基于 ARM 和 Linux 的嵌入式平台的构建

曾福振，闵联营

（武汉理工大学 计算机科学与技术学院，湖北 武汉 430063）

摘 要：首先介绍了嵌入式系统的概念，及相关硬件平台和软件版本。然后，主要介绍了嵌入式 Linux 的引导程序 U-Boot 的移植，以及开源、免费操作系统 Linux2.6.32.2 的移植。最后，构建了基于 Nand Flash 存储器的 Yaffs2 文件系统，利用 BusyBox 创建根文件系统。基于 ARM 和嵌入式 Linux 的嵌入式系统平台搭建基本完成，可以在此平台上添加更多驱动，以便更好地开发应用程序。

关键词：嵌入式；Linux；U-Boot；移植

中图分类号：TP319 **文献标识码**：A **文章编号**：1674-7720(2011)12-0051-03

Construct of embedded platform based on ARM and Linux

Zeng Fuzhen, Min Lianying

(School of Computer Science and Technology, Wuhan University of Technology, Wuhan 430063, China)

Abstract: This article firstly introduced what is embedded system, and hardware platform and software version are introduced. Whats more, this paper usesd a concrete example to introduce the porting of embedded Linux's bootloader U-Boot and open source, free operating system Linux2.6.32.2 to ARM embedded platform. Finally, this paper introduced the methof creating Yaffs2 file system base on Nand Flash memory and created a root file system using BusyBox. The basic structures of embedded systems ARM compateled based on AMR and embedded Linus. You can add some drivers on this platform to develp applications.

Key words: embedded; Linux; U-Boot; porting

进入后 PC 时代以来，伴随着设计和制造技术的发展，集成电路从当初的晶体管集成发展到现在的 IP 集成，即 SoC（System on Chip）设计技术。促使嵌入式系统渗透到了当今社会中的各个行业，并且发挥越来越重要的作用。嵌入式系统一般可定义为以应用为中心、以计算机技术为基础、软硬件可裁剪、适用于应用系统且对功能、成本、体积、功耗有严格要求的专用计算机系统，它的主要特点是嵌入、应用[1]。

随着各种嵌入式设备功能越来越强大，在设备中使用嵌入式操作系统也成为必然。Linux 操作系统具有开放源代码、易于移植、资源丰富、免费等特点，在嵌入式领域的地位越来越重要。嵌入式 Linux 和 PC 上的 Linux 是同一套内核代码，只是裁剪的程度不一样，所以，很多在 PC 上开发的软件，经过交叉编译后可以直接在嵌入式设备上运行。本文主要涉及到 Bootloader 移植和 Linux-2.6.32.2 内核的移植、根文件系统移植、在 S3C2440 平台上构建完整的嵌入式开发平台三个方面。

1 交叉开发环境的建立

在进行嵌入式软件开发之前，必须要在 PC 上建立 ARM 的交叉编译环境。交叉编译就是在 PC 平台上生成可以在 ARM 平台上运行的代码。其中主要包括 ARM 的交叉编译器 arm-elf-gcc 和交叉连接器 arm-elf-ld。本文采用的交叉编译器的版本是 gcc-3.4.5-glibc-2.3.6[2]。交叉编译流程如图 1 所示。

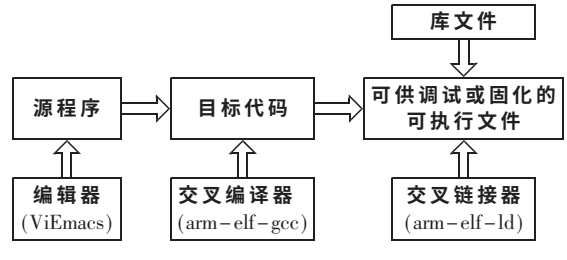

图 1 嵌入式系统交叉编译流程

2 BootLoader 引导加载程序

BootLoader 是一段在系统上电时开始执行的程序，

用以初始化硬件设备,准备好软件环境,设置好启动参数,最后引导操作系统,与 PC 上的 BIOS 程序相似。当前开放源码的 Linux 引导程序主要有 x86 架构的 LILO、GRUB,对于 ARM 架构的主要有 Vivi 和 U-Boot。本文使用 U-Boot 作为引导程序。U-Boot(Universal Boot Loader),即通用的 BootLoader,遵循 GPL 条款开放源代码。U-Boot 相对于 Vivi 功能更加强大,也更方便后续程序的调试。

BootLoader 的启动一般分为两个阶段,第一阶段的代码主要是用汇编语言编写,主要的功能是完成硬件设备的初始化,为加载第二阶段的代码准备 RAM 空间,设置好堆栈;第二阶段主要用 C 语言编写,检测内存映射,将内核映像和根文件系统从 Nand Flash 读到 RAM 中,为内核启动设置参数,引导内核。

U-Boot 的源代码可以从 ftp://ftp.denx.de/pub/u-boot/ 进行下载,本文使用的 U-Boot 版本是 U-Boot2009.08。

移植 U-Boot 的关键步骤如下:

(1)首先,将 include/configs 目录下的 smdk2410.h 复制并改名为 mini2440.h,根据 U-Boot 的说明可以知道,如果要使用开发板 board/<board_name>,则先执行“make<board_name>”_config 命令进行配置,然后执行“make all”,生成可执行文件。所以,修改 U-Boot 顶层的 Makefile 文件,添加下面一行 mini2440_config : unconfig @$ (MKCONFIG) $ (@:_config=) arm arm920t mini2440 frank s3c24x0。这里有几个重要的参数,arm 指 CPU 的架构,arm920t 指 CPU 的类型,s3c24x0 指 CPU 的型号。这样就可以使用 make mini2440_config 这条命令进行配置。

(2)本文使用的 U-Boot 是从 Nand Flash 启动的,CPU 可以直接访问 Nand Flash 中前 4 KB 代码,利用这 4 KB 代码把 U-Boot 中绝大部分代码拷贝到内存中[3]。其中下面的代码就是调用 C 语言中的 Nand Flash 的读写函数,该函数主要把 Nand Flash 中 4 KB 以后的代码复制到 RAM 中。在编写 nand_read_ll 的函数时,注意参考 Nand Flash 的数据手册,对大页和小页的 Nand Flash,其读写的命令和时序是不同的。

```
@copy U-Boot to RAM
ldr r0,=TEXT_BASE
mov r1,#0x0
mov r2,#0x60000
bl nand_read_ll
tst r0,#0x0
beq ok_nand_read
```

由于在后面加载 Linux 内核和根文件系统时,使用的是 tftp 方式,所以必须添加 DM9000EP 网卡的驱动。在 mini2440.h 文件中,其主要的配置如下:

```
#define CONFIG_DRIVER_DM9000 1
#define CONFIG_NET_MULTI 1
#define CONFIG_DM9000_NO_SROM 1
#define CONFIG_DM9000_BASE 0x20000300
#define DM9000_DATA (CONFIG_DM9000_BASE +4)
```

其中,CONFIG_DM9000_BASE 宏是最重要的,因为它定义的是网卡的地址,不同的网卡有不同的地址,DM9000EP 访问的基址为 0x20000000,之所以再偏移 0x300 是由它的特性决定的。

(3)要正确引导 Linux 内核,还需要配置下面几个重要的宏定义,这几个宏定义不同,意味着引导 Linux 内核的方式也不同。

```
#define CONFIG_BOOTARGS"noinitrd root=/dev/mtdblock3 init=/linuxrc console=ttySAC0,115200 mem=64M"
```

其中,root=/dev/mtdblock3 是由 Linux 中的 Nand Flash 分区所决定的,意味着 Nand Flash 的第 4 个分区为根文件系统。

```
#define CONFIG_BOOTCOMMAND"nand read 0x32000000 0x60000 0x560000;bootm 0x32000000"
```

这个宏定义是将 Nand Flash 中 0x60000 -0x560000(和 kernel 分区一致)的内容读到内存 0x32000000 中,然后用 bootm 命令来执行。

要正常地引导 Linux 内核,必须要具备如下几个条件[4]:

(1)CPU 寄存器

R0=0;

R1=机器类型 ID;对于 ARM 结构的 CPU,其机器类型 ID 在 linux/arch/arm/tools/mach-types;

R2=启动参数标记列表在 RAM 中起始基地址。

(2)CPU 工作模式

必须禁止中断(IRQs 和 FIQs);

CPU 必须为 SVC 模式。

(3)Cach 和 MMU 的设置

MMU 必须关闭;

指令 Cach 可以打开也可以关闭;

数据 Cach 必须关闭。

3 Linux2.6.32.2 内核的移植

3.1 内核的获取

Linux 内核的更新很快,可以从 http://www.kernel.org/pub/linux/kernel/得到最新的 Linux 内核版本,本文使用的 Linux 内核版本是 Linux-2.6.32.2,交叉编译工具使用符合 EABI 标准的 arm-linux-gcc-4.3.2。

3.2 内核的移植

可以在内核的根目录下,运行 make menuconfig 命令,对内核进行适当的裁剪,以适应硬件平台。

(1)修改 Makefile 文件

欲设置 Linux 的默认平台为 ARM 平台,需进入 Linux-2.6.32 文件夹中,修改此目录下的 Makefile 文件。

```
export KBUILD_BUILDHOST := $(SUBARCH)
ARCH ?=arm                          //使用的目标平台
CROSS_COMPILE ?=arm-linux-          //使用的交叉编译器,
                                    这里使用系统默认的编译器
```

(2)关于机器码

在启动内核时，根据 BootLoader 传入的机器码(MACH_TYPE)来决定应启动哪种目标平台[6]，本开发平台的机器码为 1999。机器码存放在文件 opt/kernel/linux-2.6.32.2/arch/arm/tools/mach-types 中。

mini2440 MACH_MINI2440 MINI2440 1999 //机器码

如果机器码不匹配，引导内核不成功，则会出现如下的错误提示：

Uncompressing Linux……done, booting the kernel.

(3)修改时钟源

将/kernel/linux-2.6.32.2/arch/arm/mach-s3c2440/目录下的 mach-smdk2440.c 文件改名为 mach-mini2440.c。因为 mini2440 和 mach-smdk2440.c 极其相似，以该文件为基础进行修改，在 mach-mini2440.c 文件中将 static void__init smdk2440_map_io(void)函数中的晶振频率修改为 mini2440 开发板上实际使用的 12000000。

(4)为内核打上 yaffs2 补丁

①Yaffs2 文件系统是专门针对嵌入式设备，特别是使用 Nand Flash 作为存储器的嵌入式设备而创建的一种文件系统，使用 yaffs2 就可以支持大页的 Nand Flash。进入 yaffs2 源代码目录执行如下命令：

#./patch-ker.sh c /opt/FriendlyARM/mini2440/linux-2.6.32.2

②配置内核以支持 Yaffs2 文件系统

在 Linux 内核源代码根目录运行 make xconfig，在“File Systems”选项中，找到“Miscellaneous filesystems”菜单项，找到“YAFFS2 file system support”并选中它，这样就在内核中添加了 yaffs2 文件系统的支持，保存并退出。然后在命令行中，执行 make zImage。

(5)修改 Nand Flash 分区信息

①在 mach-mini2440.c 文件中添加 Nand Flash 的分区信息，下面的代码将 Nand Flash 分成了 4 个分区，第 1 分区也是 BootLoader 所在的分区，对应 dev/mtdblock0；第 2 个分区是 U-Boot 的参数分区，对应 dev/mtdblock1；第 3 个分区是内核分区，对应 dev/mtdblock2；第 4 个分区为根文件系统分区对应 dev/mtdblock3。分区结构图如表 1 所示。

表 1 128 MB Nand Flash 的分区结构图

U-Boot	0x00000000-0x00040000
Param	0x00040000-0x00060000
Kernel	0x00060000-0x00560000
Root	0x00560000-0x07FFFFFF(结束)

其部分实现代码如下：

```
static struct mtd_partition mini2440_default_nand_part[] ={
[0] = {
        .name="U-boot",
    .offset= 0,
        .size= 0x00040000,
    }
```

其中 name 是分区的名字，offset 是偏移的开始地址，size 是分区的大小，其余部分的分区与此类似。

②下面代码是添加 Nand Flash 的设置表，因为板子上只有一片 Nand Flash，因此也就只有一个设置表。

```
static struct s3c2410_nand_set mini2440_nand_sets[] = {
[0] = {
        .name= "NAND",
        .nr_chips= 1,
        .nr_partitions=
ARRAY_SIZE(mini2440_default_nand_part),
        .partitions= mini2440_default_nand_part,
    }
}
```

③上面的设置完成后，还需要将 Nand Flash 设备注册到系统中。下面这段代码就是将 Nand Flash 设备添加到开发板的设备列表结构。

```
static struct platform_device *mini2440_devices[] __initdata
= {
    &s3c_device_nand,
}
```

④在 mini2440_machine_init 函数中添加平台的数据信息。

```
static void __init mini2440_machine_init(void){
s3c_device_nand.dev.platform_data=&mini2440_nand_info;
}
```

现在可以进入 kernel/linux-2.6.32.2/arch/arm/boot 目录，然后执行下面的命令，就会在该目录下生成 uImage.img 格式的、U-Boot 可以引导的内核镜象。

```
Mkimage -n 'linux-2.6.32.2' -A arm -O linux
-T kernel -C none -a 0x30008000 -e 0x30008000 -
d zImage uImage.img
```

至此，可以把生成的 uImage.img 格式的镜像文件复制到 tftp 目录下，使用 tftp 进行下载。

3.3 文件系统

所谓根文件系统，就是创建各个目录，例如在/bin、/sbin/目录下存放各种可执行的程序，在/etc 目录下存放配置文件，在/lib 目录下存放库文件。

可以利用 Busybox 工具创建根文件系统，Bosybox 是一个遵循 GPL v2 协议的开源项目，它在编写过程中对文件大小进行优化，并考虑了系统资源有限(例如内存)的情况，使用 Busybox 可以自动生成根文件系统所需的 bin、sbin、usr 目录和 linuxrc 文件，可以使用 make menuconfig 对 Busybox 的选项进行配置。

(1)进入 opt/kernel,创建一个 shell 脚本用于构建根文件系统的各个目录,并且为其增加执行权限;

(2)Linux 中的 init 进程会根据 etc/inittab 文件创建其他子进程,下面代码是 inittab 文件中的内容,说明了系统启动后首先执行的脚本文件是 rcS, 虚拟的终端是串口 0,当按下 ctr+alt+del 时重启系统,inittab 文件的作用就是控制系统启动时和启动后一些程序的运行。

```
#etc/inittab
::sysinit:/etc/init.d/rcS
s3c2410_serial0::askfirst:-/bin/sh
::ctrlaltdel:/sbin/reboot
::shutdown:/bin/umount -a-r
```

(3)创建 etc/init.d/rcS 文件,这是一个脚本文件,可以在里面添加要自动执行的一些命令。

```
#! /bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
runlevel=S                                    //运行的级别
prevlevel=N
umask 022                                     //文件夹的掩码
mount -a  //挂载/etc/fstab/文件指定的所有的文件系统
mdev-s
/bin/hostname -F /etc/sysconfig/HOSTNAME//主机的名字
```

使用 yaffs 源码提供的工具制作文件系统的映像文件。由于 128 MB 的 Nand Flash 是大页结构,所以需要使用相应的大页制作工具; 使用命令 mkyaffs2image rootfs rootfs.img 生成根文件系统映像文件。

本文通过对 U-Boot 移植和 Linux 内核移植的讨论,给出了移植 U-Boot 和 Linux 到大多数开发板的关键部分。由于移植的复杂性,不可能包括全部步骤,但通过本文的阐述可以了解移植的基本流程和关键点,为移植不同版本到其他硬件平台提供了参考,也为应用程序的开发搭建了一个比较完整的嵌入式平台。

参考文献

[1] 韦东山.嵌入式 Linux 应用开发完成手册[M].北京:人民邮电出版社,2008.

[2] 孙琼.嵌入式 Linux 应用程序开发详解[M].北京:人民邮电出版社,2006.

[3] Samsung Electronics.S3C2440A 32-bit RISC microprocessor user's manual[S].2004.

[4] RUSSELL K.ARM Linux kernel Boot requirements[EB/OL].[2002-03-18].http://www.arm.linux.org.uk/developer/booting.php.

[5] 陈莉君.深入理解 Linux 内核[M].北京:中国电力出版社,2007.

[6] JONATHAN C,ALESSANDRO R,GREG KROAH H.Linux 设备驱动程序[M].北京:中国电力出版社,2006.

(收稿日期:2011-01-25)

作者简介:

曾福振,男,1983年生,硕士研究生,主要研究方向:嵌入式系统与网络。