Kagura L87 2018-02-05 06:01:48 地址映射与共享

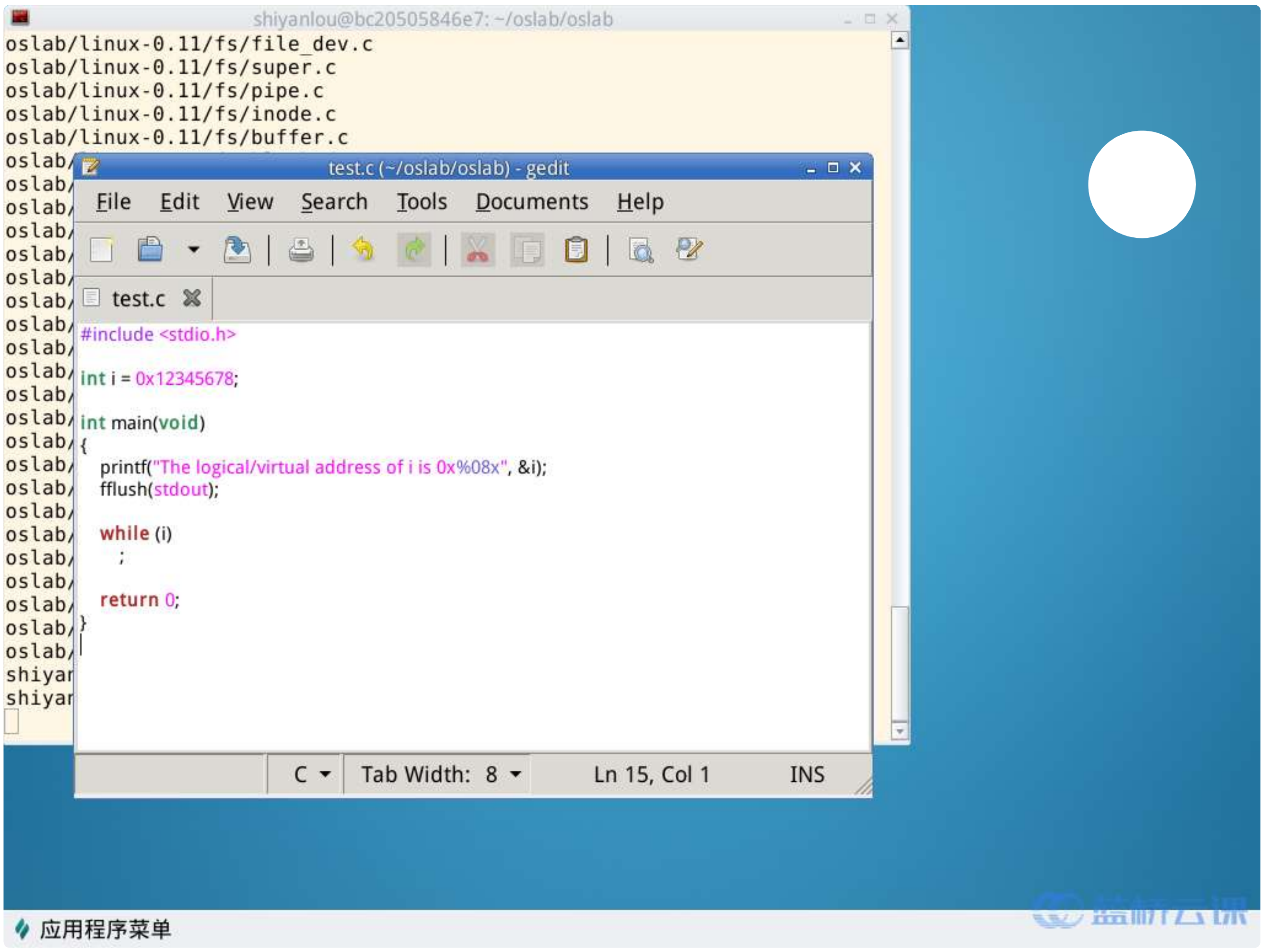
2029 0

“操作系统原理与实践”实验报告

地址映射与共享

跟踪地址翻译过程

首先把test.c程序写好

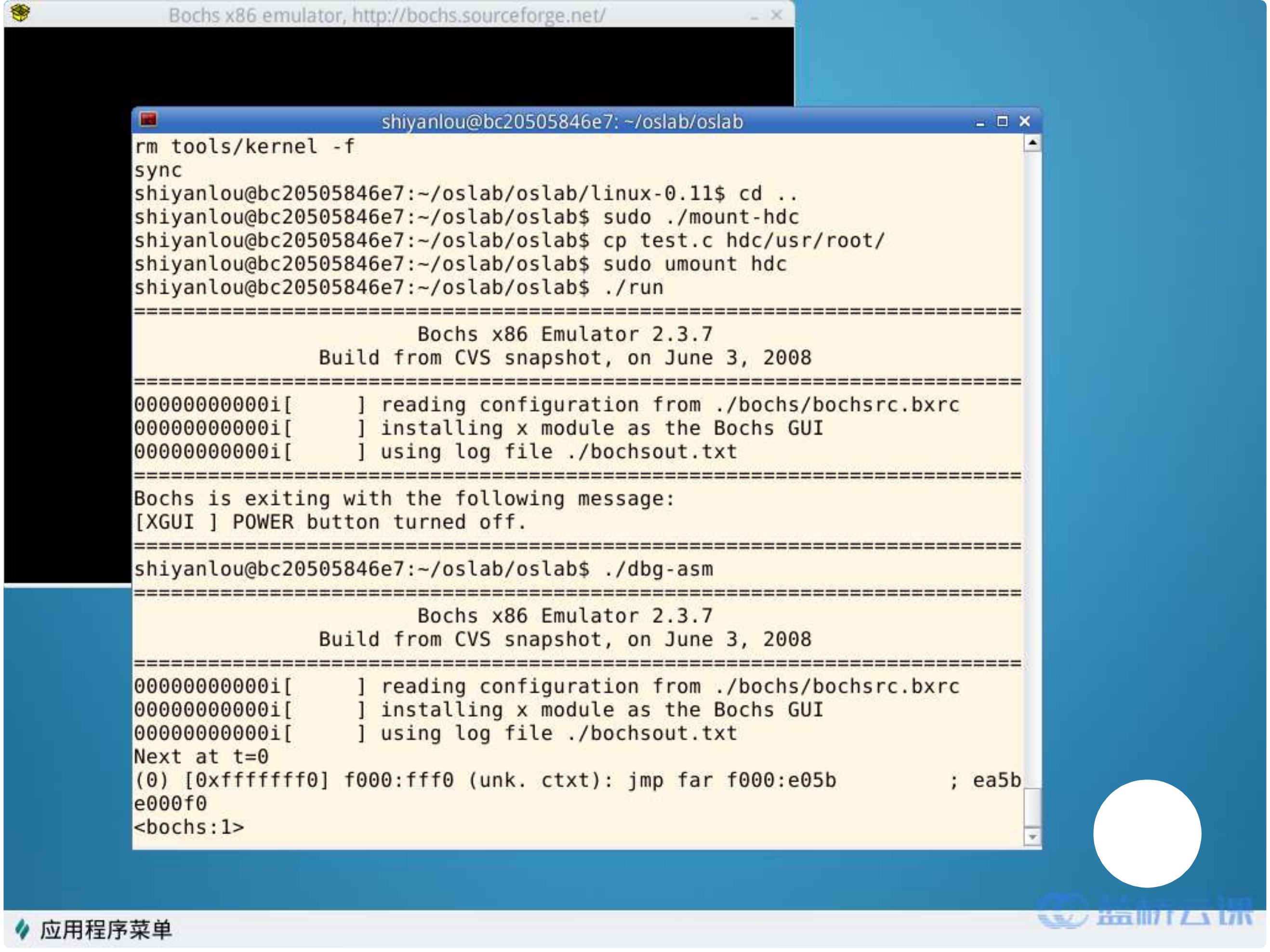


```
oslab/linux-0.11/fs/file_dev.c
oslab/linux-0.11/fs/super.c
oslab/linux-0.11/fs/pipe.c
oslab/linux-0.11/fs/inode.c
oslab/linux-0.11/fs/buffer.c
```

```
#include <stdio.h>

int i = 0x12345678;

int main(void)
{
    printf("The logical/virtual address of i is 0x%08x", &i);
    fflush(stdout);

    while (i)
        ;

    return 0;
}
```

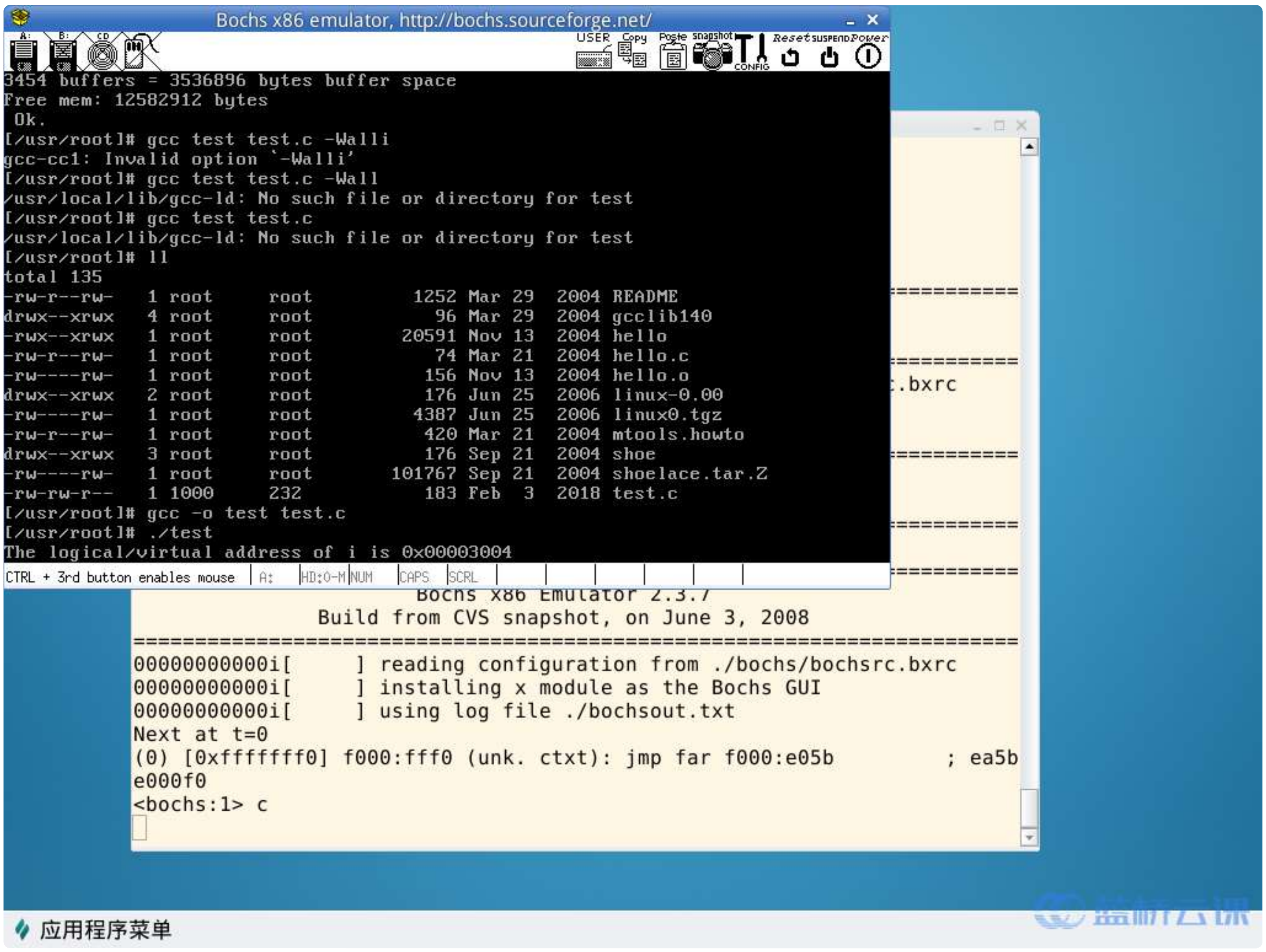
shiyanlou@bc20505846e7: ~/oslab/oslab
test.c (~/oslab/oslab) - gedit
File Edit View Search Tools Documents Help
test.c
C
Tab Width: 8
Ln 15, Col 1
INS
应用程序菜单
蓝桥云课

编译好内核，接着把test.c程序复制到linux0.11中输入`./dbg-asm`


Bochs x86 emulator, http://bochs.sourceforge.net/
shiyanlou@bc20505846e7: ~/oslab/oslab
应用程序菜单

```
rm tools/kernel -f
sync
shiyanlou@bc20505846e7:~/oslab/oslab/linux-0.11$ cd ..
shiyanlou@bc20505846e7:~/oslab/oslab$ sudo ./mount-hdc
shiyanlou@bc20505846e7:~/oslab/oslab$ cp test.c hdc/usr/root/
shiyanlou@bc20505846e7:~/oslab/oslab$ sudo umount hdc
shiyanlou@bc20505846e7:~/oslab/oslab$ ./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
========================================================================
Bochs is exiting with the following message:
[XGUI ] POWER button turned off.
========================================================================
shiyanlou@bc20505846e7:~/oslab/oslab$ ./dbg-asm
========================================================================
                       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
Next at t=0
(0) [0xfffffff0] f000:fff0 (unk. ctxt): jmp far f000:e05b         ; ea5b
e000f0
<bochs:1>
```

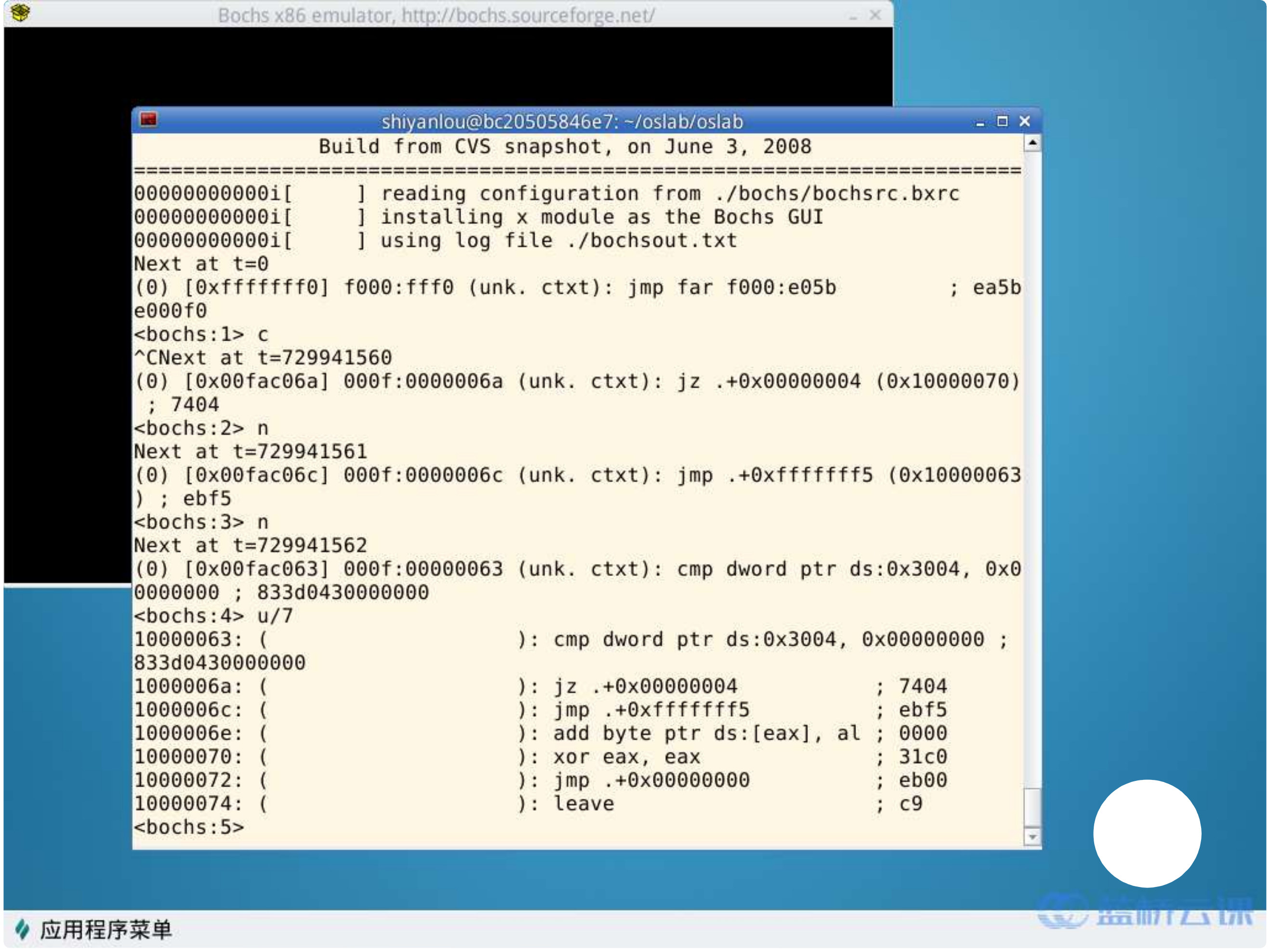
输入c运行，编译运行，打印出如下信息


Bochs x86 emulator, http://bochs.sourceforge.net/
USER Copy Paste snapshot CONFIG Reset SUSPEND Power
CTRL + 3rd button enables mouse A: HD:0-M NUM CAPS SCRL
应用程序菜单

```
3454 buffers = 3536896 bytes buffer space
Free mem: 12582912 bytes
 Ok.
[/usr/root]# gcc test test.c -Walli
gcc-cc1: Invalid option `-Walli'
[/usr/root]# gcc test test.c -Wall
/usr/local/lib/gcc-ld: No such file or directory for test
[/usr/root]# gcc test test.c
/usr/local/lib/gcc-ld: No such file or directory for test
[/usr/root]# ll
total 135
-rw-r--rw-   1 root     root         1252 Mar 29  2004 README
drwx--xrwx   4 root     root           96 Mar 29  2004 gcclib140
-rwx--xrwx   1 root     root        20591 Nov 13  2004 hello
-rw-r--rw-   1 root     root           74 Mar 21  2004 hello.c
-rw----rw-   1 root     root          156 Nov 13  2004 hello.o
drwx--xrwx   2 root     root          176 Jun 25  2006 linux-0.00
-rw----rw-   1 root     root         4387 Jun 25  2006 linux0.tgz
-rw-r--rw-   1 root     root          420 Mar 21  2004 mtools.howto
drwx--xrwx   3 root     root          176 Sep 21  2004 shoe
-rw----rw-   1 root     root       101767 Sep 21  2004 shoelace.tar.Z
-rw-rw-r--   1 1000     232           183 Feb  3  2018 test.c
[/usr/root]# gcc -o test test.c
[/usr/root]# ./test
The logical/virtual address of i is 0x00003004
```

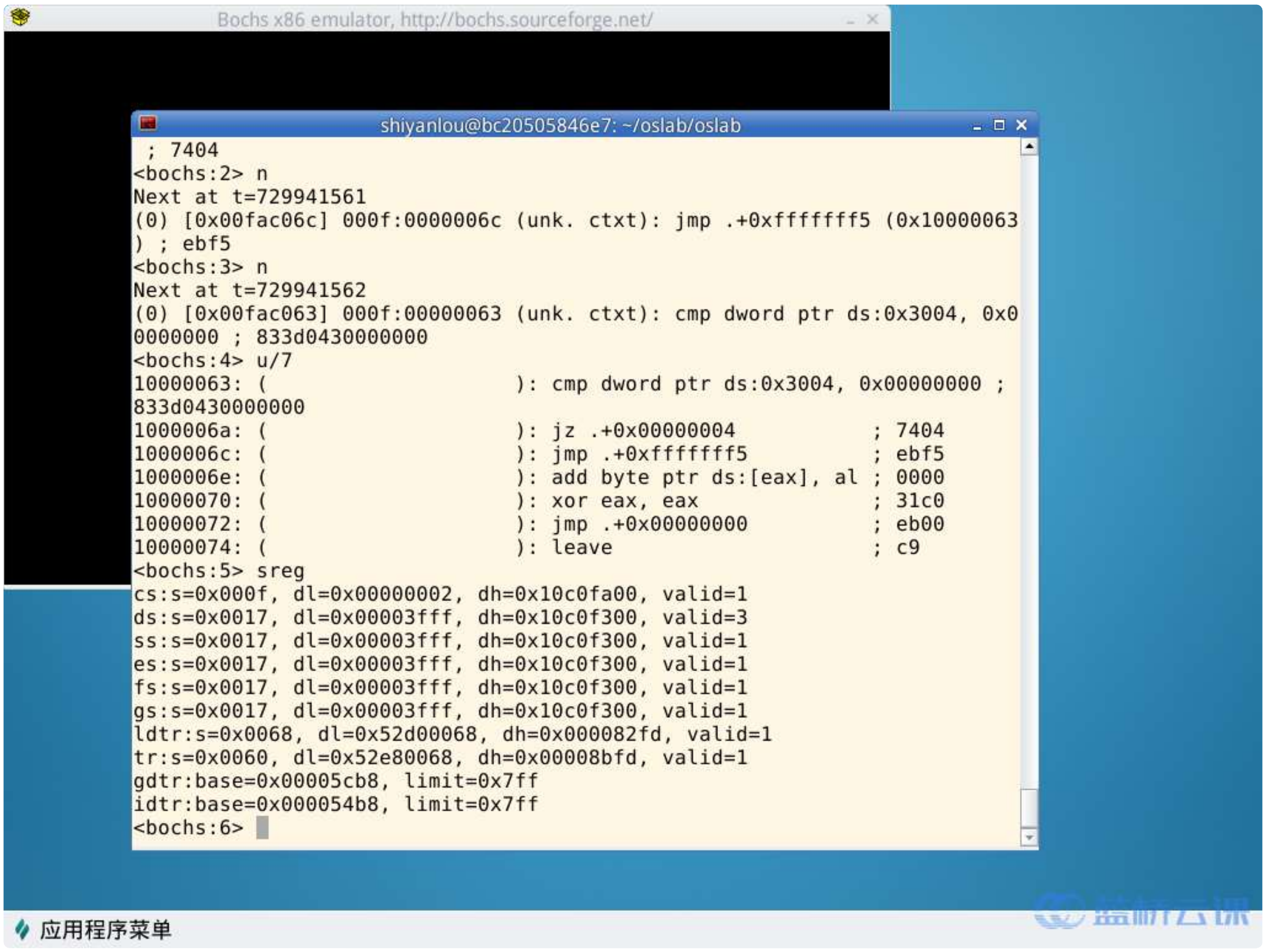
```
                       Bochs x86 Emulator 2.3.7
               Build from CVS snapshot, on June 3, 2008
========================================================================
00000000000i[     ] reading configuration from ./bochs/bochsrc.bxrc
00000000000i[     ] installing x module as the Bochs GUI
00000000000i[     ] using log file ./bochsout.txt
Next at t=0
(0) [0xfffffff0] f000:fff0 (unk. ctxt): jmp far f000:e05b         ; ea5b
e000f0
<bochs:1> c
```

单步运行至出现cmp，输入`u/7`反汇编7条汇编语句


Bochs x86 emulator, http://bochs.sourceforge.net/
shiyanlou@bc20505846e7: ~/oslab/oslab
应用程序菜单
蓝桥云课

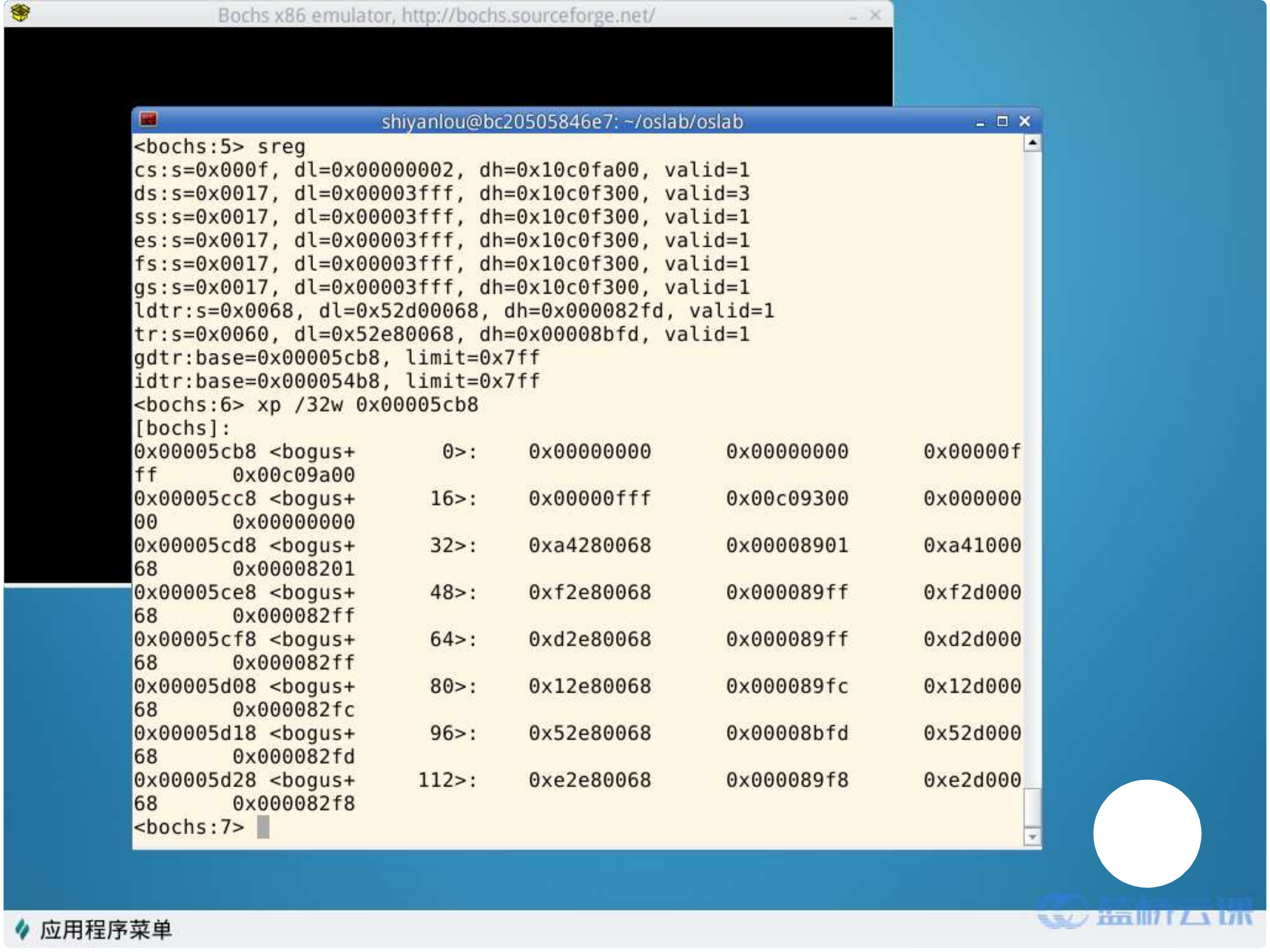
```
                Build from CVS snapshot, on June 3, 2008
=========================================================================
00000000000i[     ] reading configuration from ./bochs/bochsrc.bxrc
00000000000i[     ] installing x module as the Bochs GUI
00000000000i[     ] using log file ./bochsout.txt
Next at t=0
(0) [0xfffffff0] f000:fff0 (unk. ctxt): jmp far f000:e05b         ; ea5b
e000f0
<bochs:1> c
^CNext at t=729941560
(0) [0x00fac06a] 000f:0000006a (unk. ctxt): jz .+0x00000004 (0x10000070)
 ; 7404
<bochs:2> n
Next at t=729941561
(0) [0x00fac06c] 000f:0000006c (unk. ctxt): jmp .+0xfffffff5 (0x10000063
) ; ebf5
<bochs:3> n
Next at t=729941562
(0) [0x00fac063] 000f:00000063 (unk. ctxt): cmp dword ptr ds:0x3004, 0x0
0000000 ; 833d0430000000
<bochs:4> u/7
10000063: (                    ): cmp dword ptr ds:0x3004, 0x00000000 ;
833d0430000000
1000006a: (                    ): jz .+0x00000004          ; 7404
1000006c: (                    ): jmp .+0xfffffff5         ; ebf5
1000006e: (                    ): add byte ptr ds:[eax], al ; 0000
10000070: (                    ): xor eax, eax             ; 31c0
10000072: (                    ): jmp .+0x00000000         ; eb00
10000074: (                    ): leave                    ; c9
<bochs:5>
```

用sreg命令


Bochs x86 emulator, http://bochs.sourceforge.net/
shiyanlou@bc20505846e7: ~/oslab/oslab
应用程序菜单
蓝桥云课

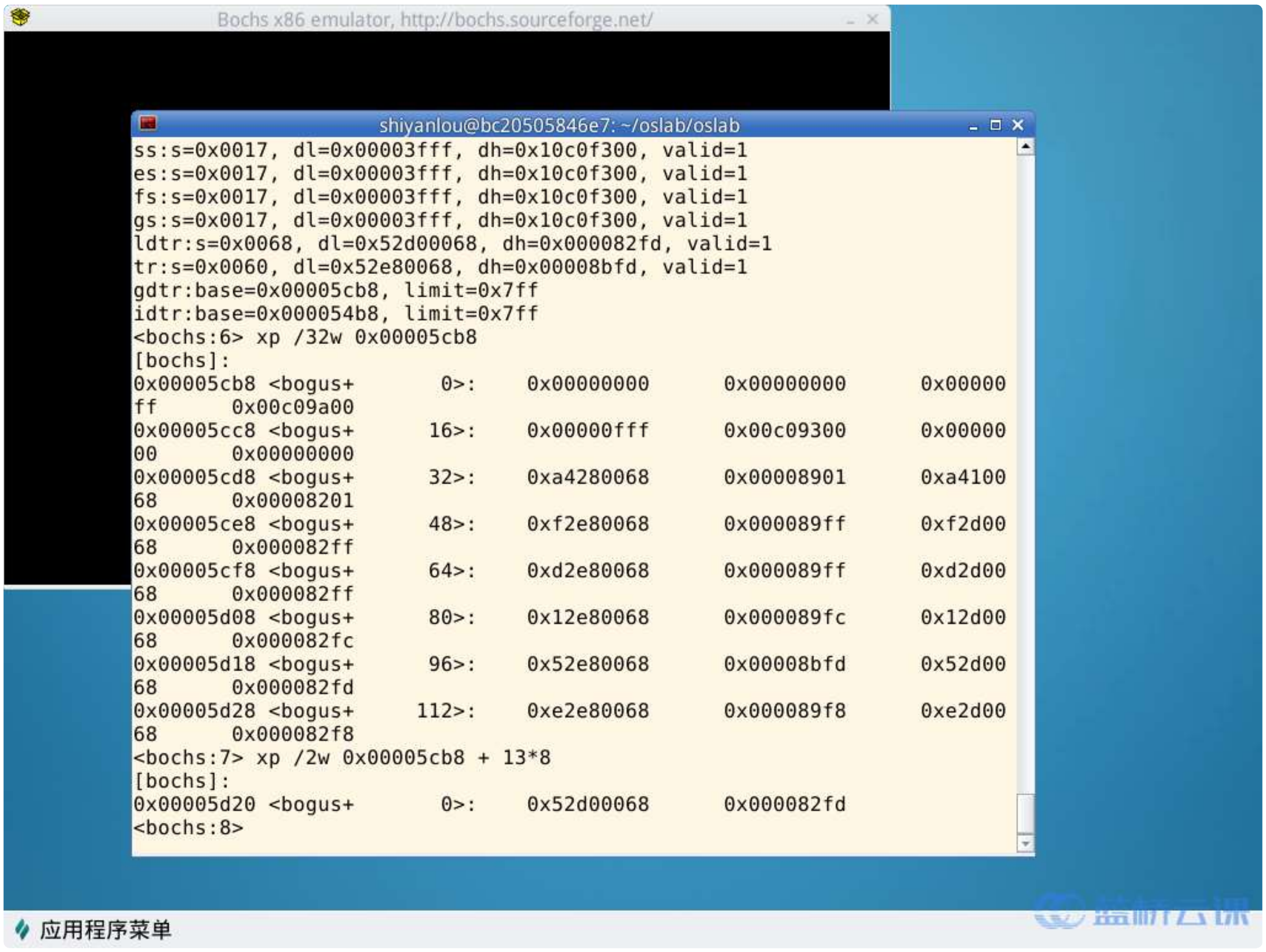
```
 ; 7404
<bochs:2> n
Next at t=729941561
(0) [0x00fac06c] 000f:0000006c (unk. ctxt): jmp .+0xfffffff5 (0x10000063
) ; ebf5
<bochs:3> n
Next at t=729941562
(0) [0x00fac063] 000f:00000063 (unk. ctxt): cmp dword ptr ds:0x3004, 0x0
0000000 ; 833d0430000000
<bochs:4> u/7
10000063: (                    ): cmp dword ptr ds:0x3004, 0x00000000 ;
833d0430000000
1000006a: (                    ): jz .+0x00000004          ; 7404
1000006c: (                    ): jmp .+0xfffffff5         ; ebf5
1000006e: (                    ): add byte ptr ds:[eax], al ; 0000
10000070: (                    ): xor eax, eax             ; 31c0
10000072: (                    ): jmp .+0x00000000         ; eb00
10000074: (                    ): leave                    ; c9
<bochs:5> sreg
cs:s=0x000f, dl=0x00000002, dh=0x10c0fa00, valid=1
ds:s=0x0017, dl=0x00003fff, dh=0x10c0f300, valid=3
ss:s=0x0017, dl=0x00003fff, dh=0x10c0f300, valid=1
es:s=0x0017, dl=0x00003fff, dh=0x10c0f300, valid=1
fs:s=0x0017, dl=0x00003fff, dh=0x10c0f300, valid=1
gs:s=0x0017, dl=0x00003fff, dh=0x10c0f300, valid=1
ldtr:s=0x0068, dl=0x52d00068, dh=0x000082fd, valid=1
tr:s=0x0060, dl=0x52e80068, dh=0x00008bfd, valid=1
gdtr:base=0x00005cb8, limit=0x7ff
idtr:base=0x000054b8, limit=0x7ff
<bochs:6>
```

可以看到ldtr的值是0x0068=0000000001101000（二进制），表示LDT表存放在GDT表的1101(二进制)=13（十进制）号位置（每位数据的意义参考后文叙述的段选择子）。而GDT的位置已经由gdtr明确给出，在物理地址的0x00005cb8。用“xp /32w 0x00005cb8”查看从该地址开始，32个字的内容，及GDT表的前16项


Bochs x86 emulator, http://bochs.sourceforge.net/
shiyanlou@bc20505846e7: ~/oslab/oslab
应用程序菜单

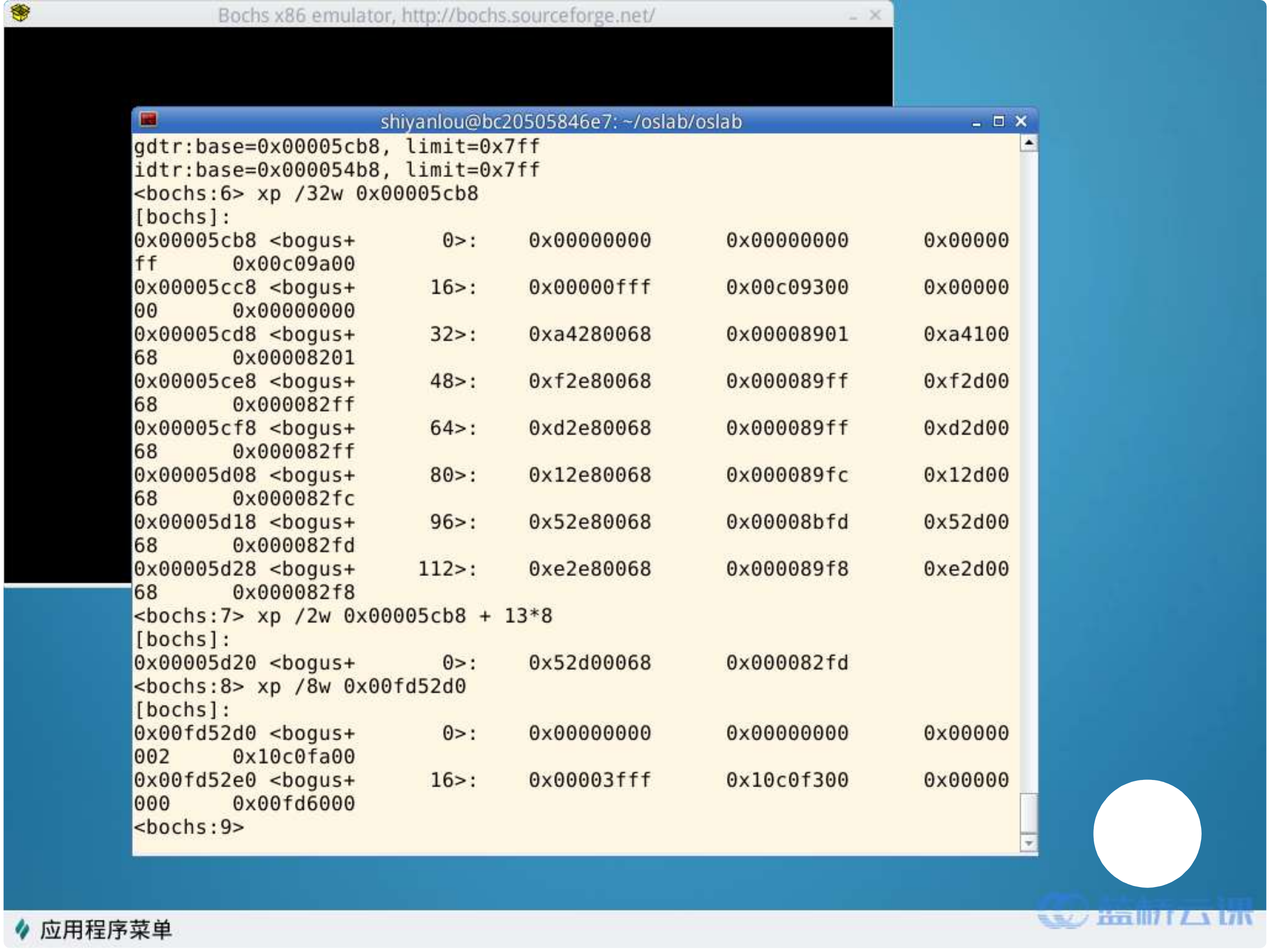
```
<bochs:5> sreg
cs:s=0x000f, dl=0x00000002, dh=0x10c0fa00, valid=1
ds:s=0x0017, dl=0x00003fff, dh=0x10c0f300, valid=3
ss:s=0x0017, dl=0x00003fff, dh=0x10c0f300, valid=1
es:s=0x0017, dl=0x00003fff, dh=0x10c0f300, valid=1
fs:s=0x0017, dl=0x00003fff, dh=0x10c0f300, valid=1
gs:s=0x0017, dl=0x00003fff, dh=0x10c0f300, valid=1
ldtr:s=0x0068, dl=0x52d00068, dh=0x000082fd, valid=1
tr:s=0x0060, dl=0x52e80068, dh=0x00008bfd, valid=1
gdtr:base=0x00005cb8, limit=0x7ff
idtr:base=0x000054b8, limit=0x7ff
<bochs:6> xp /32w 0x00005cb8
[bochs]:
0x00005cb8 <bogus+       0>:    0x00000000      0x00000000      0x00000f
ff       0x00c09a00
0x00005cc8 <bogus+      16>:    0x00000fff      0x00c09300      0x000000
00       0x00000000
0x00005cd8 <bogus+      32>:    0xa4280068      0x00008901      0xa41000
68       0x00008201
0x00005ce8 <bogus+      48>:    0xf2e80068      0x000089ff      0xf2d000
68       0x000082ff
0x00005cf8 <bogus+      64>:    0xd2e80068      0x000089ff      0xd2d000
68       0x000082ff
0x00005d08 <bogus+      80>:    0x12e80068      0x000089fc      0x12d000
68       0x000082fc
0x00005d18 <bogus+      96>:    0x52e80068      0x00008bfd      0x52d000
68       0x000082fd
0x00005d28 <bogus+     112>:    0xe2e80068      0x000089f8      0xe2d000
68       0x000082f8
<bochs:7>
```

GDT表中的每一项占64位（8个字节），所以我们要查找的项的地址是“0x00005cb8 + 13*8”


Bochs x86 emulator, http://bochs.sourceforge.net/
shiyanlou@bc20505846e7: ~/oslab/oslab
应用程序菜单

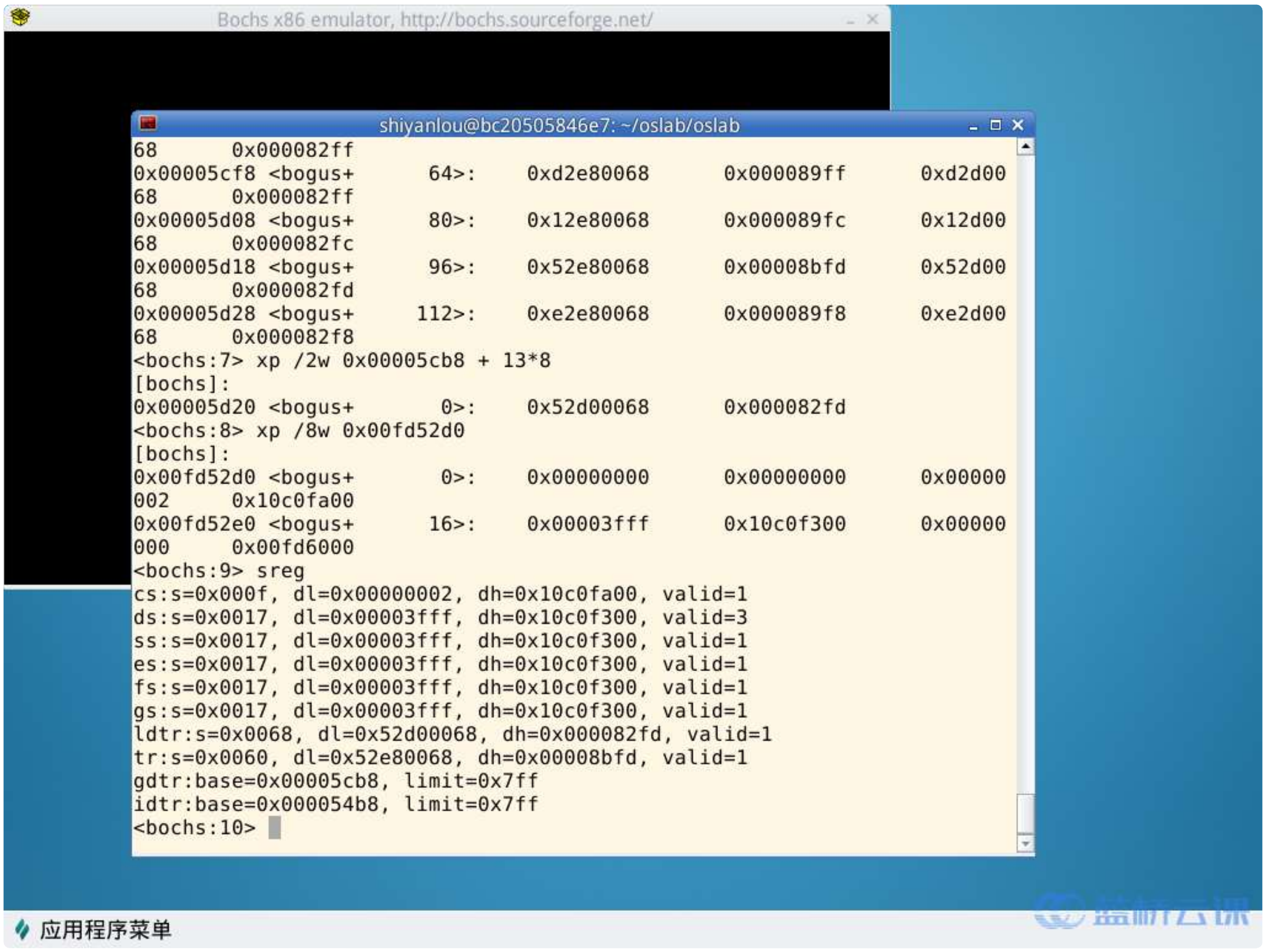
```
ss:s=0x0017, dl=0x00003fff, dh=0x10c0f300, valid=1
es:s=0x0017, dl=0x00003fff, dh=0x10c0f300, valid=1
fs:s=0x0017, dl=0x00003fff, dh=0x10c0f300, valid=1
gs:s=0x0017, dl=0x00003fff, dh=0x10c0f300, valid=1
ldtr:s=0x0068, dl=0x52d00068, dh=0x000082fd, valid=1
tr:s=0x0060, dl=0x52e80068, dh=0x00008bfd, valid=1
gdtr:base=0x00005cb8, limit=0x7ff
idtr:base=0x000054b8, limit=0x7ff
<bochs:6> xp /32w 0x00005cb8
[bochs]:
0x00005cb8 <bogus+       0>:    0x00000000      0x00000000      0x00000
ff       0x00c09a00
0x00005cc8 <bogus+      16>:    0x00000fff      0x00c09300      0x00000
00       0x00000000
0x00005cd8 <bogus+      32>:    0xa4280068      0x00008901      0xa4100
68       0x00008201
0x00005ce8 <bogus+      48>:    0xf2e80068      0x000089ff      0xf2d00
68       0x000082ff
0x00005cf8 <bogus+      64>:    0xd2e80068      0x000089ff      0xd2d00
68       0x000082ff
0x00005d08 <bogus+      80>:    0x12e80068      0x000089fc      0x12d00
68       0x000082fc
0x00005d18 <bogus+      96>:    0x52e80068      0x00008bfd      0x52d00
68       0x000082fd
0x00005d28 <bogus+     112>:    0xe2e80068      0x000089f8      0xe2d00
68       0x000082f8
<bochs:7> xp /2w 0x00005cb8 + 13*8
[bochs]:
0x00005d20 <bogus+       0>:    0x52d00068      0x000082fd
<bochs:8>
```

重新组合之后得到“0x00fd52d0”输入`xp /8w 0x00fd52d0`，得到ldt表前4项



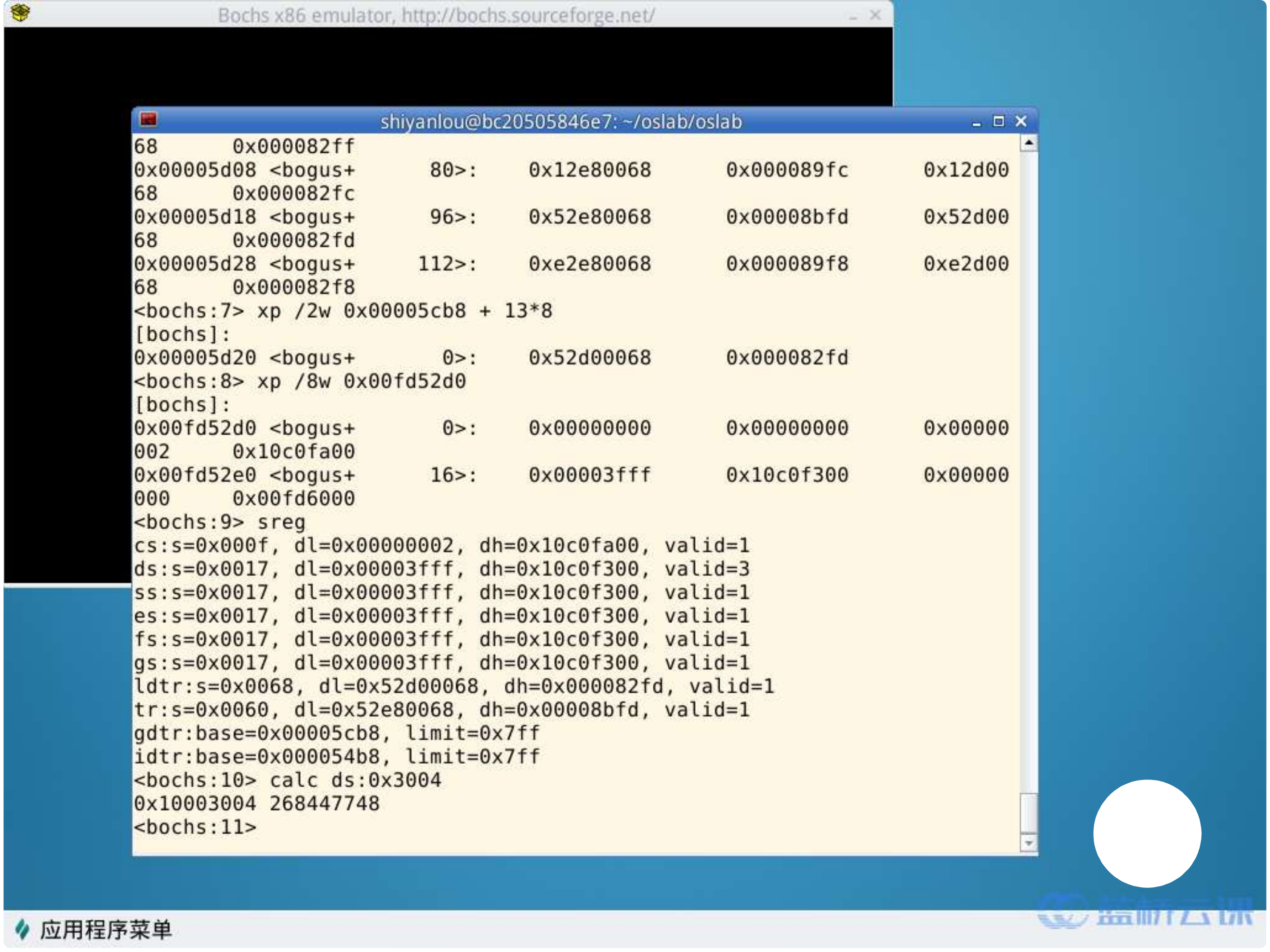
```
gdtr:base=0x00005cb8, limit=0x7ff
idtr:base=0x000054b8, limit=0x7ff
<bochs:6> xp /32w 0x00005cb8
[bochs]:
0x00005cb8 <bogus+       0>:    0x00000000      0x00000000      0x00000
ff       0x00c09a00
0x00005cc8 <bogus+      16>:    0x00000fff      0x00c09300      0x00000
00       0x00000000
0x00005cd8 <bogus+      32>:    0xa4280068      0x00008901      0xa4100
68       0x00008201
0x00005ce8 <bogus+      48>:    0xf2e80068      0x000089ff      0xf2d00
68       0x000082ff
0x00005cf8 <bogus+      64>:    0xd2e80068      0x000089ff      0xd2d00
68       0x000082ff
0x00005d08 <bogus+      80>:    0x12e80068      0x000089fc      0x12d00
68       0x000082fc
0x00005d18 <bogus+      96>:    0x52e80068      0x00008bfd      0x52d00
68       0x000082fd
0x00005d28 <bogus+     112>:    0xe2e80068      0x000089f8      0xe2d00
68       0x000082f8
<bochs:7> xp /2w 0x00005cb8 + 13*8
[bochs]:
0x00005d20 <bogus+       0>:    0x52d00068      0x000082fd
<bochs:8> xp /8w 0x00fd52d0
[bochs]:
0x00fd52d0 <bogus+       0>:    0x00000000      0x00000000      0x00000
002      0x10c0fa00
0x00fd52e0 <bogus+      16>:    0x00003fff      0x10c0f300      0x00000
000      0x00fd6000
<bochs:9>
```

看ds选择子的内容，用`sreg`命令



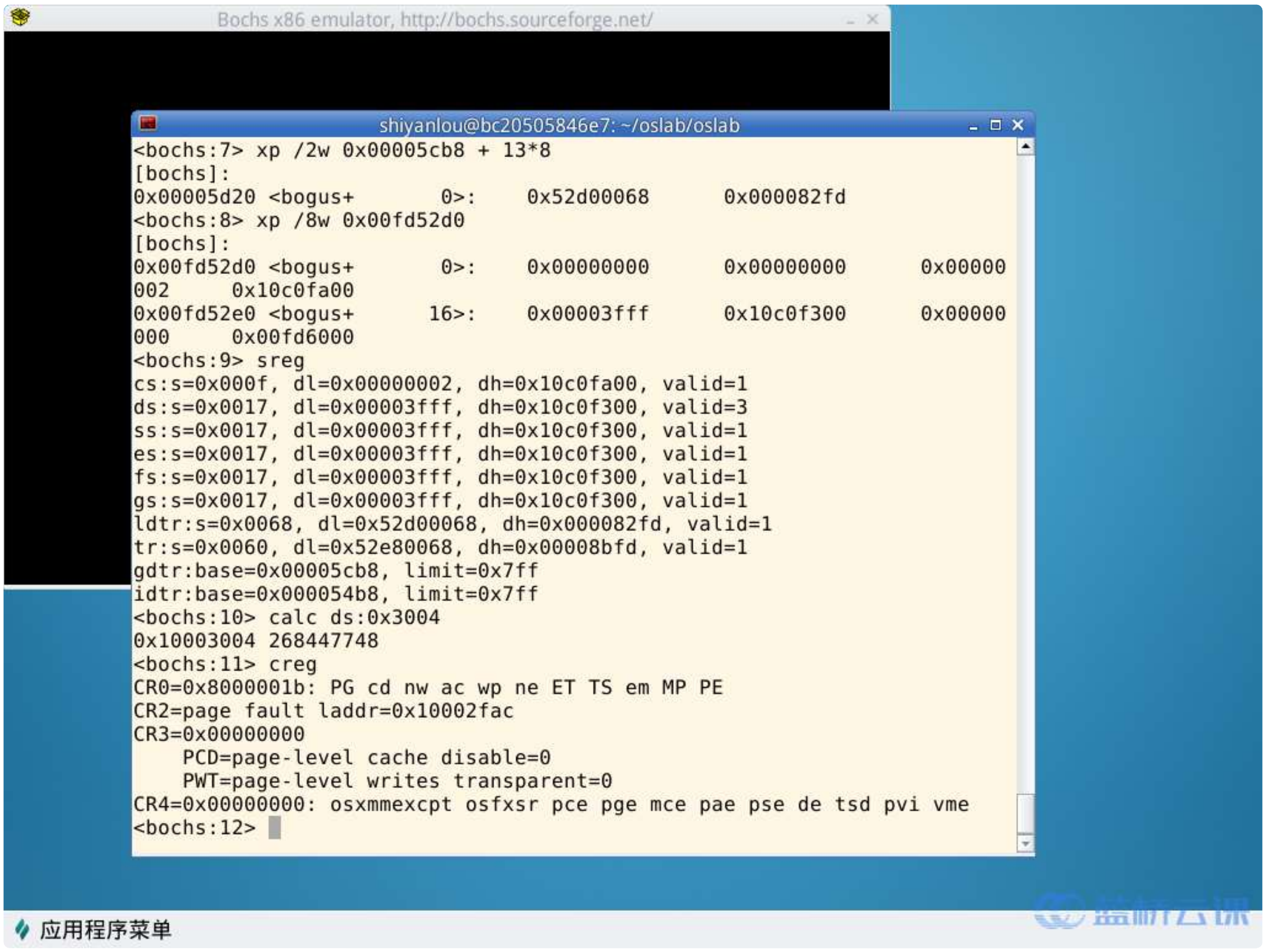
```
68       0x000082ff
0x00005cf8 <bogus+      64>:    0xd2e80068      0x000089ff      0xd2d00
68       0x000082ff
0x00005d08 <bogus+      80>:    0x12e80068      0x000089fc      0x12d00
68       0x000082fc
0x00005d18 <bogus+      96>:    0x52e80068      0x00008bfd      0x52d00
68       0x000082fd
0x00005d28 <bogus+     112>:    0xe2e80068      0x000089f8      0xe2d00
68       0x000082f8
<bochs:7> xp /2w 0x00005cb8 + 13*8
[bochs]:
0x00005d20 <bogus+       0>:    0x52d00068      0x000082fd
<bochs:8> xp /8w 0x00fd52d0
[bochs]:
0x00fd52d0 <bogus+       0>:    0x00000000      0x00000000      0x00000
002      0x10c0fa00
0x00fd52e0 <bogus+      16>:    0x00003fff      0x10c0f300      0x00000
000      0x00fd6000
<bochs:9> sreg
cs:s=0x000f, dl=0x00000002, dh=0x10c0fa00, valid=1
ds:s=0x0017, dl=0x00003fff, dh=0x10c0f300, valid=3
ss:s=0x0017, dl=0x00003fff, dh=0x10c0f300, valid=1
es:s=0x0017, dl=0x00003fff, dh=0x10c0f300, valid=1
fs:s=0x0017, dl=0x00003fff, dh=0x10c0f300, valid=1
gs:s=0x0017, dl=0x00003fff, dh=0x10c0f300, valid=1
ldtr:s=0x0068, dl=0x52d00068, dh=0x000082fd, valid=1
tr:s=0x0060, dl=0x52e80068, dh=0x00008bfd, valid=1
gdtr:base=0x00005cb8, limit=0x7ff
idtr:base=0x000054b8, limit=0x7ff
<bochs:10>
```

输入`calc ds:0x3004`，与推论一致


Bochs x86 emulator, http://bochs.sourceforge.net/
shiyanlou@bc20505846e7: ~/oslab/oslab
应用程序菜单

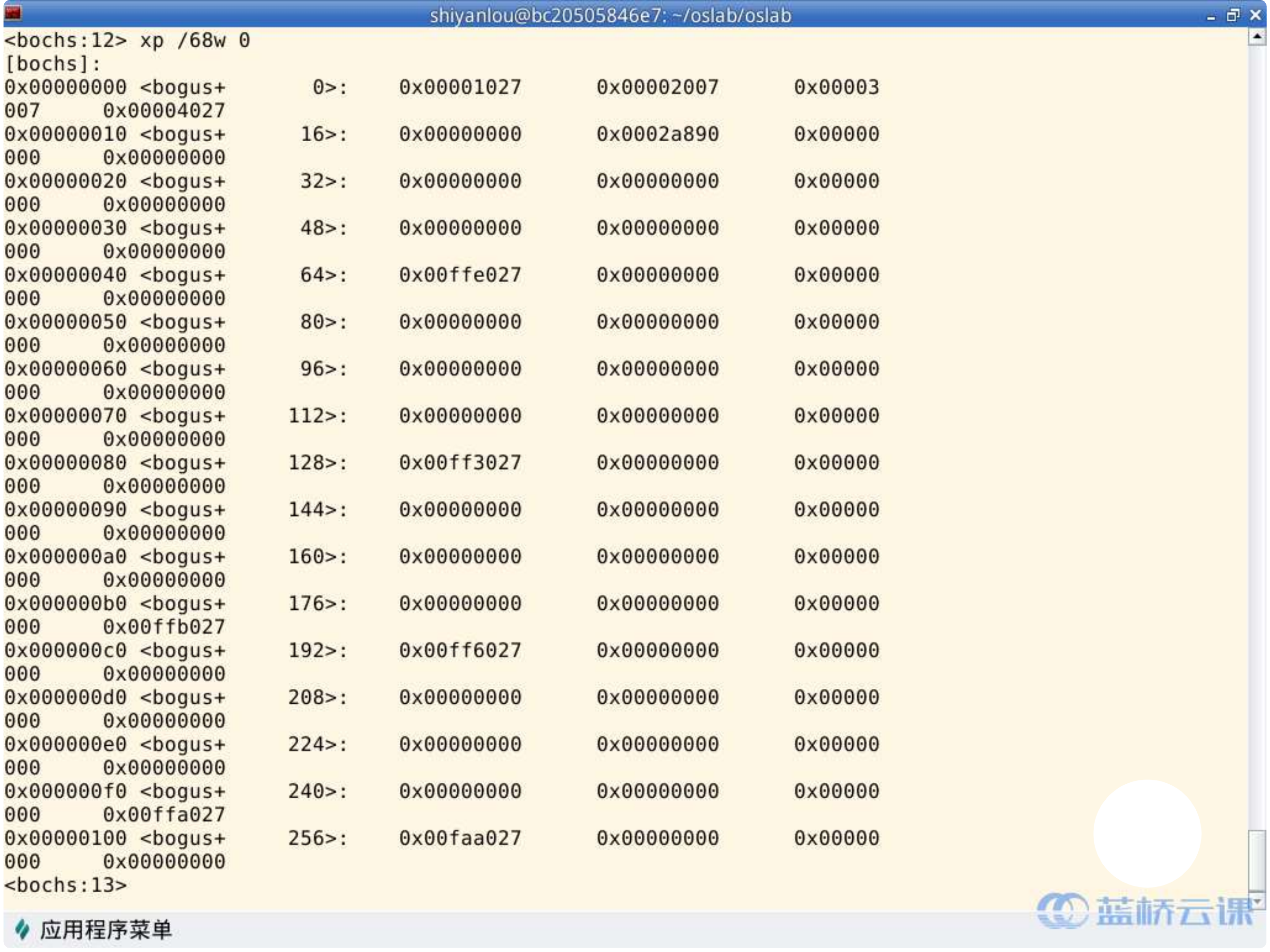
```
68      0x000082ff
0x00005d08 <bogus+      80>:    0x12e80068      0x000089fc      0x12d00
68      0x000082fc
0x00005d18 <bogus+      96>:    0x52e80068      0x00008bfd      0x52d00
68      0x000082fd
0x00005d28 <bogus+     112>:    0xe2e80068      0x000089f8      0xe2d00
68      0x000082f8
<bochs:7> xp /2w 0x00005cb8 + 13*8
[bochs]:
0x00005d20 <bogus+       0>:    0x52d00068      0x000082fd
<bochs:8> xp /8w 0x00fd52d0
[bochs]:
0x00fd52d0 <bogus+       0>:    0x00000000      0x00000000      0x00000
002     0x10c0fa00
0x00fd52e0 <bogus+      16>:    0x00003fff      0x10c0f300      0x00000
000     0x00fd6000
<bochs:9> sreg
cs:s=0x000f, dl=0x00000002, dh=0x10c0fa00, valid=1
ds:s=0x0017, dl=0x00003fff, dh=0x10c0f300, valid=3
ss:s=0x0017, dl=0x00003fff, dh=0x10c0f300, valid=1
es:s=0x0017, dl=0x00003fff, dh=0x10c0f300, valid=1
fs:s=0x0017, dl=0x00003fff, dh=0x10c0f300, valid=1
gs:s=0x0017, dl=0x00003fff, dh=0x10c0f300, valid=1
ldtr:s=0x0068, dl=0x52d00068, dh=0x000082fd, valid=1
tr:s=0x0060, dl=0x52e80068, dh=0x00008bfd, valid=1
gdtr:base=0x00005cb8, limit=0x7ff
idtr:base=0x000054b8, limit=0x7ff
<bochs:10> calc ds:0x3004
0x10003004 268447748
<bochs:11>
```

输入creg命令，得到如下图


Bochs x86 emulator, http://bochs.sourceforge.net/
shiyanlou@bc20505846e7: ~/oslab/oslab
应用程序菜单

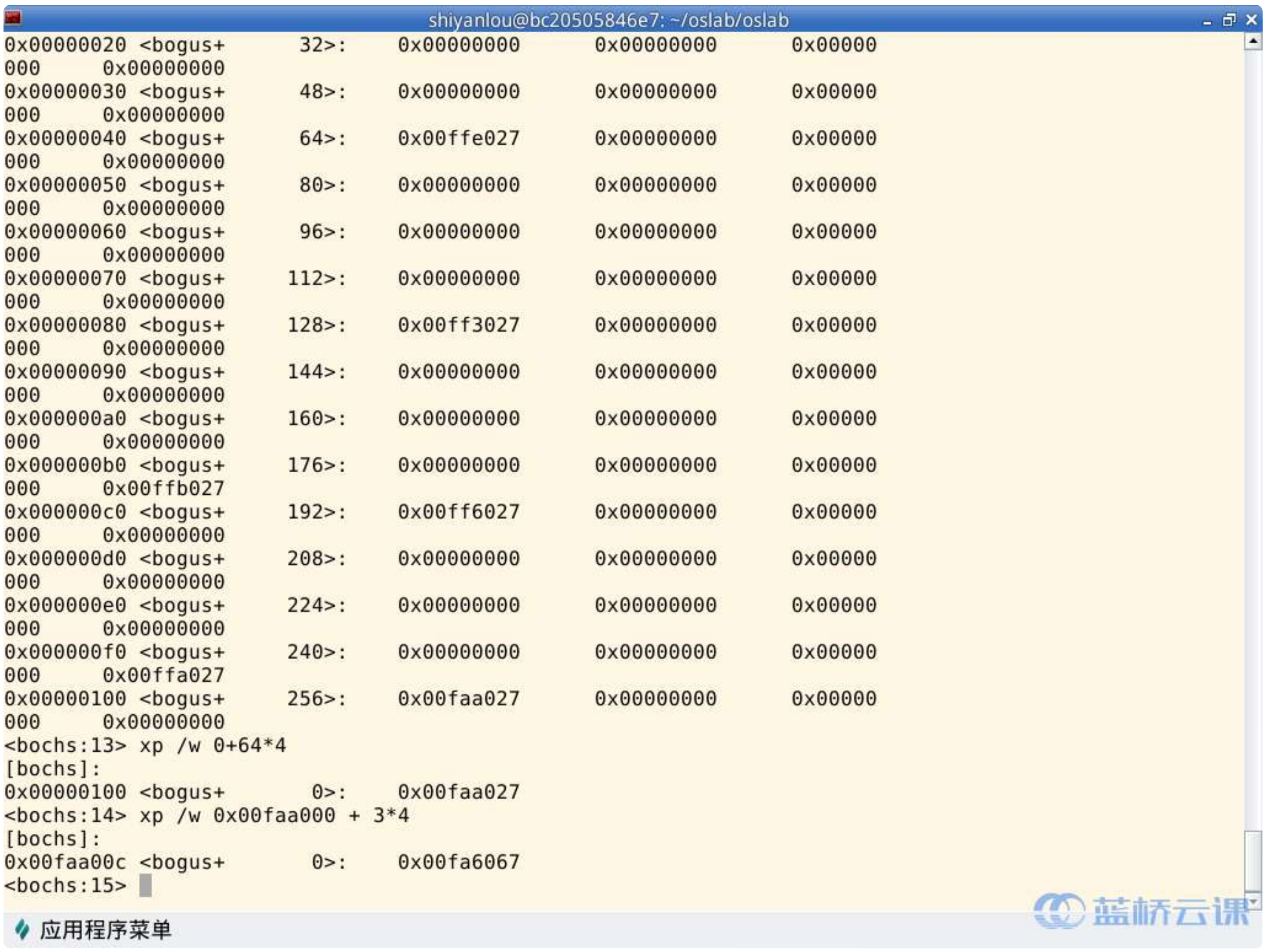
```
<bochs:7> xp /2w 0x00005cb8 + 13*8
[bochs]:
0x00005d20 <bogus+       0>:    0x52d00068      0x000082fd
<bochs:8> xp /8w 0x00fd52d0
[bochs]:
0x00fd52d0 <bogus+       0>:    0x00000000      0x00000000      0x00000
002     0x10c0fa00
0x00fd52e0 <bogus+      16>:    0x00003fff      0x10c0f300      0x00000
000     0x00fd6000
<bochs:9> sreg
cs:s=0x000f, dl=0x00000002, dh=0x10c0fa00, valid=1
ds:s=0x0017, dl=0x00003fff, dh=0x10c0f300, valid=3
ss:s=0x0017, dl=0x00003fff, dh=0x10c0f300, valid=1
es:s=0x0017, dl=0x00003fff, dh=0x10c0f300, valid=1
fs:s=0x0017, dl=0x00003fff, dh=0x10c0f300, valid=1
gs:s=0x0017, dl=0x00003fff, dh=0x10c0f300, valid=1
ldtr:s=0x0068, dl=0x52d00068, dh=0x000082fd, valid=1
tr:s=0x0060, dl=0x52e80068, dh=0x00008bfd, valid=1
gdtr:base=0x00005cb8, limit=0x7ff
idtr:base=0x000054b8, limit=0x7ff
<bochs:10> calc ds:0x3004
0x10003004 268447748
<bochs:11> creg
CR0=0x8000001b: PG cd nw ac wp ne ET TS em MP PE
CR2=page fault laddr=0x10002fac
CR3=0x00000000
    PCD=page-level cache disable=0
    PWT=page-level writes transparent=0
CR4=0x00000000: osxmmexcpt osfxsr pce pge mce pae pse de tsd pvi vme
<bochs:12>
```

用xp /68w 0 查看内容



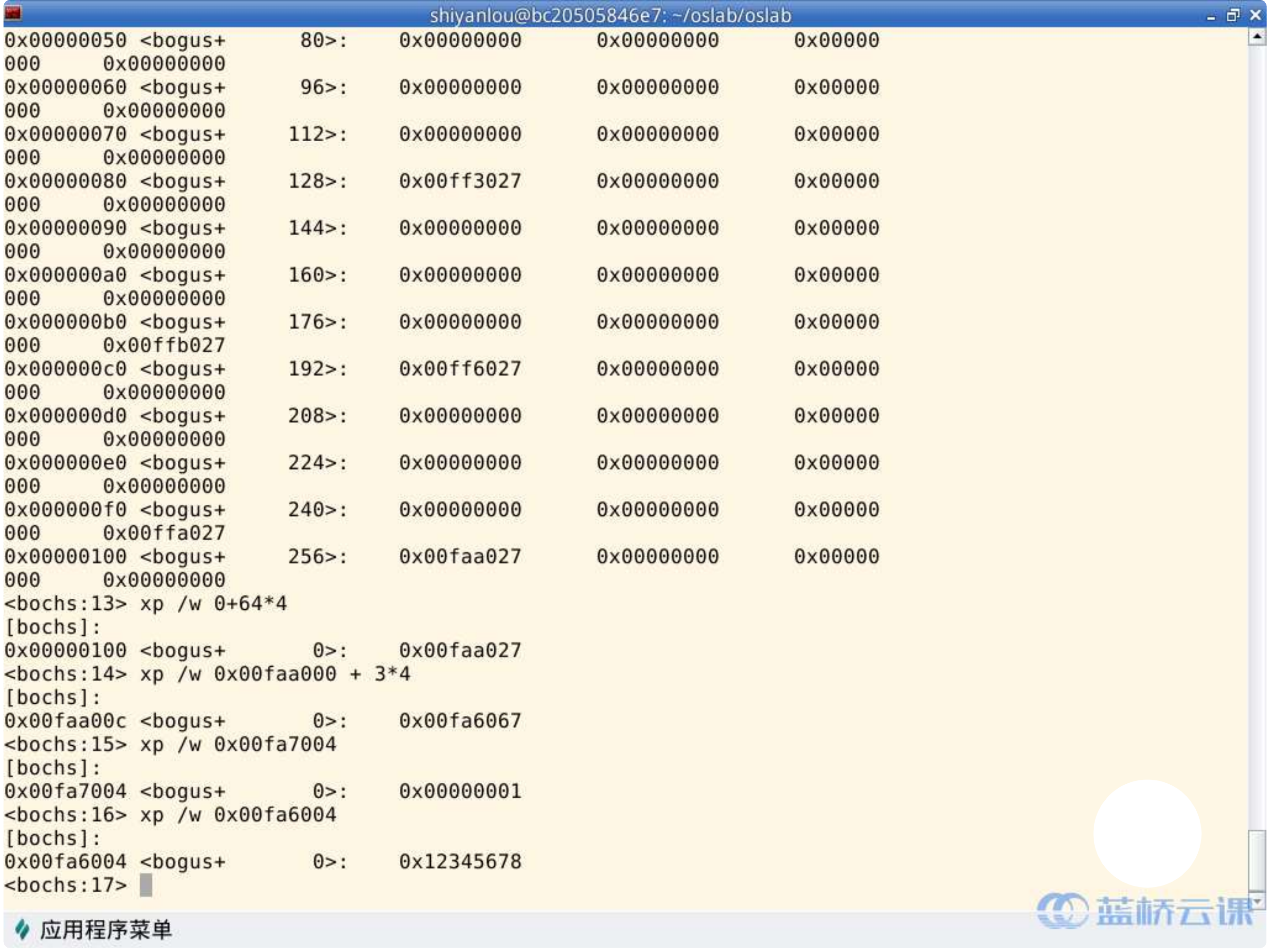
```
<bochs:12> xp /68w 0
[bochs]:
0x00000000 <bogus+       0>:    0x00001027      0x00002007      0x00003
007      0x00004027
0x00000010 <bogus+      16>:    0x00000000      0x0002a890      0x00000
000      0x00000000
0x00000020 <bogus+      32>:    0x00000000      0x00000000      0x00000
000      0x00000000
0x00000030 <bogus+      48>:    0x00000000      0x00000000      0x00000
000      0x00000000
0x00000040 <bogus+      64>:    0x00ffe027      0x00000000      0x00000
000      0x00000000
0x00000050 <bogus+      80>:    0x00000000      0x00000000      0x00000
000      0x00000000
0x00000060 <bogus+      96>:    0x00000000      0x00000000      0x00000
000      0x00000000
0x00000070 <bogus+     112>:    0x00000000      0x00000000      0x00000
000      0x00000000
0x00000080 <bogus+     128>:    0x00ff3027      0x00000000      0x00000
000      0x00000000
0x00000090 <bogus+     144>:    0x00000000      0x00000000      0x00000
000      0x00000000
0x000000a0 <bogus+     160>:    0x00000000      0x00000000      0x00000
000      0x00000000
0x000000b0 <bogus+     176>:    0x00000000      0x00000000      0x00000
000      0x00ffb027
0x000000c0 <bogus+     192>:    0x00ff6027      0x00000000      0x00000
000      0x00000000
0x000000d0 <bogus+     208>:    0x00000000      0x00000000      0x00000
000      0x00000000
0x000000e0 <bogus+     224>:    0x00000000      0x00000000      0x00000
000      0x00000000
0x000000f0 <bogus+     240>:    0x00000000      0x00000000      0x00000
000      0x00ffa027
0x00000100 <bogus+     256>:    0x00faa027      0x00000000      0x00000
000      0x00000000
<bochs:13>
```

经过一顿推断查找



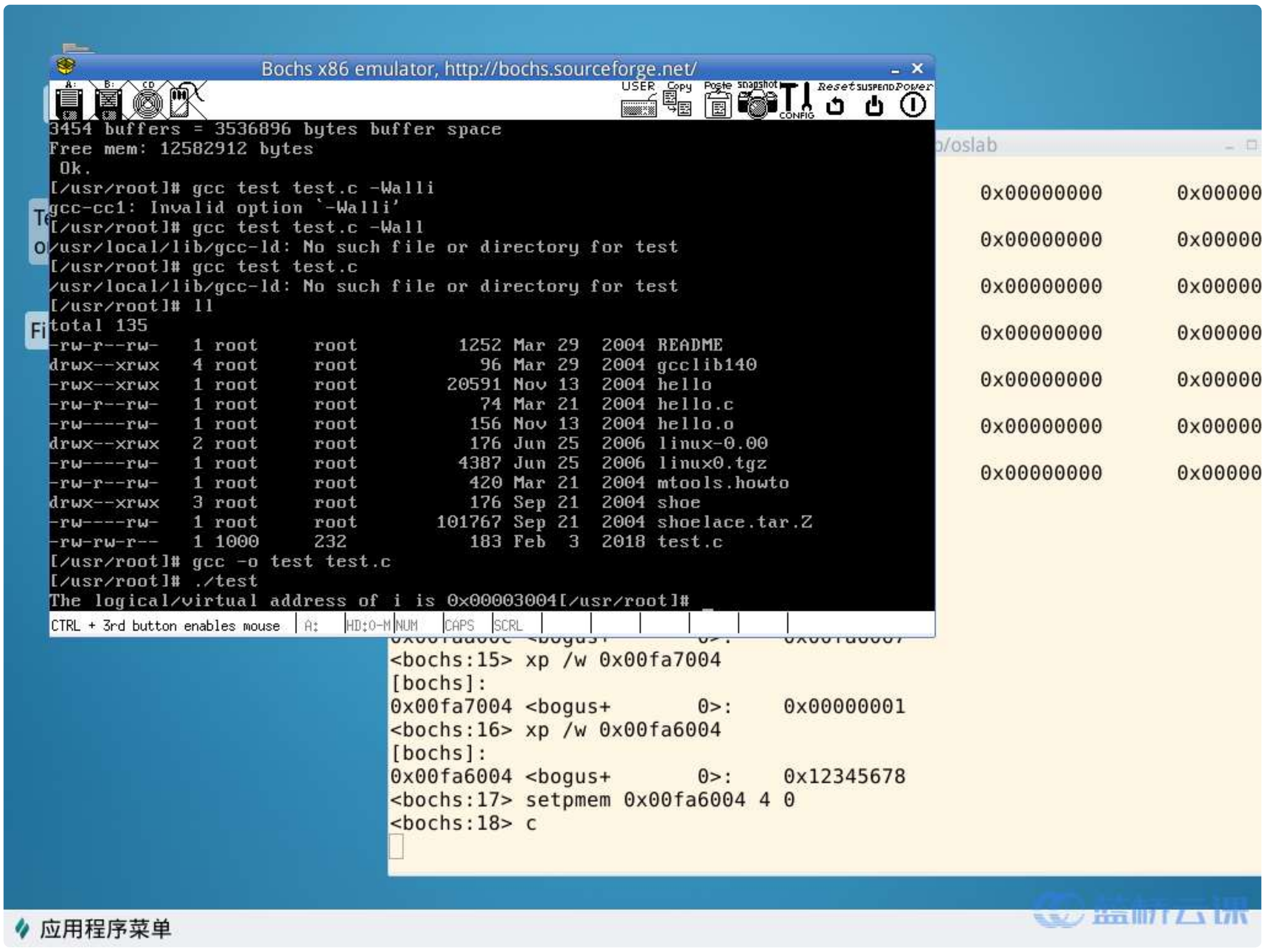
```
0x00000020 <bogus+      32>:    0x00000000      0x00000000      0x00000
000      0x00000000
0x00000030 <bogus+      48>:    0x00000000      0x00000000      0x00000
000      0x00000000
0x00000040 <bogus+      64>:    0x00ffe027      0x00000000      0x00000
000      0x00000000
0x00000050 <bogus+      80>:    0x00000000      0x00000000      0x00000
000      0x00000000
0x00000060 <bogus+      96>:    0x00000000      0x00000000      0x00000
000      0x00000000
0x00000070 <bogus+     112>:    0x00000000      0x00000000      0x00000
000      0x00000000
0x00000080 <bogus+     128>:    0x00ff3027      0x00000000      0x00000
000      0x00000000
0x00000090 <bogus+     144>:    0x00000000      0x00000000      0x00000
000      0x00000000
0x000000a0 <bogus+     160>:    0x00000000      0x00000000      0x00000
000      0x00000000
0x000000b0 <bogus+     176>:    0x00000000      0x00000000      0x00000
000      0x00ffb027
0x000000c0 <bogus+     192>:    0x00ff6027      0x00000000      0x00000
000      0x00000000
0x000000d0 <bogus+     208>:    0x00000000      0x00000000      0x00000
000      0x00000000
0x000000e0 <bogus+     224>:    0x00000000      0x00000000      0x00000
000      0x00000000
0x000000f0 <bogus+     240>:    0x00000000      0x00000000      0x00000
000      0x00ffa027
0x00000100 <bogus+     256>:    0x00faa027      0x00000000      0x00000
000      0x00000000
<bochs:13> xp /w 0+64*4
[bochs]:
0x00000100 <bogus+       0>:    0x00faa027
<bochs:14> xp /w 0x00faa000 + 3*4
[bochs]:
0x00faa00c <bogus+       0>:    0x00fa6067
<bochs:15>
```

可知0x00fa6004为其物理地址，输入命令`xp /w 0x00fa6004`



```
0x00000050 <bogus+       80>:    0x00000000      0x00000000      0x00000
000      0x00000000
0x00000060 <bogus+       96>:    0x00000000      0x00000000      0x00000
000      0x00000000
0x00000070 <bogus+      112>:    0x00000000      0x00000000      0x00000
000      0x00000000
0x00000080 <bogus+      128>:    0x00ff3027      0x00000000      0x00000
000      0x00000000
0x00000090 <bogus+      144>:    0x00000000      0x00000000      0x00000
000      0x00000000
0x000000a0 <bogus+      160>:    0x00000000      0x00000000      0x00000
000      0x00000000
0x000000b0 <bogus+      176>:    0x00000000      0x00000000      0x00000
000      0x00ffb027
0x000000c0 <bogus+      192>:    0x00ff6027      0x00000000      0x00000
000      0x00000000
0x000000d0 <bogus+      208>:    0x00000000      0x00000000      0x00000
000      0x00000000
0x000000e0 <bogus+      224>:    0x00000000      0x00000000      0x00000
000      0x00000000
0x000000f0 <bogus+      240>:    0x00000000      0x00000000      0x00000
000      0x00ffa027
0x00000100 <bogus+      256>:    0x00faa027      0x00000000      0x00000
000      0x00000000
<bochs:13> xp /w 0+64*4
[bochs]:
0x00000100 <bogus+        0>:    0x00faa027
<bochs:14> xp /w 0x00faa000 + 3*4
[bochs]:
0x00faa00c <bogus+        0>:    0x00fa6067
<bochs:15> xp /w 0x00fa7004
[bochs]:
0x00fa7004 <bogus+        0>:    0x00000001
<bochs:16> xp /w 0x00fa6004
[bochs]:
0x00fa6004 <bogus+        0>:    0x12345678
<bochs:17> 
```

现在，通过直接修改内存来改变i的值为0，命令是： `setpmem 0x00fa6004 4 0`，表示从0x00fa7004地址开始的4个字节都设为0。然后再用“c”命令继续Bochs的运行，可以看到test退出了，说明i的修改成功了，此项实验结束。



```
3454 buffers = 3536896 bytes buffer space
Free mem: 12582912 bytes
 Ok.
[/usr/root]# gcc test test.c -Walli
gcc-cc1: Invalid option `-Walli'
[/usr/root]# gcc test test.c -Wall
/usr/local/lib/gcc-ld: No such file or directory for test
[/usr/root]# gcc test test.c
/usr/local/lib/gcc-ld: No such file or directory for test
[/usr/root]# ll
total 135
-rw-r--rw-   1 root     root         1252 Mar 29  2004 README
drwx--xrwx   4 root     root           96 Mar 29  2004 gcclib140
-rwx--xrwx   1 root     root        20591 Nov 13  2004 hello
-rw-r--rw-   1 root     root           74 Mar 21  2004 hello.c
-rw----rw-   1 root     root          156 Nov 13  2004 hello.o
drwx--xrwx   2 root     root          176 Jun 25  2006 linux-0.00
-rw----rw-   1 root     root         4387 Jun 25  2006 linux0.tgz
-rw-r--rw-   1 root     root          420 Mar 21  2004 mtools.howto
drwx--xrwx   3 root     root          176 Sep 21  2004 shoe
-rw----rw-   1 root     root       101767 Sep 21  2004 shoelace.tar.Z
-rw-rw-r--   1 1000     232           183 Feb  3  2018 test.c
[/usr/root]# gcc -o test test.c
[/usr/root]# ./test
The logical/virtual address of i is 0x00003004[/usr/root]# _
```

```
<bochs:15> xp /w 0x00fa7004
[bochs]:
0x00fa7004 <bogus+        0>:    0x00000001
<bochs:16> xp /w 0x00fa6004
[bochs]:
0x00fa6004 <bogus+        0>:    0x12345678
<bochs:17> setpmem 0x00fa6004 4 0
<bochs:18> c
```

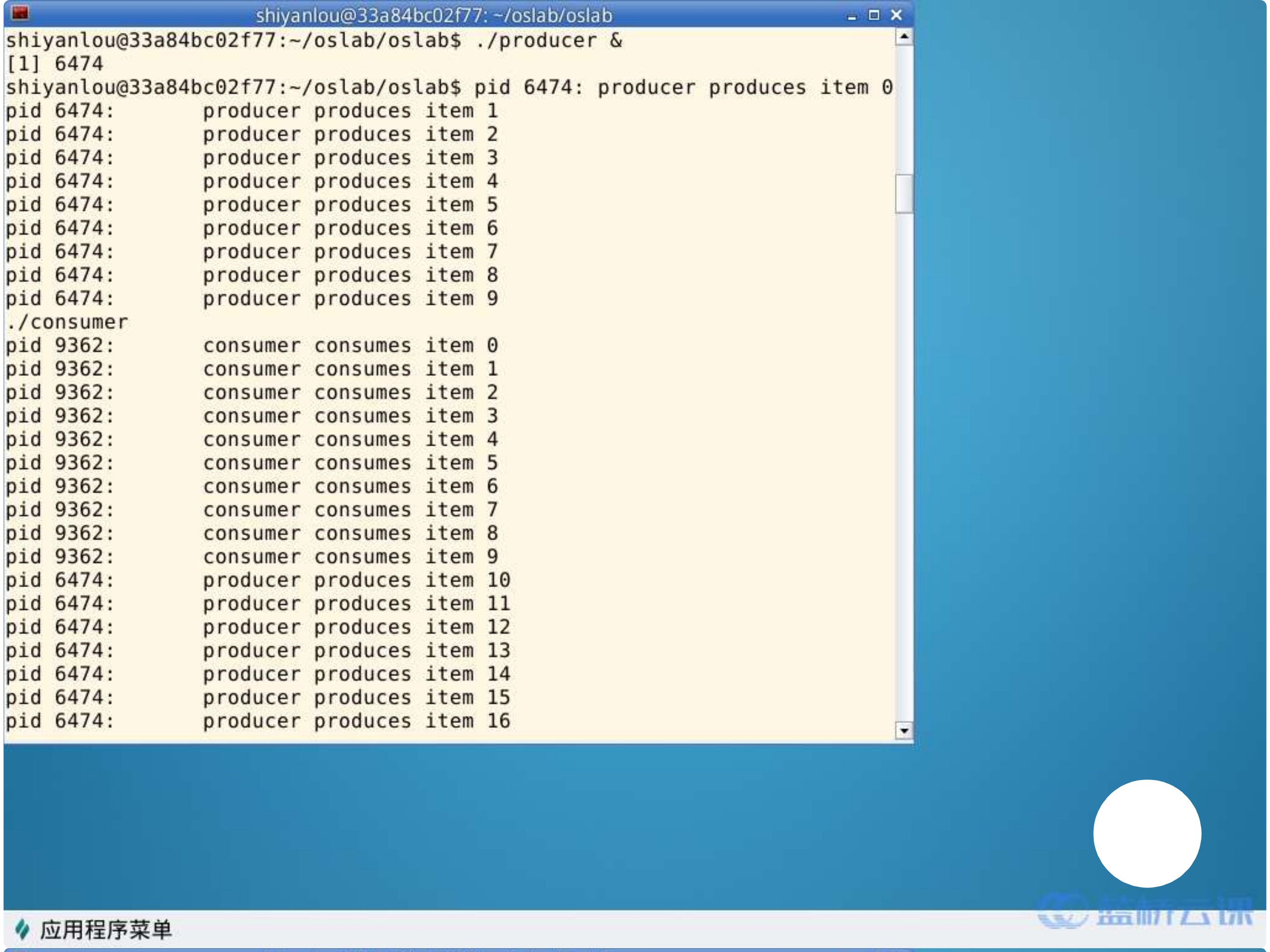
##在Ubuntu下实现基于共享内存的生产者—消费者程序 编写producer.c和consumer.c

```c
/*producer.c*/
#include <stdio.h>
#include <unistd.h>
#include <semaphore.h>
#include <sys/wait.h>
#include <sys/shm.h>
#include <sys/ipc.h>
#include <fcntl.h>

#define PRODUCE_NUM 200
#define BUFFER_SIZE 10
#define SHM_KEY 2018

int main(int argc, char* argv[])
{
    sem_t *Empty,*Full,*Mutex;
    int i, shm_id, location=0;
    int *p;

    Empty = sem_open("Empty",O_CREAT,0600,BUFFER_SIZE);
    Full = sem_open("Full", O_CREAT,0600, 0);
    Mutex = sem_open("Mutex",O_CREAT,0600, 1);

    if((shm_id = shmget(SHM_KEY, BUFFER_SIZE*sizeof(int), IPC_CREAT|0666)) == -1)
        printf("shmget failed!");

    if((p = (int * )shmat(shm_id, NULL, 0)) == -1)
        printf("shmat error!");
    for(i=0; i<PRODUCE_NUM; i++)
    {
        sem_wait(Empty);
        sem_wait(Mutex);

        p[location] = i;
        printf("pid %d:\tproducer produces item %d\n", getpid(), p[location]);
        fflush(stdout);

        sem_post(Mutex);
        sem_post(Full);
        location  = (location+1) % BUFFER_SIZE;
    }
    if(shmdt(p) == -1)
        printf("pdc shmdt error");

    return 0;
}
```

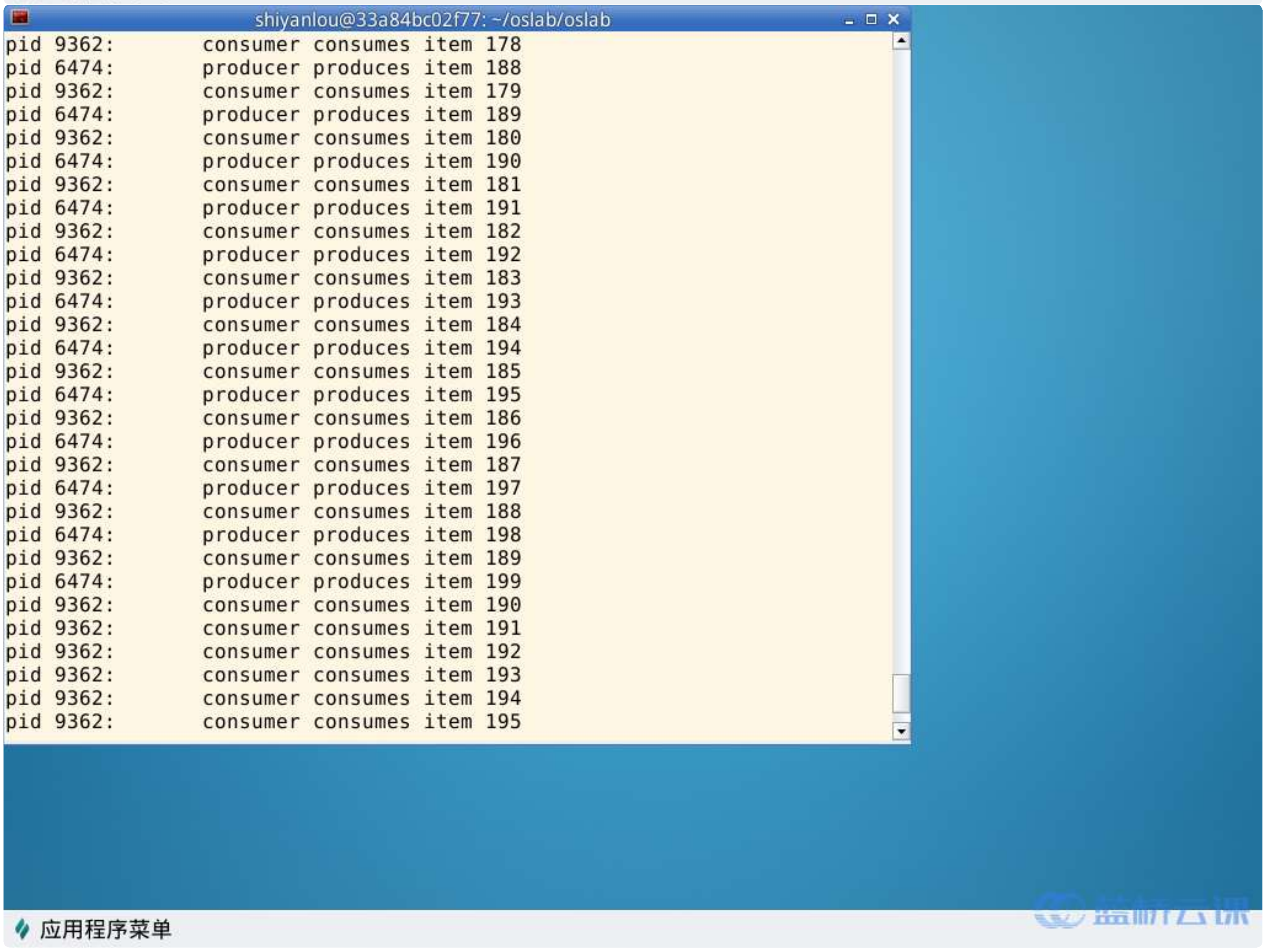
```c
/*consumer.c*/
#include <stdio.h>
#include <unistd.h>
#include <semaphore.h>
#include <sys/wait.h>
#include <sys/shm.h>
#include <sys/ipc.h>
#include <fcntl.h>

#define PRODUCE_NUM 200
#define BUFFER_SIZE 10
#define SHM_KEY 2018

int main(int argc, char* argv[])
{
    sem_t *Empty,*Full,*Mutex;
    int used, shm_id,location = 0;
    int *p;

    Empty = sem_open("Empty", O_CREAT,0600,BUFFER_SIZE);
    Full = sem_open("Full", O_CREAT,0600, 0);
    Mutex = sem_open("Mutex", O_CREAT,0600, 1);

    if((shm_id = shmget(SHM_KEY, BUFFER_SIZE*sizeof(int), IPC_CREAT|0666)) == -1)
        printf("shmget failed!\n");

    if((p = (int * )shmat(shm_id, NULL, 0)) == -1)
        printf("link error!\n");
    while(1)
    {
        sem_wait(Full);
        sem_wait(Mutex);

        printf("pid %d:\tconsumer consumes item %d\n", getpid(), p[location]);
        fflush(stdout);

        sem_post(Mutex);
        sem_post(Empty);
        location  = (location+1) % BUFFER_SIZE;

        if(++used == PRODUCE_NUM)
            break;
    }
    if(shmdt(p) == -1)
        printf("csm shmdt error.\n");
    if(shmctl(shm_id,IPC_RMID,NULL) == -1)
        printf("shmctl error.\n");

    sem_unlink("Mutex");
    sem_unlink("Full");
    sem_unlink("Empty");
    return 0;
}
```

输入命令`gcc -o producer producer.c -lpthread`和`gcc -o consumer consumer.c -lpthread`编译，输入下列指令运行即得到输出

```
# ./producer &
# ./consumer
```

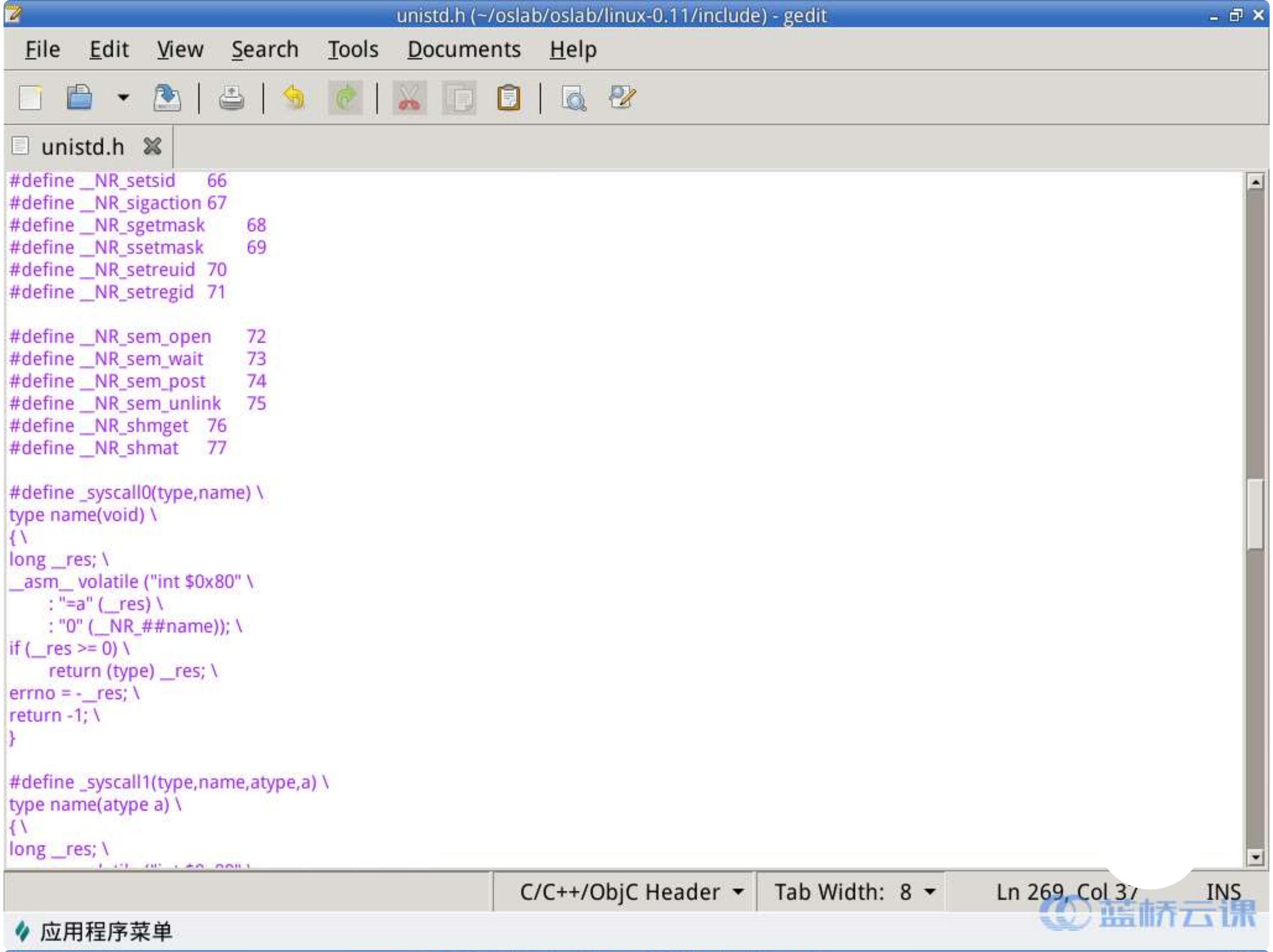


```
shiyanlou@33a84bc02f77:~/oslab/oslab$ ./producer &
[1] 6474
shiyanlou@33a84bc02f77:~/oslab/oslab$ pid 6474: producer produces item 0
pid 6474:       producer produces item 1
pid 6474:       producer produces item 2
pid 6474:       producer produces item 3
pid 6474:       producer produces item 4
pid 6474:       producer produces item 5
pid 6474:       producer produces item 6
pid 6474:       producer produces item 7
pid 6474:       producer produces item 8
pid 6474:       producer produces item 9
./consumer
pid 9362:       consumer consumes item 0
pid 9362:       consumer consumes item 1
pid 9362:       consumer consumes item 2
pid 9362:       consumer consumes item 3
pid 9362:       consumer consumes item 4
pid 9362:       consumer consumes item 5
pid 9362:       consumer consumes item 6
pid 9362:       consumer consumes item 7
pid 9362:       consumer consumes item 8
pid 9362:       consumer consumes item 9
pid 6474:       producer produces item 10
pid 6474:       producer produces item 11
pid 6474:       producer produces item 12
pid 6474:       producer produces item 13
pid 6474:       producer produces item 14
pid 6474:       producer produces item 15
pid 6474:       producer produces item 16
```

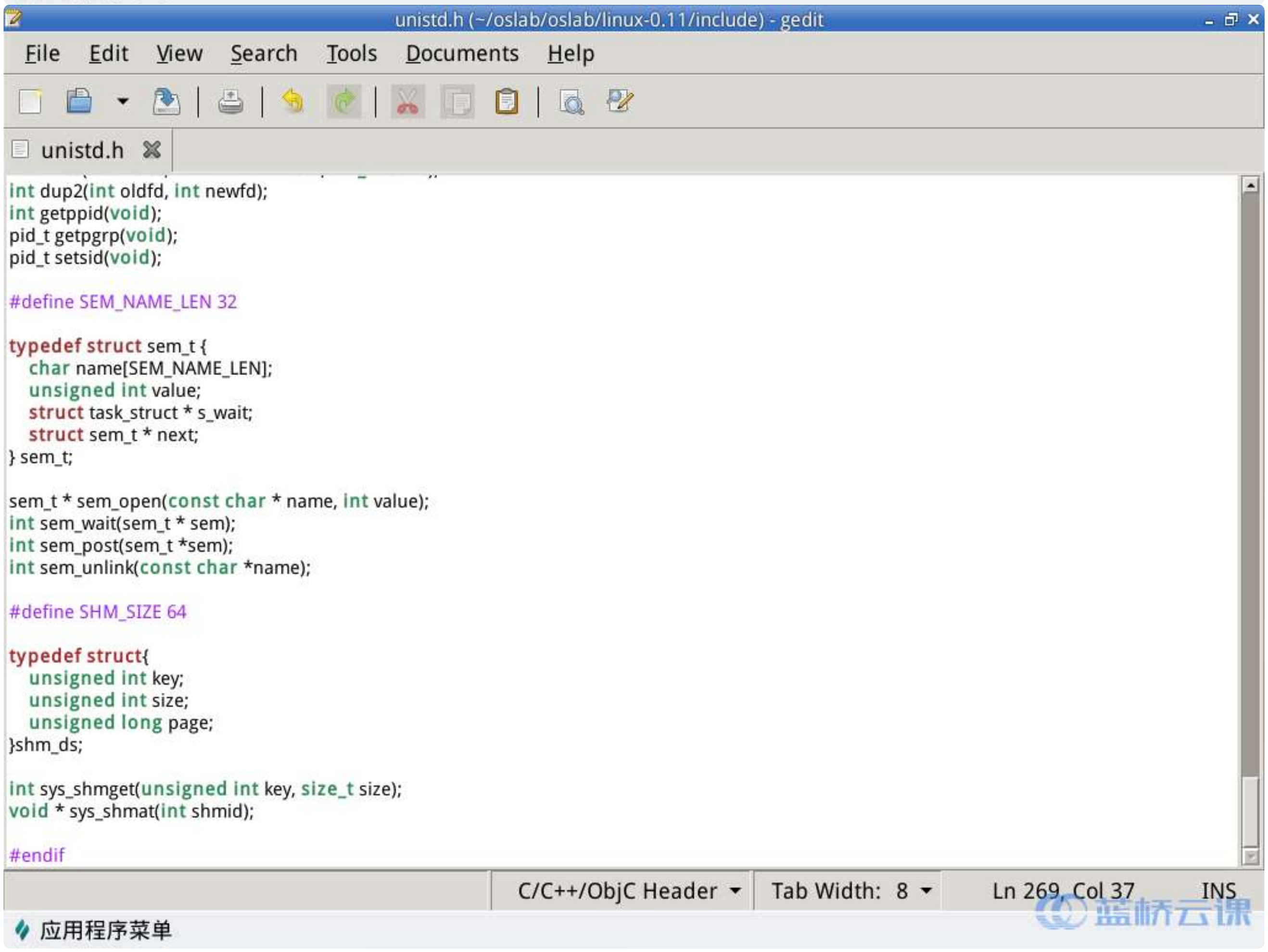


```
pid 9362:       consumer consumes item 178
pid 6474:       producer produces item 188
pid 9362:       consumer consumes item 179
pid 6474:       producer produces item 189
pid 9362:       consumer consumes item 180
pid 6474:       producer produces item 190
pid 9362:       consumer consumes item 181
pid 6474:       producer produces item 191
pid 9362:       consumer consumes item 182
pid 6474:       producer produces item 192
pid 9362:       consumer consumes item 183
pid 6474:       producer produces item 193
pid 9362:       consumer consumes item 184
pid 6474:       producer produces item 194
pid 9362:       consumer consumes item 185
pid 6474:       producer produces item 195
pid 9362:       consumer consumes item 186
pid 6474:       producer produces item 196
pid 9362:       consumer consumes item 187
pid 6474:       producer produces item 197
pid 9362:       consumer consumes item 188
pid 6474:       producer produces item 198
pid 9362:       consumer consumes item 189
pid 6474:       producer produces item 199
pid 9362:       consumer consumes item 190
pid 9362:       consumer consumes item 191
pid 9362:       consumer consumes item 192
pid 9362:       consumer consumes item 193
pid 9362:       consumer consumes item 194
pid 9362:       consumer consumes item 195
```

运行正常 ##在linux 0.11实现基于共享内存的生产者消费者 再一次进行信号量实验的套路 ###unistd.h

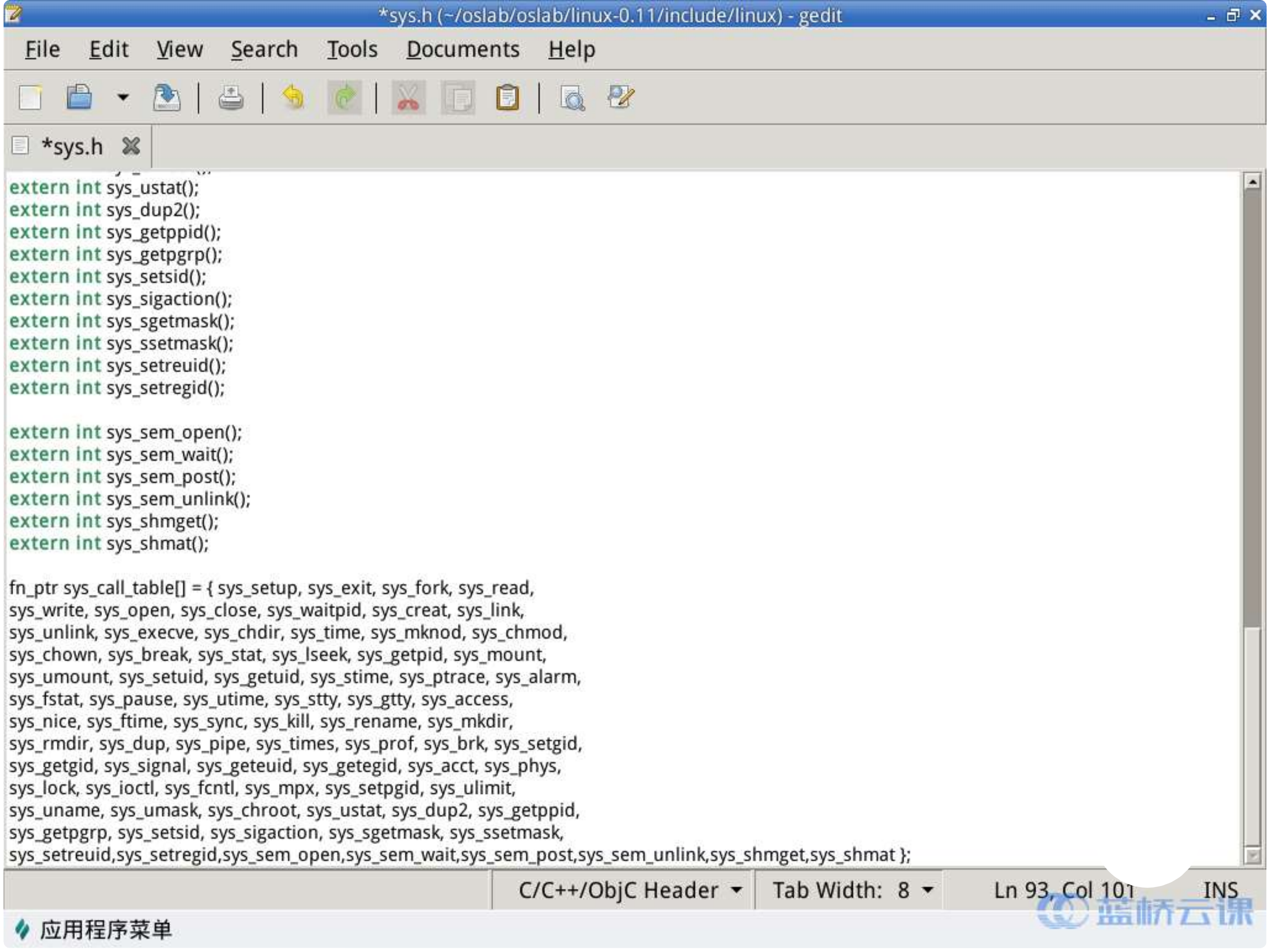

unistd.h (~/oslab/oslab/linux-0.11/include) - gedit
File Edit View Search Tools Documents Help
unistd.h
C/C++/ObjC Header
Tab Width: 8
Ln 269, Col 37
INS
应用程序菜单
蓝桥云课

```
#define __NR_setsid     66
#define __NR_sigaction 67
#define __NR_sgetmask       68
#define __NR_ssetmask       69
#define __NR_setreuid  70
#define __NR_setregid  71

#define __NR_sem_open       72
#define __NR_sem_wait       73
#define __NR_sem_post       74
#define __NR_sem_unlink     75
#define __NR_shmget   76
#define __NR_shmat      77

#define _syscall0(type,name) \
type name(void) \
{ \
long __res; \
__asm__ volatile ("int $0x80" \
        : "=a" (__res) \
        : "0" (__NR_##name)); \
if (__res >= 0) \
        return (type) __res; \
errno = -__res; \
return -1; \
}

#define _syscall1(type,name,atype,a) \
type name(atype a) \
{ \
long __res; \
```


unistd.h (~/oslab/oslab/linux-0.11/include) - gedit
File Edit View Search Tools Documents Help
unistd.h
C/C++/ObjC Header
Tab Width: 8
Ln 269, Col 37
INS
应用程序菜单
蓝桥云课

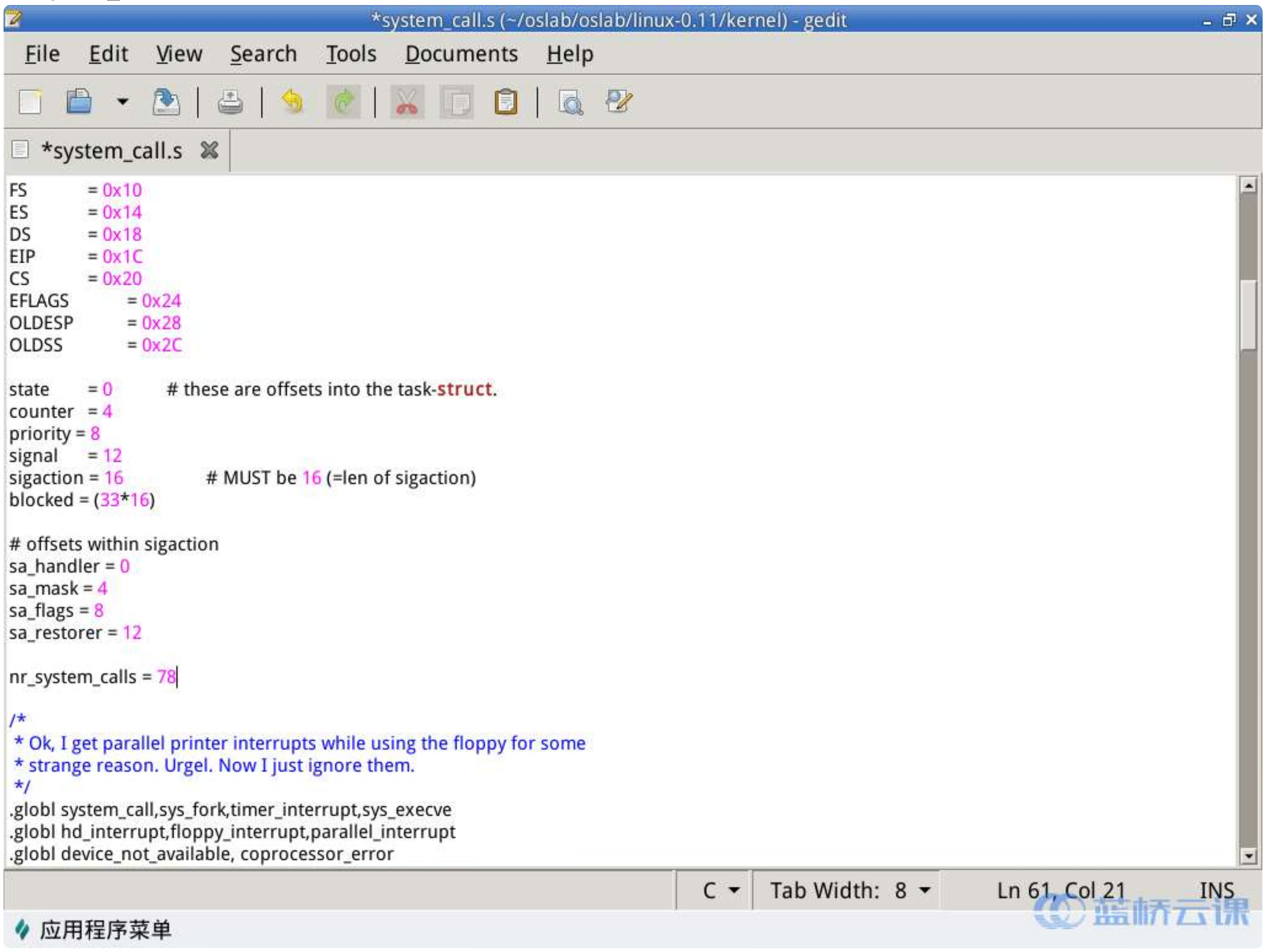
```
int dup2(int oldfd, int newfd);
int getppid(void);
pid_t getpgrp(void);
pid_t setsid(void);

#define SEM_NAME_LEN 32

typedef struct sem_t {
   char name[SEM_NAME_LEN];
   unsigned int value;
   struct task_struct * s_wait;
   struct sem_t * next;
} sem_t;

sem_t * sem_open(const char * name, int value);
int sem_wait(sem_t * sem);
int sem_post(sem_t *sem);
int sem_unlink(const char *name);

#define SHM_SIZE 64

typedef struct{
   unsigned int key;
   unsigned int size;
   unsigned long page;
}shm_ds;

int sys_shmget(unsigned int key, size_t size);
void * sys_shmat(int shmid);

#endif
```

sys.h


*sys.h (~/oslab/oslab/linux-0.11/include/linux) - gedit
File Edit View Search Tools Documents Help
*sys.h
C/C++/ObjC Header
Tab Width: 8
Ln 93, Col 101
INS
应用程序菜单
蓝桥云课

```
extern int sys_ustat();
extern int sys_dup2();
extern int sys_getppid();
extern int sys_getpgrp();
extern int sys_setsid();
extern int sys_sigaction();
extern int sys_sgetmask();
extern int sys_ssetmask();
extern int sys_setreuid();
extern int sys_setregid();

extern int sys_sem_open();
extern int sys_sem_wait();
extern int sys_sem_post();
extern int sys_sem_unlink();
extern int sys_shmget();
extern int sys_shmat();

fn_ptr sys_call_table[] = { sys_setup, sys_exit, sys_fork, sys_read,
sys_write, sys_open, sys_close, sys_waitpid, sys_creat, sys_link,
sys_unlink, sys_execve, sys_chdir, sys_time, sys_mknod, sys_chmod,
sys_chown, sys_break, sys_stat, sys_lseek, sys_getpid, sys_mount,
sys_umount, sys_setuid, sys_getuid, sys_stime, sys_ptrace, sys_alarm,
sys_fstat, sys_pause, sys_utime, sys_stty, sys_gtty, sys_access,
sys_nice, sys_ftime, sys_sync, sys_kill, sys_rename, sys_mkdir,
sys_rmdir, sys_dup, sys_pipe, sys_times, sys_prof, sys_brk, sys_setgid,
sys_getgid, sys_signal, sys_geteuid, sys_getegid, sys_acct, sys_phys,
sys_lock, sys_ioctl, sys_fcntl, sys_mpx, sys_setpgid, sys_ulimit,
sys_uname, sys_umask, sys_chroot, sys_ustat, sys_dup2, sys_getppid,
sys_getpgrp, sys_setsid, sys_sigaction, sys_sgetmask, sys_ssetmask,
sys_setreuid,sys_setregid,sys_sem_open,sys_sem_wait,sys_sem_post,sys_sem_unlink,sys_shmget,sys_shmat };
```

###system_call.s

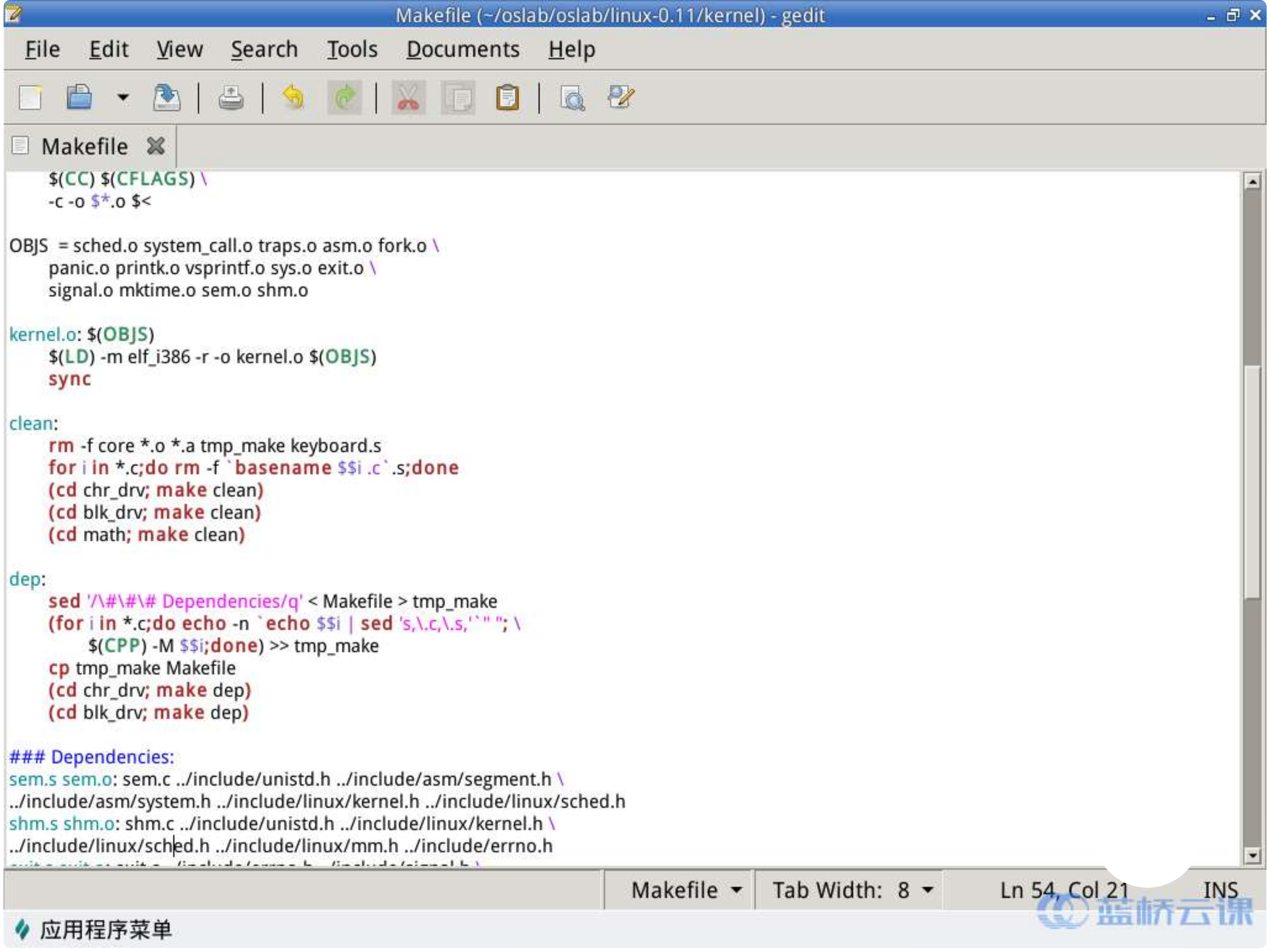

*system_call.s (~/oslab/oslab/linux-0.11/kernel) - gedit
File Edit View Search Tools Documents Help
*system_call.s
C
Tab Width: 8
Ln 61, Col 21
INS
应用程序菜单
蓝桥云课

```
FS        = 0x10
ES        = 0x14
DS        = 0x18
EIP       = 0x1C
CS        = 0x20
EFLAGS         = 0x24
OLDESP         = 0x28
OLDSS          = 0x2C

state     = 0       # these are offsets into the task-struct.
counter   = 4
priority = 8
signal    = 12
sigaction = 16           # MUST be 16 (=len of sigaction)
blocked = (33*16)

# offsets within sigaction
sa_handler = 0
sa_mask = 4
sa_flags = 8
sa_restorer = 12

nr_system_calls = 78

/*
 * Ok, I get parallel printer interrupts while using the floppy for some
 * strange reason. Urgel. Now I just ignore them.
 */
.globl system_call,sys_fork,timer_interrupt,sys_execve
.globl hd_interrupt,floppy_interrupt,parallel_interrupt
.globl device_not_available, coprocessor_error
```

###sem.c 略 ###shm.c

```c
#define __LIBRARY__
#include <unistd.h>
#include <linux/kernel.h>
#include <linux/sched.h>
#include <linux/mm.h>
#include <errno.h>

static shm_ds shm_list[SHM_SIZE] = {{0,0,0}};

int sys_shmget(unsigned int key, size_t size)
{
    int i;
    void *page;
    if(size > PAGE_SIZE)
        return -EINVAL;
    page = get_free_page();
    if(!page)
        return -ENOMEM;
    printk("shmget get memory's address is 0x%08x\n",page);
    for(i=0; i<SHM_SIZE; i++)
    {
        if(shm_list[i].key == key)
            return i;
    }
    for(i=0; i<SHM_SIZE; i++)
    {
        if(shm_list[i].key == 0)
        {
               shm_list[i].page = page;
            shm_list[i].key = key;
            shm_list[i].size = size;
            return i;
        }
    }
    return -1;
}

void * sys_shmat(int shmid)
{
    int i;
    unsigned long data_base, brk;

    if(shmid < 0 || SHM_SIZE <= shmid || shm_list[shmid].page==0 || shm_list[shmid].key <= 0)
        return (void *)-EINVAL;

    data_base = get_base(current->ldt[2]);
    printk("current's data_base = 0x%08x,new page = 0x%08x\n",data_base,shm_list[shmid].page);

    brk = current->brk + data_base;
    current->brk += PAGE_SIZE;

    if(put_page(shm_list[shmid].page, brk) == 0)
        return (void *)-ENOMEM;

    return (void *)(current->brk - PAGE_SIZE);
}
```

Makefile


Makefile (~/oslab/oslab/linux-0.11/kernel) - gedit
File Edit View Search Tools Documents Help
Makefile
Makefile
Tab Width: 8
Ln 54, Col 21
INS
应用程序菜单

```
	$(CC) $(CFLAGS) \
	-c -o $*.o $<

OBJS  = sched.o system_call.o traps.o asm.o fork.o \
	panic.o printk.o vsprintf.o sys.o exit.o \
	signal.o mktime.o sem.o shm.o

kernel.o: $(OBJS)
	$(LD) -m elf_i386 -r -o kernel.o $(OBJS)
	sync

clean:
	rm -f core *.o *.a tmp_make keyboard.s
	for i in *.c;do rm -f `basename $$i .c`.s;done
	(cd chr_drv; make clean)
	(cd blk_drv; make clean)
	(cd math; make clean)

dep:
	sed '/\#\#\# Dependencies/q' < Makefile > tmp_make
	(for i in *.c;do echo -n `echo $$i | sed 's,\.c,\.s,'`" "; \
		$(CPP) -M $$i;done) >> tmp_make
	cp tmp_make Makefile
	(cd chr_drv; make dep)
	(cd blk_drv; make dep)

### Dependencies:
sem.s sem.o: sem.c ../include/unistd.h ../include/asm/segment.h \
../include/asm/system.h ../include/linux/kernel.h ../include/linux/sched.h
shm.s shm.o: shm.c ../include/unistd.h ../include/linux/kernel.h \
../include/linux/sched.h ../include/linux/mm.h ../include/errno.h
```

consumer.c

```
/*consumer*/
#define __LIBRARY__
#include <stdio.h>
#include <unistd.h>
#include <linux/kernel.h>
#include <fcntl.h>
#include <sys/types.h>

_syscall2(sem_t *,sem_open,const char *,name,int,value);
_syscall1(int,sem_post,sem_t *,sem);
_syscall1(int,sem_wait,sem_t *,sem);
_syscall1(int,sem_unlink,const char*,name);

_syscall1(int, shmat, int, shmid);
_syscall2(int, shmget, unsigned int, key, size_t, size);

#define PRODUCE_NUM 200
#define BUFFER_SIZE 10
#define SHM_KEY 2018

int main(int argc, char* argv[])
{
    sem_t *Empty,*Full,*Mutex;
    int used = 0, shm_id,location = 0;
    int *p;

    Empty = sem_open("Empty", BUFFER_SIZE);
    Full = sem_open("Full", 0);
    Mutex = sem_open("Mutex", 1);

    if((shm_id = shmget(SHM_KEY, BUFFER_SIZE*sizeof(int))) < 0)
        printf("shmget failed!\n");

    if((p = (int * )shmat(shm_id)) < 0)
        printf("link error!\n");
    while(1)
    {
        sem_wait(Full);
        sem_wait(Mutex);

        printf("pid %d:\tconsumer consumes item %d\n", getpid(), p[location]);
        fflush(stdout);

        sem_post(Mutex);
        sem_post(Empty);
        location  = (location+1) % BUFFER_SIZE;

        if(++used == PRODUCE_NUM)
            break;
    }

    sem_unlink("Mutex");
    sem_unlink("Full");
    sem_unlink("Empty");
    return 0;
}
```

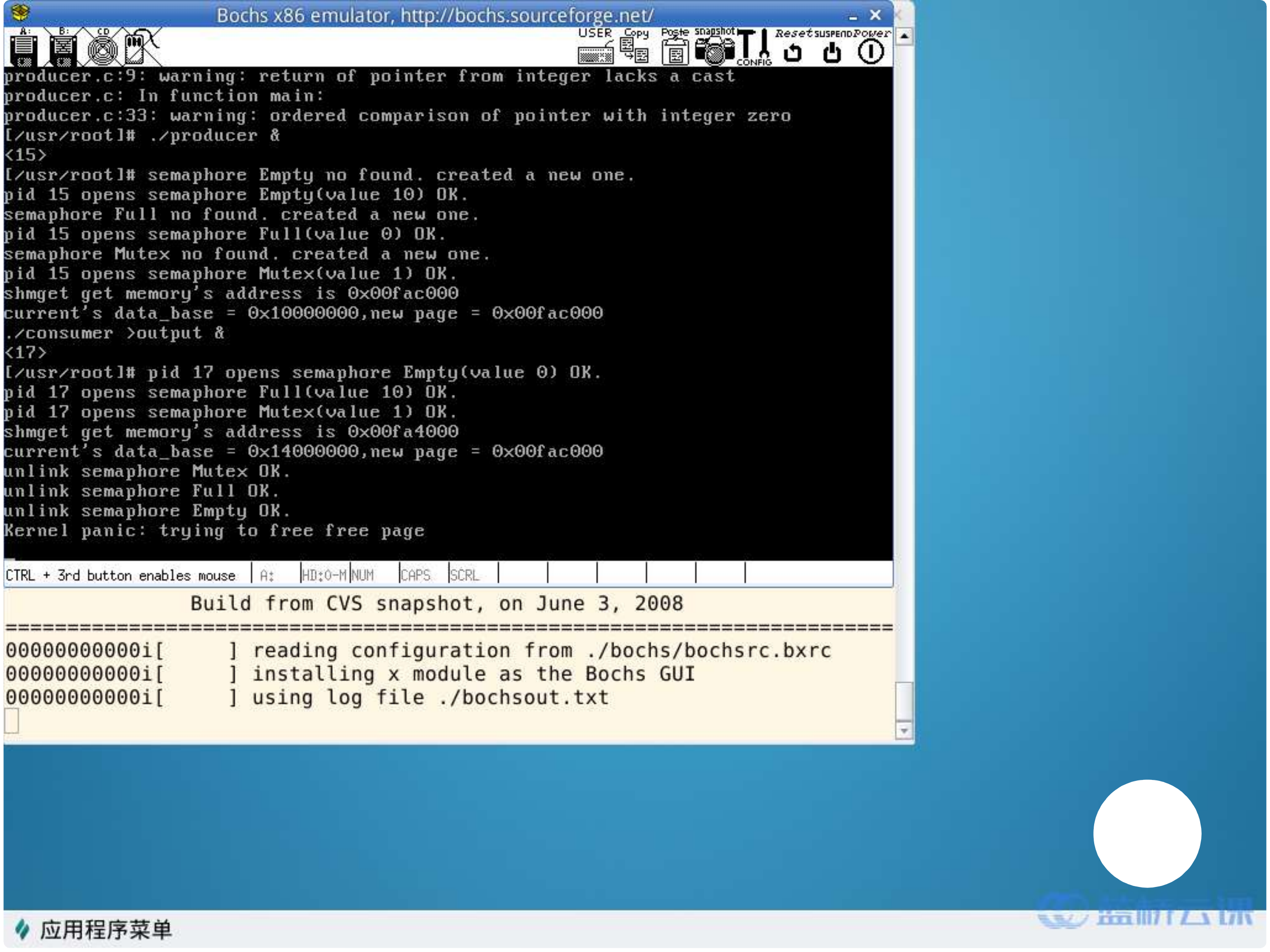
producer.c

因输出问题，此处把producer的输出删去

```
/*producer*/
#define __LIBRARY__
#include <stdio.h>
#include <unistd.h>
#include <linux/kernel.h>
#include <fcntl.h>
#include <sys/types.h>

_syscall2(sem_t *,sem_open,const char *,name,int,value);
_syscall1(int,sem_post,sem_t *,sem);
_syscall1(int,sem_wait,sem_t *,sem);

_syscall1(int, shmat, int, shmid);
_syscall2(int, shmget, unsigned int, key, size_t, size);

#define PRODUCE_NUM 200
#define BUFFER_SIZE 10
#define SHM_KEY 2018

int main(int argc, char* argv[])
{
    sem_t *Empty,*Full,*Mutex;
    int i, shm_id, location=0;
    int *p;

    Empty = sem_open("Empty", BUFFER_SIZE);
    Full = sem_open("Full", 0);
    Mutex = sem_open("Mutex", 1);

    if((shm_id = shmget(SHM_KEY, BUFFER_SIZE*sizeof(int))) < 0)
        printf("shmget failed!");

    if((p = (int * )shmat(shm_id)) < 0)
        printf("shmat error!");
    for(i=0; i<PRODUCE_NUM; i++)
    {
        sem_wait(Empty);
        sem_wait(Mutex);

        p[location] = i;

        sem_post(Mutex);
        sem_post(Full);
        location  = (location+1) % BUFFER_SIZE;
    }
    return 0;
}
```

编译，一波复制，运行系统

```
./producer &
./consumer > output &
sync
```

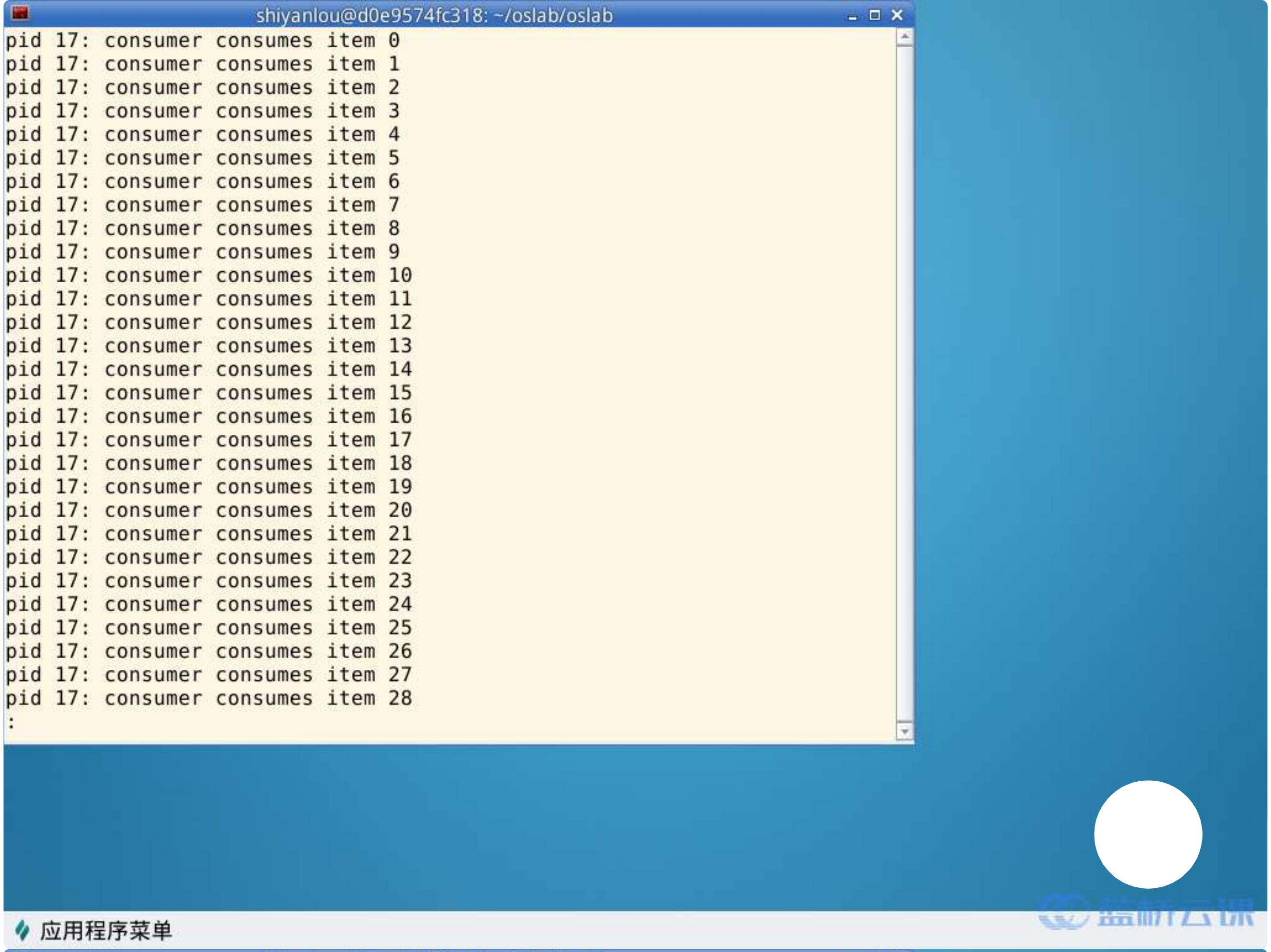


```
producer.c:9: warning: return of pointer from integer lacks a cast
producer.c: In function main:
producer.c:33: warning: ordered comparison of pointer with integer zero
[/usr/root]# ./producer &
<15>
[/usr/root]# semaphore Empty no found. created a new one.
pid 15 opens semaphore Empty(value 10) OK.
semaphore Full no found. created a new one.
pid 15 opens semaphore Full(value 0) OK.
semaphore Mutex no found. created a new one.
pid 15 opens semaphore Mutex(value 1) OK.
shmget get memory's address is 0x00fac000
current's data_base = 0x10000000,new page = 0x00fac000
./consumer >output &
<17>
[/usr/root]# pid 17 opens semaphore Empty(value 0) OK.
pid 17 opens semaphore Full(value 10) OK.
pid 17 opens semaphore Mutex(value 1) OK.
shmget get memory's address is 0x00fa4000
current's data_base = 0x14000000,new page = 0x00fac000
unlink semaphore Mutex OK.
unlink semaphore Full OK.
unlink semaphore Empty OK.
Kernel panic: trying to free free page
```

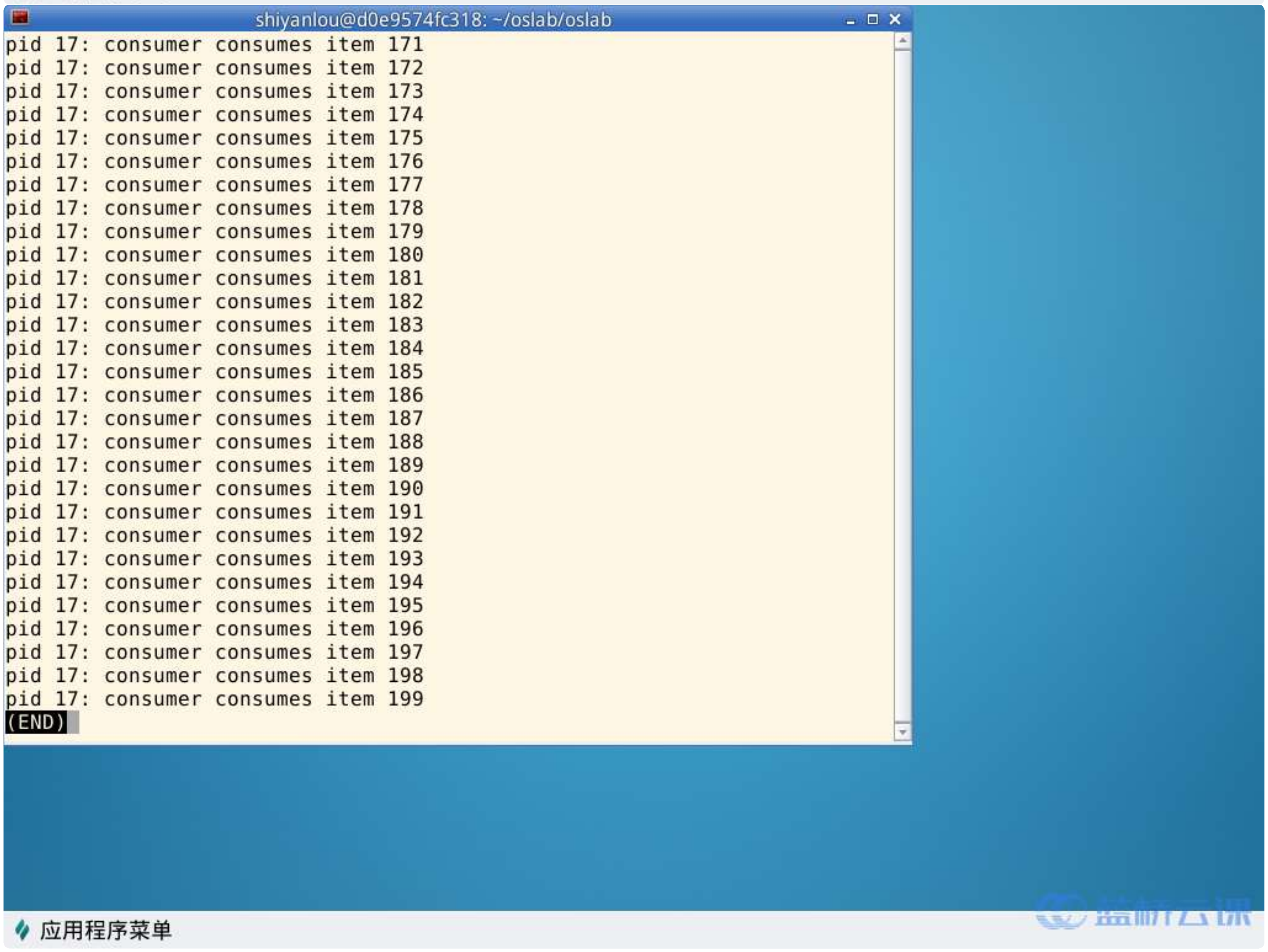
```
                    Build from CVS snapshot, on June 3, 2008
========================================================================
00000000000i[     ] reading configuration from ./bochs/bochsrc.bxrc
00000000000i[     ] installing x module as the Bochs GUI
00000000000i[     ] using log file ./bochsout.txt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

Bochs x86 emulator, http://bochs.sourceforge.net/
CTRL + 3rd button enables mouse
应用程序菜单
蓝桥云课

输入`sudo less hdc/usr/root/output`查看



```
pid 17: consumer consumes item 0
pid 17: consumer consumes item 1
pid 17: consumer consumes item 2
pid 17: consumer consumes item 3
pid 17: consumer consumes item 4
pid 17: consumer consumes item 5
pid 17: consumer consumes item 6
pid 17: consumer consumes item 7
pid 17: consumer consumes item 8
pid 17: consumer consumes item 9
pid 17: consumer consumes item 10
pid 17: consumer consumes item 11
pid 17: consumer consumes item 12
pid 17: consumer consumes item 13
pid 17: consumer consumes item 14
pid 17: consumer consumes item 15
pid 17: consumer consumes item 16
pid 17: consumer consumes item 17
pid 17: consumer consumes item 18
pid 17: consumer consumes item 19
pid 17: consumer consumes item 20
pid 17: consumer consumes item 21
pid 17: consumer consumes item 22
pid 17: consumer consumes item 23
pid 17: consumer consumes item 24
pid 17: consumer consumes item 25
pid 17: consumer consumes item 26
pid 17: consumer consumes item 27
pid 17: consumer consumes item 28
:
```



```
pid 17: consumer consumes item 171
pid 17: consumer consumes item 172
pid 17: consumer consumes item 173
pid 17: consumer consumes item 174
pid 17: consumer consumes item 175
pid 17: consumer consumes item 176
pid 17: consumer consumes item 177
pid 17: consumer consumes item 178
pid 17: consumer consumes item 179
pid 17: consumer consumes item 180
pid 17: consumer consumes item 181
pid 17: consumer consumes item 182
pid 17: consumer consumes item 183
pid 17: consumer consumes item 184
pid 17: consumer consumes item 185
pid 17: consumer consumes item 186
pid 17: consumer consumes item 187
pid 17: consumer consumes item 188
pid 17: consumer consumes item 189
pid 17: consumer consumes item 190
pid 17: consumer consumes item 191
pid 17: consumer consumes item 192
pid 17: consumer consumes item 193
pid 17: consumer consumes item 194
pid 17: consumer consumes item 195
pid 17: consumer consumes item 196
pid 17: consumer consumes item 197
pid 17: consumer consumes item 198
pid 17: consumer consumes item 199
(END)
```

正常输出，实验结束

实验问题

1. 对于地址映射实验部分，列出你认为最重要的那几步（不超过4步），并给出你获得的实验数据。

略 ####2. test.c退出后，如果马上再运行一次，并再进行地址跟踪，你发现有哪些异同？为什么？ 得到的i的物理地址可能会不同。在linux0.11中，因为有虚拟内存和段页结合的内存管理机制，get_free_page()在物理页框中找出空闲页是很随意的———只要是空闲的页框就直接拿来用。而test.exe重启后，其各个段在上一次执行时使用的物理页框很可能已经被其他进程占用了。所以这一次其data段有可能会被分配到别的物理页框中去。所以得到的i的物理地址可能会不同。

实验数据

学习时间	655分钟
操作时间	270分钟
按键次数	4413次
实验次数	4次
报告字数	9420字
是否完成	完成

评分

未评分

下一篇

上一篇

相关报告

操作系统原理与实践: 熟悉实验环境 实验报告

操作系统原理与实践: 熟悉实验环境 实验报告

操作系统原理与实践: 基于内核栈切换的进程切换 实验报告

操作系统原理与实践: 熟悉实验环境 实验报告

操作系统原理与实践: 信号量的实现和应用 实验报告