**Building HDF5 on 64-bit Windows 10 for use within the Gudrun suite of programs**

A K Soper 3rd October 2019

## Introduction

HDF5 is a general purpose data storage package which efficiently stores numerical data in a compact form in a manner that is easily and quickly accessible. It has a multitude of uses and can cope with huge data files. Unfortunately, that complexity means it can be a bit of a nightmare to build. Pre-built binaries can be downloaded for a multitude of platforms, including Windows, but those downloads may have problems, unless you are very careful:-

1) HDF5 has to be built with the same version of C and Fortran compilers that you will use for your own program, otherwise it may not to link properly to your program.

2) Gudrun is compiled in Fortran 95, but most of the HDF5 downloads do not include the Fortran interface,.

3) The MinGW set of compilers are freely available for Windows operating systems, but current versions of HDF5 do not support MinGW. Mostly it is the Intel compiler suite operating within Visual Studio that is supported by HDF5 for Windows, neither of which are free, and normally require institutional access (unless you are rich!).

4) My experience is that once built and properly linked to, HDF5 runs flawlessly.

In order to progress, it first will be noted that the demands on HDF5 placed by Gudrun (and most neutron or x-ray data analysis programs) are relatively light and only a few key features of the package are required. This means an older version of HDF5 is quite sufficient for our purposes. The last version of HDF5 for which MinGW compilers were supported is 1.8.4-patch1. This, in addition, requires a few changes to the makefiles in the ‘test’ and ‘perform’ folders, but otherwise appears to build and test fine. I have seen claims that later versions of HDF5 can be built with the MinGW compilers, but my own experience was negative – numerous fatal compile errors occurred and I am not enough of a wizard on this to know how to figure out what these were caused by. Cmake is also supposed to help the build process, but for me, it just produced another stream of undecipherable error messages, so I gave up.

## Step 1 – Obtain MinGW

MinGW – which stands for “Minimalist GNU for Windows” - is a package of Linux-like commands that will run on Windows. It also contains the set of GNU compilers, but these are 32-bit and so are not useful for the present purposes. It can be downloaded from “mingw.org”, from which an installer .exe file can be downloaded.

1. Run the installer program
2. Use the default installation folder. It should appear as C:\MinGW.
3. When downloading is complete, press “Continue”. The installation manager should appear.
4. Click on “Basic Setup”, then click on all the items listed on the right hand panel and “mark for installation”
5. Finally click on the “Installation” tab and “Apply Changes”, and then “Apply”. Wait for installation to complete.
6. Press “Close” when complete.

**Step 2 – Obtain MinGW-w64**

This package contains the 64-bit C and Fortran compilers. The installer can be obtained from "sourceforge.net/projects/mingw-w64".

1. Run the installer
2. Make sure you click "x86_64" under the Settings page - Architecture tab (other tabs can be left unchanged on this page).
3. On the Installation folder, replace the given Destination folder with: "C:\MinGW-64" (or something similar without white space)
4. Press "Next" and wait for installation to complete. Press "Next" and "Finish".

**Note**: in the past I have used the TDM-GCC-64 compilers, but the current version, 5.1.0, appears to have a bug: in a Fortran program, if you attempt to open an existing file with the "open" command, the program crashes with a segmentation fault. Although this is a known bug, apparently it hasn't been fixed so far!

**Step 3 - Set the PATH environment variable to include MinGW-w64 and msys**

In order for the compilers to run, the system needs to know where they are installed. This is controlled by the %PATH% environment variable. This can be set permanently for your computer using the Settings – System menu. However I don't recommend doing this if you are likely to be installing other programs, since you can become very confused as to which version of a program you are actually running, depending on the sequence of folders that are specified in %PATH%. Since compilation is a step you are unlikely to do very often, I prefer to set a local %PATH% environment variable which includes only the folders that you need. I have a small batch command file to do this ("setupMinGW-64.bat") that looks like this:-

```
echo off
set tdmgcc=\MinGW-64\mingw64
set msys=\MinGW\msys\1.0
set tdmgccbin=%tdmgcc%\bin
set tdmgccinclude=%tdmgcc%\include
set tdmgcclib=%tdmgcc%\lib
set msysbin=%msys%\bin
set msyslib=%msys%\lib
if not defined pathsave set pathsave=%PATH%
set path=%tdmgccbin%;%tdmgccinclude%;%tdmgcclib%;%msysbin%;%msyslib%;%pathsave%
echo Setting PATH to:- %PATH%
echo on
```

This is run once at the beginning of any build sequence. (Obviously you will need to change the second and third lines if you have installed in different folders.)

**Step 4 - Obtain and build zlib**

Before HDF5 can be built, it is necessary to make sure an operating version of the zlib libraries is available. This is essential for the so-called "Nexus" data format, since the data are compressed in this format. Zlib is often available with compiler distributions, but once again, in order for it to work with HDF5 and your program zlib must be built with the same set of C compilers. The version that worked for me is zlib-1.2.7.tar.gz, obtainable from "zlib.net/fossils/"

1. In a suitable folder (the path name to this folder should have no white space) unpack the download in a Command Prompt window:-
   :
   gzip -d zlib-1.2.7.tar.gz
   tar -xf zlib-1.2.7.tar

   This should have created a folder called “zlib.1.2.7”

2. Run the following commands in the Command Prompt:-

```
call setupMinGW-64.bat
cd zlib-1.2.7
make -fwin32/Makefile.gcc clean
make -fwin32/Makefile.gcc
copy /v /-y zlib.h %tdmgccinclude%
copy /v /-y zconf.h %tdmgccinclude%
copy /v /-y libz.a %tdmgcclib%
copy /v /-y libz.dll.a %tdmgcclib%
copy /v /-y zlib1.dll %tdmgccbin%
cd ..
```

These commands can also be put into a .bat file (“buildzlib.bat”) in case they need to be repeated. However my experience is that this normally runs quickly without any glitches or error messages. Note the copy statements at the end: these are to ensure the necessary libraries and files are in the correct places to be found when building HDF5 and Gudrun.

**Step 5 – Obtain HDF5**

hdf5-1.8.4-patch1.tar.gz can be obtained from “support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.8/hdf5-1.8.4-patch1/src/”. As before (assuming you have already run “setupMinGW-64.bat”) run gzip then tar on this file:-

```
gzip -d hdf5-1.8.4-patch1.tar.gz
tar -xf hdf5-1.8.4-patch1.tar
```

which should create the folder “hdf5-1.8.4-patch1”

**Step 6 – Modify the input files for MinGW**

Open the folder “hdf5-1.8.4-patch1”, go to the folder “release-docs” and open the file “INSTALL_MinGW.txt”. Under section 5 you will find the following instructions, copied verbatim, which should be performed as instructed:-

To remove the tests, open the given 'Makefile.in' and edit the line begining with "TEST_SCRIPT = " to remove the test script. For example, to remove the "testerror.sh" from ./test/Makefile.in:

1) Open ./test/Makefile.in

2) Find the line "TEST_SCRIPT = $(top_srcdir)/test/testerror.sh"

3) Change it to simply read "TEST_SCRIPT =", and save.

Do this for the following Makefiles and tests:

- ./test/Makefile.in: "testerror.sh"

- ./tools/h5dump/Makefile.in: "testh5dump.sh" and "testh5dumpxml.sh"

- ./tools/h5diff/Makefile.in: "testh5diff.sh"
- ./tools/misc/Makefile.in: "testh5mkgrp.sh"
- ./tools/h5copy/Makefile.in: "testh5copy.sh"
- ./tools/h5ls/Makefile.in: "testh5ls.sh"
- ./tools/h5stat/Makefile.in: "testh5stat.sh"

In addition to this you need to remove the test, “zip_perf”, which doesn’t seem to run correctly in this configuration. To do this go to the folder “hdf5-1.8.4-patch1\perform” and edit the file makefile.in.

At line 419 change:-

TEST_PROG = iopipe chunk overhead zip_perf perf_meta h5perf_serial $(BUILD_ALL_PROGS)

to

TEST_PROG = iopipe chunk overhead perf_meta h5perf_serial $(BUILD_ALL_PROGS)

Save this file.

**Note:** I can supply a suitably edited version of this distribution if that would be easier.

**Step 8 – Build HDF5**

This is achieved by running the following commands within a Command Prompt:-

```
call setupMinGW-64.bat
cd hdf5-1.8.4-patch1
set LIBS=-lws2_32
sh configure --disable-hl --prefix=/c/Users/user/Utilities/hdf5files/hdf5 --enable-fortran --disable-shared
make clean
make
make check
make install
```

The folder after the word “--prefix” in the configure command above defines the destination of where the final HDF5 libraries and executables will be installed. If you already have Gudrun installed, then this destination can be the ...\Gudrun\bin\hdf5 folder. The configure, make and make check commands take quite a long time to complete, so be patient! If any command fails, you will need to go back and check that you have followed the above procedures correctly, before proceeding to the next command.

**Step 9 – Install HDF5**

Assuming the previous steps were successful, you now only need to copy the complete hdf5 folder to the ...\Gudrunfiles\bin folder. Alternatively you can install it directly in this location by specifying the ...\Gudrunfiles\bin\hdf5 folder in the configure line in Step 8 above.

I have a couple of example read programs that will check that HDF5 is working correctly for the data Gudrun is likely to read.