

Setting up a 32/64 bit GCC Build Environment for Windows

These documents demonstrate a C/C++ software build environment for Windows, tested on Windows 8 Professional 64-bit. The build environment is capable of building both 64-bit and 32-bit software, although some manual editing of configuration files is required to switch between the two. With some fairly obvious changes, these instructions will also work for building 32-bit software using Windows XP 32-bit.

The suggested build environment includes

- 7-zip file archive tool
- Python 2.7.5 scripting language
- MSYS, from the official MinGW project
- MinGW-w64 GCC 4.7.1 compiler suite, from TDM-GCC
- gedit 2.30.1 text editor

These steps were completed by a user with administrator privileges. It is known that the Python, MinGW etc can be installed by non-administrator users, but we haven't yet tested that scenario, unfortunately. Problems may potentially arise in relation to 7-zip, gedit, NSIS or Subversion if administrator privileges are not available (but that's just guessing).

It's worth mentioning here that the 64-bit architecture (<http://en.wikipedia.org/wiki/X86-64>) for Windows that we're talking about here is alternatively referred to as x64, x64_64 and amd64. These terms all seem to be used a bit interchangeably.

First steps

- Download and install the 64-bit version of 7-zip. We used the 64-bit version, 7z920-x64.msi (<http://downloads.sourceforge.net/sevenzip/7z920-x64.msi>).
- Download and install the latest Python 2.7 release. We used the 64-bit version if 2.7.5, python-2.7.5.amd64.msi (<http://www.python.org/ftp/python/2.7.5/python-2.7.5.amd64.msi>) from this page (<http://www.python.org/download/>). Select to install for all users, and select the default installation location, c:\Python27 (although you can use another location if you already have 32-bit Python installed, but be careful to change PATH references further below if necessary).
 - Note: it is permissible to install both 32-bit and 64-bit Python in the same system, and it works fine, if you're careful about setting you PATH. If you also want to build against 32-bit Python, we suggest you download python-2.7.5.msi (<http://www.python.org/ftp/python/2.7.5/python-2.7.5.msi>) and install it in c:\Python27_32.
- Download and install the gedit (<https://projects.gnome.org/gedit/>) text editor for Windows. We used this package: gedit-setup-2.30.1-1.exe (<http://ftp.gnome.org/pub/GNOME/binaries/win32/gedit/2.30/gedit-setup-2.30.1-1.exe>) (32-bit).
 - To ensure that Windows knows that gedit is a text editor, use Windows Explorer to navigate to c:\Program Files (x86)\gedit\share\doc and right-click on README, and select Open. In the dialog, click Browse, then select c:\Program Files (x86)\gedit\bin\gedit.exe, and click OK to open the file. This will ensure that gedit appears in the list of editors for other text files you want to open in future.
 - Note: be aware that some other installers may modify your system PATH in a way that causes gedit to break. If you start to get errors when launching gedit, this is probably what has happened.

MSYS

MinGW is a minimal GNU/GCC environment for Windows. MSYS is a command-line environment based on MinGW that mimics the Linux environment sufficiently well to allow a large portion of Linux-compatible software to also be compilable under Windows. There are many different combinations of ways to set up this environment. And things are made quite a bit

more complicated by the fact that the official MinGW project does not support the 64-bit fork MinGW-w64, and as yet (Aug 2013) there is no nicely-integrated environment that supports installing everything automatically. Based on our experience in particular with difficulties in building recent versions of IPOPT, this is therefore our current preferred approach:

- Download mingw-get-setup.exe (<http://sourceforge.net/projects/mingw/files/Installer/mingw-get-setup.exe/download>) from the official MinGW (32-bit) (<http://www.mingw.org/>) project page (we used version 0.6.2-beta-20131004-1), and run it:
 - Select to install MinGW to the directory c:\MinGW\32 (this is **not** the default location)
 - Select **not** to install the graphical user interface
- Open gedit and edit the file c:\mingw\32\var\lib\mingw-get\data\profile.xml and replace the entire contents of the file with the following text:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

```
<profile project="MinGW" application="mingw-get">
```

```
<repository uri="http://prdownloads.sourceforge.net/mingw/%F.xml.lzma?download"/>
```

```
<system-map id="default">
```

```
<sysroot subsystem="mingw32" path="%R" />
```

```
<sysroot subsystem="MSYS" path="%R/../msys" />
```

```
</system-map>
```

```
</profile>
```

- From the Windows 'Start' menu open a command prompt window (<http://windows.microsoft.com/en-au/windows-vista/open-a-command-prompt-window>) and run the following, which will take a few minutes:

```
c:\mingw\32\bin\mingw-get install msys-core msys-base msys-vim msys-wget msys-patch msys-flex msys-bison
```

- Using gedit, create the file c:\mingw\msys\etc\fstab with the following content:

```
c:/mingw/32 /mingw
```

- Using Windows Explorer, create a shortcut to the file c:\mingw\msys\msys.bat and place the shortcut on the desktop. You may wish to set the icon of the shortcut to the icon file c:\mingw\msys\msys.ico.

Switchable 32- and 64-bit modes

Our setup will be a bit of a hack of the standard MinGW environment. We use the official MinGW project to provide the MSYS environment, but then we use the MinGW-w64 project (as below) to provide the actual compiler, since we want 64-bit support. Our file structure will be

- c:\ul>- **mingw** (we install *everything* under this single top-level directory)
 - **64** (the 64-bit compiler and all our 64-bit compiled-and-installed libraries will live here)
 - **32** (the mingw-get binary will already be inside here, as well as the 32-bit compiler if we install it later)

- **msys** (the MSYS environment, and loads of useful non-compiler tools can be installed here, but they shouldn't have any binary dependencies in c:\mingw\64 or c:\mingw\32, or things will get broken!)

In principle, all we need to do to switch between the 64-bit and 32-bit build environments is to edit the text file /etc/fstab (changing '32' to '64' or vice versa), then reload MSYS. In other words, if you want to now switch to 64-bit mode, you would modify /etc/fstab to read

```
c:/mingw/64 /mingw
```

After making this change, be sure to close and re-open your MinGW/MSYS command window (re-open it using the MSYS icon that you created on your desktop).

MinGW-w64

MinGW-w64 (Minimalist GNU for Windows, 64-bit) is a fork of the earlier MinGW GCC (GNU Compiler Collection: C, C++ and Fortran compilers and associated tools). MinGW-w-64 was carefully extended to include support for 64-bit Windows. Despite the name, MinGW-w64 both runs on and builds binaries for both 32-bit and 64-bit Windows, but you need to make sure you download the right package according to which platform you are running on, and which platform you want to target.

- Download the TDM-GCC 'MinGW-w64 sjlj' installer for from this page (<http://tdm-gcc.tdragon.net/download>). We used tdm64-gcc-4.7.1-3.exe (<http://sourceforge.net/projects/tdm-gcc/files/TDM-GCC%20Installer/tdm64-gcc-4.7.1-3.exe/download>).
- Install the TDM-GCC package into the directory c:\mingw\64, selecting to install **all** components. Select **not** to add MinGW-w64 to the PATH.
- After the installer completes, use the Start menu, right-click on 'Computer', select 'Properties' then 'Advanced Systems Settings' then 'Advanced' tab, click button 'Environment variables'. Under system variables, edit the variable named PATH and from then end of the path, *remove* c:\mingw\64\bin. We *don't* want this in the path because it messes with gedit, and will also cause confusion when running 32-bit compilers.
- Open the MSYS shell (make sure you are in 64-bit mode, as mentioned above) by double-clicking your new desktop shortcut, and type gcc --version. You should see

```
john@win7_64 ~$ gcc --version
gcc.exe (tdm64-1) 4.7.1
Copyright (C) 2012 Free Software Foundation, Inc. This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

- In gedit, open (or create) the file c:\mingw\msys\home\yourusername\profile (no file extension). In that file, paste in the following text, and save the resulting file.

```
#!/bin/bash# we use the reported architecture of the 'gcc' in our path to # determine which Python and other utilities we will be
using._arch=`gcc -dumpmachine`#echo "ARCH=$_arch"if grep "^x86_64-" <<< "$_arch" >/dev/null ; then          echo
"MINGW 64 BIT BUILD ENVIRONMENT"          _pydir="/c/Python27"else          echo "MINGW 32 BIT BUILD
ENVIRONMENT"          _pydir="/c/Python27_32"fi export PATH=$PATH:$pydir:$pydir/Scripts # note that mingw-get
will still install all its stuff to c:\mingw\32,# because of the contents of its profile.xml file.alias mingw-
get="/c/mingw/32/bin/mingw-get"
```

TODO: add instructions for also installing the 32-bit compilers, and instructions for getting things working on a 32-bit Windows system.

Note: the above TDM 64-bit installer does not install a fortran compiler, but a fortran compiler is required for building ASCEND. The alternative appears to be to use the latest tdm-gcc-webdl.exe (<http://sourceforge.net/projects/tdm-gcc/files/TDM-GCC%20Installer/tdm-gcc-webdl.exe/download>) downloader-installer. Select all options, but select not to install to PATH.

MinGW 32-bit compilers (*optional*)

For the 32-bit compilers, we will use the standard MinGW distribution. In MSYS set to 32-bit mode (see above), type:

```
mingw-get install gcc g++ gfortran
```

The default MinGW 32-bit compilers use win32 threading model and dwarf2 threading.

To test the installed compiler, run

```
gcc -dumpmachinegcc -v
```

And you should see output that starts with mingw32, confirming you have the 32-bit compiler running.

Note, due to current bug 1950 (<http://sourceforge.net/p/mingw/bugs/1950/>) in MinGW (Dec 2013), you may need to downgrade 'msys-core-bin' package to version 1.0.17 to avoid problems with parallel builds, eg make -j4. The necessary command which should probably be run from an standard Windows command prompt, is

```
cd c:\mingw\32\binmingw-get upgrade msys-core-bin=1.0.17-1
```

Example of building a simple program

If everything is working, you should be able to enter the following simple C program into gedit, and save it into

```
c:\mingw\msys\home\yourusername\test.c:
```

```
#include <stdio.h>int main(){ printf("Hello, world!\n"); return 0;}
```

Then compile the file and run it by typing

```
gcc test.c./a.exe
```

And you should see

```
Hello, world!
```

Build Environment for GCC compiler for other Operating System

GCC Normally inbuilt for Unix and Linux comes with Operating system.

The **GNU Compiler Collection (GCC)** is a compiler system produced by the GNU Project supporting various programming languages. GCC is a key component of the GNU toolchain. The Free Software Foundation (FSF) distributes GCC under the GNU General Public License (GNU GPL). GCC has played an important role in the growth of free software, as both a tool and an example.

Originally named the **GNU C Compiler**, when it only handled the C programming language, GCC 1.0 was released in 1987.^[1] It was extended to compile C++ in December of that year. Front ends were later developed for Objective-C, Objective-C++, Fortran, Java, Ada, and Go among others.^[5]

GCC has been ported to a wide variety of processor architectures, and is widely deployed as a tool in the development of both free and proprietary software. GCC is also available for most embedded platforms including Symbian (called *gcce*), AMCC, and Freescale Power Architecture-based chips. The compiler can target a wide variety of platforms, including video game consoles such as the PlayStation 2 and Dreamcast.

As well as being the official compiler of the GNU operating system, GCC has been adopted as the standard compiler by many other modern Unix-like computer operating systems, including Linux and the BSD family, although FreeBSD and OS X have moved to the LLVM system.^[10] Versions are also available for Microsoft Windows and other operating systems; GCC can compile code for Android and iOS.