

一个简易操作系统的设计与实现

曾建周, 李 尧

(内江师范学院 计算机与信息科学系, 四川 内江 641112)

摘 要: 根据操作系统的原理, 分析操作系统的逻辑结构, 设计出一个简易操作系统模型。并利用 NASM 汇编语言和 C 语言编写其源代码, 实现了一个可以从软盘启动计算机的简易操作系统。

关键词: 操作系统; 引导; 内核; 进程; I/O

中图分类号: TP316 **文献标识码:** A **文章编号:** 1671—1785 (2006) S1—0238—05

1 引言

操作系统 (OS) 是计算机体系里面最基础、最重要的系统软件, 它是用户与计算机硬件系统之间的接口, 是计算机系统资源的管理者, 更是用户使用的平台。真正意义上的操作系统主要有以下四个特点: 用户依靠操作系统方便的使用计算机软件、硬件设备, 提高计算机的效率, 操作系统必须具有良好的扩充性, 操作系统必须具有优良的移植性、互操作性和统一的开放环境。操作系统体系是一个非常复杂的知识体系, 它牵涉到的知识很多很复杂, 比如内核研究、进程调度和管理、存储器管理、文件管理和设备管理等技术。

目前对于操作系统教材上都只讲其原理, 不讲实际动手操作及其实现, 所以学生在学习它时感到非常抽象。国内外一些著名高校通常是鼓励学生在学这门课程的同时自己动手开发一个能实现 OS 基本功能的微型操作系统来加强对该课程的掌握。通过实践对于学生充分掌握书本知识、打下扎实的基本功有非常大的好处。在此基础上, 我提出了一个具体的操作系统。

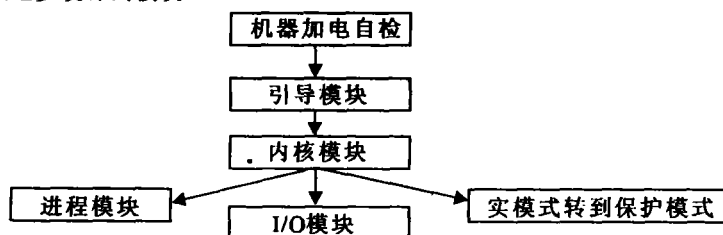
2 总体设计

2.1 问题的提出

计算机操作系统是一个结构化体系, 它主要有几大模块构成: 引导模块、模式转换模块、内核模块、进程模块、I/O 模块、设备管理模块等。

2.2 设计框架

根据软件项目工程的做法逐步设计出模块。



2.3 设计环境

一台安装有 Windows XP 和 Linux (内含 NASM、GCC 工具) 双系统 PC 机、虚拟机软件 (Virtual PC)、调试软件 (Bochs)、绝对扇区读写工具 (FloppyWriter) 等。

3 详细设计

3.1 引导模块 BOOT 设计

引导盘 (本文是以软盘为引导盘) 的设计。计算机电源被打开时, BIOS 首先会加电自检, 然后寻找位于磁盘的第

收稿日期: 2006—04—13

基金项目: 内江师范学院大学生科研项目 (05NSD—098)

作者简介: 曾建周 (1984—), 男, 四川南充人, 内江师范学院计算机与信息科学系 2003 级学生。

0面0磁道1扇区引导代码, BIOS若发现此扇区代码是以0xAA55结束, 则认为它就是一个引导扇区。此外, 为了能让微软识别引导扇区还要加上一个数据结构为BPB的头信息。写好此代码之后在Linux系统中用命令 `nasm Boot.asm -o Boot.bin` 编译成二进制文件即可运行, 然后用软盘绝对扇区读写工具写到软盘或软盘映像即可运行。

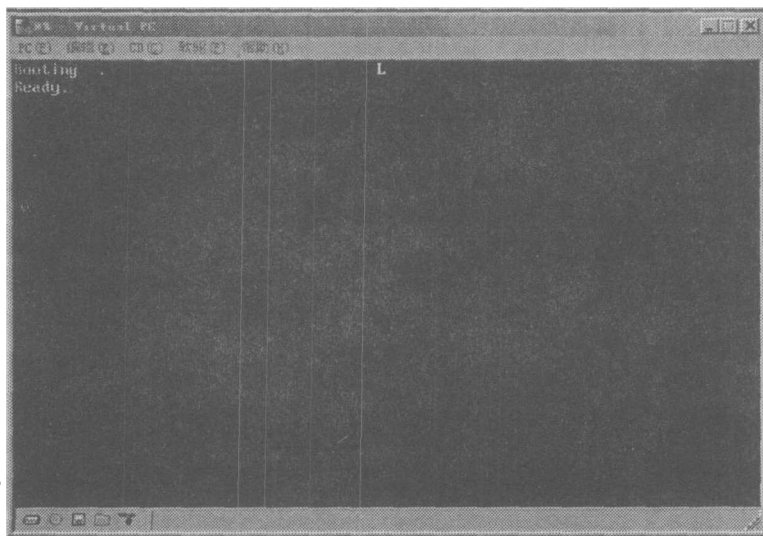
有了以上软盘引导代码, 现在需要解决的问题是将Boot.bin写进内存, 于是可用一段名为Loader.bin的代码把Boot.bin写进内存。Loader.bin生成的方法如下: 用虚拟机软件Virtual PC捕获某个*.IMG(磁盘映像文件), 并将它格式化, 于是*.IMG中留下了Loader.bin。为了简单起见, 可规定Boot.bin只能放在根目录区, 最后把它复制到软盘根目录区中即可。

这个阶段还须解决如何寻找Loader.bin。方法是在软盘的根目录区从头到尾寻找名为Loader.bin的文件, 找到后跳转到Loader.bin代码所在地址。最后软盘中文件存放位置的示意图如图1:

写好这段代码后, 在Linux系统中用命令 `nasm Boot.asm -o Boot.bin` 编译生成Boot.bin, 然后用软盘绝对扇区读写工具写到软盘或软盘映像即可运行。运行结果如图2:

数据区 (长度非固定)
根目录区 (长度非 固定)
FAT2 (长度固定)
FAT1 (长度固定)
引导扇区 (长度固定)

图1



注: 上图打印出Booting表示已经将Boot调入了内存, 然后进入了准备好状态Ready

图2

3.2 内核模块Kernel的设计

将内核加载到主存。在启动引导扇区代码以后它会自动寻找Loader代码, 并将Loader代码加入内存。接着Loader代码将加载内核到内存。在设计内核这段代码时, 主要参考了Andrew S.Tanenbaum和Albert S.Woodhull所著《操作系统: 设计与实现》以及于渊的《自己动手写操作系统》。内核代码主要包括四个文件: kernel.asm、string.asm、klib.asm、start.c, 将这些文件编译链接后得到了内核kernel.bin。

编译连接方法:

```
[root@XXX XXX]# rm -f kernel.bin
[root@XXX XXX]# nasm -f elf -o kernel.o kernel.asm
[root@XXX XXX]# nasm -f elf -o string.o string.asm
[root@XXX XXX]# nasm -f elf -o klib.o klib.asm
[root@XXX XXX]# gcc -c fno-builtin -o start.o start.c
[root@XXX XXX]# ld -s -Ttext 0x30400 -o kernel.bin kernel.o string.o start.o klib.o
```

然后在Virtual PC中用虚拟DOS将编译生成的kernel.bin复制到Boot.bin所在得到软盘或软盘映像即可运行, 到此内核模块制作完成。

3.3 实模式转到保护模式

当把内核加载进内存后就自动从实模式转到保护模式, 为了简化, 笔者设计了分页机制, 在实模式下, 描述内存的地址是逻辑地址 (Segment: Offset), 它经过计算后得到的物理地址是线性的, 即逻辑地址转换成物理地址遵循这样的计算公式:

物理地址(Physical Address)=段值(Segment)*16+偏移(Offset)

而分页机制下(保护模式),段值所在的寄存器还是没变,但是此时它变成了一个索引,这个索引指向页表,在页表中详细的定义了页面的起始地址、界限、属性等内容。分页机制示意图如图3:

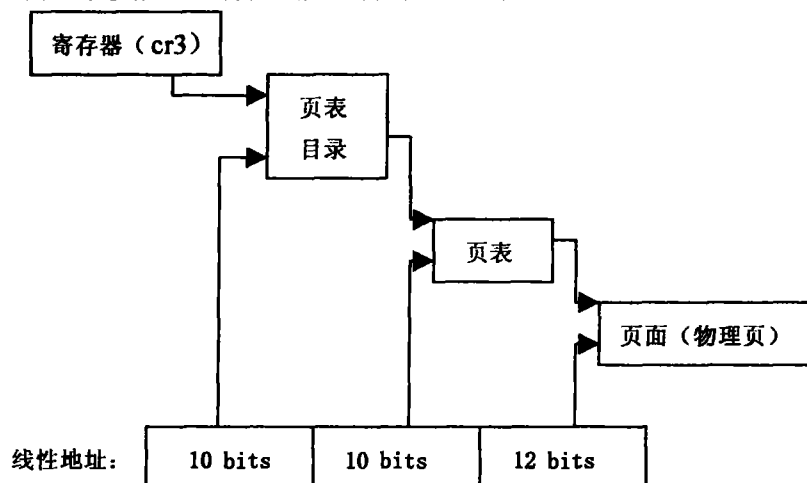
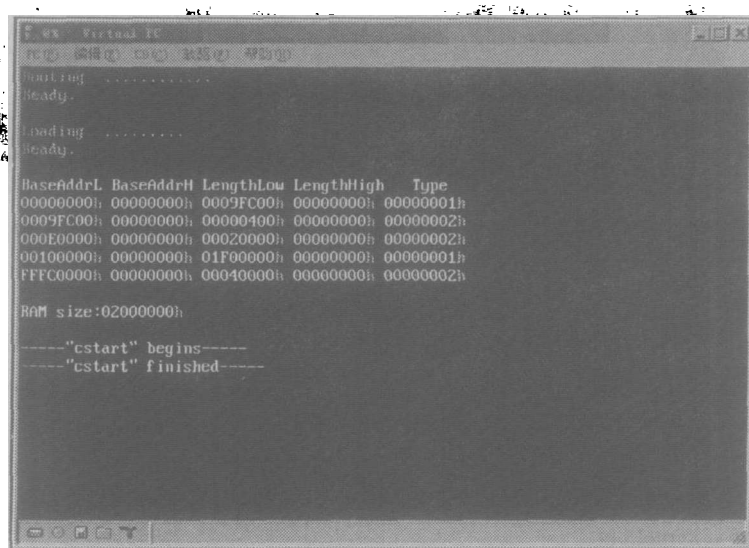


图3

按上述原理编写代码,启动分页机制,并且假设有一个任务,没有局部描述符表和中断描述符表,不允许中断,也不考虑发生异常,甚至没有使用堆栈。则该任务实例执行步骤如下:

(1) 在实模式下为进入保护模式作初始化;(2) 切换到保护模式后进入临时代码段,把部分演示代码传送到预定的内存,然后转演示代码段;(3) 建立页目录表;(4) 建立页表;(5) 启用分页管理机制;(6) 演示在分页管理机制启用后的程序执行和数据存取;(7) 关闭分页管理机制;(8) 退出保护模式,结束。

运行结果如下图:



注:上图显示总共有5段内存被列了出来。

图4

000000h~9FBFFh Type值为1表示可以被该系统使用
 9FC00h~9FFFFh Type值为2表示不能被该系统使用
 E0000h~FFFFFh
 100000h~1FFFFFFh
 FFFC0000h~FFFFFFFFh

内存(RAM)大小: 2000000h=32MB

"cstart"是一个全局函数,起一个复制页表中的属性的作用。

3.4 进程模块的设计

进程是操作系统中最重要最基本的概念之一。进程主要是由若干代码段、数据段和堆栈段和PCB等部分组成进程实体, 设计进程的关键技术有:

- (1) 进程的哪些状态需要保持;
- (2) 进程状态的完整性保存;
- (3) 如何恢复进程状态;
- (4) 进程队列的调度管理;
- (5) 进程的优先级(特权级)管理。

该模块程序主要参考了其他书籍的思想与代码。但诸多进程的性能还未实现, 有待在今后的毕业设计中继续完成。

根据进程其结构, 设计了进程A和进程B, 如图5。它们的进程调流程如下:

- (1) 进程A运行;
- (2) Inter8259发生时钟中断, 进程A被置ring1为ring0(在中断程序中设置进程的两个特权级, 分别为高特权级ring1和低特权级ring0);
- (3) 进程调度下一个要运行的进程B;
- (4) 进程B被置ring0为ring1;
- (5) 进程B运行。

根据上面流程写出其代码。从总体上讲进程模块主要包括三项代码:

- (1) 时钟中断处理程序;
- (2) 进程调度程序;
- (3) 若干进程程序。

随着源代码的增多, 编译链接代码的命令也越来越多, 又由于代码文件分

开放在不同的文件夹里, 要想编译它就比较困难。因此, 在演示程序中设计了三个打印函数交替打印, 以代替三个进程的调度问题。

运行结果如图6:

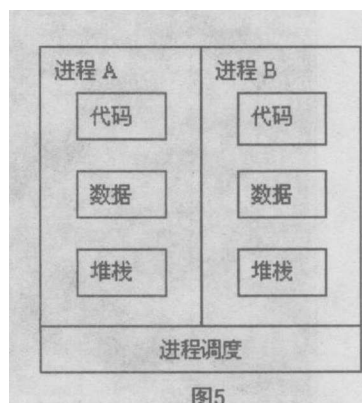


图5

```

File Virtual PC
File Edit View Help
Booting .....
Ready.

Loading .....
Ready.

BaseAddrL BaseAddrH LengthLow LengthHigh Type
00000000h 00000000h 0009FC00h 00000000h 00000001h
0009FC00h 00000000h 00000400h 00000000h 00000002h
000E0000h 00000000h 00020000h 00000000h 00000002h
00100000h 00000000h 01F00000h 00000000h 00000001h
FFFC0000h 00000000h 00040000h 00000000h 00000002h

RAM size:02000000h

-----"cstart" begins-----
-----"cstart" finished-----
-----"tinix_main" begins-----
A0x0.B0x3.C0x4.A0x65.B0x67.C0x69.A0xCB.B0xCF.C0xD0.ABx132.B0x199.C0x19C.A0x19D.C
0x200.A0x201.B0x202.C0x264.A0x265.B0x268.A0x2C9.C0x2CB.B0x2D0.C0x32F.A0x333.B0x3
34.C0x393.A0x398.B0x39B.C0x3F7.A0x3FF.B0x400.C0x462.A0x464.B0x46A.C0x4C9.A0x4CA.
B0x4D1.C0x52D.A0x532.B0x53D.C0x593.A0x597.B0x5A4.C0x5FB.A0x5FC.B0x609.C0x660.A0x
661.B0x66D.C0x6C5.A0x6C7.B0x6D1.C0x72E.A0x730.B0x736.C0x792.A0x794.B0x79F.C0x7F6
.A0x7F9.B0x805.C0x85A.A0x860.B0x86E.C0x8C2.A0x8C7.B0x8D2.C0x927.A0x92B.B0x939.C0
x98C.A0x991.B0x99D.
  
```

注: 上面演示了三个打印函数(充当三个进程)随机交替打印。

图6

3.5 I/O模块的设计

无疑现代操作系统必须将进程、I/O设备、内存管理等模块紧密联系在一起。首先, 我们从最简单的I/O设备键盘做起。从敲击键盘到显示器显示出字符大致分为这几个步骤: 敲击键盘、扫描并编码、接受并解码、与Intel8259通信、响应结束。在键盘中有一枚编码芯片, 通常是Intel8048或其它兼容芯片, 它负责监视键盘的输入, 并把数据传到计算机主板上的解码芯片Intel8042, 经过Intel8042解码后并将其放入输入缓冲区, 然后与Intel8258通信, 当允许Intel8259中断(IRQ1)时, 若键盘又有新的键被按下, 则Intel8042不会再接受, 一直到输入缓冲区被清空, Intel8042才会接受更多的扫描码。

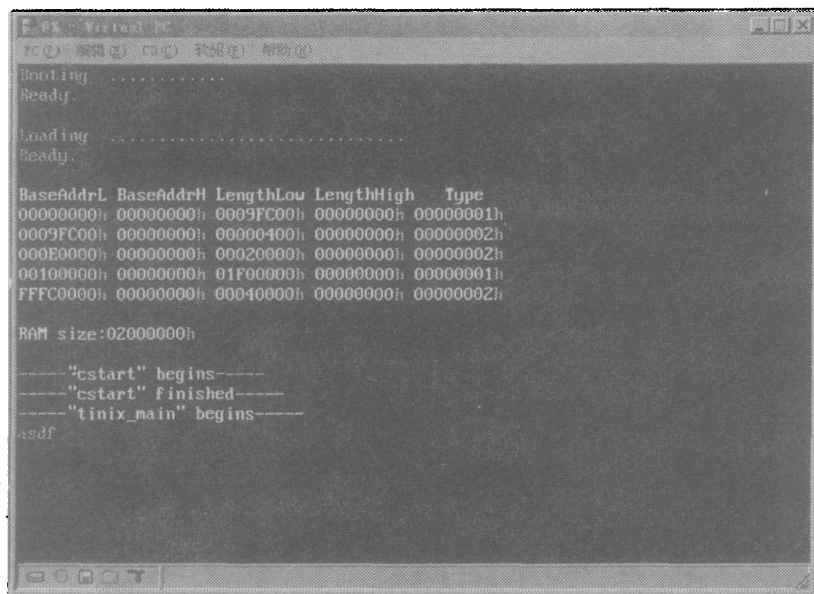
按上述原理设计I/O中断程序的流程如下:

- 1) 初始化Inter8259的IRQ1时钟中断类型, 使之响应键盘;
- 2) 通过键盘申请I/O中断;
- 3) 关中断;
- 4) 允许IRQ1中

断; 5) 设置中断内容(其中最重要的是键盘按下的键与对应字符的转换); 6) 通过显示器输出。

根据上面流程编写出代码使用makefile自动化编译, 然后同上在把生成的*.bin拷贝到软盘即可运行。

运行结果如图7:



```
PC -> Titled PC
PC -> 编辑(E)  CD(C)  软盘(D)  帮助(H)

Booting .....
Ready.

Loading .....
Ready.

BaseAddrL BaseAddrH LengthLow LengthHigh  Type
00000000h 00000000h 0009FC00h 00000000h 00000001h
0009FC00h 00000000h 00000400h 00000000h 00000002h
000E0000h 00000000h 00020000h 00000000h 00000002h
00100000h 00000000h 01F00000h 00000000h 00000001h
FFFC0000h 00000000h 00040000h 00000000h 00000002h

RAM size:02000000h

----"cstart" begins----
----"cstart" finished----
----"tinix_main" begins----
asdf
```

注: 上图中asdf是作者自己通过键盘输上去的。

图7

4 其它说明

由于篇幅有限, 文中没有给出每个模块的源代码资料。

【参 考 文 献】

- [1] 汤子瀛, 等. 计算机操作系统 [M]. 西安: 西安电子科技大学出版社, 2001. 8.
- [2] 沈美明, 等. 汇编语言程序设计 [M]. 北京: 清华大学出版社, 2001. 8.
- [3] 于渊. 自己动手写操作系统 [M]. 北京: 电子工业出版社, 2005. 8.
- [4] 梁肇新. 编程高手箴言 [M]. 北京: 电子工业出版社, 2003. 11.
- [5] 王鹏等译. 中文版操作系统: 设计与实现 [M]. 北京: 电子工业出版社, 2004.

(责任编辑: 胡 蓉)