DOI:10.13398/j.cnki.issn1673-260x.2011.08.011
第27卷 第8期 赤峰学院学报(自然科学版) Vol. 27 No. 8
2011年8月 Journal of Chifeng University (Natural Science Edition) Aug. 2011

小型微内核操作系统内核模型设计与实现

陈云龙，曲 波

(南京晓庄学院 数学与信息技术学院，江苏 南京 21117)

摘 要：系统的微内核仅仅实现一些最为基本的服务,它为整个系统的正常运作提供基础保障.它被实现在核心级,可以执行特权指令.微内核直接与底层硬件打交道,并且通过少量的应用程序编程接口向上层提供一个内核的抽象.微内核部分包括文件系统、输入输出、TTY控制以及中断的响应框架.

关键词：TTY；中断；文件系统；扫描码

中图分类号：TP316.8 文献标识码：A 文章编号：1673-260X(2011)08-0058-02

1 研究背景

随着计算机技术和通信技术的迅猛发展，计算机网络已深入到人们的工作、生活与学习中,近年来,微内核体系结构的应用已经备受关注.微内核是一个小型的操作系统的核心,只有最基本的操作系统功能才放在内核中.非基本的服务和应用程序在微内核之上构造，并在用户模式下执行.本文对微内核体系结构的设计进行了探讨,分析了模块化的设计思想和方法,提出了基于微内核系统模块的环状层次化方案,对微内核体系结构的特点和采用这种体系结构构建操作系统的优势进行了分析.

2 发展现状

操作系统的版本有很多，但是它们有一个共同点就是都是基于一个内核的,在这个内核的基础之上,可以开发出自己需要的版本,由于开发这些软件需要消耗一定的资源,在应用有些版本的时候还需要花钱. 但是LINUX基础版本是开源的,完全免费的,这些足够开发之用,而设计微内核操作系统系统模型就是在这个基础上开发的，这使得不需要花费任何费用就可以搞开发，还可以对系统模型有了更深的认识.

3 系统开发技术环境与关键技术

本系统模型的设计与实现是以LINUX(VMware Workstation + Red Hat LINUX 9 +VM0)作为平台的，涉及到VMware Workstation虚拟机、Red Hat LINUX 9、GUN工具链、Makefile文件管理、LINUX汇编器、虚拟软盘VM0等工具和技术的使用.

4 系统的具体技术实现

4.1 文件系统模块

FAT12文件系统和其它文件系统一样，都将磁盘划分为层次进行管理.从逻辑上划分,一般将磁盘划分为盘符,目录和文件;从抽象物理结构来讲,将磁盘划分为分区,簇和扇区.那么,如何将逻辑上的目录和文件映射到物理上实际的簇和扇区,就是文件系统要解决的问题.

如果让虚拟机直接读取系统生成的可启动软盘镜像，或者将TINIX.IMG软盘挂载到某个目录上，系统肯定会报出“软盘未格式化”或者“文件格式不可识别”的错误.这是因为任何系统可读取的软盘都是被格式化过的,而TINIX.IMG是一个非常原始的软盘镜像.

系统在读取一张软盘的时候，会读取软盘上存储的一些关于文件系统的信息，软盘格式化的过程也就是系统把文件系统信息写入到软盘上的过程.但是不能让系统来格式化TINIX.IMG,如果那样的话,写入的启动程序也会被擦除.所以呢,需要自己对软盘进行格式化.

FAT文件系统的主要信息,都被提供在前几个扇区内,其中第0号扇区尤其重要. 在这个扇区内隐藏着一个叫做BPB(BIOS Parameter Block)的数据结构,一旦把这个数据结构写对了,格式化过程也基本完成了.下面这个表中所示的内容,主要就是启动扇区的BPB数据结构.

FAT文件系统对存储空间分配的最小单位是“簇”,因此文件在占用存储空间时,基本单位是簇而不是字节.即使文件仅仅有1字节大小，系统也必须分给它一个最小存储单元——簇,即存储空间分配的最小单位确定了,FAT的存储空间管理是通过管理FAT表来实现的,FAT表一般位于启动扇区之后,根目录之前的若干个扇区,而且一般是两个表.从根目录区的下一个簇开始,每个簇按照它在磁盘上的位置映射到FAT表里.知道了这些,就可以从文件系统中加载文件并执行,将操作系统引导起来.

4.2 键盘输入模块

扫描码获得之后,要将扫描码和相应的字符对应起来,就要将扫描码解析让操作系统认识.

解析好键盘扫描码后,需要建立一个键盘输入缓冲区,用以防止中断程序收到的扫描码,然后就可以添加任务了.

4.3 屏幕显示控制

4.3.1 显示字符

4.3.2 处理Shift、Alt、Ctrl

现在只能输入简单的字符和数字了，但是还不能处理复杂的字符.比如Shift,按下Shift只能看到一个奇怪的字符.添加以下代码,使其响应这些功能键.

4.3.3 处理所有按键

到目前为止,系统已经可以处理大部分的按键了,但是至少还存在两个问题.

(1) 如果扫描码更加复杂一些,比如超过 3 个字符,如今的程序还不足以很好的处理.

(2) 如果按下诸如 F1、F2 这样的功能键,Tinix 会试图把它当做可打印字符来处理.

4.4 TTY 控制

运行程序的过程很清楚了,轮到每一个 TTY 时不外乎做两件事:

(1) 处理输入——查看是不是当前 TTY,如果是则从键盘缓冲区读取数据.

(2) 处理输出——如果有要显示的内容则显示它.

如果只是这样的话,TTY 任务是很简单的,箭头指的就是函数间的调用关系.Task_tty () 是一个循环,不断的调用 keyboard_read(),而 keyboard_read()从键盘缓冲区得到数据之后会调用 in_process(),将字符直接显示出来.

而系统的设计不能那么简单,需要对上面那张图的任务描述作进一步补充,主要有如下几个方面:

(1) 对于每一个 TTY,都应该有自己的读和写的动作.所以在调用 keyboard_read()时,必须让它知道是哪一个 TTY 在调用.通过为它传入一个参数来做到这一点,这个参数就是指向当前 TTY 的指针.

(2) 为了让输入和输出分离,被 keyboard_read()调用的 in_process()不应该在直接回显字符,而应该将回显的内容交给 TTY 来完成,这样就为每一个 TTY 建立一块缓冲区,用以放置将被回显的字符.

(3) 每个 TTY 回显字符时操作的 CONSOLE 是不同的,所以每个 TTY 都应该有一个成员来记载其对应的 CONSOLE 信息.

基于上面的考虑,新建了两个机构体,分别表示 TTY 和 CONSOLE.如下代码分别表示 TTY 结构和 CONSOLE 结构.

```
/* TTY 结构 */
typedef struct s_tty
{
  t_32 in_buf[TTY_IN_BYTES]; /* TTY 输入缓冲区 */
  t_32* p_inbuf_head; /* 指向缓冲区中下一个空闲位置 */
  t_32* p_inbuf_tail; /* 指向键盘任务应处理的键值 */
  int inbuf_count; /* 缓冲区中已经填充了多少 */
  struct s_console * p_console;
}TTY;
/*CONSOLE 结构 */
typedef struct s_console
{
  //struct s_tty* p_tty;
  unsigned int current_start_addr; /* 当前显示到了什么位置 */
  unsigned int original_addr; /* 当前控制台对应显存位置 */
  unsigned int v_mem_limit; /* 当前控制台占的显存大小 */
  unsigned int cursor; /* 当前光标位置 */
}CONSOLE;
```

这两个实现之后,初始化了一下 TTY,修改了一下输出函数 out_char(),为了能够看到效果,还需要一个切换控制台的函数 select_console(),这个写好之后再添加一个屏幕初始化的函数基本上就可以了.

4.5 中断

在这个模型之中,主要用的中断就是键盘中断和时钟中断

```
PUBLIC void init_clock () // 时钟中断初始化程序 kernel\clock.c
{ /* 初始化 8253 PIT */
  out_byte(TIMER_MODE, RATE_GENERATOR);
  out_byte(TIMER0, (t_8) (TIMER_FREQ/HZ) );
  out_byte(TIMER0, (t_8) ((TIMER_FREQ/HZ) >> 8));
  put_irq_handler(CLOCK_IRQ, clock_handler); /* 设定时钟中断处理程序 */
  //enable_irq (CLOCK_IRQ); /* 让 8259A 可以接收时钟中断 */
}
PUBLIC void init_keyboard () // 键盘中断初始化程序 kernel/keyboard.c
{
  kb_in.count = 0;
  kb_in.p_head = kb_in.p_tail = kb_in.buf;
  put_irq_handler (KEYBOARD_IRQ, keyboard_handler);/* 定键盘中断处理程序 */
  enable_irq(KEYBOARD_IRQ); /* 开键盘中断 */
}
```

5 结束语

这个系统看似很小,但是需要对 LINUX 操作系统、AT&T 语法、GUN 工具链和 Makefile 文件管理等有基本的了解,才能进行开发.当实现之后,你会感觉到学到的和掌握的东西很多,同时也对内核由来更深更全面的了解.

参考文献:

[1]于渊.自己动手写操作系统[M].北京:电子工业出版社,2005.

[2]Third Edition by Robert Love.Linux Kernel Development[M].北京:机械工业出版社,2011.

[3]于渊.Orange.S 一个操作系统的实现[M].北京:电子工业出版社,2009.

[4]红帽软件(北京)有限公司.RedHat Linux 用户基础[M].北京:电子工业出版社,2008.

[5]Claudia Salzberg Rodriguez 等.linux 内核编程[M].北京:机械工业出版社,2006.

[6]黄丽娜,等.Red Hat Linux9.0 基础教程[M].北京:清华大学出版社,2007.

[7]吴国伟,李张,任广臣.LINUX 内核分析及高级编程[M].北京:电子工业出版社,2008.