



实验数据

学习时间 147分钟

操作时间 36分钟

按键次数 686次

实验次数 3次

报告字数 6643字

是否完成 完成

评分

未评分

下一篇

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

“操作系统原理与实践”实验报告

地址映射与共享

暂停在cmp:

```
Bochs x86 emulator, http://bochs.sourceforge.net/
Please visit :
  http://bochs.sourceforge.net/
  http://bochs.sourceforge.net/
Bochs VBE : shiyanlou@a07f0311b3d5: ~/oslab/oslab
Bochs BIOS :
$Revision: 2.3.7
Options: ap
ata0 master:
Booting from:
Loading sys:
Partition 1:
39042/62000
19518/20666
3454 buffer
Free mem: 1
0K.
(Lusr/root)
The logical:
CTRL + 3rd button:
Bochs is exiting with the following message:
[GUI] POWER button turned off.
shiyanlou@a07f0311b3d5:~/oslab/oslab$ ./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
e00f0
<bochs:1> c
^NNext at t=75939769
(0) [0x00f98063] 000f:00000063 (unk. ctxt): cmp dword ptr ds:0x3004, 0x0
0000000 ; 833d0430000000
<bochs:2> a
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
e00f0
<bochs:1> c
^NNext at t=75939769
(0) [0x00f98063] 000f:00000063 (unk. ctxt): cmp dword ptr ds:0x3004, 0x0
0000000 ; 833d0430000000
<bochs:2> u /7
10000063: ( ) : cmp dword ptr ds:0x3004, 0x00000000 ;
833d0430000000
1000006a: ( ) : jz .+0x00000004 ; 7404
1000006c: ( ) : jmp .+0xfffffff5 ; eb5
1000006e: ( ) : add byte ptr ds:[eax], al ; 0000
10000070: ( ) : xor eax, eax ; 31c0
10000072: ( ) : jmp .+0x00000000 ; eb00
10000074: ( ) : leave ; c9
<bochs:3> u /8
10000063: ( ) : cmp dword ptr ds:0x3004, 0x00000000 ;
833d0430000000
1000006a: ( ) : jz .+0x00000004 ; 7404
1000006c: ( ) : jmp .+0xfffffff5 ; eb5
1000006e: ( ) : add byte ptr ds:[eax], al ; 0000
10000070: ( ) : xor eax, eax ; 31c0
10000072: ( ) : jmp .+0x00000000 ; eb00
10000074: ( ) : leave ; c9
10000075: ( ) : ret ; c3
<bochs:4> aocun
```

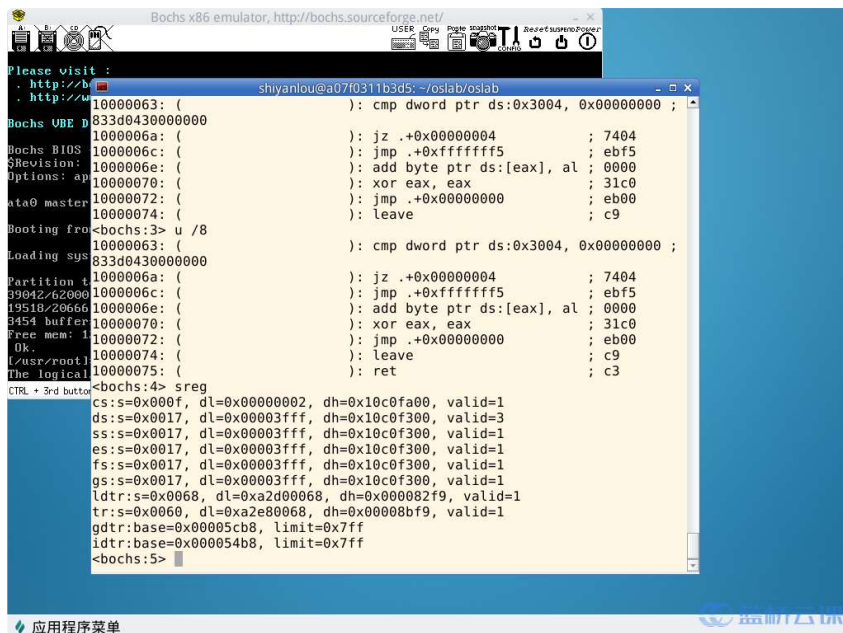
应用程序菜单

保存在0x3004

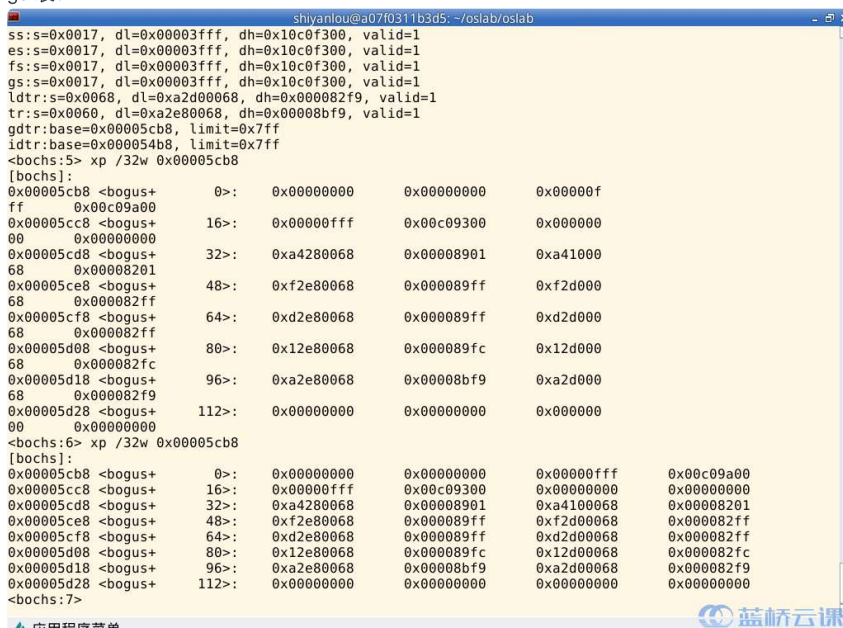
```
Bochs x86 emulator, http://bochs.sourceforge.net/
Please visit :
  http://bochs.sourceforge.net/
  http://bochs.sourceforge.net/
Bochs VBE : shiyanlou@a07f0311b3d5: ~/oslab/oslab
Bochs BIOS :
$Revision: 2.3.7
Options: ap
ata0 master:
Booting from:
Loading sys:
Partition 1:
39042/62000
19518/20666
3454 buffer
Free mem: 1
0K.
(Lusr/root)
The logical:
CTRL + 3rd button:
Bochs is exiting with the following message:
[GUI] POWER button turned off.
shiyanlou@a07f0311b3d5:~/oslab/oslab$ ./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
e00f0
<bochs:1> c
^NNext at t=75939769
(0) [0x00f98063] 000f:00000063 (unk. ctxt): cmp dword ptr ds:0x3004, 0x0
0000000 ; 833d0430000000
<bochs:2> u /7
10000063: ( ) : cmp dword ptr ds:0x3004, 0x00000000 ;
833d0430000000
1000006a: ( ) : jz .+0x00000004 ; 7404
1000006c: ( ) : jmp .+0xfffffff5 ; eb5
1000006e: ( ) : add byte ptr ds:[eax], al ; 0000
10000070: ( ) : xor eax, eax ; 31c0
10000072: ( ) : jmp .+0x00000000 ; eb00
10000074: ( ) : leave ; c9
<bochs:3> u /8
10000063: ( ) : cmp dword ptr ds:0x3004, 0x00000000 ;
833d0430000000
1000006a: ( ) : jz .+0x00000004 ; 7404
1000006c: ( ) : jmp .+0xfffffff5 ; eb5
1000006e: ( ) : add byte ptr ds:[eax], al ; 0000
10000070: ( ) : xor eax, eax ; 31c0
10000072: ( ) : jmp .+0x00000000 ; eb00
10000074: ( ) : leave ; c9
10000075: ( ) : ret ; c3
<bochs:4> aocun
```

应用程序菜单

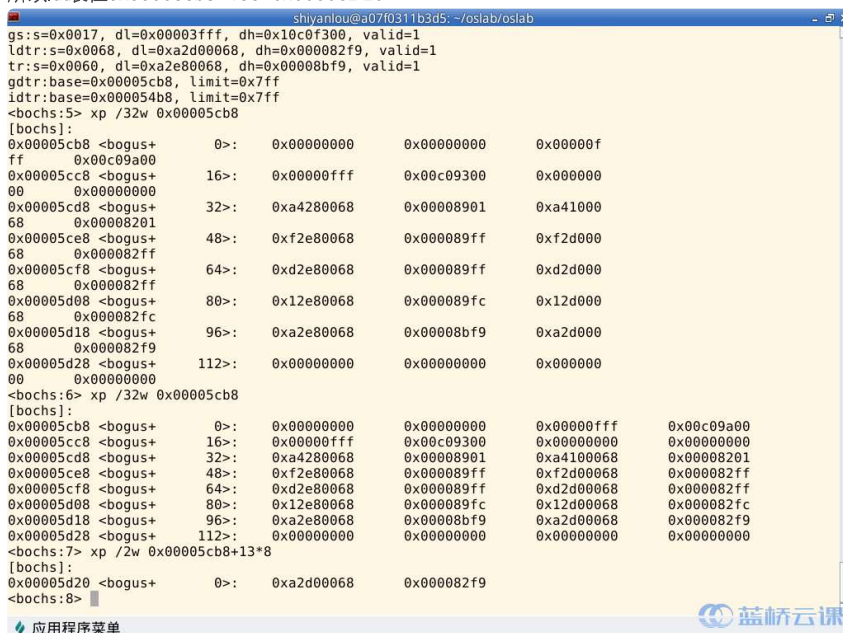
ldtr保存在0x0068=000000001101000, 即13号位置 gdt保存在0x00005cb8



gdt表:



所以ldt表在0x00005cb8+138=0x00005D20



与sreg的输出一致。组合为0x00f9c2d0,如图为ldt前4项。ds现是0x0017=000000000010111,索引为2,即0x00003fff和0x10c0f300与sreg相符则ds段的起始位置在0x10000000 ds:0x3004为0x10003004,验证如下


```
shiyianlou@a07f0311b3d5: ~/oslab/oslab
0x00005d08 <bogus+ 80>: 0x12e80068 0x000089fc 0x12d000
68 0x000082fc
0x00005d18 <bogus+ 96>: 0xa2e80068 0x00008bf9 0xa2d000
68 0x000082f9
0x00005d28 <bogus+ 112>: 0x00000000 0x00000000 0x000000
00 0x00000000
<bochs:6> xp /32w 0x00005cb8
[bochs]:
0x00005cb8 <bogus+ 0>: 0x00000000 0x00000000 0x00000fff 0x00c09a00
0x00005cc8 <bogus+ 16>: 0x00000fff 0x00c09300 0x00000000 0x00000000
0x00005cd8 <bogus+ 32>: 0xa4280068 0x00008901 0xa4100068 0x00008201
0x00005ce8 <bogus+ 48>: 0xf2e80068 0x000089ff 0xf2d00068 0x000082ff
0x00005cf8 <bogus+ 64>: 0xd2e80068 0x000089ff 0xd2d00068 0x000082ff
0x00005d08 <bogus+ 80>: 0x12e80068 0x000089fc 0x12d00068 0x000082fc
0x00005d18 <bogus+ 96>: 0xa2e80068 0x00008bf9 0xa2d00068 0x000082f9
0x00005d28 <bogus+ 112>: 0x00000000 0x00000000 0x00000000 0x00000000
<bochs:7> xp /2w 0x00005cb8+13*8
[bochs]:
0x00005d20 <bogus+ 0>: 0xa2d00068 0x000082f9
<bochs:8> xp /8w 0x00f9a2d0
[bochs]:
0x00f9a2d0 <bogus+ 0>: 0x00000000 0x00000000 0x00000002 0x10c0fa00
0x00f9a2e0 <bogus+ 16>: 0x00003fff 0x10c0f300 0x00000000 0x00f9b000
<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=0xa2d00068, dh=0x000082f9, valid=1
tr:s=0x0060, dl=0xa2e80068, dh=0x00008bf9, valid=1
gdtr:base=0x00005cb8, limit=0x7ff
idtr:base=0x000054b8, limit=0x7ff
<bochs:10> calc ds:0x3004
0x10003004 268447748
<bochs:11>
```

应用程序菜单

意义是项目序号64, 页号3, 偏移4 CR3在0x00000000

```
shiyianlou@a07f0311b3d5: ~/oslab/oslab
[bochs]:
0x00005cb8 <bogus+ 0>: 0x00000000 0x00000000 0x00000fff 0x00c09a00
0x00005cc8 <bogus+ 16>: 0x00000fff 0x00c09300 0x00000000 0x00000000
0x00005cd8 <bogus+ 32>: 0xa4280068 0x00008901 0xa4100068 0x00008201
0x00005ce8 <bogus+ 48>: 0xf2e80068 0x000089ff 0xf2d00068 0x000082ff
0x00005cf8 <bogus+ 64>: 0xd2e80068 0x000089ff 0xd2d00068 0x000082ff
0x00005d08 <bogus+ 80>: 0x12e80068 0x000089fc 0x12d00068 0x000082fc
0x00005d18 <bogus+ 96>: 0xa2e80068 0x00008bf9 0xa2d00068 0x000082f9
0x00005d28 <bogus+ 112>: 0x00000000 0x00000000 0x00000000 0x00000000
<bochs:7> xp /2w 0x00005cb8+13*8
[bochs]:
0x00005d20 <bogus+ 0>: 0xa2d00068 0x000082f9
<bochs:8> xp /8w 0x00f9a2d0
[bochs]:
0x00f9a2d0 <bogus+ 0>: 0x00000000 0x00000000 0x00000002 0x10c0fa00
0x00f9a2e0 <bogus+ 16>: 0x00003fff 0x10c0f300 0x00000000 0x00f9b000
<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=0xa2d00068, dh=0x000082f9, valid=1
tr:s=0x0060, dl=0xa2e80068, dh=0x00008bf9, 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=0x0004b090
CR3=0x00000000
PCD=page-level cache disable=0
PWT=page-level writes transparent=0
CR4=0x00000000: osxmexcpt osfxsr pce pge mce pae pse de tsd pvi vme
<bochs:12>
```

应用程序菜单

项目列表:

```
shiyianlou@a07f0311b3d5: ~/oslab/oslab
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=0xa2d00068, dh=0x000082f9, valid=1
tr:s=0x0060, dl=0xa2e80068, dh=0x00008bf9, 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=0x0004b090
CR3=0x00000000
PCD=page-level cache disable=0
PWT=page-level writes transparent=0
CR4=0x00000000: osxmexcpt osfxsr pce pge mce pae pse de tsd pvi vme
<bochs:12> xp /68w 0
[bochs]:
0x00000000 <bogus+ 0>: 0x00001027 0x00002007 0x00003007 0x00004027
0x00000010 <bogus+ 16>: 0x00000000 0x00024884 0x00000000 0x00000000
0x00000020 <bogus+ 32>: 0x00000000 0x00000000 0x00000000 0x00000000
0x00000030 <bogus+ 48>: 0x00000000 0x00000000 0x00000000 0x00000000
0x00000040 <bogus+ 64>: 0x00ffe027 0x00000000 0x00000000 0x00000000
0x00000050 <bogus+ 80>: 0x00000000 0x00000000 0x00000000 0x00000000
0x00000060 <bogus+ 96>: 0x00000000 0x00000000 0x00000000 0x00000000
0x00000070 <bogus+ 112>: 0x00000000 0x00000000 0x00000000 0x00000000
0x00000080 <bogus+ 128>: 0x00ff3027 0x00000000 0x00000000 0x00000000
0x00000090 <bogus+ 144>: 0x00000000 0x00000000 0x00000000 0x00000000
0x000000a0 <bogus+ 160>: 0x00000000 0x00000000 0x00000000 0x00000000
0x000000b0 <bogus+ 176>: 0x00000000 0x00000000 0x00000000 0x00ff0b27
0x000000c0 <bogus+ 192>: 0x00ff6027 0x00000000 0x00000000 0x00000000
0x000000d0 <bogus+ 208>: 0x00000000 0x00000000 0x00000000 0x00000000
0x000000e0 <bogus+ 224>: 0x00000000 0x00000000 0x00000000 0x00000000
0x000000f0 <bogus+ 240>: 0x00000000 0x00000000 0x00000000 0x00ffa027
0x00000100 <bogus+ 256>: 0x00f97027 0x00000000 0x00000000 0x00000000
<bochs:13>
```

应用程序菜单

第65个项目列表: 0x00f97027

```
shiyuanlou@a07f0311b3d5: ~/oslab/oslab
gs:s=0x0017, dl=0x00003fff, dh=0x10c0f300, valid=1
ldtr:s=0x0068, dl=0xa2d00068, dh=0x000082f9, valid=1
tr:s=0x0060, dl=0xa2e00068, dh=0x00008bf9, 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=0x0004b090
CR3=0x00000000
PCD=page-level cache disable=0
PWT=page-level writes transparent=0
CR4=0x00000000: osxmexcpt osfxsr pce pge mce pae pse de tsd pvi vme
<bochs:12> xp /68w 0
[bochs]:
0x00000000 <bogus+ 0>: 0x00001027 0x00002007 0x00003007 0x00004027
0x00000010 <bogus+ 16>: 0x00000000 0x00024884 0x00000000 0x00000000
0x00000020 <bogus+ 32>: 0x00000000 0x00000000 0x00000000 0x00000000
0x00000030 <bogus+ 48>: 0x00000000 0x00000000 0x00000000 0x00000000
0x00000040 <bogus+ 64>: 0x00ffe027 0x00000000 0x00000000 0x00000000
0x00000050 <bogus+ 80>: 0x00000000 0x00000000 0x00000000 0x00000000
0x00000060 <bogus+ 96>: 0x00000000 0x00000000 0x00000000 0x00000000
0x00000070 <bogus+ 112>: 0x00000000 0x00000000 0x00000000 0x00000000
0x00000080 <bogus+ 128>: 0x00ff3027 0x00000000 0x00000000 0x00000000
0x00000090 <bogus+ 144>: 0x00000000 0x00000000 0x00000000 0x00000000
0x000000a0 <bogus+ 160>: 0x00000000 0x00000000 0x00000000 0x00000000
0x000000b0 <bogus+ 176>: 0x00000000 0x00000000 0x00000000 0x00ffa027
0x000000c0 <bogus+ 192>: 0x00ff6027 0x00000000 0x00000000 0x00000000
0x000000d0 <bogus+ 208>: 0x00000000 0x00000000 0x00000000 0x00000000
0x000000e0 <bogus+ 224>: 0x00000000 0x00000000 0x00000000 0x00000000
0x000000f0 <bogus+ 240>: 0x00000000 0x00000000 0x00000000 0x00ffa027
0x00000100 <bogus+ 256>: 0x00f97027 0x00000000 0x00000000 0x00000000
<bochs:13> xp /w 0+64*4
[bochs]:
0x00000100 <bogus+ 0>: 0x00f97027
<bochs:14>
<bochs:15>
```

应用程序菜单

所以在0x00f97000+34:

```
shiyuanlou@a07f0311b3d5: ~/oslab/oslab
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=0x0004b090
CR3=0x00000000
PCD=page-level cache disable=0
PWT=page-level writes transparent=0
CR4=0x00000000: osxmexcpt osfxsr pce pge mce pae pse de tsd pvi vme
<bochs:12> xp /68w 0
[bochs]:
0x00000000 <bogus+ 0>: 0x00001027 0x00002007 0x00003007 0x00004027
0x00000010 <bogus+ 16>: 0x00000000 0x00024884 0x00000000 0x00000000
0x00000020 <bogus+ 32>: 0x00000000 0x00000000 0x00000000 0x00000000
0x00000030 <bogus+ 48>: 0x00000000 0x00000000 0x00000000 0x00000000
0x00000040 <bogus+ 64>: 0x00ffe027 0x00000000 0x00000000 0x00000000
0x00000050 <bogus+ 80>: 0x00000000 0x00000000 0x00000000 0x00000000
0x00000060 <bogus+ 96>: 0x00000000 0x00000000 0x00000000 0x00000000
0x00000070 <bogus+ 112>: 0x00000000 0x00000000 0x00000000 0x00000000
0x00000080 <bogus+ 128>: 0x00ff3027 0x00000000 0x00000000 0x00000000
0x00000090 <bogus+ 144>: 0x00000000 0x00000000 0x00000000 0x00000000
0x000000a0 <bogus+ 160>: 0x00000000 0x00000000 0x00000000 0x00000000
0x000000b0 <bogus+ 176>: 0x00000000 0x00000000 0x00000000 0x00ffa027
0x000000c0 <bogus+ 192>: 0x00ff6027 0x00000000 0x00000000 0x00000000
0x000000d0 <bogus+ 208>: 0x00000000 0x00000000 0x00000000 0x00000000
0x000000e0 <bogus+ 224>: 0x00000000 0x00000000 0x00000000 0x00000000
0x000000f0 <bogus+ 240>: 0x00000000 0x00000000 0x00000000 0x00ffa027
0x00000100 <bogus+ 256>: 0x00f97027 0x00000000 0x00000000 0x00000000
<bochs:13> xp /w 0+64*4
[bochs]:
0x00000100 <bogus+ 0>: 0x00f97027
<bochs:14> xp /w 0x00f97000+3*4
[bochs]:
0x00f9700c <bogus+ 0>: 0x00f96067
<bochs:15>
```

应用程序菜单

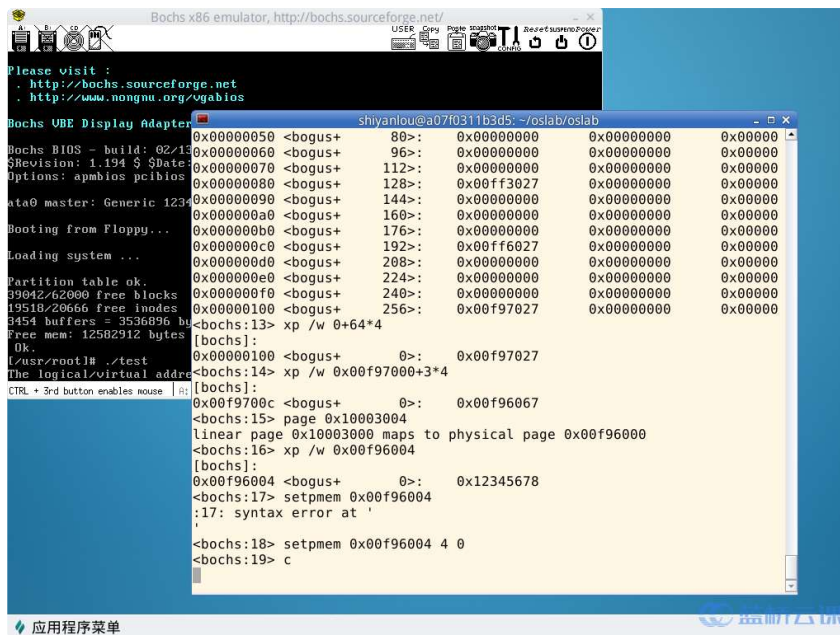
所以最后的物理地址为0x00f96000+0x0004:

```
shiyuanlou@a07f0311b3d5: ~/oslab/oslab
CR0=0x8000001b: PG cd nw ac wp ne ET TS em MP PE
CR2=page fault laddr=0x0004b090
CR3=0x00000000
PCD=page-level cache disable=0
PWT=page-level writes transparent=0
CR4=0x00000000: osxmexcpt osfxsr pce pge mce pae pse de tsd pvi vme
<bochs:12> xp /68w 0
[bochs]:
0x00000000 <bogus+ 0>: 0x00001027 0x00002007 0x00003007 0x00004027
0x00000010 <bogus+ 16>: 0x00000000 0x00024884 0x00000000 0x00000000
0x00000020 <bogus+ 32>: 0x00000000 0x00000000 0x00000000 0x00000000
0x00000030 <bogus+ 48>: 0x00000000 0x00000000 0x00000000 0x00000000
0x00000040 <bogus+ 64>: 0x00ffe027 0x00000000 0x00000000 0x00000000
0x00000050 <bogus+ 80>: 0x00000000 0x00000000 0x00000000 0x00000000
0x00000060 <bogus+ 96>: 0x00000000 0x00000000 0x00000000 0x00000000
0x00000070 <bogus+ 112>: 0x00000000 0x00000000 0x00000000 0x00000000
0x00000080 <bogus+ 128>: 0x00ff3027 0x00000000 0x00000000 0x00000000
0x00000090 <bogus+ 144>: 0x00000000 0x00000000 0x00000000 0x00000000
0x000000a0 <bogus+ 160>: 0x00000000 0x00000000 0x00000000 0x00000000
0x000000b0 <bogus+ 176>: 0x00000000 0x00000000 0x00000000 0x00ffa027
0x000000c0 <bogus+ 192>: 0x00ff6027 0x00000000 0x00000000 0x00000000
0x000000d0 <bogus+ 208>: 0x00000000 0x00000000 0x00000000 0x00000000
0x000000e0 <bogus+ 224>: 0x00000000 0x00000000 0x00000000 0x00000000
0x000000f0 <bogus+ 240>: 0x00000000 0x00000000 0x00000000 0x00ffa027
0x00000100 <bogus+ 256>: 0x00f97027 0x00000000 0x00000000 0x00000000
<bochs:13> xp /w 0+64*4
[bochs]:
0x00000100 <bogus+ 0>: 0x00f97027
<bochs:14> xp /w 0x00f97000+3*4
[bochs]:
0x00f9700c <bogus+ 0>: 0x00f96067
<bochs:15> page 0x10003004
linear page 0x10003000 maps to physical page 0x00f96004
<bochs:16> xp /w 0x00f96004
[bochs]:
0x00f96004 <bogus+ 0>: 0x12345678
<bochs:17>
```

应用程序菜单

结果正确。

setpmem 0x00f96004 4 0



linux里停止运行了。ubuntu上的程序: producer.c

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/ipc.h>
#include<sys/shm.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<semaphore.h>
#include<fcntl.h>

const int itemNum=500;
const int bufSize=10;
int main(){
    int i;
    int shmid;
    sem_t *empty,*full,*mutex;
    int* p;
    int fullvalue;
    sem_unlink("empty");
    sem_unlink("full");
    sem_unlink("mutex");
    empty=sem_open("empty",O_CREAT|O_EXCL,0666,10);
    full=sem_open("full",O_CREAT|O_EXCL,0666,0);
    mutex=sem_open("mutex",O_CREAT|O_EXCL,0666,1);
    shmid=shmget(1,(bufSize)*sizeof(int),IPC_CREAT|0777);
    if(shmid==-1){
        perror("shmget error\n");
    }
    p=(int*)shmat(shmid,NULL,SHM_EXEC);
    for(i=0;i<itemNum;i++){
        sem_wait(empty);
        sem_wait(mutex);
        *(p+i*bufSize)=i;
        sem_getvalue(full,&fullvalue);
        //printf("producer: value:%d add:%p full_value:%d\n",*
        (p+i*bufSize),p+i*bufSize,fullvalue);
        fflush(stdout);
        sem_post(mutex);
        sem_post(full);
    }
    sem_unlink("empty");
    sem_unlink("full");
    sem_unlink("mutex");
    return 0;
}
```

consumer.c



```

#include<stdio.h>
#include<stdlib.h>
#include<sys/ipc.h>
#include<sys/shm.h>
#include<sys/types.h>
#include<semaphore.h>
#include<unistd.h>
#include<fcntl.h>
#include<sys/stat.h>


const int itemNum=500;
const int bufSize=10;
int main(){
    int i;
    int shmid;
    sem_t *empty,*full,*mutex;
    int* p;
    int data;
    int fullvalue;
    freopen("out.txt","w",stdout);
    printf("start\n");
    fflush(stdout);
    empty=sem_open("empty",0);
    full=sem_open("full",0);
    mutex=sem_open("mutex",0);
    if(empty==NULL||full==NULL||mutex==NULL){
        printf("sem_open error\n");
        fflush(stdout);
        return -1;
    }
    printf("sem opened\n");
    fflush(stdout);
    shmid=shmget(1,(bufSize)*sizeof(int),IPC_CREAT|0777);
    if(shmid==-1){
        printf("shmget error\n");
    }
    p=(int*)shmat(shmid,NULL,SHM_EXEC);
    printf("begin consume\n");
    fflush(stdout);
    for(i=0;i<itemNum;i++){
        sem_wait(full);
        sem_wait(mutex);
        data=*(p+i*bufSize);
        printf("%d : %d\n",getpid(),data);
        fflush(stdout);
        sem_post(mutex);
        sem_post(empty);
    }
    return 0;
}

```

在linux0.11上的实现结果：

打开(O) ▾ 

```
4 19 : 0
5 19 : 1
6 19 : 2
7 19 : 3
8 19 : 4
9 19 : 5
10 19 : 6
11 19 : 7
12 19 : 8
13 19 : 9
14 19 : 10
15 19 : 11
16 19 : 12
17 19 : 13
18 19 : 14
19 19 : 15
20 19 : 16
21 19 : 17
22 19 : 18
23 19 : 19
24 19 : 20
25 19 : 21
26 19 : 22
27 19 : 23
28 19 : 24
29 19 : 25
30 19 : 26
```

 蓝桥云课

shm.c

```

#include<asm/segment.h>
#include<linux/kernel.h>
#include<unistd.h>
#include<linux/sched.h>
#define shmSize 10

struct shm_table{
    unsigned int key;
    unsigned int size;
    unsigned long addr;
    int used;
}shm_table[shmSize]={0};

int sys_shmget(unsigned int key, unsigned int size){
    int i;
    unsigned long tmp,addr;
    if(size>PAGE_SIZE){
        return -1;
    }
    for(i=0;i<shmSize;i++){
        if(shm_table[i].used==1&&shm_table[i].key==key)    {
            return i;
        }
    }
    for(i=0;i<shmSize;i++){
        if(shm_table[i].used!=1){
            shm_table[i].used=1;
            shm_table[i].key=key;
            shm_table[i].size=size;
            shm_table[i].addr=get_free_page();
            return i;
        }
    }
    return -1;
}

void* sys_shmat(int shmid){
    if(shm_table[shmid].used!=1){
        printk("shm not exist\n");
        return -1;
    }
    put_page(shm_table[shmid].addr,current->brk+current->start_code);
    return (void*)current->brk;
}

```

linux0.11上的生产者消费者： producer.c


```

#define __LIBRARY__
#include<unistd.h>
#include<fcntl.h>
#include<sys/types.h>
#include<stdio.h>
#include<stdlib.h>

_syscall2(sem_t *, sem_open, const char*,name,unsigned int,value);
_syscall1(int,sem_wait,sem_t *,sem);
_syscall1(int,sem_post,sem_t *,sem);
_syscall1(int,sem_unlink,const char *,name);
_syscall2(int,shmget,unsigned int,key,unsigned int,size);
_syscall1(void*,shmat,int,shmid);

const int itemNum=500;
const int bufSize=10;
int main(){
    int i;
    int shmid;
    sem_t *empty,*full,*mutex;
    int *p;
    int fullvalue;
    empty=sem_open("empty",10);
    full=sem_open("full",0);
    mutex=sem_open("mutex",1);
    shmid=shmget(1,(bufSize)*sizeof(int));
    if(shmid==-1){
        perror("shmget error\n");
        return -1;
    }
    p=(int*)shmat(shmid);
    printf("produce begin\n");
    fflush(stdout);
    for(i=0;i<itemNum+20;i++){
        sem_wait(empty);
        sem_wait(mutex);
        *(p+i*bufSize)=i;
        sem_post(mutex);
        sem_post(full);
    }
    sem_unlink("empty");
    sem_unlink("full");
    sem_unlink("mutex");
    return 0;
}

```

consumer.c

```

#define __LIBRARY__
#include<unistd.h>
#include<fcntl.h>
#include<sys/types.h>
#include<stdio.h>
#include<stdlib.h>

_syscall2(sem_t *, sem_open, const char*,name,unsigned int,value);
_syscall1(int,sem_wait,sem_t *,sem);
_syscall1(int,sem_post,sem_t *,sem);
_syscall1(int,sem_unlink,const char *,name);
_syscall2(int,shmget,unsigned int,key,unsigned int,size);
_syscall1(void*,shmat,int,shmid);

const int itemNum=500;
const int bufSize=10;
int main(){
    int i;
    int shmid;
    sem_t *empty,*full,*mutex;
    int* p;
    int data;
    int fullvalue;
    freopen("nout.txt","w",stdout);
    printf("start\n");
    fflush(stdout);
    empty=sem_open("empty",10);
    full=sem_open("full",0);
    mutex=sem_open("mutex",1);
    if(empty==NULL||full==NULL||mutex==NULL){
        printf("sem_open error\n");
        fflush(stdout);
        return -1;
    }
    printf("sem opened\n");
    fflush(stdout);
    shmid=shmget(1,(bufSize)*sizeof(int));
    if(shmid==-1){
        printf("shmget error\n");
        return -1;
    }
    p=(int*)shmat(shmid);
    printf("begin consume\n");
    fflush(stdout);
    for(i=0;i<itemNum;i++){
        sem_wait(full);
        sem_wait(mutex);
        data=*(p+i%bufSize);
        printf("%d : %d\n",getpid(),data);
        fflush(stdout);
        sem_post(mutex);
        sem_post(empty);
    }
}

```

不知道为什么，要让生产者多生产几个。不然会出现trying to free free page的警告。#回答问题
 1.大概流程就是 GDT->LDT->ds->线性地址->物理地址 2.虚拟地址不变。而线性地址和物理地址变了。再运行一次，会从0x10003004变成0x14003004，正好多了64M，对应了nr*64M的线性地址分配方法。



B *I* 🔗 “ </> 🖼️ ☰ ☰

🔗 [Markdown 语法](#)

请输入想说的话

0 / 2000

发表评论

最新评论



连接高校和企业



公司

[关于我们](#)
[联系我们](#)
[加入我们](#)

产品与服务

[会员服务](#)
[蓝桥杯大赛](#)
[实战训练营](#)
[就业班](#)
[保入职](#)

合作

[1+X证书](#)
[高校实验教学](#)
[企业内训](#)
[合办学院](#)
[成为作者](#)

学习路径

[Python学习路径](#)
[Linux学习路径](#)
[大数据学习路径](#)
[Java学习路径](#)
[PHP学习路径](#)
[全部](#)