GnuCOBOL 2.2 [06SEP2017] Build Guide for MinGW 64-bit - Minimalist GNU for Windows

cobc (GnuCOBOL) 2.2.0

Written by Keisuke Nishida, Roger While, Ron Norman, Simon Sobisch, Edward Hart.

This document was prepared by: Arnold J. Trembley (arnold.trembley@att.net)
and last updated Sunday, 24 December 2017.

Special thanks go to **Simon Sobisch**, the GnuCOBOL project leader, who built the "pacman" package and answered numerous questions for me. This procedure will build a 64-bit MinGW GnuCOBOL 2.2 compiler on Windows, with gmplib, ncurses, and Oracle Berkeley Database (for Indexed Sequential file access support).

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 real-time Anti-Virus 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 this to ANY folder on ANY drive letter, but for my example I am using the supplied 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 MSYS2 even it still has tasks still running. I usually had to use Windows Task Manager to completely stop it.

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 subfolders.

This folder will be fairly large, about 309 megabytes.

You can also update a restored MSYS2 folder by running "pacman -Syu".

STEP06 - Download and Install GnuCOBOL 2.2

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 (x86_64) in MSYS2.

Or run "pacman -U mingw-w64-i686-gnucobol mingw-w64-x86_64-gnucobol" to update them.
Or run "pacman -R mingw-w64-i686-gnucobol mingw-w64-x86_64-gnucobol" to delete them.
You can also skip the "i686" version if you only want "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 Anti-Virus program.

GnuCOBOL 2.2 Build Guide for MinGW 64-bit (draft)

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 create an exe. In my case I used a COBOL program named 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

Enter "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"

GnuCOBOL 2.2 Build Guide for MinGW 64-bit (draft)

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 create an exe. In my case I used a COBOL program named 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 32-bit GnuCOBOL.

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

close the cmd.exe window.

GnuCOBOL 2.2 Build Guide for MinGW 64-bit (draft)

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 folder name on any drive letter. For my example I will be using "C:\GC22B-64", but you can choose any name you prefer, in any path.

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

You may want to copy the following “gcshrink.cmd” file into your "C:\GC22B-64" folder and run it to remove unneeded files:

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

PAUSE


echo.
echo STEP01 install strip.exe
xcopy bin\strip.exe .

echo.
echo STEP02 install libiconv-2.dll
xcopy bin\libiconv-2.dll .

echo.
echo STEP03 strip unneeded bin and lib components
strip -p --strip-debug --strip-unneeded bin\*.dll bin\*.exe lib\*.a

echo.
echo STEP04 delete strip.exe and libiconv-2.dll
del strip.exe libiconv-2.dll
```

```
echo.
echo strip.exe completed


rem GOTO :ALLDONE

echo.
echo STEP05 delete \etc folder
rmdir /s /q \etc

echo.
echo STEP06 delete \lib subfolders
rmdir /s /q lib\gettext
rmdir /s /q lib\pkgconfig
rmdir /s /q lib\terminfo

echo.
echo STEP07 delete \share\doc subfolders
rmdir /s /q share\doc\db
rmdir /s /q share\doc\expat
rmdir /s /q share\doc\gettext
rmdir /s /q share\doc\libasprintf
rmdir /s /q share\doc\libiconv
rmdir /s /q share\doc\mpfr

echo.
echo STEP08 delete \share\licenses\ subfolders
rmdir /s /q share\licenses\bzip2
rmdir /s /q share\licenses\db
rmdir /s /q share\licenses\expat
rmdir /s /q share\licenses\gettext
rmdir /s /q share\licenses\headers
rmdir /s /q share\licenses\libiconv
rmdir /s /q share\licenses\libsystre
rmdir /s /q share\licenses\libtre
rmdir /s /q share\licenses\libwinpthread
rmdir /s /q share\licenses\winpthreads
rmdir /s /q share\licenses\bzip2
rmdir /s /q share\licenses\zlib

echo.
echo STEP09 delete \share\ subfolders
rmdir /s /q share\aclocal
rmdir /s /q share\gcc-7.2.0
rmdir /s /q share\gettext
rmdir /s /q share\gettext-0.19.8
rmdir /s /q share\info
rmdir /s /q share\locale
rmdir /s /q share\man
rmdir /s /q share\pkgconfig
rmdir /s /q share\tabset
rmdir /s /q share\terminfo

echo.
```

```
echo step10 delete \bin\bz* components
del /q bin\bz*.*

echo.
echo step11 delete \bin\gettext* components
del /q bin\gettext*.*

echo.
echo unneeded GnuCOBOL components have been removed

:ALLDONE
ECHO.
```

This will shrink the folder down to about 396 megabytes. This folder can be compressed down to about 34.4 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 compared to MinGW32 bit builds, because the directory structure is slightly different in MinGW64. But it should still work with OpenCobolIDE 4.7.6.

The MinGW64 GnuCOBOL executable programs (whether 64-bit or 32-bit) are much larger than programs compiled with MinGW32 GnuCOBOL.

Here is an example of 64-bit MinGW GnuCOBOL environment variables established by the “cobenv.cmd” script, which can be adapted for OpenCobolIDE 4.7.6:

```
C:\GC22B-64>bin\cobenv --showenv
current environment:
  PATH=C:\GC22B-64\bin;C:\Program Files\…
  COB_LIBRARY_PATH=C:\GC22B-64\lib\gnucobol
  COB_CONFIG_DIR=C:\GC22B-64\share\gnucobol\config
  COB_COPY_DIR=C:\GC22B-64\share\gnucobol\copy
C:\GC22B-64>
```

GnuCOBOL 2.2 Build Guide for MinGW 64-bit (draft)

As of 24 December 2017, the MinGW 64-bit GnuCOBOL 2.2 binary can be downloaded from the following address:

http://www.arnoldtrembley.com/GC22B-64bit-rename-7z-to-exe.7z

Due to a security restriction from my web hosting service I cannot host ".exe" files. So the new files have been renamed with ".7z" as their file extension. After downloading they can be opened using 7-Zip, or the windows file extensions can be renamed from ".7z" to ".exe", allowing them to be used as self-extracting archives. The self-extracting file will prompt you to supply a folder name for the compiler. It can also be installed to a drive other than your C: drive.

7-Zip is open source software available from http://www.7-zip.org/

In the future, new binaries will be added to the following page:

http://www.arnoldtrembley.com/GnuCOBOL.htm

For additional assistance, please feel free to try the GnuCOBOL Forums:

https://sourceforge.net/p/open-cobol/discussion/

=====end=====