【信息科学与控制工程】　　doi：10.11809/scbgxb2013.05.032

基于 S3C2410 的嵌入式 Linux 开发环境的搭建

朱　黎

（陕西工业职业技术学院，陕西 咸阳　712000）

摘要：介绍了基于 S3C2410 的嵌入式 Linux 开发环境的建立，阐述了 Linux 开发环境建立的主要过程和技术难点、关键问题，并给出了详细的过程和命令。对于 Linux 开发应用具有一定的借鉴作用。

关键词：嵌入式技术；Linux 操作系统；gcc 交叉编译器

中图分类号：TP302　　**文献标识码**：A　　**文章编号**：1006－0707(2013)05－0113－03

The Building of Embedded Linux Development Environment Based on the S3C2410

ZHU Li

(Shaanxi Polytechnic Institute, Xianyang 712000, China)

Abstract: This paper described the process of embedded Linux-based Samsung S3C2410 ARM9 development board environment, analyzed its key technology, and described in detail several key issues involved in embedded Linux system environment and the main steps.

Key words: embedded technology; Linux operating system; the gcc cross-compiler

由于嵌入式产品的大量应用和由于 Linux 具有良好的可裁剪性与可移植性，而且代码完全公开，具有丰富的网络资源及有力的技术支持和众多的研发力量。因此，嵌入式 Linux 系统的开发得到广泛的重视，成为越来越多的嵌入式系统选择和开发热点。一个完整的嵌入式 Linux 系统通常由由 Bootload、内核、文件系统 3 部分组成，目标板上电后由 Bootload 初始化硬件，引导内核和文件系统，从而启动 Linux。

嵌入式 Linux 开发环境的搭建主要包括：编译生成 Bootloader、裁剪、配置和编译 Kernel Image 和 Root File System，并将它们烧写到 Flash 中。

1　建立 Linux 交叉编译环境

本系统开发环境是在宿主机的 vmware 虚拟机中安装 Linux 操作系统实现的。其中 Linux 操作系系统的开发版本为 RedHat4.0，内核版本为 Linux2.6.24。此外，还需要在宿主机上配置 IP 地址并关闭防火墙、相关的网络服务，如 NFS 网络文件系统、TFTP 服务、Samba 服务。而对应的 ARM 开发板通常称为目标板。GNU 编译器的开发流程如图 1 所示。

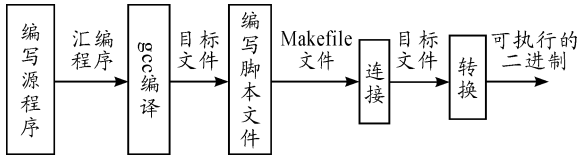

图 1　GNU 编译器的开发流程

GNU C Compiler 简写 gcc 是 Linux 系统下的功能强大、性能优越的 C 程序编译器。gcc 可以使程序员灵活地控制编译过程，通常情况下 gcc 在编译一个程序时，都要经历预处理、编译、汇编和链接 4 个阶段。每个阶段系统会自动调用不同的工具进行处理，gcc 编译程序过程如图 2 所示。

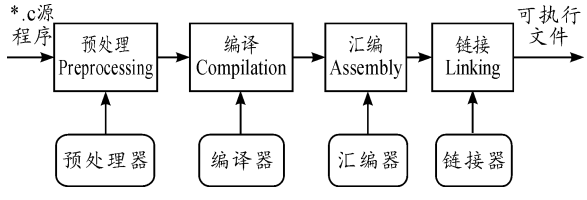

图 2　用 gcc 编译程序流程

收稿日期：2013－01－06

基金项目：中国机械工业教育协会项目(ZJJX11ZY035)。

作者简介：朱黎(1982—)，硕士，主要从事信号与信息处理、计算机控制研究。

2 Boot1oader

Bootloader 芯片复位后进入操作系统之前执行的一段程序,其作用与 PC 机上的 BIOS 类似。Bootloader 主要是为运行操作系统提供基本的运行环境,如 CPU、SDRAM、Flash、串行口等进行初始化,也可以下载文件到系统板,对 Flash 进行擦除与编程。

2.1 U-Boot 移植步骤

1) 建立目录并解压 u-boot 源码

```
[root@ vm-dev ~]# mkdir u-boot
[root@ vm-dev ~]# cd u-boot/
[root@ vm-dev u-boot]# ls
u-boot-1.3.2.tar.bz2
[root@ vm-dev u-boot]# tar -xjvf u-boot-1.3.2.tar.bz2
[root@ vm-dev u-boot]# ls
u-boot-1.3.2  u-boot-1.3.2.tar.bz2
```

2) 进入解压后的目录 u-boot-1.3.2,首先用 make distclean 命令清除原来编译环境依赖关系

```
[root@ vm-dev u-boot]# cd u-boot-1.3.2
[root@ vm-dev u-boot-1.3.2]# make distclean
```

3) 配置开发板,编译 u-boot。编译成功后会在当前目录下生成 u-boot 二进制文件。

```
[root@ vm-dev u-boot-1.3.2]# make uptech_2410class_config
Configuring for uptech_2410class board…
[root@ vm-dev u-boot-1.3.2]# make
[root@ vm-dev u-boot-1.3.2]#ls
u-boot.bin
```

4) 烧写 U-Boot

将编译得到的 u-boot.bin 拷贝到 PC 机 sjf2410-s.exe 文件所在的路径下。连接好开发板的电源、JTAG 下载线,然后打开电源。在 PC 机的 DOS 命令提示符下,进入 u-boot.bin 所在文件夹,运行命令烧写 u-boot。

```
D:》sjf2410-s.exe /f:u-boot.bin
```

在烧写中需要做一些选择,要分别输入三次 0,开始烧写,烧写完输入 2 推出。

2.2 测试 U-Boot

连接好开发板和主机之间的串口、网口,断开开发板的 JTAG 下载线,重新启动开发板。如果烧写成功,会在串口终端上出现如下内容:

```
U-Boot 1.3.2 (Dec  5 2008 - 10:35:38)
DRAM:  64 MB
Flash:512 kB
NAND:  64 MiB
*** Warning - bad CRC or NAND,using default environment
In:serial
Out:  serial
Err:  serial
Hit any key to stop autoboot:  0
[UP-2410-S #]
```

3 嵌入式 Linux 内核的裁减和移植

3.1 内核配置

1) 修改 Makefile 文件

在配置内核之前需要修改 linux-2.6.24.2 目录下的 Makefile 文件,指定交叉编译器为 arm-linux-编译器和使用 ARM 体系结构。

```
# cd  linux-2.6.24.2
# vi  Makefile
```

使用 vi 编辑器打开 Makefile 文件,作如下修改。

```
ARCH  ? = arm
CROSS_COMPILE = arm-linux-
```

2) 配置内核

内核源码必须先进行配置才能编译。通常内核的配置有以下 4 中方法:make config、make xconfi、make menuconfig 和 make gconfig。

```
# cd linux-2.6.24.2
# cp  arch/arm/configs/s3c2410_defconfig  .config
# make menuconfig
```

得到.config 文件,运行"make menuconfig"命令打开内核配置界面如图 3 所示。设置 S3C2410 Machines、Nand Flash、网卡、文件系统等相关配置信息。

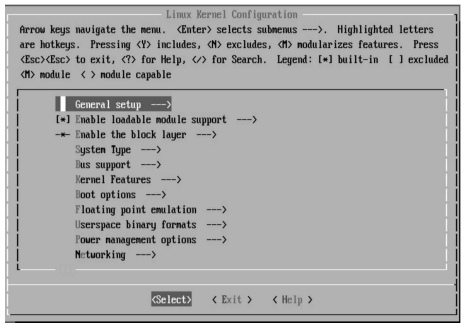

图 3 内核配置界面

3.2 内核编译

1) 编译内核映像和模块

如果内核已经编译过多次,需进入内核根目录清除原先残留的.config 和.o 文件。然后用 make 命令进行编译。编译成功,在内核源码根目录的 arch/arm/boot 下生成 zImage 文件。

```
[root@ vm-dev linux-2.6.24.4]# cd  linux-2.6.24
[root@ vm-dev linux-2.6.24.4]# make mrproper
[root@ vm-dev linux-2.6.24.4]# make
[root@ vm-dev linux-2.6.24.4]# ll arch/arm/boot/zImage
-rwxr-xr-x  1 root root 1814040  6 月 25 14:50 arch/arm/boot/zImage
```

2）生成 uImage 文件

使用由 u-boot 生成的工具 mkimage,生成 uImage 文件。执行脚本程序 make_uImage 此时会在内核源码根目录下生成 uImage 内核文件。

```
[root@ vm-dev linux-2.6.24.4]# ./make_uImage
Image Name:   Linux-2.6.24.4
Created:   Thu June 25 14:52:49 2010
Image Type:   ARM Linux Kernel Image (uncompressed)
Data Size:1814040 Bytes = 1771.52 kB = 1.73 MB
Load Address:0x30008000
Entry Point:   0x30008040
[root@ vm-dev linux-2.6.24.4]#
[root@ vm-dev linux-2.6.24.4]# ll uImage
-rw-r--r--  1 root root 1814104  6 月 25 14:52 uImage
```

3.3　烧写 Linux 内核

1）配置 IP 地址。设置宿主机即 TFTP 服务器端机器 IP 为 setenv serverip 192.168.1.12,设置 ARM 端 U-BOOT 中网络设备 IP 地址 setenv ipaddr 192.168.1.13,saveenv 保存设置。

```
[up-class2410 #]  setenv serverip 192.168.1.12
[up-class2410 #]  setenv ipaddr 192.168.1.13
[up-class2410 #]  saveenv
Saving Environment to NAND...
Erasing Nand... Writing to Nand... done
```

2）将生成的 uImage 文件拷贝到 tftpboot 目录下

```
[root@ vm-dev linux-2.6.24.4]# cp uImage /tftpboot/
cp:否覆盖'/tftpboot/uImage'? y
```

3）下载到 SDRAM。运行 tftp 0x30008000 uImage 命令,将 uImage 文件下载到 ARM 开发板的 SDRAM 中 0x30008000 开始的空间中。

4）擦除 NANDFLASH 空间,写入 Nand Flash。

3.4　引导内核

重启 ARM 开发板,执行命令"bootm",实现 U-BOOT 引导内存中的内核。启动后液晶屏左上角出现小企鹅图案。

4　文件系统

Root Filesystem(根文件系统)是 ARM Linux 正常运行的必要组成部分。创建文件系统后,应用程序对 Nand-Flash 存储设备的读写操作就好像对 MS-DOS 文件系统的磁盘设备操作一样。目前 Linux 支持多种文件系统,主要包括 Romfs、Cramfs、JFFS 和 JFFS2 等。

4.1　建立根文件系统

1）创建根文件目录 rootfs

2）使用 busybox 工具创建文件系统

a）将已有的 busybox-1.12.2 压缩包拷贝到根目录下并解压

b）修改 Makefile 文件,支持交叉编译

修改该目录下 Makefile 文件中的 ARCH 和 CROSS_COMPIL,指定交叉编译器和目标系统,与本机的路径一致。

```
CROSS_COMPILE ? = arm-linux-
…
ARCH ? = arm
```

c）编译 busybox

执行命令 make menuconfig 进入 busybox 如图 4 所示配置界面设置相关选项并保存。

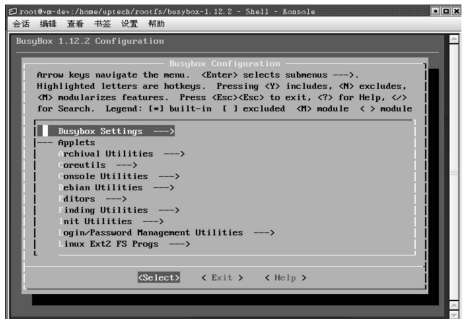

图 4　busybox 的配置界面

d）用 make,make install 进行编译生成_install 目录

```
[root@ vm-dev busybox-1.12.2]# make
[root@ vm-dev busybox-1.12.2]# make install
[root@ vm-dev busybox-1.12.2]# ls _install/
bin  linuxrc  sbin  usr
```

至此 busybox 工具编译完成,生成了文件系统需要的相关命令和工具在_install 目录下。用户也可以根据需要,在 busybox 中添加删除相关命令和工具。

创建根文件系统的其它目录结构,如 etc、dev、lib、mnt 等,并添加相关配置文件与设备节点。也可直接解压 rootfs 压缩包,生成 rootfs 根目录树。

3）复制_install 文件夹内容

将"/home/uptech/rootfs/busybox-1.12.2/_install"的全部内容复制到"/home/uptech/rootfs"中。

4）使用 mkcramfs 工具将 rootfs 文件系统目录制作成 CRAMFS 根文件系统映像生成 root.cramfs 根文件系统文件。

4.2　烧写根文件系统

1）将生成的根文件系统文件 root.cramfs 到宿主机 TFTP 服务器下载目录/tftpboot

```
[root@ vm-dev rootfs]# cp root.cramfs /tftpboot/
```

2）配置宿主机和目标机的网络 IP,启动 ARM 设备,进入 U-Boot 控制台

3）下载到 SDRAM

4.3　启动 LINUX 系统,挂载根文件系统

在 U-BOOT 中输入 boot 目录引导系统。

```
up-tech login:root
up-tech:~ #
```

输入 root 用户名称,系统顺利引导运行起来了。

(下转第 122 页)