IDE 驱动程序分析及 SATA 驱动程序移植①

洪承煜, 杨尚琴

(中国石化石油物探技术研究院, 南京 211103)

摘 要: SATA 硬盘已经是目前多媒体应用中被广泛使用的存储介质, 而有些多媒体产品的 Linux 版本中, 并不支持 SATA 驱动框架. 本文通过对 Linux 下 IDE 驱动框架及其相关的 Linux 驱动代码的分析, 提供了一种在 IDE 驱动程序框架下移植 SATA 驱动程序的方法. 最后以 SiI3512 为例, 介绍了 SATA 驱动程序的移植过程. 对于其它此类系统的移植具有很好的借鉴作用.

关键词: Linux 驱动; IDE; SATA 驱动移植

Analysis of IDE Driver and Migration of SATA Driver

HONG Cheng-Yu, YANG Shang-Qin

(SINOPEC Geophysical Research Institute, Nanjing 211103, China)

Abstract: SATA disk is Storage Media that has already been widely used in a multimedia applications currently, while many versions of Linux in Multimedia products that its driver framework does not support SATA drives. By analysing the driver framework in linux and its associated code analysis, this paper provides a way for porting SATA drives in IDE driver framework. Finally to SiI3512 example, the paper describes the porting process of SATA driver. This has a good reference for migration of other similar systems.

Key words: Linux Driver; IDE; Migration of SATA Driver

在很多嵌入式多媒体产品开发中, 目前广泛使用的 SATA 硬盘存储是一个很好的解决方案, 但是由于有些嵌入式多媒体产品受制于 BSP(板级支持包)等与 Linux 版本的相关性, 使得必须为不支持 SATA 驱动程序框架的 Linux 版本开发新的 SATA 驱动程序. 本文通过分析 Linux 2.4.15 内核的 IDE 驱动程序框架, 并从 Linux 2.4.23 内核中获取 SiI3512 驱动程序, 从而借助 ATA 转 SATA 接口芯片 SiI3512[3]来达到访问 SATA 硬盘的目的.

1 IDE驱动程序框架分析[1]

在 Linux 2.4.15 内核中, 不同的体系结构(如 x86、mips、arm 等)有着不同的 IDE 最大接口数, 一般情况下有主、次两个 IDE 接口, 而每个 IDE 接口又可以接主、从共两个 IDE 设备(如硬盘、软盘、光盘等), 即最多可以有 4 个 IDE 设备.

在 Linux 2.4.15 内核中, 用 ide_hwifs[]数组记录所有的 IDE 接口, 而数组的每个元素都是一个 ide_hwif_t 类型的结构体, 代表一个 IDE 接口. Linux2.4.15 系统初始化时如果检测到一个 IDE 接口, 就把相应表项中的 present 属性设成 1; 同时, 初始化时如果检测到某个接口上有 IDE 设备相连, 也将 ide_hwif_t 结构体内相应的类型为 ide_drive_t 的 drives[]中的 present 属性也设成 1, 并根据从系统的 CMOS 芯片中读到或检测到的各项参数设置到 ide_hwif_t 结构体中. 其中 ide_hwif_t 数据类型是对 IDE 接口的描述, 而 ide_drive_t 数据类型是对连接在具体 IDE 接口上的“IDE 设备”的描述. 比如, 如果在系统的主 IDE 接口上检测到有主、从两个 IDE 设备相连, 就把这两个 IDE 设备的参数分别填入 ide_hwifs[0]中的 drives[0]和 drives[1]中, 并把它们的 present 属性设置成 1. 又如, 如果在次 IDE 接口上连接着一个 MATSUSHITA CDROM,

① 收稿时间:2013-06-18;收到修改稿时间:2013-07-22

那就把它的参数填入 ide_hwifs[1]里面的 drives[0], 并且把 ide_hwifs[1] 中的属性 major 设置成 MATSUSHITA_CDROM_MAJOR.

另外, 在 ide_drive_t 数据类型中有个类型为 ide_driver_t 的属性 driver, 它在系统初始化过程中, 根据检测到 IDE 接口上设备的类型而设置成指向不同的 ide_driver_t 数据类型的实例. 比如, 对于 IDE 硬盘, 它指向一个数据类型为 ide_driver_t 的实例, 实例名为 idedisk_driver. 同样还有 idetape_driver、ide_cdrom_driver 以及 ide_floppy_driver 实例, 它们分别代表着可以连接到 IDE 接口上的不同类型的设备. 如果 ide_drive_t 数据类型中的属性 drives[]非空, 但是它的 driver 指针却是 NULL, 就说明初始化时虽然检测到了硬盘的存在, 但是却因某种原因未能完成对设备以及设备类型的初始化. 由以上分析可得如下图 1 所示.

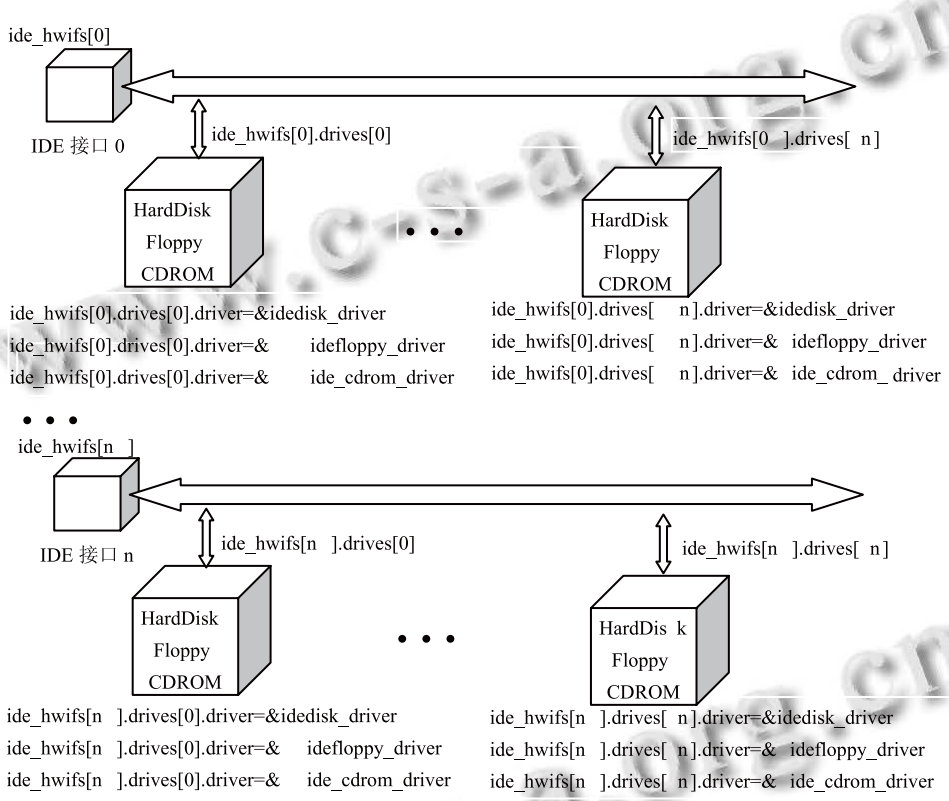

图 1 IDE 驱动硬件抽象图

从图 1 中可以看出 Linux 2.4.15 内核 IDE 驱动程序框架的初始化的顺序是: (1)初始化 IDE 接口; (2)识别挂接的 IDE 设备; (3)为相应的 IDE 设备挂接具体的驱动程序, 下面分别进行详述.

(1) 在初始化 IDE 接口时, 主要是为每个 IDE 接口(驱动程序中用 ide_hwifs[]表示)确定 10 个 IDE 地址, 通过这些地址可对 IDE 设备作一切访问. 比如, 选择 IDE 设备(一个 IDE 接口上可以接两个 IDE 设备, 这两个 IDE 设备共用这 10 个地址, 访问它们之前需要发出不同的选择命令)、发出命令、查询状态、读写数据. 即, 对 IDE 接口的初始化主要包括确定这 10 个地址, 即 ide_hwifs[].io_ports[10]的值, 就软件而言, 就是设置相应的 ide_hwifs[]实例中的属性值.

(2) 在识别挂接的 IDE 设备时, 其实在(1)中确定了 IDE 接口的地址、读写函数后, IDE 驱动程序就会通过这个 IDE 接口自动识别所挂接的 IDE 设备. 即: ①检查是否有挂接了 IDE 设备; ②识别是硬盘、光盘还是软盘; ③注册中断处理函数.

下面来详细说明一下这个 3 步过程.

一个 IDE 接口上最多可以挂接两个 IDE 设备, 可以是硬盘、光盘或软盘等——这是可以通过不同的命令序列识别出来的. 这些命令序列分两类: ATA、ATAPI. 前者对应硬盘, 后者对应光盘或软盘等.

如果我们把 IDE 设备当作一个巨大的可读写的数组

的话, 那么 IDE 设备上必然有一些只读的数据来标识它: 容量、生产厂商、扇区数/磁头数/柱面数及是否支持多扇区读写等, 现称这些信息为“IDE 设备 ID”, 大小为 512 字节, 这些信息读出后会保存起来. 从软件角度看, ide_hwifs[].drives[0]对应主 IDE 设备、ide_hwifs[].drives[1]对应从 IDE 设备: 系统如果检测到有 IDE 设备相连, 则将 ide_hwifs[].drives[].present 置 1; 然后使用 ATA/ATAPI 命令序列查询所接 IDE 设备类型(硬盘/光盘/软盘), 读取“IDE 设备 ID”, 保存在 ide_hwifs[].drives[].id 中; 最后设置 ide_hwifs[].drives[].media, 其取值可能是 ide_disk、ide_floppy、ide_cdrom 等.

最后注册中断处理函数, 对于同一个 IDE 接口下的主、从两个 IDE 设备, 它们共用一个中断线. 识别出一个 IDE 接口下所有 IDE 设备后, 发现有 IDE 设备后才注册中断.

(3) 在挂接具体驱动程序时, 会使用(2)中已经获取了 IDE 设备的参数, 包括在 ide_hwifs[].drives[].media 中标明了 IDE 设备类型等. 加载这些具体驱动程序时, 它们会一一遍历每个 IDE 设备, 即比对每个 ide_hwifs[].drives[]其 ide_hwifs[].drives[].media 项, 匹配的话就将 ide_hwifs[].drives[].driver 指向相应的结构. 比如, 对于 IDE 硬盘, 它指向一个数据类型为 ide_driver_t 的实例 idedisk_driver. 同类的实例还有 idetape_driver、ide_floppy_driver 以及 ide_cdrom_driver. 以硬盘为例, 利用 idedisk_driver 实例中提供的函数进行读写硬盘了.

最后, 就可以识别 IDE 设备分区, 并使用 IDE 设备了.

综上所述, Linux 2.4.15 内核的 IDE 驱动程序框架分为 3 层: IDE 接口层(hwifs[]); IDE 设备驱动器层(hwifs[].drives[]), 具体驱动程序(hwifs[].drives[].driver). 实际上, 移植 SiI3512 驱动程序, 主要的工作也就是让操作系统能识别到 SiI3512 对应的 IDE 接口层.

2 IDE驱动程序源码分析[2,4]

结合以上 Linux 2.4.15 IDE 驱动程序框架分析, 下面来分析一下其相应的源码.

(1) 初始化 IDE 接口.

① 在 drivers/ide/ide.c 中的 ide_init 函数, 初始化默认的 IDE 接口: ide_init->init_ide_data ->init_hwif_data;

② 在 MIPS 体系结构中, 要在 arch/mips/kernel/setup.c 的 setup_arch 中, 设置“ide_ops = &std_ide_ops”, 其 ide_init->init_ide_data->init_hwif_data 将会利用到 std_ide_ops 结构中的函数对 ide_hwifs[]数组进行默认初始化. 其它体系结构不需要此步骤. 需要说明的是每个 IDE 接口默认的 10 个地址是针对 x86 体系结构的.

③ 识别 PCI 设备上的 IDE 接口: ide_init->ide_init_builtin_drivers->probe_for_hwifs->ide_scan_pcibus->ide_scan_pcidev. 其中函数 ide_scan_pcidev 在 drivers/ide/ide-pci.c 中, 它将内核检测到的 PCI 设备与 ide_pci_chipsets[]数组(在 drivers/ide/ide-pci.c 中定义, 包含了内核 IDE 驱动所支持的 PCI 设备)中的设备相比较, 如果匹配则调用 ide_setup_pci_device 初始化此 PCI 设备上的 IDE 接口, 而 ide_setup_pci_device 在 drivers/ide/ide-pci.c 中定义, 它从数组 ide_hwifs[]中取出未用的项, 调用此设备在 ide_pci_chipsets[]数组中定义的函数来初始化其相应的 ide_hwifs[]项, 即 IDE 接口.

(2) 识别挂接的 IDE 设备.

此步骤函数调用过程是: ide_init->ide_init_builtin_drivers->ideprobe_init->probe_hwif->probe_for_drive->do_probe. 其中函数 do_probe 在 drivers/ide/ide-probe.c 中定义, 它利用前面确定的 IDE 接口的地址发出各类命令序列检测 IDE 设备, 从此函数中用到的一个宏, 可以看出是如何使用前面已确定的 IDE 接口的:

SELECT_DRIVE(HWIF(drive), drive)发出选择主/从 IDE 设备择命令, 定义如下:

```
#define SELECT_DRIVE(hwif,drive)
{     \
      if (hwif->selectproc)                    \
            hwif->selectproc(drive);           \
      OUT_BYTE((drive)->select.all, hwif->io_ports
[IDE_SELECT_OFFSET]);
}
```

由上宏可得, OUT_BYTE 用到了 IDE 接口的地址: IDE_SELECT_OFFSET, 而 OUT_BYTE 实现的是向此地址输出一个字节, OUT_BYTE 的定义通常就是 outb. 但如果访问 PCI 设备的 I/O 空间时, 其读写函数不同于内核默认的 outb, 则需要重新定义.

有前面可知, 每个 IDE 设备都有一些只读信息来标识自己, 这儿称“IDE 设备 ID”, 它的函数调用如下: ide_init->ide_init_builtin_drivers->ideprobe_init->robe_hwif->robe_for_drive->do_probe->try_to_identify->actual_try_to_identify->do_identify->ide_input_data->insw->inw

的 do_probe 函数的后面部分. 其中的 ide_input_data 函数, 它从 IDE 设备上得到字节流数据, 此函数不关心大、小字节序.

(3) 挂接具体驱动程序.

本文以硬盘驱动为例, 代码在 drivers/ide/ide-disk.c 中, 开始执行函数为 idedisk_init, 它主要完成如下 3 步: ①对所有识别出来的 IDE 设备, 如果是硬盘(ide_hwifs[].drives[].media == ide_disk), 则挂接硬盘驱动程序: de_hwifs[].drives[].driver = &idedisk_driver. 调用的函数过程如下: 判断是否硬盘: idedisk_init->ide_scan_devices; 挂接硬盘驱动程序: idedisk_init ->ide_register_subdriver. ②根据“IDE 设备 ID”作一些设置: idedisk_init->idedisk_setup, 比如获取 IDE 设备容量、确定能否多扇区操作等. ③最后调用 ide_register_module (&idedisk_module), 它首先将 idedisk_module 挂入 ide_modules 链表, 然后调用 revalidate_drives, 此函数将依次对每个检测到的 IDE 设备, 调用其驱动程序(比如对于硬盘, 驱动为 idedisk_driver)的 revalidate 函数(比如对于硬盘, 函数为 idedisk_revalidate)进行一些 IDE 设备操作(比如对于硬盘, 识别分区).

完成以上 3 步后, 在系统启动后, 就可访问 IDE 设备了. 比如对于挂接在 IDE 接口 0 上的主硬盘, 可以通过/dev/hda 访问整个硬盘, 通/dev/hda1 访问第一个分区.

3 SiI3512驱动移植[2,4]

SiI3512 是 PIC 转 SATA 的控制芯片, 它通过 PCI 总线提供了 2 个 IDE 接口, 内含 2 个 ATA 与 SATA 的转换部件. 本文就是通过 ATA 接口访问 SATA 设备.

依据上面分析, IDE 驱动有 3 层结构组成, 以下根据这 3 层结构移植 SiI3512 驱动.

(1) 初始化 IDE 接口.

① 在 MIPS 体系结构中, 要在 arch/mips/kernel/setup.c 的 setup_arch 函数中, 将“ide_ops= &no_ide_ops”修改为“ide_ops = &std_ide_ops”, 在后续的申请中断等操作中, 会用到 std_ide_ops 定义的一些函数. 其它体系结构不需要此步骤.

② 初始化默认的 IDE 接口.

函数调用顺序是: ide_init->init_ide_data->init_hwif_data, 在 init_hwif_data 函数中, 会用到 IDE 的接口地址(如 MIPS 体系结构中的 ide_ops->ide_init_hwif_ports[]), 但是这些地址默认是针对 x86 体系结构的, 如果系统不是 x86 体系结构, 后续的操作中如果使用这些地址会导致系统崩溃. 所以在 drivers/ide/ide.c 的 init_hwif_data 函数中强制设置不要探测所有的默认 IDE 接口, 即设 hwif->noprobe = 1.

③ 在 IDE PCI 设备数组 ide_pci_chipsets[]中添加一项 SiI3512 设备, 如下代码结构.

```
{
    DEVID_SiI3512,      /* 设备 ID */
    "SiI3512 Serial ATA",   /* 设备名称 */
    &init_chipset_siimage,  /* SiI3512 的初始化函数 */
    NULL,                   /* ATA CHECK*/
    &init_hwif_siimage,     /* IDE 接口的初始化函数 */
    &init_dma_siimage,      /* DMA 初始化函数*/
    {{0x00,0x00,0x00}, {0x00,0x00,0x00} },      /* enable bits */
    ON_BOARD,               /* bootable */
    0                       /* extra */
}
```

在系统识别 PCI 设备上的 IDE 接口(ide_init->ide_init_builtin_drivers->probe_for_hwifs ->ide_scan_pcibus->ide_scan_pcidev) 时, 检测到 SiI3512 之后, 会调用 init_chipset_siimage 初始化芯片, 调用 init_hwif_siimage 初始化 IDE 接口, 调用 init_dma_siimage 初始化 DMA 通道.

在 init_hwif_siimage 函数初始化 IDE 接口时, SiI3512 有 6 个 PCI BASE 空间, BASE0/BASE1 对应 IDE 接口 0, BASE2/BASE3 对应 IDE 接口 1, 它们都是 I/O 地址空间, 可以通过 inb/outb/inw/outw/inl/outl 等宏操作这些 I/O 地址空间. 但如果 I/O 地址空间的操作函数有区别, 则必须 include/asm/xxx.h 相应的文件中重新定义了这些宏.

(2) 识别挂接的 IDE 设备.

根据不同的体系结构在 linux 2.4.15 中的中断体系特点, 主要做以下修改: 在读写 IDE 设备时, 会调用 disable_irq_nosync 来禁止中断, 调用 enable_irq 来使能中断. disable_irq_nosync, 它们在 arch/xxx/kernel/irq.c 中定义. 如果 enable_irq 等有附加功能就需要在相应的文件中重新定义相应的函数.

(下转第 44 页)

任意工况数据模型, 这对复杂工业系统故障诊断和状态监测具有重要意义.

3 结语

本文主要论述了工况数据模拟生成系统的设计过程, 采用归纳法分析和比较了两种不同的工况数据模拟生成算法. 其中单模型无关系模式的工况数据模拟生成算法较简单, 且易于实现, 但其不能模拟现场设备各工况之间的关系, 也不能模拟环境中的噪声对工况数据模拟生成的影响, 还不能模拟设备运行一段时间后零部件发生衰变的状态, 具有很大的局限性. 多模型有关系模式的工况数据模拟生成算法弥补了单模型无关系模式的工况数据模拟生成算法的不足, 但由于逻辑较复杂, 造成了一定的效率损失.

参考文献

1 严新平,周强.机械系统工况监测与故障诊断.武汉:武汉理工大学出版社,2009.17–30.
2 宏宇.基于 FPGA 的风电监测系统数据采集单元设计[学位论文].北京:北京化工大学,2011.
3 宋扣生.分段函数解析.理科考试研究(高中版),2004,11(8): 8–10.
4 埃克尔. Java 编程思想.北京:机械工业出版社,2007.45–47.
5 赵铁栓,蔡应强.混凝土搅拌输送车液压驱动系统仿真.建筑机械,2005,(8).
6 刘丽珏,张龙祥.计算机软件.JDBC 与 Java 数据库程序设计.北京:人民邮电出版社,2001.
7 Geng Z, Chen J. Investigation into piston-slap-induced vibration for engine condition simulation and monitoring. Journal of Sound and Vibration, 2005, 282(3): 735–751.
8 山拜,达拉拜,黄玉划.几类非高斯噪声模型的转换研究.电子学报,2004,32(7):1090–1093.
9 贺尚红,杨昀梓.基于神经网络的混凝土泵车发动机万有特性建模与工况优化.中南大学学报(自然科学版),2010, 41(4).
10 黄虎,束鹏程,李志浩.风冷热泵冷热水机组结霜工况下工作过程动态仿真及实验验证.流体机械,2000,28(3):49–52.
11 李钟蔚,宋坤.Java 开发实战宝典.北京:清华大学出版社,2010.348–353.
12 Zhou P, Ye W. Application of JFreeChart in statistics and analysis of financial data. Journal of Chongqing Institute of Technology (Natural Science), 2008, 11: 38.

(上接第 143 页)

(3) 挂接具体驱动程序, 此项不需要修改.

通过以上 3 步即可完成 SiI3512 驱动程序的移植. 系统启动之后, 即可使用分区工具进行分区, 以便使用 SATA 硬盘存储多媒体数据.

4 结语

本文通过分析 Linux 2.4.15 内核的 IDE 驱动程序框架和其对应的源码, 摸索和总结出了一套在 Linux 下使用 IDE 框架来移植和使用 SATA 驱动程序的通用方法, 使得驱动开发者可以绕开浩瀚的驱动代码, 简单套用本文介绍的步骤, 即可完成 SATA 驱动程序的移植, 同时也对其它相关的驱动程序移植有一定的参考作用.

参考文献

1 毛德操,胡希明.LINUX 内核源代码情景分析(上、下).杭州:浙江大学出版社,2001.
2 linux 2.4.15、linux 2.4.23 内核源代码.https://www.kernel.org/pub/linux/kernel/v2.4/
3 Silicon Image. Inc.; SiI3512 Data Sheet, 2007.
4 Corbet J, Rubini A, Greg Kroah-Hartman Linux. 设备驱动程序.北京:中国电力出版社,2006.