面向嵌入式视频处理平台的 Linux移植

潘 冬,李久贤,金立左
(东南大学自动化学院,江苏省南京市 210096)

摘 要:Linux系统是嵌入式视频处理平台的关键组成部分。介绍了一种实用的 Linux移植方法,详细介绍了各个模块的设计,形成了一个完整的 Linux移植体系,包括交叉编译环境创建、BootLoader设计、Linux内核移植以及 LCD、USB设备驱动程序的开发,并以 TMS320DM6446为开发平台,实现了 Linux在其上的移植,为实时视频处理应用开发提供了良好平台。

关键词:嵌入式视频处理平台;ARM;Linux;移植
中图分类号:TP311.54

0 引 言

嵌入式系统开发已经进入 32位时代,在当前数字信息技术和网络技术高速发展的后 PC时代,嵌入式系统已经广泛地渗透到科学研究、工程设计、军事技术等各个方面。

嵌入式系统通常由硬件和软件两个大部分组成。其硬件部分的核心部件就是各类嵌入式微处理器,并配置存储器、I/O设备、通信模块等必要的外设。目前市场上主流销售的 32位嵌入式处理器有 Motorola、MIPS、ARM等系列,其中 ARM以其体积小、成本低、功耗低、性能高等特点成为嵌入式系统设计的首选。

软件部分一般由嵌入式操作系统和应用软件组成。嵌入式操作系统是一种支持嵌入式应用的操作系统软件,它负责全部软硬件资源的分配和调度、控制协调等活动。从 20世纪 80年代末开始,陆续出现了很多典型的嵌入式操作系统,如 Linux、μC/OS、Windows CE等,其中使用最广泛、最受欢迎的是 Linux,这是由于其源代码公开、可移植性好等优点[1]。

1 嵌入式视频处理平台和 Linux系统移植

本文开发的嵌入式视频处理平台在达芬奇 (Da-Vinci)数字媒体技术平台 TMS320DM6446上进行的。此平台是以嵌入式处理器 ARM为中心,由存储器、I/O设备、通信模块以及电源等必要的辅助接口组成。它的工作流程如图 1所示。摄像头将视频信号传输进来后,再通过视频采集卡转换成数字信号然后送入 TMS320DM6446,经过处理后通过视频输出接口在 LCD(液晶显示器)上显示,在此过程中可以由 USB口上所接的操纵杆进行控制,以及与存储设备进行存取操作。

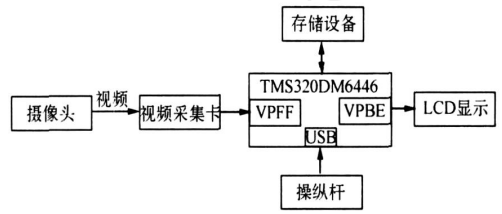

图 1 嵌入式视频处理平台的组成

此嵌入式视频处理平台主要应用于视频和图像的处理,如进行视频跟踪、图像的编解码等。

本文详细阐述如何在 TMS320DM6446平台上进行 Linux系统移植,形成了一个完整的 Linux移植体系,为后续在此平台上的开发搭建了一个良好的平台,其移植流程如图 2所示。

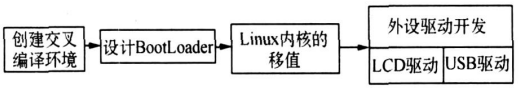

图 2 Linux系统移植流程

2 交叉编译环境的建立

开发一个嵌入式 Linux系统,首先要建立良好的交叉编译环境。所谓交叉编译环境,是由编译器、连接器和解释器组成的综合开发环境。交叉编译是嵌入式系统开发过程中的一项重要技术,它的主要特征是某机器中执行的程序代码不是在本机编译生成,而是由另一台机器编译生成。一般把前者称为目标机(target),后者称为宿主机(host)。在宿主机上编译好适合目标机运行的代码后,通过宿主机到目标机的调试通道将代码下载到目标机,然后由运行于宿主机的调试软件控制代码在目标机上运行调试,其交叉编译开发模型如图 3所示。

收稿日期:2008-06-25;修回日期:2008-08-10。

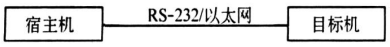

图 3　交叉编译开发模型

建立 ARM的交叉编译环境主要用到的开发工具有: binutils、gcc、glibc。其中 binutils是二进制文件的处理工具,它主要包含了一些辅助开发工具,例如 objdump显示反汇编码、nm列出符号表、readelf显示 elf文件信息及段信息等。这些工具在嵌入式开发初期,尤其是移植调试操作系统时非常有用; gcc是用来编译内核代码的工具,可以编译汇编语言和 C语言的程序,生成 ARM的代码; glibc是一个提供系统调用和基本函数的 C语言库,所有动态链接的程序都要用到它。将这些开发工具包下载到宿主机上进行编译、安装,即可创建 ARM的交叉编译环境[2]。

3　BootLoader的设计

BootLoader即引导加载程序,是在操作系统内核运行之前运行的一段程序。它建立起操作系统运行的环境,包括初始化硬件、建立存储空间映射和传递给操作系统一些基本的配置参数等。因此, Bootloader是非常重要的组成部分,它独立于操作系统,必须由用户自己设计,而且其实现高度依赖于硬件。在系统存储的空间分配结构中 BootLoader、内核启动参数、内核映像和根文件系统映像的关系如图 4所示。

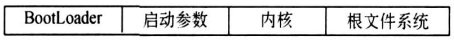

图 4　BootLoader所处的层次位置

BootLoader的作用是正确地调用内核来执行。系统开机后,执行的第 1条指令是从 Flash的 0x00地址开始的, BootLoader程序就是放在此。由于它是直接操作硬件且依据硬件环境不同而代码不同,所以适合用汇编语言写,以达到短小精悍执行效率高的目的;内核从 Flash复制到 SDRAM时,采用 C语言实现,能实现较复杂的功能,因此 BootLoader的设计分为两个阶段。用汇编语言实现的放在第 1阶段,主要完成硬件初始化,设置 SDRAM,然后把 Boot-Loader从 Flash复制到 SDRAM的起始地址,即 2M处,最后内存重映射, Flash地址从 0x00-0x1ff映射成 0x1000000-0x11fff, SDRAM地址 0x200000-0x1lfff映射成 0x00-0xfff,至此控制权交给了用 C语言实现的 loaderkernel( )函数,就进入了第 2阶段。第 2阶段是用 C语言实现的,它主要完成内核从 Flash到 SDRAM的复制,然后控制权交给 Kernel,流程如图 5所示。这样设计代码会具有很好的可读性和可移植性[3]。

本系统 BootLoader的第 1阶段设计包括:

a)关闭看门狗程序,屏蔽所有中断;

图 5　BootLoader启动流程

b)设置处理器时钟和工作频率, TMS320-DM6446中 ARM9的工作频率为 300 MHz;

c)初始化外部寄存器;

d)初始化堆栈指针;

e)复制 BootLoader的第 2阶段到 RAM空间中,使用一条跳转语句跳转到第 2阶段的 C程序如入口处。

第 2阶段用 C语言编写,具体步骤如下:

a)设置通用 I/O口参数;

b)初始化内存映射和内存管理单元;

c)初始化 mtd设备;

d)复制 Flash中的 Kernel映像和根文件系统映像到 RAM空间中;

e)跳转到内核的第 1条指令处,跳转时需要满足这些条件: R0 = 0, R1 =机器类型 ID, R2 =启动参数,同时禁止中断 (IRQ和 FIQ), CPU设置为保护模式,关闭 MMU和数据 Cache。

这样,本系统的 BootLoader就设计完成了,下面就可以进行 Linux内核移植。

4　Linux内核移植

Linux内核主要由 5个子系统构成:

a)进程调度 (Process Scheduler):负责控制进程对 CPU的使用。

b)内存管理 (Memory Manager):标准 Linux的内存管理支持虚拟内存,进程代码、数据和堆栈的总量可以超过实际内存的大小。

c)虚拟文件系统 (Virtual File System):隐藏了不同硬件的具体细节,为所有设备提供统一的接口。

d)网络接口 (Network Interface):负责支持标准的网络通信协议和各种网络硬件设备。

e)进程间通信 (Inter-Process Communica-tion):支持进程间各种通信机制。

根据嵌入式系统的特点,要使嵌入式 Linux系统具备一定的功能且保持小型化,应包括启动加载程序、

内核、初始化进程,以及硬件驱动程序、文件系统、必要的应用程序等。

不管是哪一款嵌入式处理器,完成移植工作就要修改所有与体系结构有关的代码,主要指内核入口、处理器初始化、I/O口映射等。具体操作如下[1]:

(1)修改配置文件

a) 打开根目录下的 Makefile文件,指定目标平台 ARCH = arm;指定交叉编译器 CROSS_COMPLE = arm-linux-gcc;

b) 打开 /arch/arm 目录下的 Makefile文件,添加内核起始运行地址,即 image. ram应下载的位置,该位置一般在 RAM区起始地址偏移 0x8000处;

c) 打开 /arch/arm /boot目录下的 Makefile文件,指定 Bootloader的压缩内核解压后数据的输出地址。

(2)编译 Linux内核

在完成上述工作后,开始编译 Linux内核,生成目标代码。在内核源代码目录下依次键入以下命令:

a) make clean:清除以前构造内核时生成的所有目标文件、模块和临时文件;

b) make dep:搜索 Linux输出与源代码之间的依赖关系,并以此生成依赖文件;

c) make menuconfig:调用菜单式的配置内核界面,内核配置的选项非常多,根据自己系统的具体情况选择合理的配置,在内核配置时选上相应型号的硬件;

d) make zImage:编译内核,生成压缩的 Linux内核目标代码 zImage文件;

e) make modules:编译块模块驱动程序,凡是在 menuconfig中被选为 <M >的都会在这条命令运行时被编译。

至此,已编译好能在本系统上运行的 Linux内核。

(3)创建 JFFS2文件系统

文件系统是 Linux系统的重要组成部分。本系统使用 mkfs. jffs2工具创建 JFFS2文件系统。首先建立 /bin、/sbin等目录,然后复制命令工具到 /bin文件夹,复制系统控制程序到 /sbin目录下,复制应用程序运行时所需的库到 /lib,库文件可从 PC机的交叉编译工具安装目录下复制。最后键入命令:mkfs jffs2 - o jffs2 root jffs2,生成 JFFS2根文件系统。

上述工作完成后,将 BootLoader、Linux内核、文件系统烧写到 TMS320DM6446的 Flash中,这样就能运行 Linux系统了。

5 设备驱动程序开发

5.1 Linux设备驱动程序开发步骤

Linux系统设备分为字符设备、块设备和网络设备3种。其设备驱动的开发主要包括:

a)在驱动程序源文件中定义 file_opera-tions结构,并编写出设备需要的各个操作函数,对于设备不需要的操作函数用 NULL初始化,这些操作函数将被注册到内核中。

b)定义一个初始化函数,在 Linux初始化时会调用此函数。此函数包含:初始化驱动程序要用到的硬件寄存器;初始化与设备相关的参数;注册设备;注册设备使用的中断和函数;其他一些初始化工作。

c)对于驱动程序的使用,可以进行静态编译,也可以进行动态编译。静态编译是指将设备驱动程序添加到内核中,动态编译是指将设备驱动程序编译成驱动模块。

本嵌入式系统主要用于视频处理,涉及到的外设主要是显示设备和输入设备。这里采用的显示设备是 LCD,而输入设备是通过 USB接口与系统相连的。

5.2 LCD显示驱动程序开发

LCD的设备驱动程序属于字符设备的驱动,应按照字符设备的规则编写。在 Linux下进行 LCD显示用 Framebuffer技术,这是提取图形的设备,是用户进入图形界面很好的接口。Linux内核根据硬件描述抽象出 Framebuffer设备,供用户态的进程直接进行写屏。可以将 Framebuffer看成是显示内存的一个映像,将其映射到进程地址空间之后,就可以直接进行读写操作,写操作立即反应在屏幕上。Framebuffer的设备文件一般存放在 /dev这个目录下,对此设备文件进行操作即可实现图像的显示[5]。

LCD显示驱动程序主要包括:

a)LCD驱动的文件结构:包括打开设备文件、设备文件其它操作、关闭设备文件等;

b)LCD的打开:LCD设备以读写的方式打开;

c)LCD设备的硬件初始化:包括注册 LCD设备、卸载 LCD设备等;

d)LCD相关结构的设置:以获取显存起始地址、分别率、色深等;

e)映射内存区的操作:包括初始化显存清零等,将摄像头采集到的图像数据读至显存处,以显示图像;

f)LCD控制输出:包括得到命令、画水平线、画垂直线、画圆等;

g)LCD的关闭。

将上面的内容用程序实现,进行动态编译。通过后,将 LCD驱动模块进行移植加载,一个完整的 LCD驱动就开发完毕了。

5.3 USB驱动程序开发

与 LCD设备不同,USB既不属于字符设备,也不

属于块设备,而是一个新的设备类别,设计框架和流程如下[6]:首先,提供一个“.o”的驱动模块文件,且在一开始就加载运行。USB驱动就会根据其类型向系统注册。注册成功后,系统会反馈一个主设备号,这个主设备号就是其唯一标识。USB驱动就是根据主设备号创建一个放置在 /dev目录下的设备文件。要访问此硬件,可用 open、read和 write等命令访问相应的设备文件,驱动就会接收到相应的 read或 write函数,根据模块中相对应的函数进行操作。驱动流程见图 6。

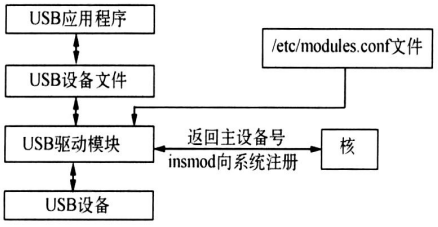

图 6　USB驱动流程

USB驱动的具体设计过程如下:

a)USB驱动的注册。USB驱动程序在注册时会发送一个命令给函数 register_chrdev,通常在驱动程序的初始化函数中。当 USB设备插入时,为了使 linux-hotplug(Linux中 USB等设备热插拔支持)系统自动装载驱动程序,需创建 MODULE_DEVICE_TABLE,在此过程中需将 USB的主设备号传递给相应的函数。

b)USB设备的打开。打开设备是通过调用 file_operations结构中的函数 open( )来完成的。其主要完成的任务是:检查设备相关错误,如果是第一次打开,则初始化硬件设备;识别次设备号;使用计数增 1。

c)USB设备的释放。释放设备是通过调用 file_operations结构中的函数 release( )来完成的。它的作用正好与 open()相反,通常要完成这样的工作:使用计数减 1,如果使用计算为 0,则关闭设备。

d)USB设备的控制信息与数据读写。USB设备驱动程序可以通过文件操作结构中的函数向应用程序提供对硬件进行控制的接口,同时读写操作也要通过此函数来完成。

e)USB驱动的注销。当从系统卸载驱动程序时,需要注销 USB设备,这样必须编写一个注销函数 unregister_chrdev。

6　结束语

本文基于 TMS320DM6446平台实现了 Linux移植,包括创建交叉编译环境、BootLoader的设计、Linux内核移植以及 LCD、USB设备驱动程序开发,为实时视频处理应用开发创建了一个良好的嵌入式平台,在此平台上可进一步进行应用程序、GUI及视频处理算法开发与测试。

参　考　文　献

[1]孙天泽,袁文菊,等. 嵌入式设计及 Linux驱动开发指南:基于 ARM9处理器[M]. 北京:电子工业出版社,2005: 1-10, 126-137.

[2]尤盈盈,孟利民. 构建嵌入式 linux交叉编译环境[J]. 计算机与数字工程,2006, 34(6): 30-32.

[3]白伟平,包启亮. 基于 ARM的嵌入式 BootLoader浅析[J]. 微计算机信息,2006, 22(4-2): 99-100, 53.

[4]徐英慧,马忠梅,等. ARM9嵌入式系统设计:基于 S3C2410与 Linux[M]. 北京:北京航空航天大学出版社,2007: 353-362.

[5]刘峥嵘,张智超,等. 嵌入式 Linux应用开发详解[M]. 北京:机械工业出版社,2004.

[6] Programming gide for Linux USB device drivers[EB/OL]. http://usb.cs.tum.edu.

潘　冬(1982-),男,硕士研究生,主要研究方向为模式识别与智能系统。

Linux System Transplanting for Embedded Video Processing Platform

PAN Dong, LI Jiuxian, JIN Lizuo

(School of Automation, Southeast University, Nanjing 210096, China)

Abstract: Linux system is an important part of embedded video processing platform. This paper presents a method of Linux system transplanting and introduces the design of each module, so as to form a complete system of Linux transplanting, including how to establish cross-compiler environment, customize the BootLoader, transplant the Linux kernel, and develop the device drivers, such as LCD and USB drivers. These functions were implemented and tested on TMS320DM6446 platform, which may provide a good environment for future real time video processing applications development.

Keywords: embedded video processing platform; ARM; Linux; transplanting