Building a cross-compiler (Cygwin)

<https://x.cygwin.com/docs/cg/cross.html>

Table of Contents

Obtaining binutils and GCC source

Obtaining Cygwin headers and libraries

Building binutils

Building GCC

**Here be dragons**: Building a cross-compiler is not often tested. You will likely encounter problems following these instructions, and require an in-depth understanding of what you are doing in order to fix those problems. These instructions are meant to be a template to be completed with your own understanding, rather than a recipe to be followed blindly. Reports to the mailing list that you followed these instructions, got an error message and are now stuck, will be ignored with harsh, uncaring indifference.

Note: Even once you have built your cross-compiler, there are a whole new class of problems that can happen when cross-compiling that are simply not an issue when building on the target. For example: you will need to somehow make the dependencies of your program available to the cross-compiler, hardly anything uses HOST\_EXEEXT correctly, etc.

This mailing list thread discusses some of the issues with building a cross-compiler, and provides an example script. It also discusses cross-compiling the Cygwin DLL itself.

You will want to read carefully the GCC installation guide (http://gcc.gnu.org/install/).

These instructions will assume you have chosen a suitable working directory, e.g. ~/cygwin/

To build a minimal cross-toolchain, we need to build binutils (for the cross-assembler and cross-linker) and GCC (for the cross-compiler).

# Obtaining binutils and GCC source

binutils and GCC releases that are known to work for Cygwin are distributed with source code by the Cygwin project. These may contain patches against the stock upstream release required to build or function correctly on Cygwin, therefore, it is highly recommended that you obtain the binutils and GCC sources from the Cygwin mirror network.

Follow these steps to download Cygwin sources:

Create a directory to store the binutils and GCC sources in, such as ~/cygwin/src/

The URL listed for your nearest mirror site should take you to the cygwin/ directory on the mirror

Download the following files from cygwin/release/, saving them to ~/cygwin/src/.

binutils/binutils-2.20.51-2-src.tar.bz2

gcc4/gcc4-4.5.0-1-src.tar.bz2

user@crosshost ~ $ mkdir -p ~/cygwin/src

user@crosshost ~ $ cd ~/cygwin/src

user@crosshost ~/cygwin/src $ wget $YOUR\_MIRROR/release/binutils/binutils-2.20.51-2-src.tar.bz2

user@crosshost ~/cygwin/src $ wget $YOUR\_MIRROR/release/gcc4/gcc4-4.5.0-1-src.tar.bz2

Por ejemplo, podemos usar:

<http://mirrors.kernel.org/sourceware/cygwin/x86/release/>

<http://mirrors.kernel.org/sourceware/cygwin/x86/release/libiconv/>

Currently GCC is provided as cygport package. You will want to examine closely the .cygport file contained in the source package which shows how GCC is configured and built natively.

You will want to read carefully the cygwin-specific READMEs installed by the corresponding binary packages into /usr/share/doc/Cygwin/, which contain important information and build instructions

# Obtaining Cygwin headers and libraries

https://x.cygwin.com/docs/cg/cross-obtaining-cygwin-headers-and-libs.html

The usual technique for building GCC cross-compilers is to:

build binutils

build a bootstrap compiler (--without-headers --enable-languages=c) that will only be used to build the C runtime library.

use the bootstrap compiler to build the C runtime library.

rebuild the final compiler, including internal libraries that need the target-specific C runtime library in order to be compiled properly.

Unfortunately, this technique cannot be applied building a cross-compiler for Cygwin, not least due to the use of C++ code in winsup/.

The simplest method of escaping from this chicken-and-egg situation is to make the Cygwin headers and libraries available at the time of building the cross-compiler, by installing them from the Cygwin binary packages containing those headers and libraries.

Headers and libraries from the following packages are required:

cygwin (needed for building libgcc)

win32api (needed for building libgcc)

iconv (needed for building libstdc++)

user@crosshost ~/cygwin/src $ wget $YOUR\_MIRROR/release/cygwin/cygwin-1.7.7-1.tar.bz2

user@crosshost ~/cygwin/src $ wget $YOUR\_MIRROR/release/w32api/w32api-3.15-1.tar.bz2

user@crosshost ~/cygwin/src $ wget $YOUR\_MIRROR/release/libiconv/libiconv-1.13-10.tar.bz2

user@crosshost ~/cygwin/src $ cd ~/cygwin

user@crosshost ~/cygwin $ tar xjf src/cygwin-1.7.7-1.tar.bz2 usr/include usr/lib

user@crosshost ~/cygwin $ tar xjf src/w32api-3.15-1.tar.bz2 usr/include usr/lib

user@crosshost ~/cygwin $ tar xjf src/libiconv-1.13-10.tar.bz2 usr/include usr/lib

user@crosshost ~/cygwin $ ln -s ../usr/include/ i686-pc-cygwin/include

user@crosshost ~/cygwin $ ln -s ../usr/lib/ i686-pc-cygwin/lib

Alternatively, these headers and libraries can be obtained by copying the contents of the /usr/lib directory and /usr/include directory of a Cygwin host, to the ~/cygwin/i686-pc-cygwin directory on your build host. Ensure the method you use to copy these files preserves symlinks.