DOI:10.13877/j.cnki.cn22-1284.2015.10.016

X86教学操作系统引导模块的设计与实现*

于江涛,陈立博
(通化师范学院 计算机学院, 吉林 通化 134002)

摘 要:文章介绍了X86平台教学操作系统引导模块的设计与实现,阐述X86平台的保护模式编程技术、教学操作系统引导模块的实现技术及编程要点.通过一个实例演示教学操作系统引导模块的启动过程.引导模块全部代码在Redhat Linux9.0操作系统环境下用AT&T汇编语言及C语言设计而成.

关键词:教学操作系统;引导模块;X86;Redhat Linux 9.0;AT&T汇编语言;C语言
中图分类号:TP316 **文献标志码**:A **文章编号**:1008-7974(2015)05-0050-04

操作系统课程是本科计算机类专业重要的主干课程,是学生必须掌握的专业基础课程.如果不了解操作系统的基本原理、内部结构及其系统调用机制,就不能有效地利用操作系统所提供的强大功能进行大型系统软件及应用软件的设计、开发与维护.如何提高操作系统课程的教学效果及教学质量,对于培养本科计算机类专业学生的专业素质具有重要的指导意义,也是计算机类专业本科教学研究的一个重要课题.

提高操作系统课程教学效果的一个重要手段就是使用教学操作系统作为操作系统课程的教学实验平台.教学操作系统"麻雀虽小、五脏俱全",具有操作系统体系结构的最基本组成部分,同时又以很小的代码规模实现,便于本科层次的计算机类专业学生研究与学习.国外学者开发的MINIX[1-2]、GeekOS等,都是教学操作系统的优秀代表.

随着近年来国内大学的不断扩招,生源质量明显下降.对于普通本科计算机类专业学生而言,研究与学习MINIX这一类的教学操作系统具有一定的困难.尽管MINIX在设计上考虑到是为了应用于操作系统课程教学,已经大量地缩减了其代码规模,但其代码量还是达到2万多行.对于国内计算机类专业的研究生及高层次本科院校的本科生来说,这样的代码规模还是可以接受的.但对于国内普通高校的计算机类专业学生而言,由于其基本能力与素质的限制,这样的代码量还显得太多,几乎很难读懂.国外另一个著名的教学操作系统GeekOS仅提供了框架结构,具体内容要由学生本人在教学实验过程中自行设计.这种思路对于激发学生深入研究、理解操作系统原理无疑具有挑战性,有利于培养学生的创造性能力,比较适合于计算机类专业的研究生及高层次本科院校的本科生使用.对于绝大多数的普通高校的计算机类专业学生而言,其很难具备这样的研究设计能力.

为解决这些问题,国内许多学者在教学操作系统方面进行了深入探索,研究设计更适合于普通高等院校计算机类专业学生的教学操作系统[3].本文阐述的内容就是笔者在这方面所做的研究.

本文介绍X86平台教学操作系统引导模块的设计与实现,阐述X86平台的保护模式编程技术、教学操作系统引导模块的实现技术及编程要点.通过一个实例演示教学操作系统引导模块的启动过程.引导模块的全部代码在Redhat Linux 9.0操作系统环境下基于VMWare虚拟机、用AT&T汇编语

* **收稿日期**:2015-08-31
作者简介:于江涛,吉林通化人,教授.

言及 C 语言设计而成.

1 X86 平台的保护模式编程技术

在 Intel 8086 时代,内存的寻址能力仅为 20 位,即 1M 字节. 在当时,这样的内存已经被认为是很大了. 在 IBM 个人计算机中,低地址的 640K 字节提供给用户的应用程序使用,其余的 384K 字节供操作系统、BIOS 及扩展设备使用.

随着存储器成本的不断下降及对内存需求的不断增长,1M 字节的内存已经成为严重问题,Intel 80286 及保护模式编程应运而生. 起初,286 的保护模式并未被广泛使用,仅用于 Xenix 及 MINIX 等. 其存在一些不足,例如由于不能从保护模式返回实模式从而无法访问 BIOS 及 DOS 系统调用,限制了保护模式编程的广泛使用. 随着 Intel 80386 的问世,先前阻碍保护模式被广泛使用的许多问题得以解决. 386 芯片具有 32 位地址总线,从而其寻址能力可达到 4G 字节. 其地址寻址的段尺寸也增加到 32 位,意味着整个 4G 字节的地址空间可直接访问,不需在多个段之间来回切换. 这时的保护模式事实上已经应用在所有基于 X86 体系结构的现代操作系统上,如 Microsoft Windows,Linux 等.

直到 Intel 80386 问世之前,Intel 80286 没有提供进入保护模式后再返回实模式的直接方法. 386 芯片通过将实模式值装入段寄存器、禁用 A20 总线并清零 CRO 寄存器 PE 位的方法退出保护模式,而不必像 286 那样实施初始设置步骤.

要进入保护模式[2],全局描述符表 GDT(Global Descriptor Table)至少有三项被创建:一个零描述符,一个代码段描述符及一个数据段描述符. 在 IBM 兼容机中,A20 总线必须设置为允许(enabled),使其能够使用全部地址总线位以便 CPU 能够访问 1M 字节以上的存储空间(主机引导后,只有前 20 位地址总线允许被使用,以确保对基于 IBM PC 及 PC/XT 模式的 Intel 8088 所书写的老旧程序的兼容性). 这两步之后,CR0 寄存器的 PE 位必须被置位(置 1)而且使用一条远跳转指令(Far jump)来清除预取指输入队列(Pre - fetch input queue).

2 教学操作系统引导模块实现技术与编程要点

2.1 基本原理

(1)通用操作系统启动过程. ①BIOS 加载并启动保存在硬盘 MBR 中的引导程序,该引导程序一般在操作系统安装时写入;②MBR 引导程序扫描所有分区表,找出活动分区;③MBR 引导程序加载并启动保存在活动分区 PBR 中的引导程序;④活动分区 PBR 中的引导程序加载并启动安装在其上的操作系统. 显然 PBR 引导程序与操作系统密切相关,一般在操作系统安装时写入.

(2)教学操作系统启动过程. 教学操作系统一般由软盘启动引导[2-3]. 其启动原理与上述过程类似.

为简化系统,笔者所设计的教学操作系统采用 Fat 系统盘方式引导,将主引导扇区启动模块(booter)写入软盘首扇区,将操作系统装载模块(loader)及内核模块(kernel)链接成一个内核镜像文件,写入软盘根目录.

启动引导过程如下:

①系统将软盘 0 面 0 道 1 扇区的启动模块程序(booter)装入内存(0x0000:7c00 处);

②启动模块程序将位于 Fat 系统引导盘根目录的内核镜像文件(kernel. img)读入内存 0x9000:0200 处,其中前半部是装载模块代码,后半部是内核模块代码;

③装载模块首先将检测必要的 BIOS 参数(如内存容量、光标位置等),将保护模式下的描述符内容存入系统高端供内核使用,然后转入保护模式,按 ELF 格式文件安装内核模块,最后跳转到内核的启动地址,运行内核程序.

2.2 实现技术

(1) 引导扇区启动模块程序 boot. s. 一般来说,由软盘引导操作系统主要有三种方式:

①用 Fat 文件系统软盘启动. 将启动模块存放在主引导扇区,装载模块及内核模块链成一个 Fat 文件,拷贝到软盘上. 其中内核模块为 ELF 文件格式. 这种启动方式的软盘格式必须严格遵循 Fat 文件规范,软盘首扇区起始位置必须保存 Fat 文件系统首部,如图 1 所示.

②按软盘逻辑扇区顺序存放启动模块、装载模块和内核模块,其中内核模块为 ELF 可执行文件格式.

③按软盘逻辑扇区顺序存放启动模块、装载模块和内核模块,其中内核模块为无格式二进制文件.

第一种及第二种方式的内核模块由于是 ELF 格式,需要在引导过程中动态安装. 但可以根据需要,设计不同的内核基址. 笔者所设计的教学操作系统采用第一种方式启动,便于生成内核镜像文件. 为方便操作系统运行过程中对 BIOS 数据区的访问,内核基址设计为 0x1000.

```
BS_OEMName:     .ascii "Mini TOS"      ; 8 bytes
BPB_BytsPerSec: .word  512
BPB_SecPerCluster:.byte  1
BPB_ResvdSecCnt: .word  1
BPB_NumFATs:    .byte  2
BPB_RootEntCnt:  .word  224
BPB_TotSec16:    .word  2880
BPB_Media:       .byte  0xf0
BPB_FATSz16:     .word  9
BPB_SecPerTrk:   .word  18
BPB_NumHeads:    .word  2
BPB_HiddSec:     .long  0
BPB_TotSec32:    .long  0
BS_DrvNum:       .byte  0
BS_Reserved1:    .byte  0
BS_BootSig:      .byte  0x29
BS_VolID:        .long  0
BS_VolLab:       .ascii  "Mini_TOS   " ; 11 bytes
BS_FileSysType:  .ascii  "FAT12   "    ; 8 bytes
```

图 1　Fat 文件系统首部定义

启动程序 boot. s 将内核镜像文件(包含装载模块和内核模块代码)装入 0x9000:0200 处,然后转向 0x9000:0200 处,将控制权交给装载模块.

启动模块的任务包括读软盘首扇区、读 Fat 文件目录、寻找内核镜像文件、读入内核镜像文件、软盘驱动器马达控制、进入保护模式、显示提示信息等.

(2)装载模块程序 load. s. 装载程序的任务是将内核模块装入内存并在保护模式下投入运行,因此,其本身必须首先在保护模式下运行.

如果内核模块使用无格式二进制文件,只需简单将其存放在指定地址处即可. 如果是 ELF 格式文件,则需根据 ELF 文件格式内容动态安装.

由于装入模块与内核模块组合在一起构成内核镜像文件,所以,装入模块还要知道自身目标代码所占用的扇区数,从而正确地将位于其后的内核模块目标代码装入指定位置. 通过在装入模块汇编程序尾部设置相应伪指令的方式就可以很容易获取其自身的目标代码长度. 关键代码如图 2 所示.

```
_start:  jmp  _start16
.align 4
ldsecs:  .word    loader_top/512
_start16:
...... (Program codes)
.align 512
loader_top = (. - _start)
```

图 2　装入模块获取自身目标代码长度的关键代码

(3)内核模块程序. 本文提供的实例只为演示引导启动过程,所以,内核模块不做实质性处理. 其中,head. s 是 AT&T 汇编程序,构建操作系统内核框架;main. c 是 C 语言程序,是内核模块实体,在本文示例中仅包含显示提示信息的代码.

2.3　编程要点

(1)检测内存容量及光标当前位置. 检测内存容量可通过调用 BIOS 中断 0x15 的 0x88、0xE801 或 0xE820 号功能实现. 如果使用 0x88 号功能,所能检测的最大容量不超过 64M 字节,因为,该功能是为早期的 16 位机设计的. 如果使用 0xE801 或 0xE820,则可检测最大容量至 4G 字节. 内存容量值保存在内存地址 0x90000 处.

当前光标位置通过调用 BIOS 中断 0x10 的 3 号功能检测,保存在内存地址 0x90004 处.

(2)开启 A20 地址线. A20 是 X86 计算机 32 位地址总线中的一位地址线,用来传送地址总线信号的第 21 位地址信号. 如果该位信号被禁用,则地址总线中第 21 位至 32 位的地址信号全为 0,因此,可实现与 16 位机的兼容. 有若干种方法开启 A20,本文示例使用设置键盘控制器端口的方法. 开启 A20 之后,通过设置机器状态字(Machine Status Word)即可进入 32 位保护模式. 关键代码如图 3 所示.

```
in   $0x92,%al      ; enable A20
or   $0x2,%al
out  %al,$0x92
mov  %cr0,%eax      ; set MSW
or   $0x1,%eax
mov  %eax,%cr0
ljmpl    $SEL_CS,$(LOAD_BASE+start_32)
```

图 3　开启 A20 及设置 MSW 的关键代码

(3)设置段描述符. 对于保护模式编程,需要设置三类段描述符表:全局描述符表(GDT)、中断描述符表(IDT)及局部描述符表(LDT). 一个系统只有一个 GDT 和一个 IDT,但具有与进程数相对应的多个 LDT. 对于每一个进程而言,在 GDT 中有两个描述符项,即 TSS 描述符及 LDT 描述符,分别指向该进程的 TSS 及 LDT.

在本文示例中,每个进程的两个 GDT 描述符表项被保存在该进程的 PCB 中,每当该进程被调度后将其动态装入 GDT. 因此,对于 GDT 而言,只需两个描述符表项就可用于所有进程. 使用这种方法,GDT 尺寸不仅可极大缩小,而且还与系统进程数量无关.

为便于存储管理,在保护模式设置之后,GDT 和 IDT 两个描述符表被移到地址 0xA0000 之下.

(4)可编程中断控制器 8259A 初始化. 对于可编程中断控制器 8259A 的初始化设置,通过向相应端口写入初始命令字 ICW 及操作命令字 OCW 实现. 本文示例所使用的关键代码如图 4 所示.

```
init_8259A:
    out_byte $0x11, $0x20  # Master 8259, ICW1
    out_byte $0x11, $0xa0  # Slave  8259, ICW1
    out_byte $0x20, $0x21  # Master 8259, ICW2
    out_byte $0x28, $0xa1  # Slave  8259, ICW2
    out_byte $0x04, $0x21  # Master 8259, ICW3
    out_byte $0x02, $0xa1  # Slave  8259, ICW3
    out_byte $0x01, $0x21  # Master 8259, ICW4.
    out_byte $0x01, $0xa1  # Slave  8259, ICW4.
    out_byte $0xff, $0x21  # Master 8259, OCW1.
    out_byte $0xff, $0xa1  # Slave  8259, OCW1.
    ret
```

图 4　可编程中断控制器 8259A 初始化汇编代码

(5)装载内核代码. 内核模块代码是 ELF 格式的可执行文件,因此,在运行前必须根据 ELF 格式将其动态装入内存. 本文示例所使用的关键代码如图 5 所示.

```
    mov  (FILE_BASE + 0x2C),%cx
    movzwl   %cx,%ecx
    mov  (FILE_BASE + 0x1C),%ebp
    add  $FILE_BASE,%ebp
1:  cmpl $0,(%ebp)
    je   2f
    push %ecx
    mov  0x10(%ebp),%ecx
    mov  0x08(%ebp),%edi
    mov  0x4(%ebp),%esi
    add  $FILE_BASE,%esi
    cld
    rep movsb
    pop  %ecx
2:  add  $0x20,%ebp
    loop 1b
    ret
```

图 5　动态装入 ELF 格式内核模块的关键代码

2.4　运行演示

利用 VMWare 虚拟机系统建立一个空的虚拟主机 TOS,将本文所示的 X86 教学操作系统引导模块全部代码编译链接之后,形成一个引导软盘镜像文件,将该镜像文件作为虚拟主机 TOS 的软盘镜像文件,即可引导启动该虚拟主机,运行结果如图 6 所示.

其中第一行是引导程序显示引导进度,第二行是装入程序显示装入进度,从第三行开始由内核模块显示. 内存尺寸由装入模块检测后存入内存 0x90000 处,由内核模块显示.

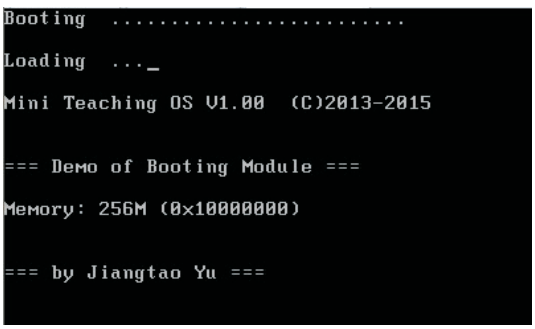

图 6　X86 教学操作系统引导模块的运行演示

3　结束语

教学操作系统对于普通高校计算机类专业的操作系统课程教学具有重要作用,有必要对教学操作系统的设计与实现作进一步研究与探讨,以便不断提高操作系统课程的教学质量和教学效果.

本文阐述的 X86 平台教学操作系统引导模块在 Linux 操作系统平台上基于 VMWare 虚拟机系统、使用 AT&T 汇编语言及 C 语言编程实现,规模很小,便于普通高校计算机类本科学生学习与研究. 在此基础上可进一步扩充教学操作系统的其他模块,具有重要的价值.

参考文献:

[1]A. S. Tanenbaum. Operating Systems: Design and Implementation, Third Edition[M]. Prentice Hall, Inc. ,2008.

[2]赵炯. Linux 内核完全注释[M]. 北京:机械工业出版社,2007.

[3]Bo Qu, Zhaozhi Wu. Design and Implementation of Tiny Educational OS, Lecture Notes in Electrical Engineering [J]. 2012,126:437 – 442.

(责任编辑:王前)