Mingw-w64 and Win-builds.org - Building for Windows

Adrien Nader

February 2, 2014

Section outline

History and motivations

- Started at OneVision by Kaï Tietz and Roland Schwingel to be able to access more than 2GB of memory
- Required changes to mingw.org, gcc, binutils
- Code donated to Kai Tietz and then upstreamed
- except for mingw.org changes (politics), thus mingw-w64
- Projects is hosted at mingw-w64.sourceforge.net

Philosophy and differences with mingw.org

- Headers are built from MSDN too
- But information is also obtained through reverse-engineering (helps when MSDN is insufficient or wrong).
- Everything is done upstream: GCC, binutils, libtool, cygwin, ...
- Active (and welcoming) community and mailing-lists where you're allowed to bottom-post
- Trying to match MSVC's C++ ABI (getting closer)

What comes with a mingw-w64 tarball

- Headers, including information for Windows 8
- Some headers come from Wine (DirectX) or ReactOS (Driver Development Kit)
- C99, C11 support libraries: correct printf and scanf, libm (also faster than MS'), functions that are mandated by standards and missing in msvcrt.dll;
- Support libraries like winpthreads, to replace pthreads-win32

Environments to build with mingw-w64

- Cross-compilation from Linux or from a POSIX-like env on Windows
- Cygwin
- MSYS, a fork of cygwin's code from more than a decade ago; only goal is to run ./configure
- MSYS2, same but forked cygwin less than a year ago
- UWIN, similar to Cygwin but not the same?
- Native compilation possible too but you get no sh.exe; makes the most sense from an IDE

Some binary toolchain and package providers

- OpenSuse: Linux, good support, has existed for a long time, has many packages
- Fedora: Linux, good support and many packages too
- MXE: Unix, source-based, many packages, watch out for static linking of LGPL libraries though
- Rubenvb (retired, for now): some "exotic" builds with various options, clang and maybe soon LLVM.
- Cygwin
- Mingw-builds
- coApp: Windows, a few dozen packages
- Win-builds: super awesome

Section outline

Project goals

- Reduce work for other developers, ease deployment
- Smaller, faster and more up-to-date binaries
- Help bridge gap between Windows and a FOSS kernel: drop proprietary components, use free libraries, get portability for free

Current status

- Runs and builds on GNU/Linux.
- Runs from Windows XP (please, let this OS die) to Server 2012.
- Around 60 library packages.
- Architecture and infrastructure now mostly stable.
- Version 1.3 released early January 2014.
- Version 1.4 planned for release in 2 to 3 months.

Available compilers

- C, C++ with C11 and C++11 support, Lua (not lua-jit)
- Ada/Fortran/Perl/Ruby not tried yet: which applications should they be tested against?
- ObjC not kept in the current release for the same reason
- GCJ (Java) and ObjC GC support are dying! GCC's internal copy of Boehm's GC out-of-date for Windows 64 support
- OCaml as a cross-compiler (patches still being upstreamed)
- Python still requires MSVC to build AFAIU

Available libraries

- GTK+ (and its deps), 2.x for now and limited to 32b
- Enlightenment libraries
- libogg, theora, vorbis
- Qt missing (thanks to qmake); will be worked on in the coming weeks
- xz, ffmpeg (still requires more codecs), x264
- Several various more libraries

Planned features - short term

- More packages
- Fill-in inter-package dependencies
- Wine-based testsuite for regression; something similar with ReactOS?
- Smaller installer (currently $<$1MB) and self-contained .exe
- GUI for the installation

Planned features - longer term

- Package manager as an installer: `cat yypkg.exe $payload > payload_install.exe`
- More packages again
- Signed packages
- Wine on Windows :)
- Cross-compilers to Linux running on Windows

Pointers to start using win-builds

- Documentation on win-builds.org
- Trying to dogfood doc regularly
- When using on Linux, prefer http://win-builds.org/1.4-dev1
- On Windows, use 1.3.x; it has nice howtos and installation scripts

How to get involved

- Get it, use it and provide feedback
- Build your own code, regularly if possible
- Provide new packages
- Test, on Windows too
- Test pre-releases too (and on Windows too!)
- Link to our websites to increase visibility

Users so far (according to Apache stats)

- Anywhere from 1000 to 2000 installations during January
- 30% to 50% of people download the .zip file but don't run any installation script
- Around 3% of installations are from Linux, around 5% of installations are from Cygwin
- Getting feedback from around 0.1% of users
- Most visitors use recent versions of Windows and browsers
- Possibly as many as 90% of users running 64b
- Next yypkg version will help collect OS version, bitness and environment (msys, cywin, native) stats

Section outline

Yypkg, the package manager

- Stands for Yellow Yeti ... (and "Yahoo" at first but that's trademarked)
- Written from scratch
- Most of the work is: `tar xfv > list && xargs rm > list`
- A few ad-hoc additions, mostly symlink fallback
- 2k lines of OCaml: it compiles to native code and has a solid Windows port

Build scripts

- Scripts from Slackware Linux and its community
- Plain shell scripts
- Very easy to customize; "porting" takes less than one minute
- when there's no issue with the Windows support, no weird build system and there is cross-compilation support

Light maintenance you can do if your software is already packaged

- If it is already packaged, get logs at http://win-builds.org/1.3.0/logs and fix your warnings.
- Scary stats: 6525 warnings when building 32 bits, 9384 warnings for 64 bits! LLP64 issues?
- Check your own code: fetch win-builds sources and run `git grep '!PATCH'` (starting tomorrow).

Some (common) issues

- No cross-compilation support in the build system.
- Use of `libpng-config` (e.g.) which lacks cross-compilation support instead of `pkg-config`.
- Missing ".exe" extension (openssl and expat).
- Missing includes: libsoup required `#include <string.h>` for strchr().
- Several files which names only differ in case: `SSL.3.gz` and `ssl.3.gz` (openssl).
- DLL files have to go into `$(bindir)`, not `$(libdir)`.

Questions?

- Mingw-w64: mingw-w64.sf.net, sourceforge's bug tracker, source repository and mailing-lists
- Win-builds: win-builds.org, bug tracker at win-builds.org/bugs (ask for the packages you want there)

LLVM

- LLVM has a Windows port too.
- We found out about it the same way everyone else did.
- Bit frustrating especially considering there's a large project overlap and license often shouldn't be an issue.
- Uses MSVC's headers which are non-free.
- LLVM actually tries to be a drop-in replacement fro MSVC's command-line tools and would still be called from Visual Studio.