Windows toolchains for compiling Python extension with MingW-W64

Four dedicated native MingW-W64 based toolchains for 32 bit and 64 bit architectures. The main objective of these toolchains is the usage as a convenient build tool for Python extensions usable with the standard MSVC based Windows Python distributions. Supported Python versions are: 2.6, 2.7, 3.2, 3.3, 3.4.

History

2015-04

- upgrade to OpenBLAS-0.2.14
- revised patches for numpy-1.9.2 and scipy-0.15.1 for the MingW-W64 OpenBLAS builds
- upgrade of the test wheels for numpy-1.9.2 and scipy-0.15.1 uploaded on binstar

2015-01

- use of `mingwpy` as package name
- upgrade to OpenBLAS-0.2.13
- added `-DMS_WIN64` to the specs for the 64bit builds
- added `libgcc_eh.a` stub library to prevent link errors under certain circumstances

2014-11

- upgrade to GCC version 4.9.2
- use of MingW-W64 trunk
- mingw-w64-crt FPU control word patch for issue5194
- more conservative choice of the threading and exception model
- toolchains are now part of the Winpython distribution, see release 2014-12 follow-up

2014-07

- moved from google code to bitbucket

Todo

- remaining test failures during numpy.test() and scipy.test() runs are due to to unresolved MingW-W64 math library issues
- deploy mingwpy as wheel and conda package on binstar
- move to mingwpy on github

download

mingwpy x86-64 toolchain (64 bit)

python2.6-3.2: mingwpy_amd64_vc90.tar.xz or mingwpy_amd64_vc90.7z
python3.3-3.4: mingwpy_amd64_vc100.tar.xz or mingwpy_amd64_vc100.7z

mingwpy i686 toolchain (32 bit)

python2.6-3.2: mingwpy_win32_vc90.tar.xz or mingwpy_win32_vc90.7z
python3.3-3.4: mingwpy_win32_vc100.tar.xz or mingwpy_win32_vc100.7z

features

- gcc-4.9.2 build with patched mingw-build scripts
- language support: gfortran, gcc, g++, lto
- statically linked gcc runtimes for easy deployment of binary extensions
- uses MingW-W64 trunk
- win32 thread model configuration
- pthread support with winpthreads library (MIT license)
- seh structured exception handling model for C++ (64bit)
- sjlj exception handling model for C++ (32 bit)
- ensure correct *exclusive* MSVC runtime linkage
- automated linkage of manifest resources to binaries
- `ftime` patches to support OpenMP with `MSVCR90` or `MSVCR100` runtimes
- patches for MSVC compatible default FPU control word
- OpenBLAS-0.2.14 BLAS/LAPACK library included

OpenBLAS-0.2.14

- OpenBLAS-0.2.14 development release fb02cb0
- BLAS and Lapack routines with thread support
- compiled with `DYNAMIC_ARCH=1` for CPU runtime detection of optimized BLAS kernels
- the 32 bit OpenBLAS binary is compiled for modern CPU's and *doesn't work on legacy hardware without SSE2 support* (non-SSE2 targets removed from build)

Download and Install

1. download and install the appropriate toolchain (s. above) for your python installation and make sure it is added to your `PATH` environment variable.
2. download the appropriate import libraries and copy the import libraries into the `libs` folder of your Python installation:

- libpython-cp26-none-win_amd64.7z for Python-2.6 64bit
- libpython-cp26-none-win32.7z for Python-2.6 32bit
- libpython-cp27-none-win_amd64.7z for Python-2.7 64bit
- libpython-cp27-none-win32.7z for Python-2.7 32bit
- libpython-cp32-none-win_amd64.7z for Python-3.2 64bit
- libpython-cp32-none-win32.7z for Python-3.2 32bit
- libpython-cp33-none-win_amd64.7z for Python-3.3 64bit
- libpython-cp33-none-win32.7z for Python-3.3 32bit
- libpython-cp34-none-win_amd64.7z for Python-3.4 64bit
- libpython-cp34-none-win32.7z for Python-3.4 32bit