

基于 ARM 的嵌入式 Linux 操作系统移植的研究

The Reseach of Porting Embedded Linux Operating System Based on ARM

刘名博 邓中亮 (北京邮电大学电子工程学院 北京 100876)

摘要:随着 ARM 技术的广泛应用,建立基于 ARM 构架的嵌入式操作系统已成为当前研究的热点。本文结合实例论述了基于 ARM9 核的微处理器上的嵌入式 Linux 操作系统的移植技术,讨论了 uClinux 内核的移植过程中内核代码裁剪,介绍了移植 uClinux 的基本实现过程。

关键词:ARM 嵌入式操作系统 移植 Linux

1 前言

目前,在嵌入式系统里基于 ARM 核的嵌入式处理器已经成为市场主流。随着 ARM 技术的广泛应用,建立面向 ARM 构架的嵌入式操作系统也就成为当前研究的热点问题。在众多的嵌入式操作系统里,开发人员选择比较多的是 Linux,这是因为它公开源代码,可以让任何人将其修改移植到自己的目标平台系统里使用。ARM Linux 支持包括 ARM7、ARM9、StrongARM 等系列的 ARM 处理器,这些 ARM 处理器都带有内存管理单元(MMU),而对于不带有内存管理单元的 CPU,一般采用 uClinux 作为其操作系统。本文就是根据实际项目,研究如何将 uClinux 内核(使用的 uClinux 版本:2.4.18)移植到 ARM 构架的目标平台上。

2 目标平台介绍

本文讨论的嵌入式目标平台采用的是大唐微电子科技有限公司开发的基于 ARM946E 核的 COMIP 嵌入式处理器,COMIP 内部集成了 LCD 控制器,支持 SRAM、SDRAM 和 FLASH 等多样大容量外部存储器。本文所用平台在系统的统一编址空间里,经重映射后的地址空间分配为:SRAM 位于 0 ~ 0x00080000,FLASH 位于 0x02000000 ~ 0x02200000,SDRAM 位于 0x08000000 ~ 0x08800000。其中 FLASH 地址烧写了 Bootloader、uClinux kernel 以及 Jffs2 文件系统。

3 uClinux 移植分析

3.1 uClinux 启动的 Bootloader 介绍

uClinux 的 Bootloader 有许多开源代码,如 Blob、Redboot、U-boot 等,Bootloader 有以下功能:

- (1) 禁止所有中断;
- (2) 设置存储器参数;
- (3) 设置 GPIO 参数;
- (4) 设置处理器的时钟,初始化各个模式下寄存器与堆栈;
- (5) 使能 I catch, D catch;
- (6) 跳转到 c 代码,初始化串口,设置 uClinux 启动参数;
- (7) 然后跳转到 uClinux 的内核启动地址。

3.2 uClinux 启动参数

uClinux 启动可以在 Bootloader 中输入相应的启动参数,启动参数放入 SDRAM 中固定的地址,以方便 uClinux 在启动过程中取得相应的地址的数据,并解释执行。启动参数的定义在:linux/include/asm/setup.h。

Linux 2.4.x 以后的内核都期望以标记列表(tagged list)的形式来传递启动参数。启动参数标记列表以标记 ATAG_CORE 开始,以标记 ATAG_NONE 结束。每个标记由标识被传递参数的 tag_header 结构以及随后的参数值数据结构来组成。数据结构 tag 和 tag_header 定义在 Linux 内核源码的 include/asm/setup.h 头文件中。

在嵌入式 Linux 系统中,通常需要由 Bootloader 设置的常见启动参数有:ATAG_CORE、ATAG_MEM、AT-

AG_CMDLINE、ATAG_RAMDISK、ATAG_INITRD 等。

Linux 内核在启动时可以以命令行参数的形式来接收信息,利用这一点我们可以向内核提供那些内核不能自己检测的硬件参数信息,或者重载(override)内核自己检测到的信息。比如,我们用这样一个命令行参数字符串“console = ttyS0,115200n8”来通知内核以 ttyS0 作为控制台,且串口采用“115200bps、无奇偶校验、8 位数据位”这样的设置。

3.3 uClinux 内核自解压过程

COMIP 在经过 Bootloader 初始化后,跳转到 uClinux 的内核起始地址,即 head - comip. S,一般 uClinux 有两个入口的说法,一个就是 Head. S(本版本为 head - comip. S),另一个是 head - armv. S 的,其实真正的入口是 head - armv. S。这是由于 uClinux 编译完成后有两个版本,一个是 .ram 版本,一个是 .rom 版本,.rom 版本是 uClinux 可执行文件的压缩版本,.ram 为未压缩版本。Head. S 为 .rom 版本的入口地址,head - armv. S 为 .ram 版本的入口地址。

head. S 可以说属于 Bootloader 的范畴,主要功能是初始化自解压所需要的空间,获得内核解压后的首地址 ZRELADDR(由 ZRELADDR 定义所得,本系统是 0x08008000);然后调用 decompress_kernel,(该函数也有初始化串口,提供串口的输入输出的函数,在 include/asm - armnommu/arch - comip/uncompress.c 中定义),在该函数中调用 gunzip(),对系统压缩内核解压,解压的内核地址放在 ZRELADDR 上。最后跳转到 ZRELADDR 的地址区,开始真正的 uClinux 内核启动过程。

3.4 uClinux 内核裁剪

前面的内容介绍了移植 uClinux 一些基本原理和准备工作,然而 uClinux 移植需要更改与添加的内容比较复杂,具体要修改的文件查看表 1。其中如何将 COMIP 处理器的支持加入到 uClinux 中是成功移植 uClinux 到 COMIP 平台的关键。

用户通过改写各个文件目录的 config. in 与 Makefile 文件可以将新的源文件添加到 uClinux 的系统中^[1]。(config. in 为系统的配置文档,主要是配置相关的宏定义、系统信息等等,这样在 Makefile 中根据这些宏定义来决定编译连接那些程序^[3]。至于 uClinux 如何规定代码,数据映象的地址分配,可以参考 \linux\

arch \armnommu \vmlinux. lds^[4]。

表 1 部分内核源代码修改列表

路径与文件(以下都在 /linux - 2. 4. x 下)	修改内容
/arch/armnommu/config. in	添加 COMIP 芯片的定义
/arch/armnommu/Makefile	添加 comip 芯片的运行地址的定义
/arch/armnommu/tools/mach - types	添加 comip 的芯片与体系结构标识的定义
/arch/armnommu/mm/proc - arm6, 7. S	添加 comip 芯片硬件信息到系统支持的芯片列表当中,以及相关信息
/arch/armnommu/kernel/entry - armv. S	添加 comip 检测中断源的宏定义
/arch/armnommu/mach - comip	该目录定义了 uClinux 对 comip 芯片结构的支持
/include/asm - armnommu/arch - comip	该目录定义了 uClinux 对 comip 芯片板级体系结构的支持
/drivers/block/blkmem. c	增加 comip 芯片的 romfs? 的支持
/arch/armnommu/boot/Makefile	自解压的相关存储器的定义
/arch/armnommu/boot/compressed/Makefile	添加 head - comip. S 到编译链表中
/drivers/char/tty. lo. c	添加 console 的初始化程序,在 start_kernel 中能调用到
/include/asm/proc	更改 Vector_base() 为 0

4 uClinux 内核启动

4.1 内核启动入口

head - armv. S 为 uClinux 内核启动的真正入口函数,这段汇编代码很短,程序过程如下:

- (1) 设定当前程序状态,禁止 FIQ、IRQ 进入 SVC 模式;
- (2) 清空 BSS 段;
- (3) 判断保存 processor 类型与 Architecture 类型;
- (4) 跳转到 start_kernel。

4.2 内核启动流程

从 FLASH 里把压缩的内核映象复制到 SDRAM 内,并且把根盘文件系统所在地址参数传递给内核。内核复制到 SDRAM 之后,进行内核解压,执行内核启动。内核启动过程中根据 Bootloader 传递过来的地址参数去寻找根盘文件系统,将其加载到嵌入式系统上。这样,整个 uClinux 被引导启动起来,进入正常工作状态。

(下转第 44 页)

5 结束语

本文根据 COMIP 目标平台,分析了如何将 uClinux 移植到 ARM9 构建的嵌入式系统上的主要技术和基本流程,掌握这些移植的技术和流程,对于开发嵌入式系统是十分重要的。目前已经成功移植 uClinux 到 COMIP 目标平台,并能顺利启动和稳定运行。

参考文献

1 Daniel P. Bovet, Understanding the Linux Kernel

(2nd Edition) [M]. O'Reilly, 2002.

- 2 刘峥嵘等编著,嵌入式 Linux 应用开发详解[M],北京:机械工业出版社,2004.7。
- 3 王学龙编著,嵌入式 Linux 系统设计与应用[M],北京:清华大学出版社,2001.8。
- 4 李明,ARMLinux 的移植过程及分析[J],电子设计应用,2003.7。