

题目：使用 bochs 调试 MBR

Date: 2010.03.27

Author: Cryin'

Blog: <http://hi.baidu.com/justear>

一、环境配置：

操作系统：Microsoft Windows XP Professional Service Pack 3

调试工具：bochs 2.4.0.0

1、安装 bochs

Bochs 是一种十分轻便的使用 c++编写的开源 IA-32(x86)电脑模拟器，可以运行在最受欢迎的平台上。它仿真英特尔 x86 CPU、常见的 I/O 设备、和定制的 BIOS。目前，Bochs 可以被编译仿真 386、486、Pentium/PentiumII/PentiumIII/Pentium4 或 x86-64 位的 CPU，包括可选的 MMX,SSEx 和 3DNow 指令。在 Bochs 仿真环境里能够运行许多操作系统，比如 Linux、DOS、Windows 95/98/NT/2000/XP 或者 Windows Vista。Bochs 是由凯文·劳顿编写的，目前由“<http://bochs.sourceforge.net>”的 Bochs 项目组维护。

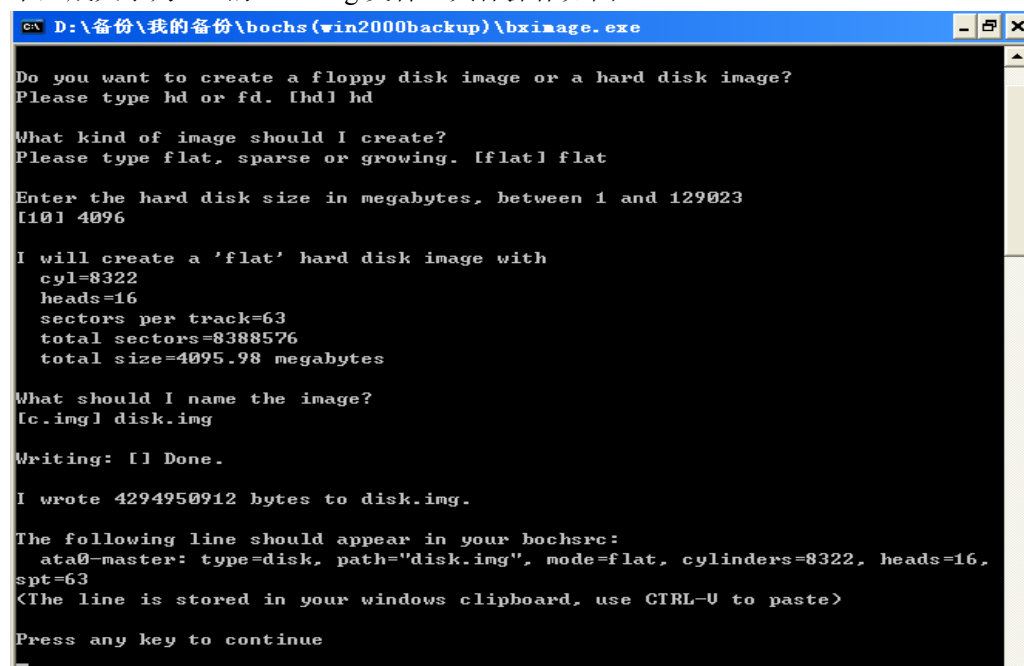
如果论性能的话 bochs 远不及 VMware 等其他虚拟机，但在调试方面，bochs 却有其它虚拟机所不及的优越之处。本文主要介绍如何配置 bochs 调试环境以及如何用 bochs 调试 MBR 程序。重点介绍调试环境配置方法和简单的 MBR 实例调试。并不对调试 MBR 做详细分析。主要是做一个文档型的记录。因为本人在使用 bochs 调试这个过程中摸索了相当长时间，也走了不少弯路。所以如果这个文档能给一些人带来些许帮助，也算是助人为乐，再者，几乎每次重新搭建 bochs 时总是很乱，所以写此文以备将来再次配置调试环境做参考！

Bochs 的最新发布版本可以从 bochs 主页 <http://bochs.sourceforge.net> 下载，也可以从该站点获取到 bochs 相关说明文档以及源代码。本例使用版本为 bochs 2.4.0.0。

配置 Bochs 需要相关的文件有 bochs.exe、bochsgdb.exe、bximage.exe、DEBUG32.EXE 以及一些原本没有可能需要自己下载的文件 niclist.exe 和 WinPcap_4_0_2.exe。既然 bochs 本身是虚拟机那首先要装一个系统，本例使用 win 2k 系统。

1) 创建虚拟硬盘：

双击运行 bximage.exe 创建一个 4G、flat 模式的虚拟硬盘文件 disk.img;这样就会在当前目录下生成大小为 4G 的 disk.img 文件。具体操作如图：



```

C:\D:\备份\我的备份\bochs (win2000backup)\bximage.exe

Do you want to create a floppy disk image or a hard disk image?
Please type hd or fd. [hd] hd

What kind of image should I create?
Please type flat, sparse or growing. [flat] flat

Enter the hard disk size in megabytes, between 1 and 129023
[10] 4096

I will create a 'flat' hard disk image with
  cyl=8322
  heads=16
  sectors per track=63
  total sectors=8388576
  total size=4095.98 megabytes

What should I name the image?
[c.img] disk.img

Writing: [ ] Done.

I wrote 4294950912 bytes to disk.img.

The following line should appear in your bochsrc:
  ata0-master: type=disk, path="disk.img", mode=flat, cylinders=8322, heads=16,
  spt=63
<The line is stored in your windows clipboard, use CTRL-U to paste>

Press any key to continue

```

2) 安装系统:

用 win 2k 系统盘直接安装或者创建一个 ISO 镜像的文件, 鉴于 bochs 虚拟机的速度, 还是不推介自己安装系统, 最好和别人直接拷贝一份已经安装好操作系统 disk.img 文件。

3) 创建 win2k.bxrc:

主要内容:

#win2k.bxrc 配置信息

#设置默认系统 BIOS ROM 模块

romimage: file=\$BXSHARE/BIOS-bochs-latest

#设置 CPU 参数

cpu:count=1,ips=22100000,reset_on_triple_fault=1,cpuid_limit_winnt=0,msrs="msrs.def"

#设置内存, 可以选择 8、16、32、64、128 或者 512

megs: 512

#设置默认 VGA ROM 模块

vgaromimage: file=\$BXSHARE/VGABIOS-lgpl-latest

#vga: extension

vga: extension=vbe

#选择并设置软驱 A, 注意此处设置, 在调试时将 mbr 文件改为 a.img 即可进行调试

floppya: 1_44=a.img, status=inserted

#ATA controller for hard disks and cdroms

ata0: enabled=1, ioaddr1=0x1f0, ioaddr2=0x3f0, irq=14

ata1: enabled=1, ioaddr1=0x170, ioaddr2=0x370, irq=15

ata2: enabled=0, ioaddr1=0x1e8, ioaddr2=0x3e0, irq=11

ata3: enabled=0, ioaddr1=0x168, ioaddr2=0x360, irq=9

#选择引导设备

boot: floppy, disk

Enables or disables the 0xaa55 signature check on boot floppies

floppy_bootsig_check: disabled=0

#关闭日志

log:nul

LOG CONTROLS

panic: action=ask

error: action=report

info: action=report

debug: action=ignore

DEBUGGER_LOG:

debugger_log: -

#设置建立串口通道, 可以与 windbg 相连. 本例中不需要可以注释掉

com2: enabled=1, mode=pipe-server, dev=\\.\pipe\com_2

VGA_UPDATE_INTERVAL

vga_update_interval: 300000

KEYBOARD_SERIAL_DELAY

keyboard_serial_delay: 250

KEYBOARD_PASTE_DELAY

```

keyboard_paste_delay: 100000
# MOUSE
mouse: enabled=0
#private_colormap
private_colormap: enabled=0
#设置网卡信息
ne2k:ioaddr=0x300,irq=10,mac=00:1B:77:59:AC:28,ethmod=win32,
ethdev=\Device\NPF_{5175FF64-AD7E-4B75-A4E0-540FA4AAF493}
# pnics: Bochs/Etherboot pseudo-NIC
pnics: enabled=1, mac=00:26:55:36:ec:ae, ethmod=vnet
# KEYBOARD_MAPPING
keyboard_mapping: enabled=0, map=
##设置硬件设备显卡、网卡.
i440fxsupport: enabled=1, slot1=pcivga, slot2=ne2k
#设置 PCI 设备码、厂商码.貌似可以随便写
pcidev: vendor=0x1234, device=0x5678
#end of win2k.bxrc 配置信息

```

上面关于 win2k.bxrc 配置信息的内容中有一处需要设置网卡信息,这个就可以使用我上面提到的 niclist.exe 这个程序了, 而使用这个程序需要安装 WinPcap, 本例使用 WinPcap_4_0_2.exe;双击 niclist.exe 即可获得相关信息, 本例使用本机的网卡信息, 图上显示数条网卡信息因安装 VMware 缘故。如图:

```

D:\备份\我的备份\bochs(win2000backup)\niclist.exe

1: Adapter for generic dialup and UPN capture
   Device: \Device\NPF_GenericDialupAdapter
2: MS Tunnel Interface Driver
   Device: \Device\NPF_{A9F1485A-F26D-4AB4-A883-74640E88B602}
3: VMware Virtual Ethernet Adapter
   Device: \Device\NPF_{12CE0B6D-4B78-43CF-8660-3C2CA29A2CF3}
4: VMware Virtual Ethernet Adapter
   Device: \Device\NPF_{F4243120-037F-4E13-8030-49023BFC5FE2}
5: Realtek 10/100/1000 Ethernet NIC (Microsoft's
   Packet Scheduler)
   Device: \Device\NPF_{5175FF64-AD7E-4B75-A4E0-540FA4AAF493}

Example config for bochsrc:
ne2k: ioaddr=0x300, irq=3, mac=b0:c4:20:00:00:00, ethmod=win32, ethdev=\Device\NPF_GenericDialupAdapter

Press any key to continue

```

4) 创建运行 bochs 的批处理文件

创建运行 bochs 批处理文件"运行.bat", 主要内容:

set BXSHARE=F:\bochs #此处为 bochs 调试器 bochs.exe 的路径

%BXSHARE%\bochsdbg.exe -q -f win2k.bxrc

rem %BXSHARE%\debug32.exe -q -f win2k.bxrc

#win2k.bxrc 为上一步创建的 win2k.bxrc 配置信息文件

5) 使用 bochs 开始调试:

到这一步就可以使用 bochs 进行调试了, 双击"运行.bat"批处理文件即可打开 bochs 调试

器, 如果有 a.img 文件, bochs 会以光盘启动方式启动操作系统; 如果没有 a.img 文件, bochs 会以硬盘形式启动操作系统, 当然 a.img 的名称可以在 win2k.bxrc 配置信息文件中按照个人喜好随意修改。

二、使用 bochs 调试 MBR

1) 编写 MBR 程序:

本例中使用我之前写过的一个 MBR 小程序"基于 MBR 的系统登录密码验证程序"进行演示使用 bochs 调试 MBR 的方法, 具体可参见 <http://hi.baidu.com/justear/blog/item/3be1e953295501838c5430d2.html>, 首先将汇编代码用 NASM 编译生成的 bin 文件拷贝到 bochs 文件夹下并更名为 a.img, 这样 bochs 就会以 a.img 作为光盘启动操作系统。

2) Bochs 调试命令:

鉴于文章主题, 这里只简单介绍一些最常用的 bochs 调试命令, 作为入门这些就足够了, 更多的可以参考 bochs 相关的说明文档。网上也有很多 bochs 调试命令的资料。下面只介绍本例中用到的若干命令:

命令 c 运行程序, 相当于 windbg 的 g 以及 OD 的 F9

命令 s 即 step, 单步执行程序

命令 p 单步执行, 步过函数

命令 q 退出 bochs 并关闭虚拟机

命令 b 下断点命令, 如本例中: b 0x7c00

命令 blist 显示断点状态

命令 watch 显示当前所有读写断点

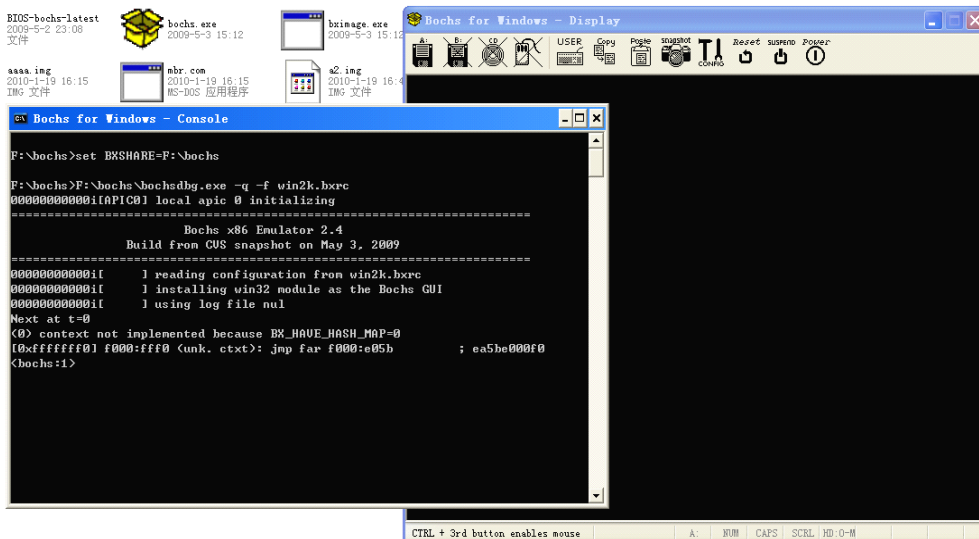
命令 r 显示寄存器的值

命令 u 反汇编代码, 可设定起始和结束位置

命令 info 根据参数不同显示相关信息

3) 开始调试 MBR

将第一步编写的 MBR 程序拷贝到 bochs 调试器文件夹下并更名为 a.img, 双击运行.bat 开始调试, 看到如下界面就说明一切工作正常, 就可以开始调试了:



下面是具体的调试代码:

```
F:\bochs>set BXSHARE=F:\bochs
```

```
F:\bochs>F:\bochs\bochsdbg.exe -q -f win2k.bxrc
000000000000i[APIC0] local apic 0 initializing
```

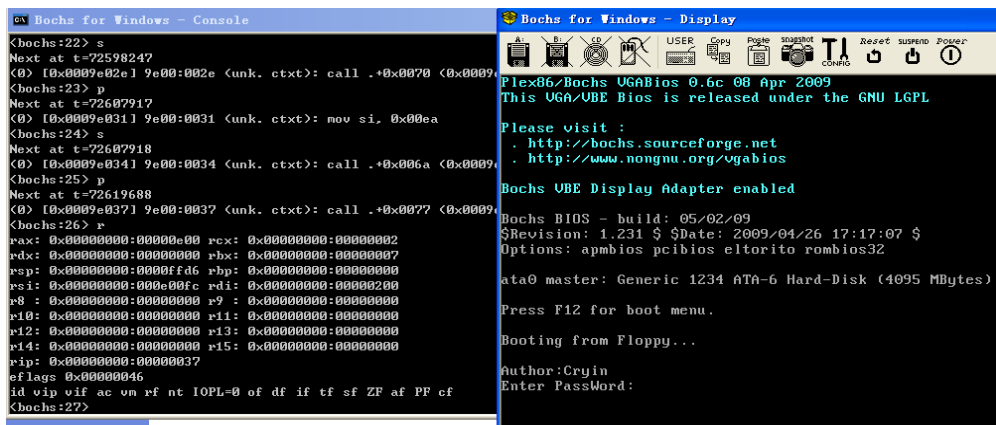
Bochs x86 Emulator 2.4
Build from CVS snapshot on May 3, 2009

```
000000000000i[      ] reading configuration from win2k.bxrc
000000000000i[      ] installing win32 module as the Bochs GUI
000000000000i[      ] using log file nul
Next at t=0
(0) context not implemented because BX_HAVE_HASH_MAP=0
[0xffffffff] f000:fff0 (unk. ctxt): jmp far f000:e05b          ; ea5be000f0
<bochs:1> b 0x7c00          //此处 MBR 被加载到 0x7c00 处断点
<bochs:2> c                  //运行程序
(0) Breakpoint 1, 0x0000000000007c00 in ?? ()
Next at t=72597973
(0) [0x00007c00] 0000:7c00 (unk. ctxt): xor ebx, ebx          ; 6631db
<bochs:3> s                  //单步执行
Next at t=72597974
(0) [0x00007c03] 0000:7c03 (unk. ctxt): mov ds, bx            ; 8edb
<bochs:4> s
Next at t=72597975
(0) [0x00007c05] 0000:7c05 (unk. ctxt): mov ax, word ptr ds:0x413 ; a11304
<bochs:5> s
Next at t=72597976 //40:13, BIOS 数据区保存常规的内存大小
(0) [0x00007c08] 0000:7c08 (unk. ctxt): and al, 0xfc          ; 24fc
<bochs:6> s
Next at t=72597977
(0) [0x00007c0a] 0000:7c0a (unk. ctxt): sub ax, 0x0004        ; 2d0400
<bochs:7> s
Next at t=72597978 //开辟一段内存, 实现程序的驻留
(0) [0x00007c0d] 0000:7c0d (unk. ctxt): mov word ptr ds:0x413, ax ; a31304
<bochs:8> s
Next at t=72597979
(0) [0x00007c10] 0000:7c10 (unk. ctxt): shl ax, 0x06          ; c1e006
<bochs:9> s
Next at t=72597980
(0) [0x00007c13] 0000:7c13 (unk. ctxt): mov es, ax            ; 8ec0
<bochs:10> s
Next at t=72597981
(0) [0x00007c15] 0000:7c15 (unk. ctxt): mov si, 0x7c00        ; be007c
<bochs:11> s
Next at t=72597982
```

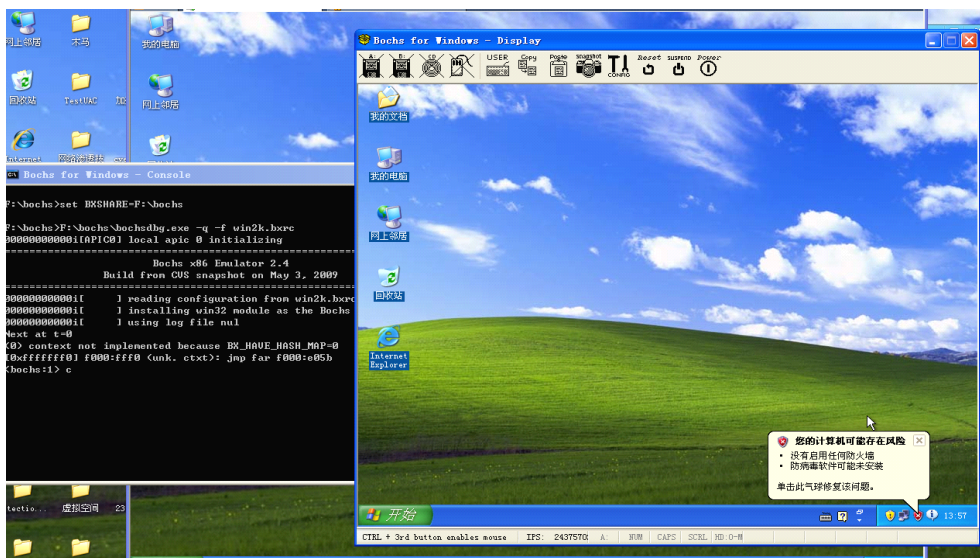
```

(0) [0x00007c18] 0000:7c18 (unk. ctxt): xor di, di ; 31ff
<bochs:12> s
Next at t=72597983 //拷贝 512
(0) [0x00007c1a] 0000:7c1a (unk. ctxt): mov cx, 0x0100 ; b90001
<bochs:13> s
Next at t=72597984 //拷贝代码到驻留内存中执行
(0) [0x00007c1d] 0000:7c1d (unk. ctxt): rep movsw word ptr es:[di], word ptr ds:
[si] ; f3a5
<bochs:14> p //单步步过
Next at t=72598240
(0) [0x00007c1f] 0000:7c1f (unk. ctxt): mov ax, 0x0201 ; b80102
<bochs:15> s
Next at t=72598241
(0) [0x00007c22] 0000:7c22 (unk. ctxt): mov cl, 0x02 ; b102
<bochs:16> s
Next at t=72598242
(0) [0x00007c24] 0000:7c24 (unk. ctxt): cdq ; 6699
<bochs:17> s
Next at t=72598243
(0) [0x00007c26] 0000:7c26 (unk. ctxt): push es ; 06
<bochs:18> s
Next at t=72598244
(0) [0x00007c27] 0000:7c27 (unk. ctxt): push 0x002b ; 682b00
<bochs:19> s
Next at t=72598245
(0) [0x00007c2a] 0000:7c2a (unk. ctxt): retf ; cb
<bochs:20> p //单步步过
Next at t=72598246
(0) [0x0009e02b] 9e00:002b (unk. ctxt): mov si, 0x00db ; bedb00
<bochs:21> s
Next at t=72598247
(0) [0x0009e02e] 9e00:002e (unk. ctxt): call .+0x0070 (0x0009e0a1) ; e87000
<bochs:22> p //单步步过显示信息的函数
Next at t=72607917
(0) [0x0009e031] 9e00:0031 (unk. ctxt): mov si, 0x00ea ; beea00
<bochs:23> s
Next at t=72607918
(0) [0x0009e034] 9e00:0034 (unk. ctxt): call .+0x006a (0x0009e0a1) ; e86a00
<bochs:24> p //单步步过显示信息的函数
Next at t=72619688
(0) [0x0009e037] 9e00:0037 (unk. ctxt): call .+0x0077 (0x0009e0b1) ; e87700
<bochs:25> //此时系统要求输入密码
程序执行到这里，系统要求用户输入密码，如图所示虚拟机界面：

```



在虚拟机里面输入自己在 MBR 程序里面设置的密码就可以接着在 bochs 里面一步一步继续往下面调试了！从 MBR 被载入并执行开始到 NTLDR 被载入并执行一直到操作系统开启的整个过程都可以一步步跟踪调试。系统启动后 bochs 就想别的虚拟机一样进入一个虚拟的操作系统中了，如图：



如果自己动手开发操作系统，那么使用 bochs 调试是必不可缺少的。网上有很多几十行的一个 hello world 的最小的操作系统的代码，现在就可以来调试看看，也可以通过慢慢调试自己亲自动手写一个操作系统！