2019/10/19 Markdoc Preview

实验报告

实验目标

- 熟悉 hit-oslab 实验环境
- 建立对操作系统引导过程的深入认识
- 掌握操作系统的基本开发过程
- 能对操作系统代码进行简单的控制,揭开操作系统的神秘面纱

实验内容

- 1. 阅读《Linux 内核完全注释》的第 6 章,对计算机和 Linux 0.11 的引导过程进行初步的了解
- 2. 按照下面的要求改写 0.11 的引导程序 bootsect.s
- 3. 有兴趣同学可以做做进入保护模式前的设置程序 setup.s

改写 bootsect.s 主要完成如下功能:

1. bootsect.s 能在屏幕上打印一段提示信息"XXX is booting...",其中 XXX 是你给自己的操作系统起的名字,例如 LZJos、Sunix 等(可以上论坛上秀秀谁的 OS 名字最帅,也可以显示一个特色 logo,以表示自己操作系统的与众不同。)

改写 setup.s 主要完成如下功能:

- 1. bootsect.s 能完成 setup.s 的载入,并跳转到 setup.s 开始地址执行。而 setup.s 向屏幕输出一行"Now we are in SETUP"。
- 2. setup.s 能获取至少一个基本的硬件参数(如内存参数、显卡参数、硬盘参数等),将其存放在内存的特定地址,并输出到屏幕上。
- 3. setup.s 不再加载 Linux 内核,保持上述信息显示在屏幕上即可。

思考题

有时,继承传统意味着别手蹩脚。x86 计算机为了向下兼容,导致启动过程比较复杂。请找出 x86 计算机启动过程中,被硬件强制,软件必须遵守的两个"多此一举"的步骤(多找几个也无妨),说说它们为什么多此一举,并设计更简洁的替代方案。

- 1. x86CPU启动时为了向下兼容16位使用实模式: 纯16位无保护执行环境。对于80286或以上的CPU通过A20 GATE来控制A20地址线。 技术发展到了80286,虽然系统的地址总线由原来的20根发展为24根,这样能够访问的内存可以达到2^24=16M,但是Intel在设计80286时提出的目标是向下兼容,所以在实模式下,系统所表现的行为应该和8086/8088所表现的完全一样,也就是说,在实模式下,80386以及后续系列应该和8086/8088完全兼容仍然使用A20地址线。 所以高级芯片为了运行以前的程序,不得不保留实模式。 所以说80286芯片存在一个BUG: 它开设A20地址线。 如果程序员访问100000H-10FFEFH之间的内存,系统将实际访问这块内存。进入实模式多此一举,可以直接进入保护模式。 解决方案: 是不向下兼容直接进入32位的保护模式,经管在Intel 80286手册中已经提出了虚地址保护模式,但实际上它只是一个指引,真正的32位地址出现在Intel 80386上。 保护模式本身是80286及以后兼容处理器序列之后产成的一种操作模式,它具有许多特性设计为提高系统的多道任务和系统的稳定性。 例如内存的保护,分页机制和硬件虚拟存储的支持。现代多数的x86处理器操作系统都运行在保护模式下。
- 2. 当PC的电源打开后,80x86结构的CPU将自动进入实模式,并从地址0xFFFF0开始自动执行程序代码,这个地址通常是ROM—BIOS中的地址。 PC机的BIOS将执行某些系统的检测,并在物理地址0处开始初始化中断向量。此后将启动设备的第一个扇区512字节读入内存绝对地址0x7C00处。 因为当时system模块的长度不会超过0x80000字节大小512KB,所以bootsect程序把system模块读入物理地址0x10000开始位置处时并不会覆盖在0x90000处开始的bootsect和setup模块,多此一举的是system模块移到内存中相对靠后的位置,以便加载系统主模块。 解决方案:在保证操作系统启动引导成功的前提下尽量扩大ROM—BIOS的内存寻址范围,以达到不需要读入靠后的位置处。

实验步骤

1. 完成 bootsect.s 的屏幕输出功能

```
shiyanlou@0834348fb075: ~/oslab/linux-0.11/boot
 start:
    mov ah,#0x03
xor bh,bh
int 0x10
    mov cx,#25
    mov bx, #0x0007
    mov bp, #msg1
    mov ax,#0x07c0
    mov es,ax
    mov ax,#0x1301 int 0x10
inf loop:
     jmp inf_loop
msg1:
              13,10
"YPYos is booting..."
     .byte
     .ascii
              13,10,13,10
org 510
boot_flag:
              0xAA55
"bootsect.s" 21L, 398C
                                                                 1.1
                                                                                 全部
```

2. 编译和运行

♦ 应用程序菜单

2019/10/19 Markdoc Preview

```
Bochs x86 emulator, http://bochs.sourceforge.net/
                                                    USER Copy Poste Snapshot T ResetsuspenD.Poster
 lex86/Bochs V
            VGABios current-cvs 07 Jan
This UGA/UBE Bios is released under the GNU LGPL
Please visit :
  http://bochs.sourceforge.net
  http://www.nongnu.org/vgabios
Bochs UBE Display Adapter enabled
Bochs BIOS - build: 02/13/08
$Revision: 1.194 $ $Date: 2007/12/23 19:46:27 $
Options: apmbios pcibios eltorito rombios32
ata0 master: Generic 1234 ATA-6 Hard-Disk (  60 MBytes)
                                                                                  ect of=
Booting from Floppy...
YPYos is booting...
                                                                                  age
                                                                                  ======
                         HD:0-M NUM CAPS SCRL
CTRL + 3rd button enables mouse A:
                         Build from CVS snapshot, on June 3, 2008
         ------
        00000000000i[
                             ] reading configuration from ../../bochs/bochsrc.bxrc
        000000000001[
                            ] installing x module as the Bochs GUI
        00000000000i[
                            ] using log file ../../bochsout.txt
 ♦ 应用程序菜单
```

3. bootsect.s读入setup.s

```
shiyanlou@633ee8422070: ~/oslab/linux-0.11/boo
SETUPSEG=0x07e0
entry _start
_start:
    mov ah,#0x03
xor bh,bh
int 0x10
    mov cx,#25
     mov bx,#0x0007
     mov bp,#msg1
    mov ax,#0x07c0
    mov es,ax
mov ax,#0x1301
int 0x10
load_setup:
    mov dx,#0x0000
     mov cx,#0x0002
    mov bx #0x0200
mov ax #0x0200+SETUPLEN
     int 0x13
    jnc ok_load_setup
mov dx,#0x0000
     mov ax,#0x0000
     int 0x13
     jmp load_setup
ok_load_setup:
              0, SETUPSEG
     jmpi
msq1:
     .byte 13,10
.ascii "YPYos is booting..."
              13,10,13,10
.org 510 boot_flag:
              0xAA55
     .word
"bootsect.s" 34L, 655C
                                                                                                        1,1
                                                                                                                         全部
 ♦ 应用程序菜单
```

4. setup.s显示硬件参数

```
INITSEG = 0x9000
entry _start
_start:
! Print "NOW we are in SETUP"
```

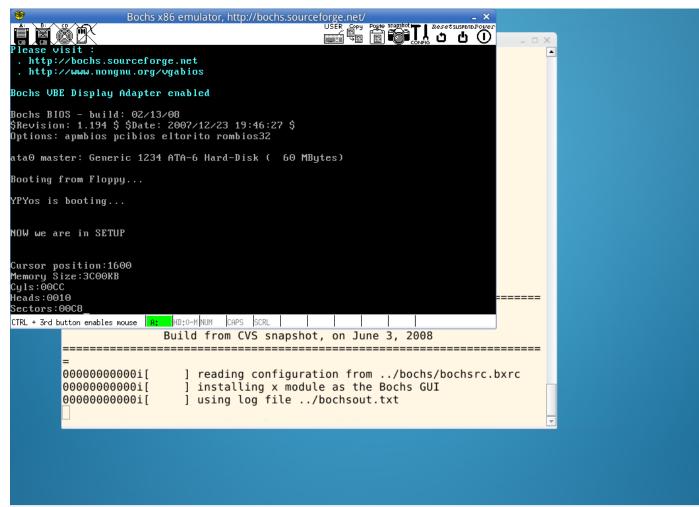
```
mov ah,#0x03
   xor bh,bh
   int 0x10
   mov cx,#25
   mov bx,#0x0007
   mov bp,#msg2
   mov ax,cs
   mov es,ax
   mov ax,#0x1301
   int 0x10
   mov ax,cs
   mov es,ax
! init ss:sp
   mov ax,#INITSEG
   mov ss,ax
   mov sp,#0xFF00
! Get Params
   mov ax,#INITSEG
   mov ds,ax
   mov ah,#0x03
   xor bh,bh
   int 0x10
   mov [0],dx
   mov ah,#0x88
   int 0x15
   mov [2],ax
   mov ax,#0x0000
   mov ds,ax
   lds si,[4*0x41]
   mov ax,#INITSEG
   mov es,ax
   mov di,#0x0004
   mov cx,#0x10
   rep
   movsb
! Be Ready to Print
   mov ax,cs
    mov es,ax
   mov ax,#INITSEG
   mov ds,ax
! Cursor Position
   mov ah,#0x03
   xor bh,bh
   int 0x10
   mov cx,#18
   mov bx,#0x0007
   mov bp,#msg_cursor
   mov ax,#0x1301
   int 0x10
   mov dx,[0]
   call print_hex
! Memory Size
   mov ah,#0x03
   xor bh,bh
   int 0x10
   mov cx,#14
   mov bx,#0x0007
   mov bp,#msg_memory
   mov ax,#0x1301
   int 0x10
   mov dx,[2]
           print_hex
   call
! Add KB
   mov ah,#0x03
   xor bh,bh
   int 0x10
   mov cx,#2
   mov bx,#0x0007
   mov bp,#msg_kb
   mov ax,#0x1301
   int 0x10
! Cyles
   mov ah,#0x03
   xor bh,bh
   int 0x10
   mov cx,#7
   mov bx,#0x0007
   mov bp,#msg_cyles
   mov ax,#0x1301
   int 0x10
   mov dx,[4]
   call print_hex
! Heads
   mov ah,#0x03
   xor bh,bh
   int 0x10
   mov cx,#8
   mov bx,#0x0007
   mov bp,#msg_heads
```

mov ax,#0x1301

```
int 0x10
   mov dx,[6]
   call
          print_hex
! Secotrs
   mov ah,#0x03
   xor bh,bh
   int 0x10
   mov cx,#10
   mov bx,#0x0007
   mov bp,#msg_sectors
   mov ax,#0x1301
   int 0x10
   mov dx,[12]
   call print_hex
inf_loop:
   jmp inf_loop
print_hex:
         cx,#4
   mov
print_digit:
  rol
        dx,#4
          ax,#0xe0f
   mov
         al,dl
   and
          al,#0x30
   add
         al,#0x3a
   cmp
   jl
          outp
         al,#0x07
   add
outp:
          0x10
   int
         print_digit
   loop
   ret
print_nl:
         ax,#0xe0d ! CR
   mov
   int
         0x10
         al,#0xa
                    ! LF
   mov
          0x10
   int
   ret
msg2:
    .byte 13,10
    .ascii "NOW we are in SETUP"
    .byte 13,10,13,10
msg_cursor:
    .byte 13,10
   .ascii "Cursor position:"
msg_memory:
    .byte 13,10
    .ascii "Memory Size:"
msg_cyles:
   .byte 13,10
    .ascii "Cyls:"
msg_heads:
   .byte 13,10
    .ascii "Heads:"
msg_sectors:
   .byte 13,10
   .ascii "Sectors:"
msg_kb:
   .ascii "KB"
.org 510
boot_flag:
    .word 0xAA55
```

1. 再次编译运行

2019/10/19 Markdoc Preview



♦ 应用程序菜单

实验总结

- 了解了Linux系统启动流程
- 了解如何实现bootsect.s和setup.s
- 了解了make命令,了解了如何配置build