|  |  |  |
| --- | --- | --- |
| Bochs Developers Guide | | |
| [Prev](http://docs.google.com/coding.html) | Chapter 4. Coding | [Next](http://docs.google.com/webmastering.html) |

4.2. Building a Bochs release

4.2.1. Preparing source files and SVN

Update version number and strings in configure.in.

VERSION="2.5"  
VER\_STRING="2.5"  
REL\_STRING="Build from SVN snapshot on November 27, 2011"

In the README file you have to update version number and date. Add some information about new features if necessary.

Bochs x86 Pentium+ Emulator  
Updated: Sun Nov 27 16:55:00 CET 2011  
Version: 2.5

In the file bochs.manifest you have to update the version number for the Windows build.

version="2.5.0.0"

Check date, update/sumup info in CHANGES. Run autoconf to regenerate configure and check them in. Create an SVN tag that contains all files of the revision that was used in the release. For prereleases I create a normal SVN tag like this:

svn mkdir tags/REL\_2\_5\_pre1\_FINAL  
 svn copy trunk/bochs tags/REL\_2\_5\_pre1\_FINAL/bochs  
 svn commit

But for a real release, I make an SVN branch tag AND a normal tag.

svn mkdir tags/REL\_2\_5\_FINAL  
 svn copy trunk/bochs tags/REL\_2\_5\_FINAL/bochs  
 svn mkdir branches/REL\_2\_5  
 svn copy trunk/bochs branches/REL\_2\_5/bochs  
 svn commit

The tag marks where the branch split off of the main trunk. This is very useful in maintaining the branch since you can do diffs against it.

svn diff tags/REL\_2\_5\_FINAL/bochs trunk/bochs  
 svn diff tags/REL\_2\_5\_FINAL/bochs branches/REL\_2\_5  
 etc.

All bugfix-only releases after the final release should be created from the REL\_2\_5 branch. Now you can start building packages with the sources from the created release tag.

4.2.2. Anonymous SVN checkout and platform-independent sources

An anonymous SVN checkout from the release tag is the base for all official release packages. Do this checkout from the release tag and specify a not yet existing directory name with the version number as the destination. Then create the source package from this new directory. These steps can be done both on Linux and Windows (Cygwin).

svn co http://svn.code.sf.net/p/bochs/code/tags/REL\_2\_5\_FINAL/bochs bochs-2.5  
 tar czvf bochs-2.5.tar.gz --exclude=.svn bochs-2.5

The source TAR file bochs-2.5.tar.gz is ready for upload.

4.2.3. Building the release on Linux

The RPM will be building using the configuration in .conf.linux with a few parameters from **build/redhat/make-rpm**. Make any last minute changes to .conf.linux. Any changes will go into the source RPM. The DLX Linux demo package will be downloaded from the Bochs website to the Bochs root directory if it is not already present there.

./build/redhat/make-rpm | tee ../build.txt

This produces two rpm files in the current directory. Test and upload.

4.2.4. Building the release on win32

These instructions require cygwin and MSVC++. Use the Bochs sources from the SVN checkout or unpack the TAR file.

In Cygwin:

sh .conf.win32-vcpp # runs configure  
 make win32\_snap # unzip workspace, make a win32 source ZIP

The source ZIP is present in the parent directory of the Bochs root and now ready for upload. To build the binary package, copy it to a Windows machine, if necessary.

Open up Visual C++ and load the workspace file Bochs.sln. Check the Build:Set Active Project Configuration is set the way you want it. For releases I use "Win32 Release".

To create "bochsdbg.exe" with Bochs debugger support, manually change these lines in config.h to turn on the debugger and the enhanced debugger gui.

#define BX\_DEBUGGER 1  
#define BX\_DISASM 1  
#define BX\_DEBUGGER\_GUI 1

One of the optimization features must be turned off, since it is currently not compatible with the debugger.

#define BX\_SUPPORT\_HANDLERS\_CHAINING\_SPEEDUPS 0

VC++ will rebuild Bochs with debugger and overwrite bochs.exe. To avoid trashing the non-debug version, move it out of the way while the debugger version is being built. Then rename the debugger version to bochsdbg.exe.

cd obj-release  
 mv bochs.exe bochs-normal.exe  
 (build again with BX\_DEBUGGER=1 this time)  
 mv bochs.exe bochsdbg.exe  
 mv bochs-normal.exe bochs.exe

Do make *install\_win32* into the NSIS folder in the Bochs source tree:

make install\_win32 INSTDIR=./build/win32/nsis/bochs-2.5

This downloads and unpacks both the DLX Linux demo and the HTML docs from the Bochs website, copies all the files into *./build/win32/nsis/bochs-2.5* and then creates a binary ZIP file in the NSIS folder.

Now make the NSIS installer package (the current script is known to work with NSIS 2.44)

cd build/win32/nsis  
 make

That gives an installer called Bochs-2.5.exe. Test and upload it.

4.2.5. Creating a file release and uploading files on SF

When you are ready with creating release packages you have to upload them using the SF file manager feature. Create a subdirectory with the version number in the bochs directory. Point the download destination to the new directory and start uploading packages. The top of the CHANGES file should be used as the release notes. After setting up the file properties the new release is ready for download.

After having all files set up in the download area, don't forget to post an announcement containing a brief summary of changes to the bochs-announce mailing list and the "Project News" section on SF.

|  |  |  |
| --- | --- | --- |
| [Prev](http://docs.google.com/coding.html) | [Home](http://docs.google.com/index.html) | [Next](http://docs.google.com/webmastering.html) |
| Coding | [Up](http://docs.google.com/coding.html) | Webmastering |