基于 ARM 的嵌入式 Linux 系统构建①

冷玉林 钟 将 (重庆大学 计算机学院 重庆 400044)

摘 要: 详细论述了在基于 ARM920T 核心的 S3C2410 平台上构建嵌入式 Linux 系统的过程，包括交叉开发环境的建立，引导加载程序 U-Boot、Linux 操作系统内核针对特定目标平台的移植，以及根文件系统的建立等。试验结果显示系统在目标平台上运行稳定、可靠，对其它嵌入式系统的开发具有参考意义。

关键词: 嵌入式系统; ARM; S3C2410 微处理器; Linux

Building Embedded Linux System Based on ARM

LENG Yu-Lin, ZHONG Jiang

(Department of Computer Science, Chongqing University, Chongqing 400044, China)

Abstract: This paper discusses the process of building embedded Linux system on the ARM920T core-based S3C2410 platform, including the establishment of cross-development environment, the transplanting of U-Boot and Linux kernel to a specific target platform, and the building of root file system. The final system proves to be stable and reliable after several testing. Also it's helpful to the exploiting of other embedded systems.

Keywords: embedded system; ARM; S3C2410 microprocessor; Linux

1 引言

随着信息产业的发展和集成电路技术的进步，嵌入式系统已经广泛地应用到移动计算设备、网络设备、工控设备、信息家电和仪器仪表等领域。嵌入式系统，是以应用为中心、以计算机技术为基础、软硬件可裁剪，能适应应用系统对功能、可靠性、成本、体积、功耗严格要求的专用计算机系统。这种系统具有软件代码少、响应速度快、高度自动化等特点，用于实现对其它设备的控制、监视或管理等功能，特别适用于要求实时的和多任务的应用[1]。ARM 嵌入式芯片是一种高性能、低功耗的 RISC 芯片。他由英国 ARM 公司设计，世界上几乎所有的主要半导体生产商都生产基于 ARM 体系结构的通用芯片，或在其专用芯片中应用相关 ARM 技术，S3C2410 是韩国 Samsung 公司基于 ARM 公司的 ARM920T 处理器核,采用 0.18um 制造工艺的 32 位嵌入式微处理器。Linux 是免费发行的、快速高效的操作系统，它的出现在计算机世界引发了一场革命。Linux 以其代码开放、功能强大又易于移植等特点成为嵌入式操作系统的新兴力量。在所有的嵌入式操作系统中，Linux 是发展最快、应用最广泛的。利用 Linux 搭建嵌入式系统是近年来出现的最令人振奋的方案之一[2,3]。嵌入式 Linux 是按照嵌入式操作系统的要求设计的一种小型操作系统，由一个内核以及一些根据需要进行定制的系统模块组成。其内核很小，一般只有几百 kb，即使加上其他必要的模块和应用程序，所需的存储空间也很小，非常适合于嵌入式系统。

本文将详细论述在基于 ARM920T 核心的 S3C2410 平台上构建嵌入式 Linux 系统的过程，包括交叉开发环境的建立，引导加载程序 U-Boot、Linux 操作系统内核针对特定目标平台的移植，以及根文件系统的建立等。

① 收稿时间:2010-03-19;收到修改稿时间:2010-04-18

2　交叉开发环境的建立

嵌入式软件开发中所采用的编译为交叉编译，所谓交叉编译就是在一个平台上生成可以在另一个平台上执行的代码。搭建交叉编译环境是嵌入式开发的第一步，也是必备的一步，选择合适的交叉编译器对于嵌入式开发是非常重要的。交叉编译器的安装一般涉及到多个软件的安装，这包括 binutils、gcc、glibc 等软件。其中，binutils 主要用于生成一些辅助工具，如 objdump、as、ld 等；gcc 是用来生成交叉编译器，主要生成 arm-linux-gcc 交叉编译工具；而 glibc 主要是提供用户程序所使用的一些基本的函数库。社区的开发者和一些芯片厂商已经编译出了常用体系结构的工具链，使用这些工具链，将使得工作量大大降低。

将工具链压缩包 arm-linux-gcc-3.4.5-glibc-2.3.6.tar.bz2 解压至/workspace/tools 目录：

```
#cd /workspace/tools
#tar jxvf arm-linux-gcc-3.4.5-glibc-2.3.6.tar.bz2
```

在/etc/bashrc 文件末尾加入如下一行:

```
PATH=$PATH:/workspace/tools/gcc-3.4.5-glibc-2.3.6/bin
```

3　嵌入式Linux系统的构建

3.1 引导加载程序 U-Boot 的移植

系统上电之后，需要一段程序来进行初始化：关闭 WATCHDOG、改变系统时钟、初始化存储控制器、将更多的代码复制到内存中等。如果它能将操作系统内核复制到内存中运行，无论从本地(如 Flash)还是从远端(如网络)，就称这段程序为 BootLoader。简单地说 BootLoader 就是在操作系统内核或用户应用程序运行之前运行的一段小程序。通过这段小程序，可以初始化硬件设备、建立内存空间的映射图,从而将系统的软硬件环境带到一个合适的状态，以便为最终调用操作系统内核或用户应用程序准备好环境。对于一个嵌入式系统来说,可能有的包括操作系统,有的小型系统也可以只包括应用程序,但是在这之前都需要 BootLoader 为它准备一个正确的环境。通常,BootLoader 是依赖于硬件而实现的,特别是在嵌入式领域,为嵌入式系统建立一个通用的 BootLoader 是很困难的。U-Boot，全称为 Universal Boot Loader，即通用 BootLoader，是遵循 GPL 条款的开放源代码项目，并且已经成为 ARM 平台事实上的标准 BootLoader，其名字“通用”有两层含义：可引导多种操作系统、同时支持多种架构 CPU[4]。本文所用 U-Boot 版本为 u-boot-1.3.4，将所获取的源码压缩包 u-boot-1.3.4.tar.bz2 解压得到全部源码，首先需要分析一下它已支持的开发板,比较出硬件最接近的开发板。本文将在 u-boot-1.3.4 已经支持的 smdk2410 板的基础上进行移植。

3.1.1 新建、修改目标板对应目录及文件

在 board 目录下将 smdk2410 复制为 my2410 目录，将此目录下的 smdk2410.c 文件重命名为 my2410.c；将 include/configs/smdk2410.h 复制为 my2410.h；在顶层 Makefile 中添加如下两行

```
my2410_config:    unconfig
    @$(MKCONFIG) $(@:_config) arm arm920t my2410 NULL s3c24x0
```

然后在 board/my2410/Makefile 中修改 COBJS：

```
COBJS  :=my2410.o flash.o
```

3.1.2 修改 SDRAM 配置

本文将目标板 HCLK 设为 100MHz，根据 SDRAM 芯片的具体参数重新计算 REFCNT 寄存器的值，对 board/my2410/lowlevel_init.S 文件作如下部分修改：

```
#define REFCNT 0x4f4 /*period=7.8125us, HCLK=100Mhz, (2048+1-7.8125*100)*/
```

3.1.3 支持目标板 NAND Flash

对 NAND Flash 的支持有新旧两套代码，新代码在 drivers/mtd/nand 目录下。选择用此新代码来支持目标板 NAND Flash。要让 U-Boot 支持 NAND Flash,首先在 include/configs/my2410.h 的宏 CONFIG_COMMANDS 中增加 CONFIG_CMD_NAND，然后在 include/configs/my2410.h 中选择不定义 CONFIG_NAND_LEGACY 宏(若定义则为使用旧代码)。在 include/configs/my2410.h 中作如下定义：

```
#define CONFIG_CMD_NAND
#define CFG_MAX_NAND_DEVICE 1     //目标板上有一块 NAND Flash 设备
#define CFG_NAND_BASE 0x4e000000 //NAND Flash 控制器基址
```

3.1.4 支持 CS8900

本文目标板网卡芯片 CS8900 的连接方式与样板

smdk2410 一致，U-Boot 已能够支持 CS8900，其驱动程序为 drivers/cs8900.c。所以在使用网络之前设置其 IP 地址、MAC 地址、宿主机 IP 地址即可。

3.1.5 若干默认配置参数的设置

在 include/configs/my2410.h 中增加以下默认配置参数：

```
#define CONFIG_SETUP_MEMORY_TAGS 1
#define CONFIG_CMDLINE_TAG    1
#define CONFIG_BOOTARGS " noinitrd root=/dev/mtdblock1 rootfstype=jffs2 console=ttySAC0"
#define CONFIG_ETHADDR 08:00:3e:26:0a:5b
#define CONFIG_NETMASK 255.255.255.0
#define CONFIG_IPADDR 222.198.131.33
#define CONFIG_SERVERIP 222.198.131.32
```

上述参数分别为设置向内核传递的命令行参数，以及目标板，宿主机 IP 的设置等。

3.1.6 U-Boot 的编译、烧写

执行以下命令配置、编译，得到二进制映象文件 u-boot.bin。

```
#make my2410_config
#make all
```

最后，通过 JTAG 与宿主机并口相连,在主机烧写程序 H-JTAG 的支持下将 u-boot.bin 文件烧写到 NOR Flash。烧写完成后,复位实验板,串口终端显示 u-boot 的启动信息。

3.2 Linux 内核的移植

标准 Linux 内核对于资源受限的嵌入式系统来说过于庞大,要将其移植到嵌入式系统上,就需要将 Linux 内核根据目标平台的情况进行剪裁、配置。本文将配置、修改 linux-2.6.22.6 内核，使得它在能够支持 s3c2410 板的同时能够支持 JFFS2、YAFFS 文件系统，并修改 MTD 设备分区，以支持挂载 NAND Flash 上的文件系统。

3.2.1 MTD 分区的修改

MTD(Memory Technology Device)，即内存技术设备，是 Linux 中对 ROM、NOR Flash、NAND Flash 等存储设备抽象出来的一个设备层，它向上提供统一的访问接口：读、写、擦除等，屏蔽了底层硬件的操作、各类存储设备的差别。本文修改 arch/arm/plat-s3c24xx/common-smdk.c 中的 smdk_nand_part 结构，改变分区信息：

```
Static struct mtd_partition smdk_default_nand_part[] = {
    [0] = {
    .name = "kernel",
    .size = SZ_2M,
    .offset = 0,
    },
    [1] = {
    .name = "jffs2",
    .offset = MTDPART_OFS_APPEND,
    .size = SZ_8M,
    },
    [2] = {
    .name = "yaffs",
    .offset = MTDPART_OFS_APPEND,
    .size = MTDPART_SIZ_FULL,
    }
    };
```

本文所划出的分区情况：前 2MB 存放内核，接下来 8MB 存放 JFFS2 文件系统，剩下的以后可用来存放 YAFFS 文件系统。

3.2.2 添加支持 YAFFS 文件系统

YAFFS(yet another flash file system)是类似于 JFFS/JFFS2、专为 NAND Flash 而设计的一种嵌入式文件系统，适于大容量的存储设备。它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的一致性和完整性的影响[5]。可从 http://www.aleph1.co.uk/cgi-bin/viewcvs.cgi/获取其源码压缩包 root.tar.gz，解压之，并使用其中的文件 patch-ker.sh 来修改内核。进入 yaffs2 源码目录：

```
#cd/workspace/system/Development/yaffs2
#./patch-ker.shc/workspace/system/linux-2.6.22.6
```

3.2.3 内核配置、编译与烧写

现在，内核支持 s3c2410，同时能够支持 JFFS2、YAFFS 文件系统，且在修改了 MTD 设备分区后能支持挂载 NAND Flash 上的文件系统。现对内核进行配置、编译，进入 Linux 内核源码所在目录，执行如下命令：

```
#cd /workspace/system/linux-2.6.22.6
```

```
#cp
arch/arm/configs/s3c2410_defconfig ./.config
#make menuconfig
#make uImage
```

先将 arch/arm/configs/ 下的 s3c2410_defconfig 文件复制为内核源码根目录下的.config 文件，在此基础上对若干配置选项作进一步的选择，主要包括以下几项配置:处理器类型,板级支持,对 RAM disk、设备驱动及文件系统的支持。编译完成后在 arch/arm/boot/目录下生成 U-Boot 格式的内核映象文件 uImage。

```
#tftp 0x30000000 uImage
#nand erase 0x0 0x200000
#nand write.jffs2 0x30000000 0x0 $(filesize)
```

上述命令将 uImage 置于主机 tftp 目录下，通过 tftp 方式下载并将之烧写到 NAND Flash 前 2M 空间中。

3.3 根文件系统的构建

Linux 系统由内核与文件系统共同构成，而根文件系统则是 Linux 启动时使用的第一个文件系统。没有根文件系统，Linux 将无法正常启动。根文件系统由一系列目录组成，目录中包含了应用程序、C 库、以及相关的配置文件[6]。所谓制作根文件系统，就是创建各种目录，并且在里面创建各种文件。比如在/bin、/sbin 目录下存放各种可执行程序，在/etc 目录下存放配置文件，在/lib 目录下存放库文件等。

本文首先使用 Busybox 来创建/bin、/sbin 等目录下的可执行文件。Busybox 是一个遵循 GPL v2 协议的开源项目，它将众多的 UNIX 命令集合进一个很小的可执行程序中,其中各种命令与相应的 GNU 工具相比，所能提供的选项较少，但是能够满足一般应用。

3.3.1 配置 Busybox

将获取的 Busybox 源码压缩包解压之得到源码目录 busybox-1.7.0。首先通过配置 Busybox 来选择所需要的命令、定制某些命令的功能(选项)、指定 Busybox 的连接方法，以及其安装路径。在 busybox-1.7.0 目录下执行：#make menuconfig，退出并保存所做配置。然后分别修改 Busybox 根目录下 Makefile 中 ARCH 与 CROSS_COMPILE 变量的值为 arm 与 arm-linux-，进行编译并安装至/workspace/nfs_root 目录：

```
#make
#makeCONFIG_PREFIX=/workspace/nfs_root install
```

3.3.2 使用 glibc 库

本文使用前面制作交叉编译工具链时生成的 glibc 库，其位置为 /workspace/tools/gcc-3.4.5-glibc-2.3.6/arm-linux/lib,根文件系统中只需要其中的加载器与动态库，将其复制到根文件系统中：

```
#mkdir –p /workspace/nfs_root/lib
#cd/workspace/tools/gcc-3.4.5-glibc-2.3.
6/arm-linux/lib#cp*.so*/workspace/nfs_root/lib –d
```

3.3.3 构建 etc 目录

该目录用于存放各种配置文件，init 进程将根据/etc/inittab 文件创建其它子进程，如配置 IP 地址、挂接文件系统，以及最后的 shell 启动等。本文创建 3 个文件：etc/inittab(为 init 进程创建其它子进程的依据)、etc/init.d/rcS(自动运行的命令)、etc/fstab(文件系统)。

3.3.4 构建 dev 目录

在/dev 目录下创建各设备文件，这涉及的设备有：/dev/mtdblock*(MTD 块设备)、/dev/ttySAC*(串口设备)、/dev/console、/dev/null 等，而其它设备文件可当系统启动起来后，通过 cat /proc/devices 查看内核中注册的设备，并一一创建相应的设备文件。

最后建立若干需要的空目录,如 proc、mnt、tmp、sys、root 等。现在 nfs_root 下得到一最小根文件系统，将其制作为映象文件 nfs_root.jffs2 后，烧入 NAND Flash 中。

```
#tftp 0x30000000 nfs_root.jffs2
#nand erase 0x200000 0x800000
#nand write.jffs2 0x30000000 0x200000 $(filesize)
```

4 结语

本文详细论述了在基于 ARM920T 核心的 S3C2410 平台上构建嵌入式 Linux 系统的过程，包括交叉开发环境的建立，引导加载程序 U-Boot、Linux 操作系统内核针对特定目标平台的移植，以及根文件系统的建立等。最终系统在

(下转第 31 页)

(上接第 26 页)

目标平台上运行稳定、可靠，结果证明方法可行，这对其它嵌入式系统的开发同样具有参考意义。本文创新部分在于其针对特定嵌入式硬件平台的 BootLoader 的程序实现和 Linux 的系统移植，因为平台独特的硬件环境，一些程序代码要严格依赖硬件设备进行设计。

参考文献

1 李善平,刘文锋等编著.Linux 与嵌入式系统.北京:清华大学出版社, 2003.

2 廖日坤编著.ARM 嵌入式应用开发技术白金手册.北京:中国电力出版社, 2005.

3 杜春雷编著. ARM 体系结构与编程.北京:清华大学出版社, 2004.

4 康一梅等编著.嵌入式软件设计.北京:机械工业出版社, 2007.

5 Bovet DP. Understanding the Linux Kernel(3rd Edition). O'Reilly, 2002.

6 刘名博,邓中亮.基于 ARM 的嵌入式 Linux 操作系统移植的研究.计算机系统应用, 2006,15(11):87－88.