# 嵌入式 Linux 系统在 ARM 平台上的构建①

李宗海[1] 陈蜀宇[2] 李海伟[1]

(1.重庆大学 计算机学院 重庆 400044；2.重庆大学 软件学院 重庆 400044)

**摘 要：** 嵌入式系统在人们的日常生活中使用越来越广泛。主要研究了如何在 ARM 平台上构建嵌入式系统，讲述了 u-boot 的工作原理、启动流程，详细介绍了 Linux 内核、u-boot 的裁剪和编译以及根文件系统的制作过程，最后在 SBC2410 硬件平台上成功的构建了嵌入式 Linux 系统。

**关键词：** ARM; 嵌入式 Linux; 内核移植; u-boot 移植; busybox;

## Construction of the Embedded Linux System Based on ARM Platform

LI Zong-Hai[1], CHEN Shu-Yu[2], LI Hai-Wei[1]

(1.Computer Science Department, Chongqing University, Chongqing 400044, China;

2.Software Engineer Department, Chongqing University, Chongqing 400044, China)

**Abstract:** The embedded system, installed in the micro electronic products, is used widely in our daily life. The paper illustrates the process of building a scalable embedded system on the ARM platform effectively. In the beginning, the theory of u-boot and Linux kernel is introduced. Then, the paper gives some details about u-boot, such as initialization, adjusting and so on. After that, the process of building a root file system is given as well. Finally, an experimental embedded Linux system based on ARM SBC2410 platform is constructed successfully.

**Keywords:** ARM; embedded Linux; transplantation of the Linux kernel; transplantation of u-boot; busybox

## 1 引言

嵌入式系统已经成为当今最为热门的领域之一，它迅猛的发展势头引起了社会各方面人士的关注，广泛渗透到人们工作、生活中的各个领域，与我们的生活息息相关，嵌入式处理器已经占分散处理器市场份额的 90%以上。

与此同时，嵌入式 Linux 操作系统也在嵌入式领域中蓬勃发展，它不仅继承了 Linux 源码开放、内核稳定高效、软件丰富等优势，而且还具备支持广泛的处理器和硬件平台、占有空间小、成本低廉、结构紧凑等特点[1]，是嵌入式操作系统的理想选择。本文主要研究在 ARM9 平台上搭建嵌入式 Linux 系统。

## 2 嵌入式Linux系统的构成

在嵌入式系统中，操作系统内核是不能够直接运行的，在操作系统内核运行之前必须运行一段程序，我们称之为 bootloader，它类似于 PC 机中的 BIOS 程序，通过这段程序，可以完成硬件设备的初始化，并建立内存空间的映射图，从而将系统的软硬件带到一个合适的状态，为操作系统内核的运行做准备。

嵌入式 Linux 系统要想正常运行，除了操作系统内核外，还必须有根文件系统，二者缺一不可，整个嵌入式系统的构成如图 1 所示。

① 基金项目:重庆市自然科学基金(CSTC2008BB2307)
收稿时间: 2010-02-07;收到修改稿时间:2010-03-25

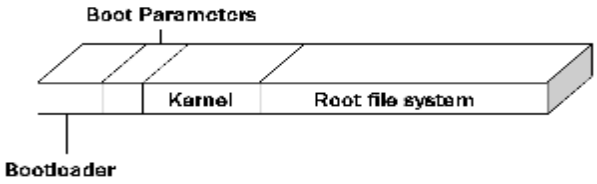


图 1　嵌入式系统构成

## 3　Bootloader移植

系统加电或复位后，所有的 CPU 通常都从某个由 CPU 制造商预先安排的地址上取指令，而基于 CPU 构建的嵌入式系统通常都有某种类型的固态存储设备(如本平台中的 Flash)被映射到这个预先安排的地址上，该地址上存放的代码就是 bootloader。通常，bootloader 是严重依赖于硬件实现的，因此，在嵌入式世界里建立一个通用的 bootloader 几乎是不可能的。

### 3.1 U-boot 介绍

常见的 bootloader 有很多种，而 u-boot 是使用最广泛的 bootloader 之一。U-boot，全称 Universal Boot Loader，是遵循 GPL 条款的开放源码项目，由 FADSROM、8xxROM、PPCBOOT 逐步发展演化而来。其源码目录、编译形式与 Linux 内核很相似；U-boot 不仅支持嵌入式 Linux 系统的引导，还支持 NetBSD、VxWorks、QNX、RTEMS、ARTOS、LynxOS 等嵌入式操作系统的引导[2]。另外，硬件方面，u-boot 支持 PowerPC、MIPS、 x86、ARM、NIOS、Xscale 等诸多常用系列的处理器。U-boot 项目的开发目标就是支持尽可能多的嵌入式处理器和嵌入式操作系统。

### 3.2 U-boot 启动流程

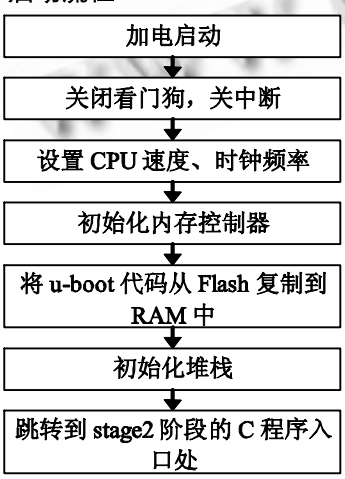


图 2　Stage1 工作流程

与大多数 bootloader 一样，u-boot 的启动流程也分为 stage1 和 stage2 两个阶段，stage1 阶段通常用汇编语言实现，主要完成基本硬件初始化、设置堆栈等工作，为执行 stage2 阶段的 C 语言代码做好准备，其具体的工作流程如图 2 所示。

Stage2 阶段通常用 C 语言实现，以便于实现更复杂的功能和更好的代码可读性及可移植性，并引导操作系统内核，其工作流程如图 3 所示：

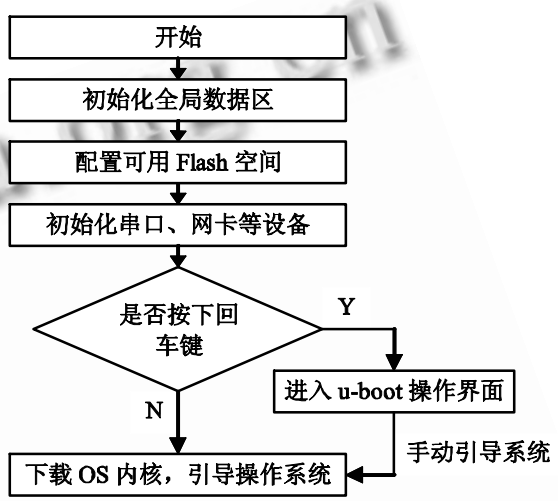


图 3　Stage2 工作流程

### 3.3 U-boot 源代码组织

board ：平台依赖，存放开发板相关的目录文件，每一套板子对应一个目录，如 smdk2410(arm920t)；

cpu ： 平台依赖，存放 CPU 相关的目录文件，每一款 CPU 对应一个目录，如 arm920t、i386 等目录；

lib_arm ： 平台依赖，存放对 ARM 体系结构通用的文件，主要用于实现 ARM 平台通用的函数；

common ： 通用，通用的多功能函数实现，如环境、命令、控制台相关的函数实现；

include ： 通用，头文件和开发板配置文件，所有开发板的配置文件都在 configs 目录下；

lib_generic ： 通用，通用库函数的实现；

net ： 通用，存放网络协议的程序；

drivers ： 通用，通用的设备驱动程序，主要有以太网接口的驱动，nand 驱动等；

### 3.4 与开发板 sbc2410 相关的重要文件

SBC2410 开发板，其硬件资源为：型号为 S3C2410X 的 ARM9 处理器、2 片 32M SDRAM、一

片 64M Nand Flash、一片 2M Nor Flash、一个串口 COM0，一个标准 JTAG 接口等等。

include/s3c24x0.h ：该文件中定义了 s3x24x0 芯片的各个特殊功能寄存器（SFR）的地址；

cpu/arm920t/start.s ：该文件包含了 u-boot 中 stage1 阶段的代码，负责初始化硬件环境，把 u-boot 从 flash 加载到 RAM 中去，然后跳到 lib_arm/board.c 中的 start_armboot 中去执行；

lib_arm/board.c ：该文件包含了 u-boot 中 stage2 阶段的代码，初始化全局数据结构以及设备和控制台；

board/smdk2410/flash.c ：该文件中包含 sbc2410 开发板的资源配置，如内存地址，flash 型号，外围芯片如网络芯片等；

顶层目录下的 Makefile 文件 ：该文件负责 u-boot 整体编译配置，每一种开发板在 Makefile 中都有自己配置的定义，例如 smdk2410 开发板的定义如下：

```
smdk2410_config : unconfig
    @/mkconfig $(@:_config=) arm arm920t smdk2410 NULL s3c24x0
```

其中各项的含义如下：

arm：CPU 架构

arm920t：CPU 类型

smdk2410：开发板型号

NULL：开发者

S3c24x0：片上系统 SOC

按照上面的定义我们可以在 Makefile 文件中增加自己的开发板信息

```
sbc2410_config : unconfig
    @./mkconfig $(@:_config=) arm arm920t sbc2410 NULL s3c24x0
```

### 3.5 通过交叉编译生成 u-boot.bin 二进制文件

```
make sbc2410_config
make CROSS_COMPILE=arm-linux-
```

### 3.6 烧写

使用 Flash 烧写工具 sjf2410，通过 JTAG 接口将 u-boot.bin 烧写到开发板 SBC2410 的 Nor Flash 中。

## 4 编译内核

Linux 操作系统是一个单内核系统，运行在单独的内核地址空间，具有单内核系统简单和高性能的特点[3]；同时，Linux 系统也汲取了微内核的精华，引入了独具特色的模块化设计、抢占式内核，支持内核线程及动态加载内核模块，从而又具有微内核系统的优点，且避免了微内核性能损失的缺陷，同时兼具二者的优点[4]。

Linux 操作系统的模块化设计使得用户可以根据具体的应用编译出满足要求的最精简的操作系统内核，大大降低了嵌入式系统的资源消耗，使其更适合嵌入式领域的应用[5]。通过下面的步骤可以编译出一个 Linux 操作系统内核。

进入 Linux 源代码的顶层目录，执行以下命令：

```
make mrproper ARCH=arm
/*清除原有内核的配置与中间文件*/
make menuconfig ARCH=arm
/* 在图形界面中根据自己的需要选择模块，对于不清楚的按照其默认设置，完成后保存退出 */
make dep ARCH=arm CROSS_COMPILE=arm-linux-
/* 建立依赖关系 */
make zImage ARCH=arm CROSS_COMPILE=arm-linux-
/* 编译内核 */
```

编译操作完成后，最终生成操作系统内核[6]

arch/arm/boot/zImage

## 5 构建根文件系统

根文件系统是 Linux 系统启动时必须挂载的文件系统，它和普通的文件系统的区别在于它包含了 Linux 启动时所必须的目录和关键性的文件，例如 Linux 启动时都需要有 init 目录下的相关文件，在 Linux 挂载分区时 Linux 一定会找/etc/fstab 这个挂载文件等，根文件系统中还包括了许多的应用程序的 bin 目录等。任何包括这些 Linux 系统启动所必须的文件的系统都可以成为根文件系统。

### 5.1 创建根文件系统目录结构

```
mkdir bin dev lib proc sbin sys usr mnt tmp var usr/bin usr/lib usr/sbin lib/modules [7]
/* 创建目录结构 */
mknod -m 666 console c 5 1 /* 创建设备文件 */
```

mknod －m 666 null c 1 3　　/*创建设备文件*/

cp etc-linux/*　etc/ -rf　　/* 复制配置文件 */

5.2 安装内核模块

进入 linux 内核源代码顶层目录，执行下面命令：

make modules ARCH=arm CROSS_COMPILE=arm-linux-

make modules_install ARCH=arm INSTALL_MOD_PATH

=/home /rootfs

模块安装完后在文件系统的 lib/modules 下会产生内核生成的模块。

5.3 编译安装 busybox

BusyBox 被称作是嵌入式开发的瑞士军刀：实用、短小、稳定。

Busybox 可以理解为一个 Linux 的命令集合，我们在进行 Linux 操作时所需要的常用命令，都可以在 busybox 里找到，但 busybox 又不是简单的将所有的命令集合在一起，它采用了一种非常巧妙的方式，即“使用一个程序完成所有的事”。

平时我们用 ls、vi 等命令，都要用到 glibc 的相关调用，如果每个命令都静态链接这些调用，势必每个命令都会很大，因此在通常的发行版中，都会动态链接 glibc，可是 glibc 的动态库本身就很大，这在 PC＋Linux 平时上还可以接受，但在嵌入式系统中，这就太大了，而且又不是所有的库函数都使用。

一般采用两种解决办法，一种是裁剪 glibc，另一种就是使用 busybox，busybox 把 ls、vi 等程序的 main 函数改名后，全部链接在一起，然后静态链接 glibc，这样，只有需要的调用的代码才会链接进来，从而使整个 busybox 程序可能都比 glibc 的动态库小。

5.3.1 裁剪 busybox

make clean　清除旧的编译文件

make menuconfig　执行该命令以后，进入配置主菜单。

首先进入菜单 Build Options，选择“Do you want to build BusyBox with a Cross Compiler?”选项，并在“Cross Compiler prefix”栏中输入交叉编译器安装的位置，如 /usr/local/arm/2.95.3/bin/arm-linux-，注意最后是 arm- linux-，不能加其他字符或空格；

然后在“Any extra CFLAGS options for the compiler”栏中输入需要匹配的 arm-linux 系统源码目录中头文件所在的位置，如“/home/linux /include”，然后返回主菜单；

再根据系统需要，进入所需命令工具所在的子菜单中，选择相应的命令项，如需要 insmod 命令，就在“Linux Module Utilities”子菜单中选中；

选择完成所需的全部命令后，保存配置文件并退出；

5.3.2 编译并安装 busybox

Make

make install

5.4 制作 ramdisk

genext2fs －b 8192 －d /home/roofts ramdisk

gzip -9 －r ramdisk

根文件系统构建完成，最终生成文件 ramdisk.gz

## 6 下载OS内核并配置开发板环境变量

6.1 首先配置宿主机和开发板的 IP 地址，并启动 tftp 服务器

6.2 将 zImage 和 ramdisk.gz 下载到开发板的 Nor Flash 中

erase 0x30000 1fffff　/*擦出 NorFlash 为下载软件腾出空间*/

tftp 0x30008000 zImage　/* 使用 tftp 将操作系统内核下载的开发板内存中 */

cp.b 0x30008000 0x30000 0xba07c　/* 使用 u-boot 命令将操作系统内核从内存写入 Nor Flash 中 */

tftp 0x30800000 ramdisk.gz　/* 使用 tftp 将根文件系统下载的开发板内存中 */

cp.b 0x30800000 0xeb000 5f4fc　/* 使用 u-boot 命令将根文件系统从内存写入 Nor Flash 中 */

setenv bootcmd cp.b 0x30000 0x30008000 0xba07c \;cp.b 0xeb000 0x30800000 5f4fc \;go 30008000

saveenv　　/* 设置开发板启动参数 */

## 7 结束语

嵌入式系统的使用越来越广泛，嵌入式软件开发也随处可见，而针对各种不同的硬件环境，嵌入式软件系统的移植与构建，是每个嵌入式程序员的必须掌握的技能，本文详细介绍了嵌入式 Linux 系统在 ARM 平台上的移植过程，为嵌入式系统后续软件开发做好准备，希望对嵌入式开发人员有一定的参考价值。

嵌入式 Linux 系统成功移植到 SBC2410 开发板，通过超级终端可以看到系统成功启动，如图 4 所示：


图 4 SBC2410 平台上的嵌入式 Linux 系统

**参考文献**

1 Bovet DP, Cesati M. Understanding the Linux Kernel. 3rd Edition, O'Reilly Media, 2005.10－12.

2 孙琼.嵌入式 Linux 应用程序开发详解. 北京：人民邮电出版社, 2006.155.

3 于渊. Orange’S:一个操作系统的实现.第二版.北京：电子工业出版社, 2009.301－302.

4 Love R. Linux 内核设计与实现.第二版.北京:机械工业出版社, 2006.5－6.

5 Rubini A, Corbet J. Linux 设备驱动程序.第三版.北京:中国电力出版社, 2006.22.

6 宋宝华.Linux 设备驱动开发详解.北京:人民邮电出版社, 2008.50－55.

7 陈海明.基于 U 盘 Linux 系统的制作及应用[硕士学位论文].北京:中国地质大学, 2009.