2017-11-29-Wed 22:07 USA Central Time (notes by Arnold Trembley)

How to install 64-bit GnuCOBOL for Windows using MSYS2 (MinGW64-bit)

Special thanks to Simon Sobisch who built the "pacman" package and answered numerous questions for me. This procedure will build 64-bit MinGW GnuCOBOL 2.2 on Windows, with gmplib, ncurses, and Oracle Berkeley Database.

step01 - Download MSYS2/MinGW64

go to http://www.msys2.org/ and download "msys2-x86_64-20161025.exe"

step02 - Disable Anti-Virus

Close all web browsers and disable realtime AntiVirus scanning.

step03 - Install MSYS2/MinGW64

Install "msys2-x86_64-20161025.exe" into the default "C:\msys64" folder.
Accept the default start menu shortcut folder "MSYS2 64bit".
(You can actually install to any folder on any drive letter, but for my example I will use the default of "C:\msys64".)
Then click on "run MSYS2 64-bit now" and click on finish.
This should open the bash shell window for you.

step04 - Apply MSYS2 Updates

run "pacman -Syu"
Answer "Y" to "Proceed with installation?"

Wait for the following message:

warning: terminate MSYS2 without returning to shell and check for updates again
warning: for example close your terminal window instead of calling exit

After you see the message above, shut down MSYS2 using the X in the upper right hand corner, or by typing alt-PF4, or by killing the task using Windows Task manager.

You need to kill it even it has tasks still running. I usually had to use Task Manager to completely kill it off.

step05 - Finish Applying MSYS2 Updates

Then restart "MSYS2 MSYS" from "MSYS2 64bit" in the start menu.

run "pacman -Su" (to update MSYS2).

answer "Y" to "proceed with installation?"

When it finishes you may want to make a backup of this folder, so you can restore to this point. If you choose to do so, close MSYS2 (by typing "exit") and make a backup of the "C:\msys64" folder and all its sub-folders.

This folder will be fairly large, about 309 megabytes.

step06 - Download and Install GnuCOBOL 2.2

Then restart "MSYS2 MSYS" from "MSYS2 64-bit" in the start menu. You must have internet access to download the packages, but you should probably leave your Anti-Virus program disabled.

run "pacman -S mingw-w64-i686-gnucobol mingw-w64-x86_64-gnucobol"

```
This installs both 32-bit (i686) and 64-bit GnuCOBOL in MinGW64 MSYS64.
or run "pacman -U mingw-w64-i686-gnucobol mingw-w64-x86_64-gnucobol" to update them.
Use "pacman -R mingw-w64-i686-gnucobol mingw-w64-x86_64-gnucobol" to delete them, if needed.
You can also skip the "w64-i686" version if you only want "w64-x86_64" for 64-bit GnuCOBOL.
You can also run "pacman -Ss gnucobol" to discover what GnuCOBOL versions are available.
```

Answer Y or All to prompts when they appear. This step runs for about 10 minutes.

Once this is done, it should be safe to re-enable your AntiVirus program.

step07 - Test GnuCOBOL in the MinGW64 bash shell

This step is optional. It is for testing GnuCOBOL in the bash shell.

Start "MSYS2 MinGW 32-bit" from "MSYS2 64bit" in the start menu. This will be a bash shell.

Enter "cd /mingw32"

At this point you can run "cobc --info" and "cobcrun --info" to verify this is 32-bit GnuCOBOL.

Enter ". cobenv.sh" (or "source cobenv.sh") to set environment variables. There must be a blank space after the period/full stop in ". cobenv.sh".

Now run a test compile to exe. In my case I used testfunc.cob.

Run "cobc -x -Wall -debug -fnotrunc -t testfunc.lst testfunc.cob" to create testfunc.exe

Note to windows users: you can use "ls" to list files in a bash shell window.

Enter "./testfunc" to run testfunc.exe

Now compile the same program as a dll file:

run "cobc -m -Wall -debug -fnotrunc -t testfunc.lst testfunc.cob" to create testfunc.exe

run "cobcrun TESTFUNC" (note that uppercase letters are required for the COBOL dll name) to run testfunc.dll.

type "exit" to close "MSYS2 MinGW 32-bit"

Start "MSYS2 MinGW 64-bit" from "MSYS2 64bit" in the start menu.

run "cd /mingw64"

The rest of this test is the same commands used for testing \mingw32.

Type "exit" to close "MSYS2 MinGW 64-bit"

step08 - Test GnuCOBOL in Windows cmd.exe

This step is optional. It is for testing GnuCOBOL in Windows cmd.exe.

Open a cmd.exe window and navigate to "C:\msys64\mingw32" as the current directory

Enter "bin\cobenv --verbose" (or “bin\cobenv -v”) This sets the environment variables.

At this point you can run "cobc --info" and "cobcrun --info" to verify this is 32-bit GnuCOBOL.

Now run a test compile to exe. In my case I used testfunc.cob

run "cobc -x -Wall -debug -fnotrunc -t testfunc.lst testfunc.cob" to create testfunc.exe

type "testfunc" to run testfunc.exe

Now compile the same program as a dll file:

run "cobc -m -Wall -debug -fnotrunc -t testfunc.lst testfunc.cob" to create testfunc.exe

run "cobcrun TESTFUNC" (note that uppercase letters are required for the COBOL dll name) to run testfunc.dll.

close the cmd.exe window.

Open a cmd.exe window and navigate to "C:\msys64\mingw64" as the current directory

Enter "bin\cobenv --verbose" (or bin\cobenv -v) This sets the environment variables.

At this point you can run "cobc --info" and "cobcrun --info" to verify this is 64-bit GnuCOBOL.

Run the same tests in the "C:\msys64\mingw64" folder as in the "C:\msys64\mingw32" folder.

close the cmd.exe window.

step09 - Build a 64-bit GnuCOBOL Compiler Folder.

Copy the entire "C:\msys64\mingw64" folder to a new folder. I suggest naming the new folder "C:\GC22B-64" but you can choose any name you like on any drive letter. For my example I will be using "C:\GC22B-64".

This folder is quite large, about 508 megabytes.

Once you create your "C:\GC22B-64" folder, you can strip out a few files to make it smaller.

You can run "rmdir /s C:\GC22B-64\share\doc\db" to remove 88 megabytes of Berkeley Database documentation manuals.

You can run "rmdir /s C:\GC22B-64\share\doc\gettext" to remove 3.5 megabytes of gettext documentation

C:\GC22B-64\share\doc\gettext

Run the following in a .bat or .cmd file:

```
@echo off
echo strip out unneeded GnuCOBOL components
echo.

PAUSE

copy bin\strip* . && copy bin\libiconv* . && strip -p --strip-debug --strip-unneeded bin\*.dll
bin\*.exe lib\*.a && del strip* libiconv*

echo unneeded GnuCOBOL components have been removed
echo.
```

This will shrink the folder down to about 418 megabytes. This folder can be compressed down to about 40 Megabytes using 7-Zip with maximum compression.

NOTE: You can use the same procedure to build a 32-bit GnuCOBOL compiler, just start by copying the "C:\msys64\mingw32" directory. But it is also quite large, about 508 megabytes.

The "cobenv.cmd" file builds slightly different environment variables for 64-bit MinGW GnuCOBOL than my MinGW32 bit builds, because the directory structure is slightly different in MinGW64. But it should still work with OpenCobolIDE 4.7.6.

The 64-bit GnuCOBOL executable programs are much larger than programs compiled with MinGW32 GnuCOBOL.

Here are the typical environment variables created by running “bin\cobenv -v” in C:\msys64\ming64:

```
C:\msys64\mingw64>bin\cobenv -v
Setup GnuCOBOL environment - MinGW
environment saved:
  PATH=C:\ProgramData\Oracle\Java\javapath; (rest skipped)
  COB_LIBRARY_PATH=
  COB_CONFIG_DIR=
  COB_COPY_DIR=
environment set:
  PATH=C:\msys64\mingw64\bin\..\bin;C:\ProgramData\Oracle\Java\javapath; (rest skipped)
  COB_LIBRARY_PATH=C:\msys64\mingw64\bin\..\lib\gnucobol
  COB_CONFIG_DIR=C:\msys64\mingw64\bin\..\share\gnucobol\config
  COB_COPY_DIR=C:\msys64\mingw64\bin\..\share\gnucobol\copy
C:\msys64\mingw64>
```

=====end=======