dejaneli/downloads 目录下(其中/dejaneli 是普通用户的主目录), 最后通过命令: #tar zxvf cross tool-0.43.tar.gz 将 cross-0.43.tar.gz 解压在/home/dejaneli 目录下[5].

### 2.2 建立自己的编译脚本

以目录/home/dejaneli/crosstool-0.43/下的 demo-arm.sh 文件为模板, 将其复制并改名为 dejaneli-arm.sh, 修改该脚本, 具体操作如下:

```
#!/bin/sh
set－ex
TARBALLS_DIR=$HOME$/downloads
#定义工具链所存放的位置
RESULT_TOP=/home/dejaneli/crosstool
#定义工具链的安装目录
export TARBALLS_DIR RESULT_TOP
GCC_LANGUAGES=”c,c++”
#定义支持 c, c++语言
export GCC_LANGUAGES
mkdir－p $RESULT_TOP
#创建/home/dejaneli/crosstool 目录
Eval‘cat arm.dat gcc-4.3.3-glibc-2.8.dat’sh all.sh－notest
#编译工具链
echo Done
```

### 2.3 建立配置文件

首先在 crosstool-0.43 目录下编辑 arm.dat 和 gcc-4.3.3-glibc-2.8.dat, 它们主要用于定义配置文件和定义生成编译工具链的名称以及编译选项等. 然后修改 arm.dat 中的 “TARGET=arm-none-unkonwn-linux” 为 “TARGET=arm-linux”, 而 gcc-4.3.3-glibc-2.8.dat 的内容不作修改.

### 2.4 执行脚本文件

将 crosstool 的脚本文件和配置文件准备好之后，开始执行 arm.sh 脚本来编译交叉编译工具，命令如下:

```
#cd /home/dejaneli/crosstool-0.43
#./dejaneli-arm.sh
```

大约经过三小时左右的漫长编译, 便会在/home/dejaneli/crosstool 目录下生成新的交叉编译工具, 其内容包括 arm-linux-g++、arm-linux-ld、arm-linux-size、arm-linux-ar、arm-linux-gcc、arm-linux-nm 等.

### 2.5 添加环境变量并使其生效

用 vi 编辑器打开/etc/profile 文件, 然后在其末尾添加命令: #export PATH=/home/dejaneli/crosstool/gcc-4.3.3-glibc-2.8/arm-linux/bin:$PATH, 保存退出后在虚拟机终端运行命令#source /etc/profile 使其生效[9].

## 3 Bootloader的移植

### 3.1 裁剪获取的源码文件

u-boot 的裁剪一般删除 board 目录下除 smdk2410 外的其他文件; 删除 cpu 目录下除 arm920t 以外的其他文件; 删除根目录下 lib_XXX 的库文件目录, 只留下 lib_arm 和 lib_generic; 删除 include 目录下 asm-XXX 的文件, 只留下 asm-arm; 删除 include/configs 目录下除 smdk2410.h 以外的其它所有配置头文件[2].

### 3.2 建立自己的目标板

```
#cd /opt/EmbedSky/u-boot-1.1.6/
#进入 u-boot-1.1.6 目录
#cp－f board/smdk2410 /board/my2440
#建立自己的目标板目录 my2440
#cp － f board/my2440/smdk2410.cboard/my2440/my2440.c
#cp include/configs/smdk2410.h include/configs/my2440.h
#建立配置头文件
```

### 3.3 修改 Makefile 文件

先修改 my2240 目录下 Makefile 文件的 28 行为:

```
COBJS := my2440.o flash.o
```

然后进入顶层 MakeFile 文件, 在 1881 行添加如下内容[1]:

```
my2440_config :unconfig
@$(MKCONFIG) $(@:_config=) arm arm920t my2440 NULL s3c24x0
```

最后再将顶层 Makefile 的 128 行修改为：

```
feq ($(ARCH),arm)
CROSS_COMPILE=/opt/EmbedSky/crosstools_4.3.3_softfloat/gcc-4.3.3-glibc-2.8/arm-linux/bin/arm-linux
```

endif

### 3.4 编译出镜像文件

执行命令:#make all ARCH=arm CROSS_COMPILE=arm-linux-生成 U-boot.bin 的可执行文件[8].

## 4 Linux内核的移植

本文选择具有新特性的 Linux-2.6.30.4 来进行说明. 首先下载内核源码 linux-2.6.30.4.tar.bz2, 将其解压到 PC 的根目录. 然后进到内核源码, 将 Makefile 文件的 193 行和 194 行修改为“ARCH=arm”和“CROSS_COMPILE?=arm-linux-”. 再在内核目录输入: #make menuconfig 进入内核菜单配置模式, 并根据自己的需求进行配置. 最后进入内核目录输入: #make zImage 进行编译, 完成后会在内核源码的“arch/arm/boot”目录下生成一个名为“zImage”的镜像[10].

## 5 制作根文件系统

根文件系统是 Linux 系统启动的一个重要组成部分, 在启动时内核需要根文件系统来挂载, 下面我们针对一般应用的需求阐述如何利用 Busybox 软件构建文件系统.

### 5.1 安装 Busybox

首先在 http://www.busybox.net/downloads 下载 busybox 源码并将其解压到自己的工作目录. 然后进入源码中,修改 Makefile 文件的 164 行和 189 行分别为: CROSS_COMPILE=arm-linux-和 ARCH=arm. 再输入命令: #make menuconfig, 进入配置菜单进行配置. 最后使用命令: #make; make install 进行编译安装, 编译结束后会在 busybox-1.13.0 目录下面生成一个名为“_install”的目录.

### 5.2 构建文件系统

#### 5.2.1 在内核中添加对 yaffs 的支持

首先到 http://www.aleph1.co.uk/cgi-bin/viewcvs.cgi 下载 yaffs2 文件系统的补丁, 然后向内核打上补丁, 具体命令如下:

#tar zxvf cvs-root.tar.gz

#cd cvs/yaffs2/

#./patch-ker.sh c /opt/EmbedSky/linux-2.6.30.4/

之后你会在内核源码的“fs/”目录下看到一个名为“yaffs2/”的目录, 同时“fs/”目录下面的 Makefile 和 Kconfig 文件也添加了 yaffs2 的配置和编译条件.

#### 5.2.2 建立目录框架

首先创建一个文件系统目录, 取名为“root_2.6.30.4”, 复制 busybox 的“_install”目录的内容到该目录下. 然后新建“dev”、“etc”、“home”、“lib”、“mnt”、“opt”、“proc”、“root”、 “sddisk”、“sys”、“tmp”、“usdik”、“var”和“web”目录, 同时在原有的“usr”目录下面新建一个“lib”和“share”目录[3-7].

#### 5.2.3 向目录中添加内容

首先需要在“dev”目录下静态创建两个设备文件, 否则文件系统启动时会警告没有打开控制台, 方法如下:

#mknod console c 5 1

#mknod null c 1 3

然后在“etc”目录下分别创建“fstab”、“group”、“inittab”、“shadow”、“password”、“profile”、“mdev.conf”、“resolv.conf”、“init.d”、“rc.d”、“boa”目录用来存放一些系统的配置文件. 接着从 4.3.3 的编译器中提取常用的库文件拷贝到“lib”目录下, 具体方法如下:

#cd /opt/EmbedSky/root_2.6.30.4

#cp –f /home/dejaneli/crosstool/gcc-4.3.3-glibc-2.8/arm-linux/arm-linux/lib/*.so* lib -a

#rm –f lib/linwv* lib/libuniconf*

#### 5.2.4 编译并烧写 Yaffs2 文件系统

在文件系统目录下输入如下命令:

#mkyaffs2image root_2.6.30.4 root_2.6.30.4.bin 然后便会在文件系统目录下生成 root_2.6.30.4.bin 的镜像文件, 将其通过 USB 烧写到开发板中, 就可以运行了.

## 6 实验结果

经过对 Bootloader、内核以及根文件系统的编译和烧写, 最终将 Linux 系统移植到 S3C2440 目标板上, 同时可以看到串口终端的打印信息, 系统运行稳定.

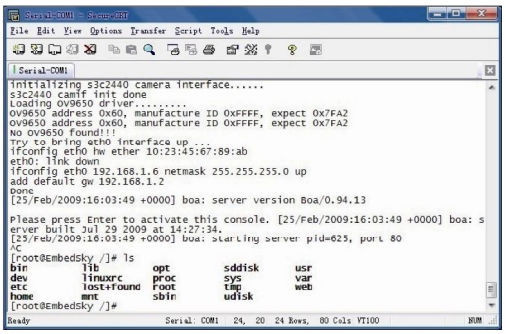


图 2　串口打印信息

（下转第 219 页）