

Bochs User Manual

[Prev](#)[Next](#)

Chapter 1. Introduction to Bochs

Table of Contents

[What is Bochs?](#)[Who uses Bochs?](#)[Is Bochs right for me?](#)[Will it work for me?](#)[Bochs License](#)[Third Party Software Licensing and Temporary Files](#)[Features](#)[Supported Platforms](#)[FAQ](#)

What is Bochs?


Bochs is a program that simulates a complete Intel x86 computer. It can be configured to act like a 386, 486, Pentium, or Pentium Pro. Bochs interprets every instruction from power-up to reboot, and has device models for all of the standard PC peripherals: keyboard, mouse, VGA card/monitor, disks, timer chips, network card, etc. Because Bochs simulates the whole PC environment, the software running in the simulation "believes" it is running on a real machine. This approach allows Bochs to run a wide variety of software with no modification, include most popular x86 operating systems: Windows 95/98/NT, all Linux flavors, all BSD flavors, and more.

Bochs is written in the C++ programming language, and is designed to run on many different host platforms^[1], including x86, PPC, Alpha, Sun, and MIPS. No matter what the host platform is, Bochs still simulates x86 software. In other words, it does not depend on the native instructions of the host machine at all. This is both a strength and a weakness, and it's the major difference between Bochs and many other x86 emulation software such as plex86, VMware, etc. Because Bochs uses software simulation for every single x86 instruction, it can simulate a Windows application on an Alpha or Sun workstation. However, the downside of Bochs's approach is simulation performance. To model the processor accurately, Bochs must run many instructions for every simulated x86 instruction, and this makes the simulated machine many times slower than the physical machine. Commercial PC emulators (VMware, Connectix, etc.) can achieve much high emulation speed using a technique called virtualization^[2], but they are neither portable to non-x86 platforms nor open

source. The [Plex86](#) project is working toward an open-source x86 simulator with virtualization.

To do anything interesting in the simulated machine, Bochs needs to interact with the operating system on the host platform (the host OS). When you press a key in the Bochs display window, a key event goes into the device model for the keyboard. When the simulated machine needs to read from the simulated hard disk, Bochs reads from a disk image file on the host machine. When the simulated machine sends a network packet to the local network, Bochs uses the host platform's network card to send the packet out into the real world. These interactions between Bochs and the host operating system can be complicated, and in some cases they are host platform specific. Sending a network packet in FreeBSD requires different code than sending the packet in Windows 95, for example. For this reason, certain features are supported on some host platforms and not others. On GNU/Linux, Bochs can simulate a network card that communicates with the world, but on BeOS the simulated network card may not work because the communication code between the device model and the BeOS operating system has not been written.

Bochs was written by Kevin Lawton starting in 1994. It began as a commercial

product, which you could buy with source code for ...  [3] Finally, in March 2000, Mandrakesoft bought Bochs and made it open source under the GNU LGPL. In March 2001, Kevin helped a few developers to move all Bochs activities from bochs.com to a new site at bochs.sourceforge.net. Since then the Bochs Project has settled into its new home, and around release times has even hit #1 most active project of the week at Source Forge.

Notes

- [1] Since Bochs can run on one kind of machine and simulate another machine, we have to be clear in our terminology to avoid confusion. The host platform is the machine that runs the Bochs software. The guest platform is the operating system and applications that Bochs is simulating.
- [2] Virtualization takes advantage of simulating x86 instructions on an x86 machine, allowing large portions of the simulation to take place at native hardware speed. Whenever the simulated machine talks to the hardware or enters certain privileged modes (such as in kernel code), the simulator typically takes control and simulates that code in software at much slower speed, just like Bochs does.
- [3] We need a Bochs historian to help out here. For background, it would be interesting to know how much Bochs used to cost and what it was used for. I thought I saw an interview out there somewhere where Kevin says why he started it and some more background information.

[Prev](#)

Bochs User Manual

[Home](#)

[Next](#)

Who uses Bochs?