

基于嵌入式 μ CLinux 设备驱动程序的实现

肖竞华¹, 夏红霞²

(1. 武汉科技大学 计算机学院, 湖北 武汉 430081;

2. 武汉理工大学 计算机学院, 湖北 武汉 430070)

摘 要:文中通过在嵌入式 μ CLinux 上实现字符型设备驱动程序的添加的实例,介绍了嵌入式 Linux 系统的设备管理、设备驱动程序的框架和实现设备驱动程序的添加的方法,以能够提供给大家一个开发 μ CLinux 下设备驱动程序的向导。使之在开发各种设备的驱动程序中得到很好的应用。

关键词:嵌入式系统; μ CLinux; 驱动程序

中图分类号: TP311

文献标识码: A

文章编号: 1673-629X(2006)07-0177-03

Realization Device Driver Added in Embedded μ CLinux

XIAO Jing-hua¹, XIA Hong-xia²

(1. Computer School, Wuhan University of Science and Technology, Wuhan 430081, China;

2. Computer School, Wuhan University of Technology, Wuhan 430070, China)

Abstract: Introduces management in embedded μ CLinux system of driver and an approach of adding driver through an instance of realizing device driver added in embedded Linux system. Hope that can supply a programming guide for μ CLinux device drivers. And this has been very useful for developing device drivers.

Key words: embedded system; μ CLinux; driver program

随着信息技术和网络技术的快速发展,嵌入式系统已成为一个备受关注的热点。嵌入式系统融合了计算机软硬件技术、通信技术和微电子技术,用户可以根据具体应用需求,把相应的微处理器直接嵌入到具体的应用系统中。然而在嵌入式开发中,经常会遇到许多新的外部设备需要系统支持的情况,在嵌入式 μ CLinux 上实现设备驱动程序的添加的方法,成为开发人员十分关注的一个问题。

1 μ CLinux 嵌入式系统

μ CLinux 是针对控制领域的嵌入式 Linux 操作系统,由 Linux 2.0, 2.4 内核派生而来,沿袭了主流 Linux 的绝大部分特性,适合不具备内存管理单元的微处理器(MMU)。有无 MMU 支持是 Linux 与 μ CLinux 的基本差异。最小的嵌入式 μ CLinux 仅需 3 个基本元素:内核引导实用程序、 μ CLinux 微内核、初始化过程。同时,根据需要添加:硬件驱动程序、1 个或多个应用进程(以提供所需功能)、1 个文件系统(可能在 ROM 或者 RAM 内)、存储半瞬态数据和提供交换空间的磁盘、TCP/IP 网络栈等。

μ CLinux 与 Linux 最大区别在于 Linux 采用了虚拟存

储技术^[1]。虚拟存储技术的实现基于局部性原理。一个程序在运行之前,没有必要全部装入内存,而是将那些当前需要运行的部分页面装入内存运行,其余暂时留在硬盘上。如果发现所需页面不在内存中,操作系统将产生一个页失效异常,导致操作系统把需要运行的部分页加载到内存中。虚拟存储技术能提供内存保护,进程不能以非授权方式访问或修改页面,内核保护单个进程的数据与代码,以防止其他进程修改它们。

Linux 是针对有内存管理单元的处理器而设计的,虚拟内存地址是经过 MMU 进行地址转换后映射为实际的物理地址。 μ CLinux 没有 MMU,不能使用虚拟存储技术,但仍然采用存储器分页管理机制,系统启动时把实际存储器分页,在加载应用程序时分页加载。由于没有 MMU 管理,所以 μ CLinux 采用了实存储器策略。 μ CLinux 对内存的访问是直接的,所有程序访问的地址都是实际的物理地址。操作系统对内存空间没有保护,各个进程实际上是共享一个运行空间。一个进程在运行前,系统必须为进程分配足够的连续地址空间,然后全部载入到主存储器连续空间中。

2 设备驱动程序

2.1 嵌入式系统的设备管理

Linux 的设备管理系统是操作系统的重要组成部分,

收稿日期:2005-10-25

作者简介:肖竞华(1955-),女,湖北武汉人,副教授,硕士研究生导师,研究方向为系统软件开发、操作系统、信息安全研究。

是输入输出子系统。系统分为上下两个部分:一部分是下层的、设备相关的,即所谓的设备驱动程序,它直接与相应的设备打交道,并向上提供一组访问接口;另一部分是上层的、与设备无关的,这部分根据输入输出请求,通过特定设备驱动提供的接口,与设备进行通信^[2]。

目前绝大多数嵌入式系统的软件系统存放在 Flash 上,包括 μ CLinux 操作系统、应用系统的程序以及各种设备的驱动程序都被直接固化到 Flash 上,系统启动时,程序代码直接在 Flash 上开始运行。

根据设备性质的不同设备驱动程序通常可归结为 4 种类型:字符设备(char)、块设备(block)、网络接口(net)和其他设备驱动程序模块,它们都在 Linux drivers 目录下的子目录,如 char, block, net, cdrom, scsi, sound 等。Linux 系统通过这种文件系统实现软件设备驱动程序。所有的字符和块设备的驱动程序都支持文件操作接口,因此用户对任何一个设备的存取如同对文件操作一样——只要特定的设备驱动程序支持这一抽象的文件接口即可。对于 Linux 系统来说,虽然设备的种类繁多,但是为了便于使用,将各种设备都作为(特殊)文件来处理,对设备可以进行 open, read 和 write 等操作。

2.2 设备驱动程序的框架

Linux 的设备驱动程序与内核的接口可分为 3 部分:与系统启动代码的接口对设备进行初始化;与内核接口通过 file-operation 来完成;与设备的接口对设备进行读写操作。

每一个设备驱动程序实质上是用来完成特定任务的一组函数集。驱动程序拥有一个称为 file-operation 的数据结构,其中包含指向驱动程序内部大多数函数的指针。引导系统时,内核调用每一个驱动程序的初始化函数,将驱动程序的主设备号以及程序内部的函数地址结构的指针传输给内核。这样,内核就能通过设备驱动程序的主设备号索引访问驱动程序内部的子程序,完成打开、读、写等操作。

μ CLinux 系统引导时,通过 sys-setup(linux/fs /file systems.c)进行系统初始化。而 sys-setup 又调用 device-setup 进行设备初始化。字符型初始化由 chr-dev-init(uclinux /Linux/drivers/char/mem.c)完成。

为 Linux 内核编写驱动程序,一般的工作方式都是在现成的驱动程序的基础上针对特殊的硬件设备作相应的改动,并编写几个基本的函数并向虚拟文件系统(VFS)注册。当上层应用要使用该设备时,VFS 就会调用相应的设备函数。

3 实现设备驱动程序的添加

驱动程序可以按照两种方式编译:一种是静态编译进内核;另一种是编译成模块以供动态加载。由于 μ CLinux 不支持模块动态加载,因而设备驱动程序只能静态编译进 μ CLinux 内核^[3]。下面以在嵌入式 μ CLinux 系统中需新

增一个字符型设备为例,介绍设备驱动程序的添加的一系列步骤(设其驱动程序为 test.c)。

3.1 改动设备驱动程序 test.c 源代码

第一步,将原来的:

```
# include <linux/module.h>
# include <linux/version.h>
char kernel-version[] = UTS-RELEASE;
```

改为:

```
# ifdef MODULE
# include <linux/module.h>
# include <linux/version.h>
char kernel-version[] = UTS-RELEASE;
# else
# define MOD-INC-USE-COUNT
# define MOD-DEC-USE-COUNT
# endif
```

第二步,新建函数 init init-test(void),将设备注册写在此:

```
result = register-chardev(254, "test", &test-fops);
```

3.2 复制改动好的驱动程序到嵌入式 μ CLinux 系统

将 test.c 复制到 μ CLinux /Linux/driver/char 目录下,在该目录下 Makefile 中增加如下代码:

```
ifeq ($ (CONFIG-TESTDRIVE), y)
L-OBJS += test.o
Endif
```

3.3 新设备驱动程序的注册

在/uclinux/romdisk/romdisk/dev/目录下创建设备:

```
@ test c 254 0
```

其中 test 是设备名, c 表示字符设备, 254 是主设备号, 0 是次设备号。做法为:

```
$ touch @ test c 254 0
```

在/uclinux/linux/drivers/char 目录下 mem.c 中,

init-char-dev-init()函数中增加如下代码:

```
bool 'support for testdrive' CONFIG-TESTDRIVE y
```

3.4 重新编译内核

在/uclinux/linux/目录下运行 menuconfig 进行编译(在 menuconfig 字符设备选项中可以看到刚添加的'support for testdrive'选项,并且已被选中);make clean;make dep;make linux;make linux.test;make linux.data;cat linux.text linux.data>linux.bin.

通过 make 的一系列编译,所有内核和应用程序源代码将被编译,并会产生 imagez.bin 二进制影像文件和 clv4 文件。

3.5 添加驱动程序到 Flash

下载的方法如下,通过串口进入目标系统:

在 μ CLinux 目录下运行 make sloaf 将新的 imagez.bin 写入 Flash 中。

目标系统重新启动完成后,在目标系统的 dev 目录下增加了一个名称为 test 的设备,在 bin 目录下增加了一个

test 文件,到这里,在 μ CLinux 中添加设备驱动程序的工作可以说是完成了,你就可以使用自己的新设备 test 了。

4 设备驱动机制的关键问题

4.1 内存操作

在设备驱动程序中动态开辟内存,不是用 malloc 函数,而是用 kmalloc 函数。这个函数运行很快,除非它被阻塞。或者用 get-free-pages 直接申请页。释放内存用的是 kfree 或 free-pages。

4.2 中断处理

设备驱动程序通过调用 request_irq() 函数申请中断,将一个硬件处理函数挂到相应的处理队列中^[4]。 μ CLinux 系统中对中断的处理属于系统的核心部分,因此,如果外部设备与系统之间以中断方式进行数据交换,就必须把该设备的驱动程序作为系统内核的一部分。通过 request_irq 函数调用就可以把相关的中断号和具体的中断处理程序相关联。

```
Int request_irq(unsigned int irq,
Void (* handler)(int,void *,struct pt_regs *),
Unsigned int log flags,
Const char * device);
```

其中:handler 是中断处理子程序指针,中断产生时由操作系统自动调用此函数;参数 irq 为中断号;pt_regs 为中断发生时寄存器的内容;device 为设备名称;flags 确定了中断处理程序的一些特性。Flags=SA_INTERRUPT 表示该中断处理函数是快速中断,在其运行时所有中断都被屏蔽;不设置 SA_INTERRUPT 表示慢速处理程序,这种程序运行时除了正在处理的中断外,其他中断都没有被屏蔽;flags=SA_SHIRQ 表示共享此中断处理程序。函数调用成功返回 0 值,返回-INVAL 表示中断号超出范围或者 handler=NULL,返回-BUSY 表示中断已经被占用且

不能共享。

4.3 用户空间和内核空间

设备驱动程序是在“内核空间”中运行的,而一般应用程序则是在“用户空间”中运行。在 μ CLinux 系统中,操作系统内核程序在最高级执行(也称为“管理员态”),在内核空间可以执行对硬件的任何操作^[5],而一般的应用程序则运行在最低级(“用户空间”)。在用户空间操作,系统禁止对硬件的直接访问和对内存的未授权访问。

μ CLinux 通过系统调用和硬件中断完成从用户空间到内核空间的控制转移。执行系统调用的内核代码在进程的上下文上执行(它代表调用进程操作,而且可以访问进程地址空间的数据)。但与此不同,处理中断的函数代码相对进程而言是异步的,而且与任何一个进程都无关。

5 结束语

文中通过介绍在 μ CLinux 操作系统下设备驱动的相关问题,使用户了解嵌入式系统设备驱动的开发流程。借助 μ CLinux 的开放性,能够灵活快捷开发嵌入式应用系统。

参考文献:

- [1] 邹思秩.嵌入式 Linux 的设计与应用[M].北京:清华大学出版社,2002.
- [2] Rubini A,Corbet J. LINUX 设备驱动程序[M].魏永明,骆刚,姜君译.北京:中国电力出版社,2002.
- [3] 贾明,严世贤.Linux 下的 C 编程[M].北京:人民邮电出版社,2001.
- [4] 翟鸿鸣.Linux 系统实时性能增强方法的研究[J].微机发展,2003,13(S):1-3.
- [5] 王俊峰,冯志彪.如何编写设备驱动程序[J].微机发展,2005,15(2):25-27.

(上接第 113 页)

同时投影任意输出主元空间的某一个平面上而形成的二维等高图。Contour 图的优点直观,尤其是网络输出主元维数较低的时候,但随着主元空间维数的升高,全面反映系统分布的图将会越来越多而且精确性会受到影响。如主元空间为三维时需要 3 张图,四维时候需 6 张图。图 5 是正常数据联合概率密度的前两个输出主元的二维投影图,内外线分别为模型样本输出的 95% 和 99% 置信线,测试正常样本几乎都落在了 95% 置信区间内。

5 结束语

目前分布式系统在当前各个领域应用广泛,它的介入既让这些领域取得了新的发展与变革,也给自己充实了新的概念与方法。文中将分布式和神经网络结合起来,将其应用到工业企业内部便于对工业设备的监测。考虑到系统的异构性,采用 CORBA 规范来实现,充分保证了系统

的跨平台性和服务端的实时性两大优点。从对 FCCU 数据的实验结果分析,软件具有一定的实用价值。

参考文献:

- [1] 陈前.关于大系统的状态监测与故障诊断[J].振动、测试与诊断,2002,22(3):163-174.
- [2] 刘道敏,陈前.多元统计过程状态监测分布式系统的实现[J].工业控制计算机,2004(11):21-23.
- [3] 虞和济,陈长征,张省,等.基于神经网络的智能诊断[M].北京:冶金工业出版社,2000.
- [4] 朱三元,钱乐秋,宿为民.软件工程技术概论[M].北京:科学出版社,2002.
- [5] Borland/Inprise 公司.VisiBroker for C++[M].北京:机械工业出版社,2000.
- [6] McFarlane R C, Reineman R C, Bartee J F, et al. Dynamic Simulator for a Model IV Fluid Catalytic Cracking Unit[J]. Computers Chem Engng, 1993, 17(3): 275-300.