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。

2.实验内容 此次实验的基本内容是：

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；

按照下面的要求改写0.11的引导程序bootsect.s

有兴趣同学可以做做进入保护模式前的设置程序setup.s。

3.评分标准

- bootsect显示正确，30%
- bootsect正确读入setup，10%
- setup获取硬件参数正确，20%
- setup正确显示硬件参数，20%
- 实验报告，20%

实验报告 4.1完成bootsect.s的屏幕输出功能

tar –zxvf 文件名解压，

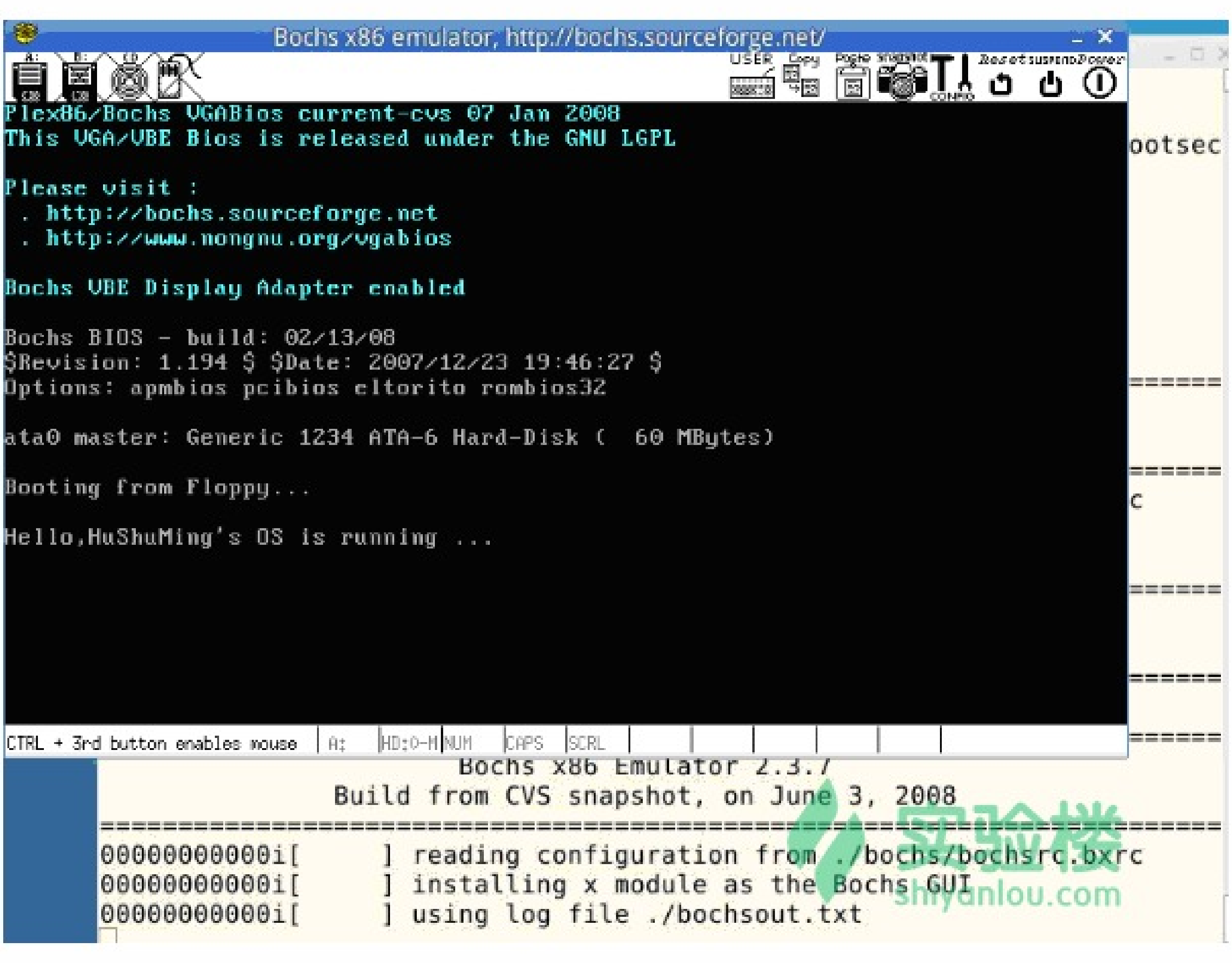
进入 ./oslab/oslab/linux-0.11目录下，修改bootsect.s，

cd ..返回上一目录，输入make，

再进入boot目录，输入dd bs=1 if=bootsect of=Image skip=32，生成Image文件；

将linux-0.11/boot下Image文件覆盖linux-0.11下Image文件；

cd ..返回目录，输入 ../run运行，结果如下：

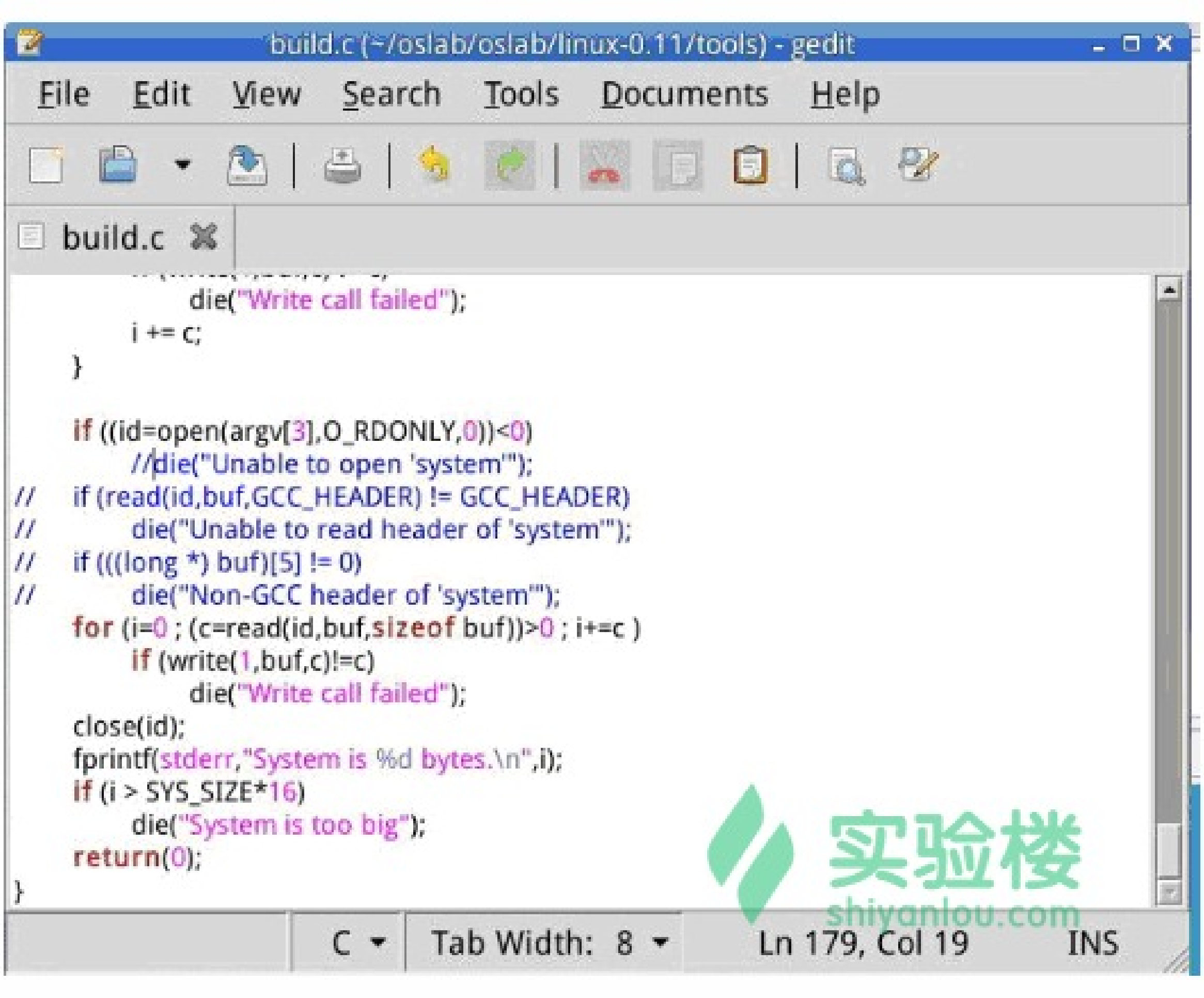


```
Plex86/Bochs VGABios current-cvs 07 Jan 2008
This VGA/VBE Bios is released under the GNU LGPL

Please visit :
 . http://bochs.sourceforge.net
 . http://www.nongnu.org/vgabios

Bochs VBE 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)

Booting from Floppy...

Hello,HuShuMing's OS is running ...
```

```
                 Bochs x86 Emulator 2.3.7
         Build from CVS snapshot, on June 3, 2008
========================================================================
00000000000i[     ] reading configuration from ./bochs/bochsrc.bxrc
00000000000i[     ] installing x module as the Bochs GUI
00000000000i[     ] using log file ./bochsout.txt
```

4.2修改setup.s，并屏幕输出

修改tools目录下的build.c文件，注释掉第179行，如下图所示；

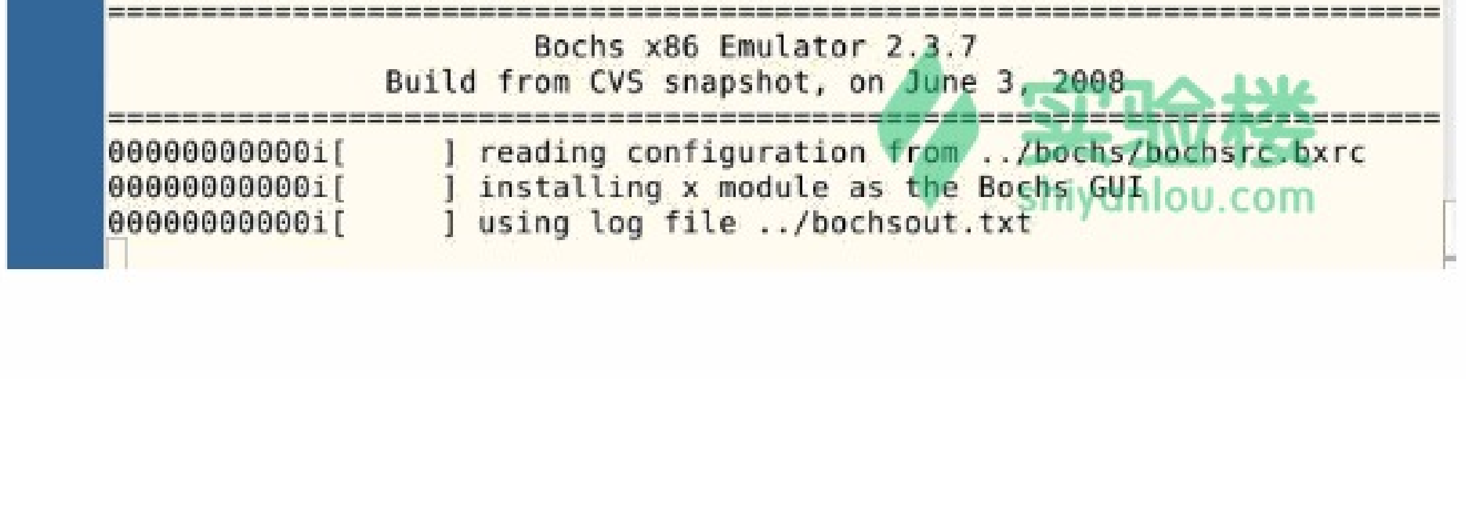


```
			die("Write call failed");
		i += c;
	}

	if ((id=open(argv[3],O_RDONLY,0))<0)
		//die("Unable to open 'system'");
//	if (read(id,buf,GCC_HEADER) != GCC_HEADER)
//		die("Unable to read header of 'system'");
//	if (((long *) buf)[5] != 0)
//		die("Non-GCC header of 'system'");
	for (i=0 ; (c=read(id,buf,sizeof buf))>0 ; i+=c )
		if (write(1,buf,c)!=c)
			die("Write call failed");
	close(id);
	fprintf(stderr,"System is %d bytes.\n",i);
	if (i > SYS_SIZE*16)
		die("System is too big");
	return(0);
}
```

修改setup.s，修改后源码如下：

```
!
!      setup.s          (C)  1991  Linus  Torvalds
!
! setup.s   is  responsible    for  getting   the  system  data  from  the  BIOS,
! and  putting   them  into  the  appropriate    places  in  system  memory.
! both  setup.s   and  system  has  been  loaded  by  the  bootblock.
!
! This  code  asks  the  bios  for  memory/disk/other     parameters,    and
! puts  them  in  a  "safe"   place:   0x90000  - 0x901FF ,  ie  where  the
! boot  - block  used  to  be.  It  is  then  up  to  the  protected   mode
! system  to  read  them  from  there  before  the  area  is  overwritten
! for  buffer  - blocks.
!

! NOTE!  These  had  better   be  the  same  as  in  bootsect.s!

.globl   begtext,   begdata,   begbss,   endtext,   enddata,   endbss
.text
begtext:
.data
begdata:
.bss
begbss:
.text

entry   start
start:
INITSEG   = 0x9000      !  we  move  boot  here  -  out  of  the  way
SYSSEG    = 0x1000      !  system   loaded   at  0x10000   ( 65536 ).
SETUPSEG = 0x9020       !  this   is  the  current   segment

! ok,  the  read  went  well  so  we  get  current   cursor   position    and  save  it  for
! posterity.

     mov     ax,  #INITSEG      !  this   is  done  in  bootsect    already,    but...
     mov     ds,ax
     mov     ah,  #0x03       !  read   cursor   pos
     xor     bh,bh
     int     0x10             !  save  it  in  known  place,   con_init   fetches

         mov     ax,  #SETUPSEG
         mov     es,ax
         mov     ah,  #0x03
         xor     bh,bh
         int     0x10
         mov     cx,  #24
         mov     bx,  #0x0007
         mov     bp,  #msg2
         mov     ax,  #0x1301
         int     0x10
     mov     [ 0 ],dx            !  it   from   0x90000  .

!  Get  memory  size   (all    mem,  kB)
     mov     ah,  #0x88
     int     0x15
     mov     [ 2 ],ax

!Print     memory  size
     mov     ah,  #0x03
     xor     bh,bh
     int     0x10
     mov     cx,  #20
     mov     bx,  #0x0007
     mov     bp,  #msg3
     mov     ax,  #0x1301
     int     0x10
     mov     dx,[   2 ]
     add     dx,  #0x0400
     mov     cx,  #4
     jmp  print_hex

print_hex:
     mov     ax,  #0x0E0F
     rol     dx,  #4
     and     al,dl
     cmp     al,  #0x0A
     jb        ltA
     jnb       nltA

ltA:
     add     al,  #0x30
     jmp     ctnp

nltA:
     add     al,  #0x37
     jmp  ctnp

ctnp:
     int     0x10
     loop      print_hex

msg2:
     .byte        13 ,  10
     .ascii         "Now  we  are   in   SETUP"
     .byte        13 ,  10 ,  13 ,  10

msg3:
     .byte        13 ,  10
     .ascii         "Total     Memory   Size:"
     .byte        13 ,  10 ,  13 ,  10

.text
endtext:
.data
enddata:
.bss
endbss:
```

linux-0.11目录下输入 make BootRoot; 再输入 ../run 运行，结果如下：



```
Plex86/Bochs VGABios current-cvs 07 Jan 2008
This VGA/VBE Bios is released under the GNU LGPL

Please visit :
 . http://bochs.sourceforge.net
 . http://www.nongnu.org/vgabios

Bochs VBE 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)

Booting from Floppy...

Hello,HuShuMing's OS is running ...


Now we are in SETUP


Total Memory Size:4000
```

```
========================================================================
shiyanlou@27724de4d3a5:~/oslab/oslab/linux-0.11$ ../run
========================================================================
                 Bochs x86 Emulator 2.3.7
         Build from CVS snapshot, on June 3, 2008
========================================================================
00000000000i[     ] reading configuration from ../bochs/bochsrc.bxrc
00000000000i[     ] installing x module as the Bochs GUI
00000000000i[     ] using log file ../bochsout.txt
```