The Bochs PC Emulator

Soft Hardware

Although the Open Source community has developed free alternatives to complex Office suites, PC emulators have proved challenging so far. Despite poor performance, the Bochs project is at least interesting for test purposes. BY BERNHARD BABLOK

The Bochs (say "box") project has been around since 1994. Its aim is to provide a portable x86 emulator [1]. In this case portable means not restricted to Linux or to the Intel platform. The code has been implemented entirely in C++ for this purposes, so Bochs runs on various platforms, from Linux via Windows to the Macintosh. Bochs even ran on the Linux/zSeries – as is evidenced by Sourceforge bug reports, strangely enough.

Platform independence is the main difference to the commercial alternative, VMware. x86 commands are simulated and are thus available on other architectures, whereas virtualization, which allows simulated instructions to run almost at hardware speeds, is restricted to x86 only emulators. Having said that, Bochs is a lot slower than VMware.

If you have been keeping track of Bochs developments over the course of the last few releases, you will have noticed the considerable progress that has been achieved with each new release, both with respect to the supported instruction sets and peripherals, and to speed. The project is extremely active and downloading the latest version is definitely worthwhile.

Hardware

Bochs provides a simple PC with the following specs:

- 286 through Pentium Pro instruction set (depending on the configuration)
- VGA graphics chip
- two floppy drives (1.44 Mbytes or 2.88 Mbytes)
- four ATA channels with up to eight devices
- mouse and keyboard support
- SoundBlaster emulation
- NE 2000 support
- simulation of up to 15 processors is now possible.

The VGA emulation only supports low resolutions and network support is limited to an emulation of a NE 2000 network card, no matter which card is installed on the host system. The hardware specs are definitely not state of the art, but more than sufficient to install and use a guest operating system.

Bochs does not place any constrictions on the host operating system, but the more powerful your hardware the better: an emulator program, implemented in C++ and running entirely in userspace simply can't get enough. Before you can use Bochs, the powers that be dictate that you must first download and configure the program.

Download and Installation

The source package and pre-compiled binaries, including RPM format packages, are available on the Bochs homepage at [1]; they contain a disk image of a simple Linux system ("dlx", with a 1.3 kernel). This article discusses the 2.0.0-pre2 version, although 2.0.0 should be available by the time this issue of Linux Magazine hits the shops.

With respect to functionality, there is no real reason to download and compile the sources as of version 1.4.1, although previously, it did make sense for users requiring non-US keyboard support. However, the sources comprise a series of patches that have not, or not yet, found there way into the official codebase. So, if you want to do some experimenting of your own, or simply intend to try out the latest features, you might like to compile your own version of Bochs. This is in fact quite trivial, as Bochs supports the typical "./configure options; make; make install”. Selecting the “configure” options will typically take longer than actually compiling the sources. If you do not have a quick and permanent Internet connection, you might like to edit the make file before running “make install”, as the command will otherwise attempt to download an entire Linux disk image.

There were two useful “configure” options missing in the Linux binary package, although they no longer work in the 2.0.0-pre2 version. First, there is the “--enable-slowdown” option that allows you to synchronize the Bochs clock with the host system’s internal clock. Without this option, the system tends to apply its own notion of time. The second useful option is “--enable-idle-hack”, which reduces the CPU load caused by the emulator when idling – you have to live with permanent full load otherwise.

“make install” installs the program below “/usr/local/bochs/latest/”, a link to the current version. This allows you to install multiple parallel versions. The 1.4.1 version is supplied with the current SuSE Linux 8.1, however, this version installs outdated documentation and does not provide adequate non-US keyboard support; so you can look forward to some manual tweaking.

Setting Up Bochs

From the user’s viewpoint, Bochs comprises of the actual emulator, “/usr/local/bin/bochs”, and a configuration file that specifies the hardware and error handling. The program searches for the configuration file at the following locations and in the following order:

- “./.bochsrc”
- “./bochsrc”
- “./bochsrc.txt”
- “ ~ /.bochsrc”
- “/etc/bochsrc” (as of 2.0).

SuSE have patched the source code to search in “/etc/bochsrc” first, a debatable modification, as this prevents multiple users from working with multiple parallel configurations in separate directories.

The configuration file itself has a simple format and is exhaustively commented to boot. Listing 1 shows the most important configuration items. As most parameters are self-explanatory, let’s take a closer look at lines 20 (“com1”) and 27 (“ips”): “com1” can either be a genuine serial line, or alternatively an X terminal (“ > pty”). To allow this, you need to launch two XTerms: one for Bochs and the other for your “com1” output. The second terminal uses the “tty” command to query “pty” output. After launching Bochs, the entire serial output will be passed to the XTerm.

Benchmarking

The “ips” configuration option means instructions per second and refers to the number of instructions emulated per second. This figure affects the emulator’s behavior with respect to the “vga_update_interval” or the various keyboard delay rates. The figure actually controls the timing of the emulator, as is made evident by the system clock of the emulated systems – if there is nothing to do, the clock just races ahead.

Unfortunately, the value of “ips” is not constant, but depends on the complexity of the emulated instructions. Experiments with Linux as a guest operating system have shown that there is very little deviation with constant loads, although varying loads can cause deviations of between 1 and 2 powers of ten.

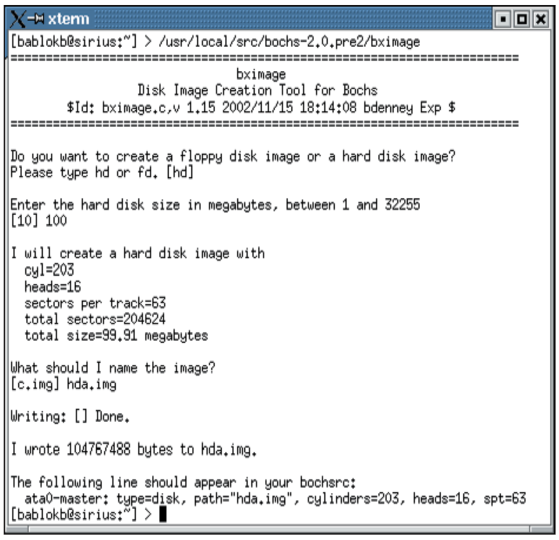
To discover an exact value for “ips”, you need to compile the source code yourself and enable an option in the “config.h” file created by “configure” to output the value periodically. If you do not want to take the trouble of discovering the value this way, you will have

Listing 1: “bochsrc”

```
01: romimage: file=/usr/local/bochs/latest/BIOS-bochs-latest,
address=0xf0000
02: megs: 64
03: vgaromimage: /usr/local/bochs/latest/VGABIOS-elpin-2.40
04: floppya: 1_44=/dev/fd0, status=inserted
05: floppyb: 1_44=/var/bochs/floppy_b.img, status=ejected
06: ata0: enabled=1, ioaddr1=0x1f0, ioaddr2=0x3f0, irq=14
07: ata1: enabled=1, ioaddr1=0x170, ioaddr2=0x370, irq=15
08: ata2: enabled=0, ioaddr1=0x1e8, ioaddr2=0x3e8, irq=11
09: ata3: enabled=0, ioaddr1=0x168, ioaddr2=0x368, irq=9
10: ata0-master: type=disk, path=/data/bochs/hda.10M, cylinders=306,
heads=4, spt=17
11: ata0-slave: type=cdrom, path=/dev/cdrom, status=ejected
12: ata1-master: type=disk, path=/data/bochs/hdb.10M, cylinders=306,
heads=4, spt=17
13: boot: disk
14: floppy_bootsig_check: disabled=0
15: log: /var/log/bochs.log
16: panic: action=report
17: error: action=report
18: info: action=report
19: debug: action=ignore
20: com1: dev=/dev/pts/1
21: parport1: enable=1, file="/var/bochs/parport.out"
22: sb16: midimode=1, midi=/dev/midi00, wavemode=1, wave=/dev/dsp,
loglevel=2, log=sb16.log, dmatimer=600000
23: vga_update_interval: 300000
24: keyboard_serial_delay: 250
25: keyboard_paste_delay: 100000
26: floppy_command_delay: 500
27: ips: 2000000
28: private_colormap: enabled=0
29: ne2k: ioaddr=0x280, irq=9, mac=b0:c4:20:00:00:00, ethmod=linux,
ethdev=eth0
30: keyboard_mapping: enabled=1,
map=/usr/local/bochs/latest/keymaps/x11-pc-de.map
```

to make do with the comments in the sample "bochsrc" file.

On a Duron 700 system, Bochs is capable of emulating approximately five million instructions per second (this was measured when loading and initializing a Linux 2.0 kernel). On starting up Linux indicates a value of 4.96 bogomips, whereas the Linux host system is capable of 1400 bogomips. Bochs 1.4.1 only managed two million IPS, so the new Bochs release is obviously a lot quicker.



```
[bablokb@sirius:~] > /usr/local/src/bochs-2.0.pre2/bximage
========================================================================
                                bximage
                      Disk Image Creation Tool for Bochs
          $Id: bximage.c,v 1.15 2002/11/15 18:14:08 bdenney Exp $
========================================================================

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

Enter the hard disk size in megabytes, between 1 and 32255
[10] 100

I will create a hard disk image with
  cyl=203
  heads=16
  sectors per track=63
  total sectors=204624
  total size=99.91 megabytes

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

Writing: [] Done.

I wrote 104767488 bytes to hda.img.

The following line should appear in your bochsrc:
  ata0-master: type=disk, path="hda.img", cylinders=203, heads=16, spt=63
[bablokb@sirius:~] >
```

Figure 1: Hard disk configuration

Floppies and CDs

Bochs uses the physical floppy drive (which is configured in line 4 of Listing 1), or a floppy image (line 5 includes sample instructions for a second drive). Images of this type are quite easy to create:

```
dd if=/dev/zero of=floppy.img ↵
bs=1k  count=1440
mkdosfs floppy.img
```

Of course, you can use similar steps to create a floppy with an Ext-2 or Minix filesytem. Floppy images also offer the advantage of being a lot quicker than the physical drive. CDs can also be physical drives or ISO images. As modern CD ROM drives are typically quite fast, there is very little difference compared with a CD image.

Bochs version 1.4 or later supports booting CD ROMs, as the BIOS includes El Torito support. This is an enormous gain, as you can now launch the installation CDs of a Linux distribution directly without needing to create boot disks.

Hard Disk Images

Hard disks are emulated by image files just like floppies or CDs; although you may also be able to use a raw partition, this approach is not recommended. The values for the "bs" and "count" parameters of the "dd" command are included in the old documentation below "docs-html/install.html". However, you might prefer to use the new "bximage" tool that prompts you for the size and name of the image file, creates the file and generates an appropriate entry for the "bochsrc" configuration file (Figure 1).

Hard disk images can be partitioned using a minimal floppy based Linux system within Bochs. Alternatively, you can launch "fdisk" with the image file as an argument. In this case, you will need to open the expert menu of fdisk before partitioning the file, in order to supply the cylinder, sector and head values manually.

The Bochs homepage offers complete images of pre-installed free systems from Minix, through FreeDOS to Debian. The current Debian image weighs in at 77 Mbytes. ATA channels and hard disks are configured as shown in Listing 1 (lines 6 through 12). Again, this is an enormous gain in comparison to the old "diskc" and "diskd" syntax in version 1.4.1.

Although this fact is little known, you can use hard disks without partitions like enormous floppies. The following command

```
mke2fs -F hdb.img
```

creates a filesystem on the image. This is particularly useful for a second hard disk, as an image created this way can also be mounted via the loop device of the host system to support data transfers between the host and guest systems. In version 2.0 you can now access the host system via the tuntab interface. Mounting image partitions via the loop device is somewhat more complex. The loop device will need the correct offset in the image file, and there is some danger of data loss if you miscalculate.

Launching the Emulation

After using the configuration file to define the hardware and creating any floppy, CD and hard disk images you might need, you can simply type the "bochs" command to launch the emulator. The configured device will now attempt to load an operating system. Figure 2 shows Bochs launching the current Debian installation CD.

You can clearly see the configured hardware (the 10 Mbyte hard disk and an ATAPI CD ROM, for example). The available devices are shown top left in the emulator, however, the icons at the top right do not work, with the exception of

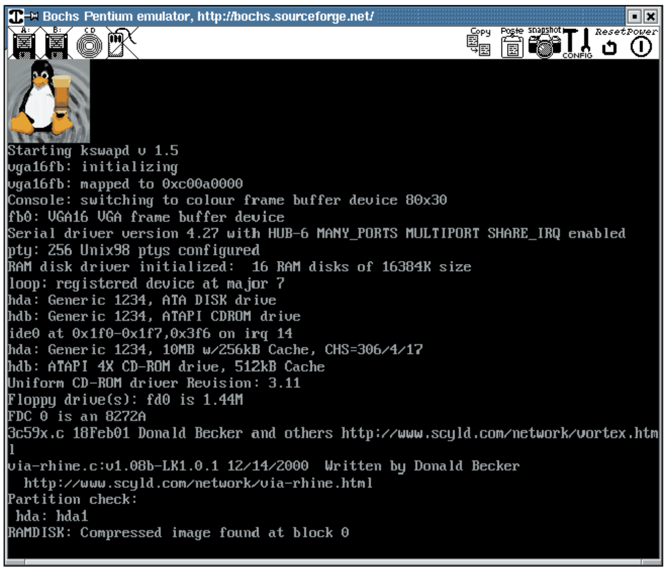
The History of Bochs

Kevin Lawton started working on Bochs as a commercial product in 1994 (!). Early in 2000 MandrakeSoft bought the code and placed it under the LGPL. At this time, Kevin Lawton started working for Mandrake on the Plex86 project (the new homepage is located at [2]) which is intended to provide an emulator with similar features to VMware.

In March 2001 Bochs became a Sourceforge project with numerous developers involved – the developer mailing list has over 300 members. Anyone who has kept in touch with the bugfixes and patches released since then, will be aware that there is lot going on here. When the New Economy bubble burst, Mandrake was forced to tread more carefully, and dismissed a number of employees, including Kevin Lawton. This effectively put the Plex86 project on ice.

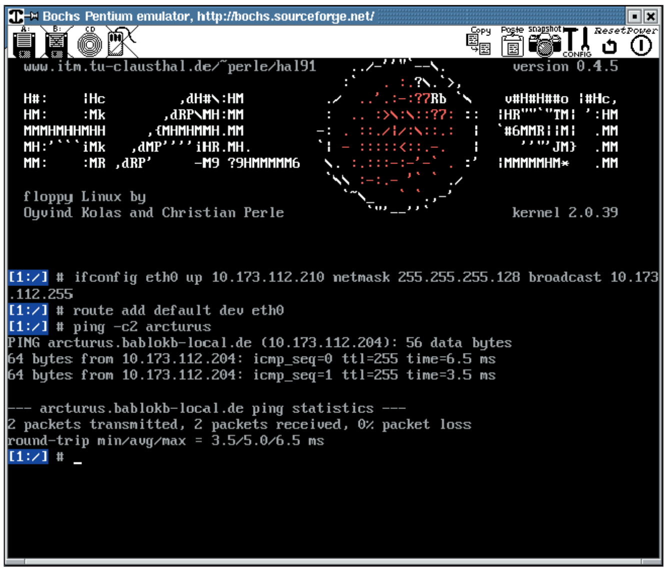
Today, the project has moved back to its old homepage *www.plex86.org* [2], a Sourceforge clone. The news section suggests that work is continuing on the project, but this is contradicted by a lack of new code in the CVS repository. Unfortunately, the last code release is also broken – Linux has made considerable progress in the last few years, and Kevin Lawton is no longer registered as a Plex86 developer, although he has taken up work on Bochs.

There would seem to be some close contact between Bochs and Plex86. Ideally, Plex86 should look to virtualizing the simulated x86 instructions provided by Bochs and moving them from userspace to kernel mode, which would certainly ensure an enormous performance boost.



```
Starting kswapd v 1.5
vga16fb: initializing
vga16fb: mapped to 0xc00a0000
Console: switching to colour frame buffer device 80x30
fb0: VGA16 VGA frame buffer device
Serial driver version 4.27 with HUB-6 MANY_PORTS MULTIPORT SHARE_IRQ enabled
pty: 256 Unix98 ptys configured
RAM disk driver initialized:  16 RAM disks of 16384K size
loop: registered device at major 7
hda: Generic 1234, ATA DISK drive
hdb: Generic 1234, ATAPI CDROM drive
ide0 at 0x1f0-0x1f7,0x3f6 on irq 14
hda: Generic 1234, 10MB w/256kB Cache, CHS=306/4/17
hdb: ATAPI 4X CD-ROM drive, 512kB Cache
Uniform CD-ROM driver Revision: 3.11
Floppy drive(s): fd0 is 1.44M
FDC 0 is an 8272A
3c59x.c 18Feb01 Donald Becker and others http://www.scyld.com/network/vortex.htm
l
via-rhine.c:v1.08b-LK1.0.1 12/14/2000  Written by Donald Becker
  http://www.scyld.com/network/via-rhine.html
Partition check:
 hda: hda1
RAMDISK: Compressed image found at block 0
```

Figure 2: Debian under Bochs



```
 www.itm.tu-clausthal.de/~perle/hal91                    version 0.4.5

 floppy Linux by
 Oyvind Kolas and Christian Perle                         kernel 2.0.39


[1:/] # ifconfig eth0 up 10.173.112.210 netmask 255.255.255.128 broadcast 10.173
.112.255
[1:/] # route add default dev eth0
[1:/] # ping -c2 arcturus
PING arcturus.bablokb-local.de (10.173.112.204): 56 data bytes
64 bytes from 10.173.112.204: icmp_seq=0 ttl=255 time=6.5 ms
64 bytes from 10.173.112.204: icmp_seq=1 ttl=255 time=3.5 ms

--- arcturus.bablokb-local.de ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 3.5/5.0/6.5 ms
[1:/] # _
```

Figure 3: Network connections on a local area network

the power switch and the configuration icon. You can temporarily overwrite configuration options by stipulating command line parameters. The format is the same as the configuration file. As the options contain shell metacharacters, you will need to enclose them in quotes. This is a bit annoying as it prevents your from using features such as filename completion.

Scripted or Interactive Configuration

A shell script provides an easy solution to this, allowing you to pass parameters to the program in typical Unix fashion. A script is included in the "patches" directory of the source distribution. The script additionally defines the size and geometry of floppy and hard disks. Depending on your configure options, Bochs may also include an interactive configuration program, although the tool is unfortunately not exactly intuitive. The tool is either launched on starting the emulator or via the config icon at the top right of the emulator.

A note for KDE users, although Gnome probably has the same issue: you may have difficulty switching consoles in the Linux guest system, as KDE will tend to grab keyboard shortcuts, such as [Alt] + [F1]; [Alt] + [F2] launches a minimal command line for example. Some re-mapping is required, for example [Alt] + [Shift] + [F2] instead of [Alt] + [F2] for the command line. Unfortunately, any communication between the guest and the host system will need to be indirect in the 2.0.0-pre2 version. You can exchange data via floppies, CDs or hard disks (as described above). The hard disk method allows you to exchange larger amounts of data, but will require you to terminate and re-launch the emulator. The new tuntab interface now takes care of talking to the host system

If your network adapter is supported, the guest system will also be able to communicate with other systems on the network, of course (see Figure 3). The network emulation uses the physical network adapter. To exchange data on the network you will need a second machine running NFS or Samba with mounts both on the host and on the guest system.

Conclusion

The configuration, launch, and operations of the Bochs PC emulator are no longer an issue, and the rapid progress the project has made indicates that the Open Source community is still working on a useable emulator. However, the fact that Bochs does without kernel modules has a serious impact on the emulator's performance (an old 486 that I still possess was 20 time quicker than an emulated PC on a Duron 700). Still, it would be wrong to condemn Bochs as a waste of time. Of course, it is incapable of emulating Windows in order to run a special program that is not available on Linux. This has not stopped me from using Bochs regularly and productively for several projects. These projects are typically CD installation and restore procedures – or creating CD images to test whether they are bootable, and sometimes destructive operations, such as restoring data to hard disks. In these cases pure performance never has been an issue. And Bochs even offers some advantages over VMware as the environment can be controlled via the command line, and thus lends itself to automated test scripts.

It remains to be seen that the Plex86 project (see Insert "The History of Bochs") will see the current performance problems being resolved in time. Till then, users for whom Bochs' performance is out of the question will need to turn to one of the commercial alternatives.

INFO

[1] Bochs homepage: *http://bochs.sourceforge.net/*

[2] New Plex86 homepage: *http://savannah.nongnu.org/projects/plex86*

THE AUTHOR

Bernhard Bablok works as a Group-Leader of the Datawarehouse Group for Allianz Versicherungs AG based at Munich, Germany. His PC-career started with OS/2 version 1.0. After IBM turned down OS/2, he switched to Linux in 1996. Since then, none of his PCs had to suffer anymore under an operating system from Redmond.