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

cobc (GnuCOBOL) 2.2.0
Copyright (C) 2017 Free Software Foundation, Inc.
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 Tuesday, 12 September 2017.

Please refer to the excellent manual written by Gary Cutler (CutlerGL@gmail.com):

OpenCOBOL-1.1-06FEB2009-Build-Guide-For-MinGW.pdf

http://www.arnoldtrembley.com/OpenCOBOL-1.1-06FEB2009-Build-Guide-For-MinGW.pdf

I used Gary Cutler's document as a guide when building GnuCOBOL 1.1 and GnuCOBOL 2.0
using MinGW. This document will be much less detailed, and will describe how to build
GnuCOBOL 2.2 rather than older versions of OpenCOBOL or GnuCOBOL.

Simon Sobisch of the GnuCOBOL project on Sourceforge.net was extremely helpful whenever I
encountered difficulties building GnuCOBOL, especially in running the compiler test suites and
VBISAM. **Brian Tiffin** also provided assistance and encouragement. The
GnuCOBOL/OpenCOBOL project can be found at:

https://sourceforge.net/projects/open-cobol/

GnuCOBOL 2.2 Build Guide for MinGW (draft)

Required Components:

Based on Gary Cutler's build guide, you will need to find and install the following components in order to build the GnuCOBOL 2.0 compiler in Windows:

1. MinGW - Minimalist GNU for Windows
2. GNU Multiple-Precision Arithmetic package (gmplib)
3. PDCurses 3.4 - used for screen read and write.
4. Berkeley Database (BDB) package from Oracle ==OR==
5. VBISAM2 by Roger While, from Sourceforge
6. GnuCOBOL 2.0 source code from the OpenCOBOL project on Sourceforge.net

You may want to download all these packages first and make your own backups before starting the GnuCOBOL build process.

Licensing:

This document will not discuss open source licensing in detail. Please refer to Gary Cutler's document. The GnuCOBOL 2.0 compiler is licensed under the GNU General Public License (GPL) version 3, and the runtime libraries are licensed under the GNU Lesser General Public License (LGPL) version 3. The Oracle Berkeley Database (BDB) package, used for indexed sequential file access, has some license restrictions related to distribution of compiled GnuCOBOL programs that could require distributing your COBOL source code or else paying a license fee to Oracle. There are no similar license restrictions if the VBISAM package is used for indexed sequential file access (instead of BDB), or if no indexed sequential file access will be supported (NODB).

Download the packages:

The **MinGW** software package can be downloaded from:
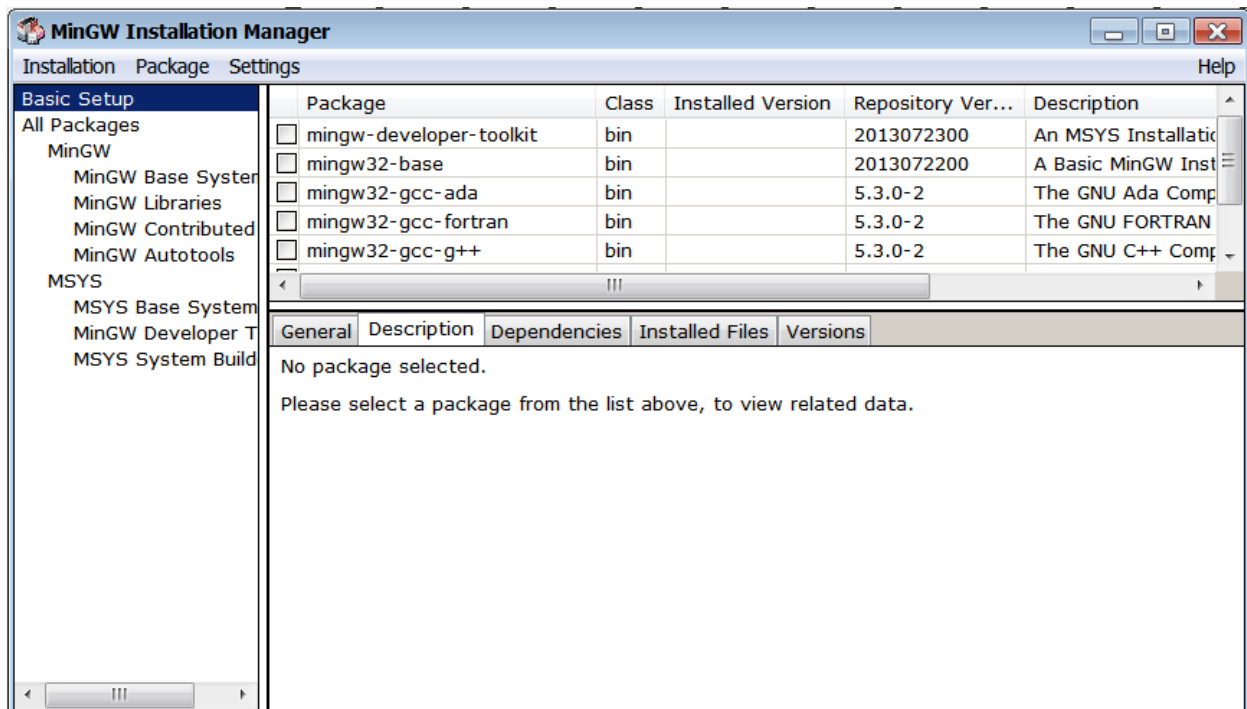https://sourceforge.net/projects/mingw/files/

Download the file named "mingw-get-setup.exe". This should be the 32-bit version of MinGW. MinGW is a Unix-like environment for Windows needed to run GCC (the GNU Compiler Collection) to build the GnuCOBOL compiler. The GnuCOBOL compiler will translate COBOL source code into C source code, and the embedded MinGW GCC compiler will translate the intermediate C code into executable code.

GnuCOBOL 2.2 Build Guide for MinGW (draft)

Installing MinGW and the MSYS bash shell

First, create a wrapper folder to hold MinGW and keep it separate from any other MinGW installation on the same PC. For this I created **C:\GC22-build\MinGW.** When you run "mingw-get-setup.exe" it will default to installing in **C:\MinGW** and you should override this with **C:\GC22-build\MinGW**. The folder must already exist in order to be selected.

Continue with "mingw-get-setup.exe" and let it download the catalog of current components. When that is complete, use the GUI interface to select the "**mingw32-base**" and "**msys-base**" packages from the "Basic Setup" menu.

GnuCOBOL 2.2 Build Guide for MinGW (draft)

Then select the following components from the "All Packages" menu. Some of them will already be selected with ming32-base and msys-base.

mingw32-autoconf.bin	
mingw32-automake.bin	
mingw-bzip2.bin	???
mingw32-dos2unix.bin	???
mingw32-gettext.bin	/* for translatable strings */
mingw32-gettext.dev	/* for translatable strings */
mingw32-gettext.dll	/* for translatable strings */
mingw32-gmp	???
mingw32-libtool.bin	
mingw32-make.bin	
msys-help2man.bin	/* for GnuCOBOL make */
msys-m4.bin	/* for building Berkeley DB */
msys-make.bin	
msys-perl.bin	/* for NIST COBOL85 testing */
msys-wget.bin	/* for NIST COBOL85 testing */

Note: MinGw GCC 6.3.0 did not appear to include "msys-wget.bin" component, and I had to manually copy in the "newcob.val" file for the NIST COBOL85 test suite. But as of 29JUL2017 the msys-get GUI does include "msys-wget.bin" as a selectable component.

Then have "mingw-get" apply all changes that were previously marked. The installation may take 5 to 15 minutes.

Immediately after installing **C:\GC22-build\MinGW,** copy the **cc1.exe** file to an additional location:

Copy "C:\GC22-build\MinGW\libexec\gcc\mingw32\6.3.0**cc1.exe**"
to "C:\GC22-build\MinGW\bin**cc1.exe**".

In the example above, "6.3.0" is the version of GCC in MinGW. This version number may be different when you download the "mingw-get" GUI interface installer.

Next, verify that **C:\GC22-build\MinGW\bin** contains a file named **mingwm10.dll.**

GnuCOBOL 2.2 Build Guide for MinGW (draft)

Then verify that a directory named "**C:\GC22-build\MinGW\MSYS\1.0\etc**" exists and that it contains a file named "**fstab**" in it with no file extension. View this file with Notepad or Wordpad and verify that it has the following line in it:

```
c:/GC22-build/MinGW /mingw
```

This "fstab" file may contain comment lines which begin with "#", and they can be ignored. Since Unix file and folder names are case-sensitive, this fstab file tells MinGW/MSYS to treat the Windows "**C:\GC22-build\MinGW**" folder as the Unix/Linux mount point named "/mingw".

```
# /etc/fstab -- mount table configuration for MSYS.
# Please refer to /etc/fstab.sample for explanatory annotation.

# MSYS-Portable needs this "magic" comment:
# MSYSROOT=C:/GC22-Build/MinGw/msys/1.0

# Win32_Path                    Mount_Point
#-------------------------------------------
C:/GC22-Build/MinGw             /mingw
```

MSYS will only use the "/mingw" mount point, even though it is "**C:\GC22-build\MinGW**" to Windows. If you selected a different folder on the MinGW startup screen, it should be automatically built into the "fstab" file.

GnuCOBOL 2.2 Build Guide for MinGW (draft)

Finally, you will need to create a shortcut on your windows desktop to:

`C:\GC22-build\MinGW\MSYS\1.0\msys.bat -norxvt`

Do not forget the "-norxvt" parameters in the shortcut. This tells the msys.bat file to use the "sh" shell instead of the "rxvt" shell. You may also want to change the icon for the shortcut to use "C:\GC22-build\MinGW\MSYS\1.0\msys.ico".

Make sure the properties for this shortcut use:

Target: C:\GC22-Build\MinGw\msys\1.0\msys.bat -norxvt
Start in: C:\GC22-Build\MinGW\MSYS\1.0

Also, make sure the Security properties of the MSYS shortcut include full administrator access (Full Control, Modify, Read & Execute, Read, and Write).

You should now have an icon "**MSYS**" on your desktop. Double-click it to start it.

Configure the BASH shell, from Gary Cutler's document:

The window started by the "MSYS" icon resembles a Windows console, but is actually an MSYS "bash" shell; Use the window's "Properties" command as you would do with a normal window to change the window size to 190 columns by 60 rows - make sure the buffer size has a "height" of at least 600; you'll also want to change the font to something that enables that window to fit on your screen (I use "Lucida Console" with a font "size" of "10").

*If you are running Windows Vista or Windows 7, close the bash window and restart it again, this time giving it Administrator authority via "**Run As Administrator**".*

I strongly recommend changing the font type and size first (lucida console 10), and then changing the MSYS window to 190 columns by 60 rows, and making the "height" at least 2000 lines (or even more). When running "make" commands many components display thousands of messages in the MSYS window, and you may want to scroll back to view them.

If you are running MinGW on a widescreen laptop you may want to limit the window height to 50 or 55 rows for your convenience. You should also set the MSYS properties "edit options" to enable both "quick edit mode" and "insert mode". That will allow you to paste commands into the MSYS bash shell using the right mouse button (instead of Ctrl-V), and copy blocks of text messages from the MSYS bash shell window and paste them into other documents.

GnuCOBOL 2.2 Build Guide for MinGW (draft)

Unexpected Errors

If unexpected errors occur anytime during the GnuCOBOL build process, first check that all steps are run in the correct order and no steps are skipped. For example, accidentally omitting any "make install" command will prevent the final GnuCOBOL from building. Then check that all MSYS commands have been entered correctly, without any typing mistakes.

It is still possible to have unexpected errors and for those you should join the GnuCOBOL forum "Help Getting Started" and ask questions there. Be prepared to show exactly what errors occurred. It will also help if you register with the OpenCOBOL Sourceforge project, so your posts can appear immediately without having to wait to be moderated.

Here are the links to the Sourceforge project and the discussion forums:

https://sourceforge.net/projects/open-cobol/files/nist/

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

Some of the errors I encountered included messages like this:

```
rm: cannot lstat `conftest.exe': Permission denied
```

Based on comments in the GnuCOBOL forums, the best solution to this problem is to disable Windows file indexing (for search), and also to deactivate your internet security /anti-virus product temporarily. But you should still leave Windows Defender (formerly Microsoft Security Essentials or MSE) activated.

GnuCOBOL 2.2 Build Guide for MinGW (draft)

Anti-Virus Considerations

I found that Windows Defender/Microsoft Security Essentials did not appear to have conflicts with MinGW GnuCOBOL builds, while McAfee Antivirus and freeware Avast Anti-virus sometimes caused problems. One user in the GnuCOBOL/OpenCOBOL Sourceforge forum reported that Norton Anti-virus quarantined MinGW output as possible viruses.

If your anti-virus system tries to block any programs generated by MinGW GCC, you may want to disconnect from the internet and disable your anti-virus protection. Or you may want to exclude the entire C:\GC22-build\MinGW* folder from anti-virus scanning if that is an option. I also had to create exclusions for dummy.exe in the Windows %User%\temp folder.

Some of the errors I encountered said "permission denied", and it I reran that configure or make step, the errors disappeared.

I recommend running with your anti-virus disabled (except for Windows Defender/MSE). This will allow the build to run much more quickly and also prevent problems during the build.

If you intend to run the NIST COBOL85 test suite, you either need an internet connection to download the "**newcob.val**" file during "make test" testing, or you need to download it from Sourceforge before building the components:

https://sourceforge.net/projects/open-cobol/files/nist/

You will need to choose one of the archive files (newcob.zip, newcob.7z, or newcob.val.tar.gz), and expand it to extract the "**newcob.val**" file. It is 26 megabytes and contains all the NIST COBOL85 test programs.

GnuCOBOL 2.2 Build Guide for MinGW (draft)

Building GMP (GNU Multiple-Precision Arithmetic Library)

The **GMP 6.1.2** GNU Multiple-Precision Arithmetic Library can be downloaded from:
https://gmplib.org/#DOWNLOAD

As of July 20, 2017, The most current GMP version is in a file named "gmp-6.1.2.tar.lz", but previous versions of MSYS could not unpack a "tar.lz" file. I used the free "lzip.exe" for windows to convert "gmp-6.1.2.tar.lz" into "gmp-6.1.2.tar".

Lzip.exe can be downloaded as "lzip-1.11" from:
http://www.softpedia.com/get/Compression-tools/Lzip.shtml

Once you have decompressed "gmp-6.1.2.tar.lz" into "gmp-6.1.2.tar" you will need to copy the "tar" file into your /mingw/MSYS folder using Windows Explorer or the Windows CMD.EXE command shell. Then use the MSYS bash shell to execute the following commands to unpack the gmp tarball:

```
cd /mingw/MSYS
tar xf gmp*.tar      /* unpack non-compressed tar */
rm gmp-*.tar         /* remove the "tar" file  */
```

At this point you have a Windows folder named C:\GC22-build\MinGW\MSYS\gmp-6.1.2. The next five MSYS commands will build compile gmp 6.1.2 for us:

```
cd gmp*
./configure --prefix=/mingw --enable-shared --disable-static
make
make check           /* Test gmp 6.1.2*/
make install
```

You may find it easier to copy and paste these commands into the MSYS bash shell one line at at time, especially for the complicated commands. The four commands to build GMP take 10 to 20 minutes to run. Commands can also be concatenated using the && operator, for example "make && make check && make install". Processing will stop early if any errors are found.

GnuCOBOL 2.2 Build Guide for MinGW (draft)

The "./configure" command took a long time to run and produced hundreds of messages.

The "make" command took a very long time to run and produced thousands of messages.

The "make check" command ran for a very long time and produced thousands of messages. It performs hundreds of tests against the generated gmplib components

The "make install" command runs very quickly. It can be run before "make check", but do not forget to run "make install" to prevent problems building the final COBOL compiler.

After "make install" completes, run the following command in MSYS to verify that a file named "**/mingw/bin/libgmp-10.dll**" exists:

```
ls /mingw/bin/libgmp*.dll
```

GnuCOBOL 2.2 Build Guide for MinGW (draft)

Building PDCurses

The **PDCurses 3.4** package is in a file named "**PDCurses-3.4.tar.gz**" which can be downloaded from:
https://sourceforge.net/projects/pdcurses/files/pdcurses/3.4/

The PDCurses package is used for COBOL SCREEN-SECTION and extended console input-output support (DISPLAY/ACCEPT AT/WITH). This package did not need any changes from Gary Cutler's original instructions.   As of March, 2017, PDCurses 3.4 built on 2008-09-08 is still the most current version.

Copy "PDCurses-3.4.tar.gz" into your **C:\GC22-build\MinGW\MSYS** folder using Windows commands, and then use the following bash shell commands to uncompress it:

```
cd /mingw/MSYS
tar xzf PDC*.tar.gz
rm PDC*.tar.gz
```

Then build PDCurses using these commands:

```
cd PDC*/win32
make -f gccwin32.mak DLL=Y
```

Then use the following command to verify that a "pdcurses.dll" exists in the
"**C:\GC22-build\MinGW\MSYS\PDCurses-3.4\win32**" folder.

```
ls *.dll
```

You can also verify this using Windows Explorer.

Gary Cutler's document said "*there is no 'make install' available for PDCurses, so you'll have to deploy the PDCurses components yourself*". Here are the MSYS commands to do that, and this is another instance where it may help you to paste these commands into the MSYS bash shell window:

```
cp pdcurses.dll /mingw/bin/.
cp pdcurses.a /mingw/lib/libpdcurses.a
cd ..
cp *.h /mingw/include/.
cp curses.h /mingw/include/pdcurses.h
```

GnuCOBOL 2.2 Build Guide for MinGW (draft)

Process Improvement Checkpoint

At this point in the build process I normally backup the entire "c:\GC22-Build" folder so that I can restore it with gmplib and PDCurses already built. This saves me a lot of time when building multiple versions of GnuCOBOL with either Berkeley DataBase, no indexed sequential file support, or VBISAM 2.01. The backup folder includes the MinGW get program and the shortcut to MSYS:

```
 Directory of C:\GC22-Build-BKUP

07/20/2017  01:53 AM    <DIR>          .
07/20/2017  01:53 AM    <DIR>          ..
07/20/2017  02:01 AM    <DIR>          MinGw
09/07/2014  03:53 AM            86,528 mingw-get-setup.exe
11/09/2016  07:44 PM             2,211 msys.bat - Shortcut.lnk
             2 File(s)         88,739 bytes
             3 Dir(s)  199,516,729,344 bytes free
```

For example, if I have already built GnuCOBOL with BDB and I want to build another version with VBISAM 2.0, I simply restore the C:\GC22-Build folder from the backup and copy the MSYS shortcut to the desktop. Then I can start MSYS with gmplib and PDCurses already built.

Backing up the C:\GC22-Build folder is completely optional at this point. But if you build GnuCOBOL frequently then restoring the build folder can save you an hour or two on every subsequent build.

It should be possible to use MinGW get to install pre-built gmplib and curses support, but my first and only attempt at this was not successful. One of my goals is to test building GnuCOBOL with pre-built MinGW gmplib and curses support. If that can be made to work reliably, then these instructions can be simplified to build MinGW GnuCOBOL much more quickly and easily.

At this point the next step is to build the optional indexed sequential file support (either BDB, no indexed sequential file support, or VBISAM 2.0), before building from GnuCOBOL 2.0 source code files.

GnuCOBOL 2.2 Build Guide for MinGW (draft)

Building Berkeley Database (BDB)

The **Berkeley Database (BDB)** file is named "db-6.2.32.NC.tar.gz", which is the most current version as of June, 2017, and it can be downloaded from:
http://www.oracle.com/technetwork/database/database-technologies/berkeleydb/downloads/index.html

The Berkeley Database (BDB) package provides indexed file access for the GnuCOBOL compiler. You should bypass this step if you intend to build GnuCOBOL with VBISAM 2 instead of BDB, or if you intend to build GnuCOBOL with no Indexed Sequential file support (NODB).

Copy "**db-6.2.32.NC.tar.gz**" into your **C:\GC22-build\MinGW\MSYS** folder and then use the following MSYS bash commands to decompress it:

```
cd /mingw/MSYS
tar xzf db*.tar.gz
rm db*.tar.gz
```

At this point we need to make a **source code patch** to BDB before continuing. Locate the file named "C:\GC22-build\MinGW\MSYS\db-6.2.32.NC\src\os_windows**os_stat.c**" and search for "**_tcsclen**". There should be only one instance of it. Replace "**_tcsclen**" with "**strlen**", and save the file. If you are cautious you may want to first make a backup of that "os_stat.c" file, and then a separate backup of the patched version of that file.

Then build BDB using the following commands:

```
cd db*/build_unix
../dist/configure --enable-mingw --prefix=/mingw --enable-compat185 LIBCSO_LIBS=-lwsock32
make
make install
```

The "configure" command runs fairly quickly and produces hundreds of messages. The "make" command runs for a fairly long time. The "make install" command runs very quickly.   Unlike other components the Berkeley Database package does not support "make check" or "make test" to validate the build.

2017-09-07 – the configure failed with an lstat error, permission denied. I ran the configure again and it completed successfully. Turning off McAfee Internet Security also prevents this problem.

GnuCOBOL 2.2 Build Guide for MinGW (draft)

It appeared as if the Berkeley DataBase "make" command failed because there was a long wait after displaying the following messages:

```
libtool: install: cp -p .libs/db_printlog.exe /mingw/bin/db_printlog.exe
libtool: install: cp -p .libs/db_recover.exe /mingw/bin/db_recover.exe
libtool: install: cp -p .libs/db_replicate.exe /mingw/bin/db_replicate.exe
libtool: install: cp -p .libs/db_stat.exe /mingw/bin/db_stat.exe
libtool: install: cp -p .libs/db_tuner.exe /mingw/bin/db_tuner.exe
libtool: install: cp -p .libs/db_upgrade.exe /mingw/bin/db_upgrade.exe
libtool: install: cp -p .libs/db_verify.exe /mingw/bin/db_verify.exe
Installing documentation: /mingw/docs ...
```

If your BDB build appears to hang on this step, simply allow it run for a long time, at least 20 minutes. It should finish eventually.   In later builds I found little or no wait for installing documentation.

After completing the "make install" step, run the following command to verify that "**libdb-6.2.dll**" was generated in "**C:\GC22-build\MinGW\bin**":

`ls /mingw/bin/libdb*.dll`

If you are using a different version of BDB, the "6.2" in the "libdb" name will match the version number of the BDB package.

As of 01 June 2017, the most current version of BDB from the download site is: "db-6.2.32.NC.tar.gz".

GnuCOBOL 2.2 Build Guide for MinGW (draft)

Building VBISAM 2.0

The alternative Indexed Sequential component is **VBISAM 2.01**, which has a less restrictive license than Oracle Berkeley DataBase. You should bypass this step if you plan to build GnuCOBOL with Oracle Berkeley DataBase (BDB), or NODB (no ISAM support) instead of VBISAM.

NOTE: The best version of VBISAM that I found is "vbisam-2.0.1", posted anonymously in the GnuCOBOL SourceForge discussion forums. I am hosting it on my website in two different archives (it's the same source code):

http://www.arnoldtrembley.com/vbisam_install_guide_v5.1.zip

http://www.arnoldtrembley.com/vbisam-2.0.1.zip

You can use either the "vbisam-2.0.1.zip" from the install guide package, or the "vbisam-2.0.1.zip" file by itself. Simply extract "vbisam-2.0.1.zip" into your MSYS folder as a "vbisam-2.0.1" folder. Most of the source modules have a 2016-04-27 05:01:50 date-time stamp.

You may now return to your MSYS bash shell and enter the following commands to build VBISAM:

```
cd /mingw/MSYS
cd vbi*
./configure --prefix=/mingw
make
make check
make install
```

The "configure" command runs fairly quickly and produces about 100 messages. The "make" command runs a bit longer and produces more messages. The "make check" command runs very quickly. The "make install" command should also run very quickly.

There are other sources for VBISAM 2.0, and you should check the GnuCOBOL SourceForge discussion groups to see if a newer version is available, and if the NIST COBOL85 tests will run.

GnuCOBOL 2.2 Build Guide for MinGW (draft)

GnuCOBOL 2.2 Release Candidate 1 source code was downloaded from the following sourceforge link:

https://sourceforge.net/projects/open-cobol/files/gnu-cobol/2.2/

The downloaded file was named "**gnucobol-2.2.tar.xz**" and dated 2017-09-06. Copy that file into your MSYS folder. Then use the following MSYS bash shell commands to unpack the GnuCOBOL 2.0 Release Candidate 2 source code:

```
cd /mingw/MSYS
tar xf gnu*.tar.xz
rm gnu*.tar.xz
```

When you are finished you should have a subfolder named "gnucobol-2.2" in your **/MinGW/MSYS** folder.

If you are using "gnucobol-2.2.tar.gz" the unpack command is "tar xzf gnu*.tar.gz" and the delete command is "rm gnu*.tar.gz".

Applying Patches

As of 2017-07-20 there are no source code patches to be applied to the GnuCOBOL 2.2 files.

If you are not connected to the internet and you also want to run the NIST COBOL85 test suite, you should download the "**newcob.val**" file and copy it into this folder:

C:\GC22-build\MinGW\msys\gnucobol-2.2-rc\tests\cobol85

The "newcob.val" file can be downloaded in several different compressed formats from here:

https://sourceforge.net/projects/open-cobol/files/nist/

GnuCOBOL 2.2 Build Guide for MinGW (draft)

THIS STEP IS NO LONGER NEEDED FOR THE FINAL GC22 RELEASE!!

When building GnuCOBOL 2.2 Release Candidate (20JUL2017) several GnuCOBOL tests in the "make check" step failed due to the test scripts needing specific customization for the MinGW environment. Thanks to Simon Sobisch, this problem was solved by downloading the "**data_binary.at**" and "**run_misc.at**" files and copying them into the "**\MinGW\msys\gnucobol-2.2-rc\tests\testsuite.src**" folder.

This corrected problems running the GnuCOBOL "make check" test scripts in the MinGW environment. Those files can be downloaded from the following locations:

https://sourceforge.net/p/open-cobol/code/1858/tree//branches/gnucobol-2.x/tests/testsuite.src/data_binary.at

https://sourceforge.net/p/open-cobol/code/1858/tree//branches/gnucobol-2.x/tests/testsuite.src/run_misc.at

THIS STEP IS NO LONGER NEEDED FOR THE FINAL GC22 RELEASE (06SEP2017)!!

GnuCOBOL 2.2 Build Guide for MinGW (draft)

Build the GnuCOBOL 2.0 compiler

The next step is to build the GnuCOBOL compiler from source code, using the following commands in MSYS, depending upon whether you want to build with BDB, or VBISAM, or without any ISAM support at all (--without-db):

```
cd /mingw/MSYS/gnu*

./configure --prefix=/mingw --disable-rpath
      /* to build with BDB */
```

-OR-

```
./configure --prefix=/mingw --with-vbisam --disable-rpath
     /* to build with VBISAM 2.01 */
```

-OR-

```
./configure --prefix=/mingw --without-db --disable-rpath
     /* to build without Indexed Sequential access (NODB) */
```

```
make                   /* Build the GnuCOBOL compiler */
make check             /* Test the GnuCOBOL build */
make test              /* NIST COBOL 85 tests      */
make install           /* Prepare Install format  */
```

The default or typical GnuCOBOL path is to build the compiler with Oracle Berkeley DataBase. The next comments assume that was your choice.

The "./configure" step runs for less than one minute and generates one or two screens of diagnostic or informational messages.

The "make" step runs for about 10 minutes and generates hundreds of messages.

The "make check" command performs 769 basic tests against the GnuCOBOL 2.2 compiler, and takes 15-25 minutes to run.

"make test" (NIST COBOL85 test suite) generates hundreds of messages and runs for about 10 minutes.

GnuCOBOL 2.2 Build Guide for MinGW (draft)

Here is the "make check" test results summary from building GnuCOBOL 2.2 RC with BDB:

```
## ------------- ##
## Test results. ##
## ------------ ##

756 tests behaved as expected.
10 tests were skipped.

203: ACUCOBOL 32bit literal size                 skipped (syn_misc.at:3200)
231: Constant Expressions (2)                    expected failure (syn_misc.at:4422)
260: Optional SCREEN for BLANK / ERASE           expected failure (syn_screen.at:442)
294: Multiple programs in one compilation group  expected failure (listings.at:1657)
384: ACCEPT OMITTED (SCREEN)                      skipped (run_accept.at:210)
451: SORT: table sort (3)                         skipped (run_misc.at:2167)
485: ON EXCEPTION clause of DISPLAY               skipped (run_misc.at:4964)
486: EC-SCREEN-LINE-NUMBER and -STARTING-COLUMN   skipped (run_misc.at:4989)
487: LINE/COLUMN 0 exceptions                     skipped (run_misc.at:5030)
498: C API Test                                   expected failure (run_misc.at:5719)
518: ASSIGN with BASED data item                  expected failure (run_file.at:481)
663: Complex OCCURS DEPENDING ON (6)              skipped (run_extensions.at:660)
664: Complex OCCURS DEPENDING ON (7)              skipped (run_extensions.at:704)
692: SORT ASSIGN KEYBOARD to ASSIGN DISPLAY       expected failure (run_extensions.at:2133)
729: System routine CBL_GC_FORK                   skipped (run_extensions.at:3801)
730: System routine CBL_GC_WAITPID                skipped (run_extensions.at:3856)
```

This was a successful outcome, without any unexpected test failures.

Here are the results from running the "make test" step for NIST COBOL85 tests with Berkeley Database:

```
------ Directory Information -------    --- Total Tests Information ---
Module Programs Executed Error Crash    Pass Fail Deleted Inspect Total
------ -------- -------- ----- -----    ----- ---- ------- ------- -----
NC         95       95      0     0     4393    0      4       1   4398
SM         17       17      0     0      291    0      2       1    294
IC         25       25      0     0      246    0      4       0    250
SQ         85       85      0     0      518    0      0      89    607
RL         35       35      0     0     1827    0      5       0   1832
ST         40       40      0     0      288    0      0       0    288
SG         13       13      0     0      310    0      0       0    310
OB          7        7      0     0       39    0      0       0     39
IF         45       45      0     0      733    0      0       0    733
DB         16       16      0     0      418    0      4      27    449
IX         42       42      0     0      507    0      1       0    508
------ -------- -------- ----- -----    ----- ---- ------- ------- -----
Total     420      420      0     0     9570    0     20     118   9708
```

This was a very good result, with 9570 tests passing out of 9708, and no program abends or failed tests.

These "make test" results were identical when building GnuCOBOL 2.2 with vbisam 2.0.1.

GnuCOBOL 2.2 Build Guide for MinGW (draft)

Here are the NIST COBOL85 test results for GnuCOBOL 2.2 with no ISAM support (NODB):

```
------ Directory Information -------    --- Total Tests Information ---
Module Programs Executed Error Crash   Pass Fail Deleted Inspect Total
------ -------- -------- ----- -----   ----- ---- ------- ------- -----
NC           95       95     0     0   4393    0       4       1  4398
SM           17       17     0     0    291    0       2       1   294
IC           25       25     0     0    246    0       4       0   250
SQ           85       85     0     0    518    0       0      89   607
RL           35       35     0     0   1827    0       5       0  1832
ST           40       40     0     0    288    0       0       0   288
SG           13       13     0     0    310    0       0       0   310
OB            7        7     0     0     39    0       0       0    39
IF           45       45     0     0    733    0       0       0   733
DBNOIX       15       15     0     0    400    0       4      25   429
------ -------- -------- ----- -----   ----- ---- ------- ------- -----
Total       377      377     0     0   9045    0      19     116  9180
```

This was a very good result, with 9045 tests out of 9180 passing, and no program abends or failed tests.

GnuCOBOL 2.2 Build Guide for MinGW (draft)

The vbisam 2.0.1 version used for this build was posted anonymously in the GnuCOBOL forums , and can be downloaded from here:

https://sourceforge.net/p/open-cobol/discussion/help/thread/51d48303/5d39/attachment/vbisam_install_guide_v4.7z

This version of vbisam is based on VBISAM 2.0 by Roger While, found in SourceForge at:

https://sourceforge.net/projects/vbisam/files/vbisam2/

But it has patches from Sergey Kashyrin at www.kiska.net, and some additional edits by Mario Matos, and some changes by the anonymous poster.

GnuCOBOL 2.2 Build Guide for MinGW (draft)

Packaging the C:\GnuCOBOL folder

Assuming all the previous steps completed successfully, especially "**make install**", here are the logical steps for building the C:\GnuCOBOL folder containing GnuCOBOL 2.0:

Create the C:\GnuCOBOL folder   (mkdir C:\GnuCOBOL)

copy C:\GC22-build\MinGW\bin   to C:\GnuCOBOL\bin
copy C:\GC22-build\MinGW\share\gnu-cobol\config   to C:\GnuCOBOL\config
copy C:\GC22-build\MinGW\share\gnu-cobol\copy   to C:\GnuCOBOL\Copy
copy C:\GC22-build\MinGW\lib   to C:\GnuCOBOL\lib
copy C:\GC22-build\MinGW\libexec   to C:\GnuCOBOL\libexec
copy C:\GC22-build\MinGW\include   to C:\GnuCOBOL\include

Note that these are logical instructions, not explicit copy command syntax. You can also use Windows Explorer to copy these folders. The important thing is to be sure that all subfolders are copied.

The Windows CMD.EXE commands would look like this, assuming "C:\GnuCOBOL" is the name chosen for your compiler folder, and "C:\GC22-build\MinGW" is the name of the Windows folder where MinGW resides:

```
mkdir C:\GnuCOBOL
xcopy C:\GC22-build\MinGW\bin\*.* c:\GnuCOBOL\bin\ /s /e
del c:\GnuCOBOL\bin\auto*.*
xcopy C:\GC22-build\MinGW\share\gnucobol\config\*.* C:\GnuCOBOL\config\ /s /e
xcopy C:\GC22-build\MinGW\share\gnucobol\copy\*.* C:\GnuCOBOL\copy\ /s /e
xcopy C:\GC22-build\MinGW\lib\*.* C:\GnuCOBOL\lib\
xcopy C:\GC22-build\MinGW\lib\gcc\*.* C:\GnuCOBOL\lib\gcc\ /s /e
xcopy C:\GC22-build\MinGW\libexec\gcc\*.* C:\GnuCOBOL\libexec\gcc\ /s /e
xcopy C:\GC22-build\MinGW\include C:\GnuCOBOL\include\ /s /e
del c:\GnuCOBOL\include\autosp*.*
xcopy C:\GC22-Build\MinGW\msys\gnucobol-2.2\extras\*.* C:\GnuCOBOL\extras\ /s /e
```

Note the in the final "xcopy" command, the name "gnucobol-2.2" may vary depending upon the actual name of that folder in MSYS after unpacking.

These commands could be built into a .BAT or .CMD file if this step will be done more than once.

GnuCOBOL 2.2 Build Guide for MinGW (draft)

Simon Sobisch provided commands for removing unneeded components from the generated compiler to reduce the size of the download file. This ".cmd" file would need to be executed in the "C:\GnuCOBOL" folder:

```
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*
```

Note that the last two lines are a single statement that concatenates four separate commands.

The resulting C:\GnuCOBOL folder is sufficient for compiling COBOL programs, but some additional files should also be added.

GnuCOBOL 2.2 Build Guide for MinGW (draft)

The following files were copied from the MinGW GnuCOBOL 1.1 compiler included in the Windows OpenCobolIDE install package, or from other sources:

AUTHORS.txt
ChangeLog.txt   copied from C:\GC22-Build\MinGw\msys\gnucobol-2.2
ChangeLog_cobc.txt   copied from C:\GC22-Build\MinGw\msys\gnucobol-2.2\cobc
ChangeLog_libcob.txt copied from C:\GC22-Build\MinGw\msys\gnucobol-2.2\libcob
config.log   copied from C:\GC22-build\MinGW\MSYS\gnucobol-2.2\config.log
COPYING.txt
COPYING_DOC.txt
COPYING_LESSER.txt
GnuCOBOL 2.0 Manual.pdf   copied from https://open-cobol.sourceforge.io/doc/gnucobol.pdf
NEWS.txt
README.txt   Modified for this package
set_env.cmd   Modified for this package (written by Simon Sobisch)
summary.log
testsuite.log   Copied from C:\GC22-build\MinGW\MSYS\gnucobol-2.2\tests\testsuite.log
THANKS.txt

The "config.log", "summary.log", and "testsuite.log" files are copied from the MinGW MSYS build of GnuCOBOL 2.2, and used to diagnose compiler build problems.

The COPYING, AUTHORS, NEWS, README, and THANKS files can also be downloaded from the GnuCOBOL sources on the project page:

https://sourceforge.net/p/open-cobol/code/HEAD/tree/branches/gnucobol-2.x/

Then there are several files I added for quickly testing GnuCOBOL compiler installation:

gcx.cmd   Command file to compile a .cob or .cbl source file and create .exe and .lst
gcmf.cmd   Command file to compile a .cob or .cbl source file and create .dll and .lst
                            "gcmf.cmd" assumes the input COBOL source file is in free-format
                            instead of fixed-format 80-byte lines.
testfunc.cob   Sample COBOL program to display date compiled and current date
TestGC.cmd   Command file to compile and execute testfunc.cob, once as .exe and once
                            as .dll, and delete the generated exe, dll, and lst files.

GnuCOBOL 2.2 Build Guide for MinGW (draft)

In actual practice, I created a full folder with all the added files, then make a copy of it before running the "strip" commands supplied by Simon Sobisch.

Then the smaller "C:\GnuCOBOL" folder was packaged as a self-extracting file.

Based on recommendations from Simon Sobisch, I have changed the packaging of the compiler to use open source 7-Zip self-extracting archives which provide better compression ratios.

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/

As of September 9, 2017, the newest MinGW binaries for GnuCOBOL 2.2 (06SEP2017) can be downloaded from the following addresses:

 http://www.arnoldtrembley.com/GC22-BDB-rename-7z-to-exe.7z

 http://www.arnoldtrembley.com/GC22-NODB-rename-7z-to-exe.7z

 http://www.arnoldtrembley.com/GC22-VBI-rename-7z-to-exe.7z

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

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

GnuCOBOL 2.2 Build Guide for MinGW (draft)

OpenCobolIDE is a GUI (Graphical User Inteface) written in Python and used to edit, compile, and test GnuCOBOL programs. It is compatible with Unix/Linux, Windows, and Mac OSX. The OpenCobolIDE preferences can be changed to use a different GnuCOBOL compiler instead of the GnuCOBOL 2.0 RC2 (BDB) compiler embedded in the Windows installer for OpenCobolIDE.

The OpenCobolIDE install package for Windows can be downloaded from this site:
https://launchpad.net/cobcide/4.0/4.7.6/+download/OpenCobolIDE-4.7.6_Setup.exe

Additional information about OpenCobolIDE can be found here:
http://OpenCobolIDE.readthedocs.io/en/latest/download.html

The "GnuCOBOL 2.2 Manual.pdf" was downloaded from the GnuCOBOL sourceforge website code repository, which includes Programmers Guide, Quick Reference, and Sample Programs in both A4 and USA Letter size PDF's:
https://sourceforge.net/p/open-cobol/code/HEAD/tree/external-doc/guide/PDFs/

The parent page for all GnuCOBOL manuals and related documentation downloads is:
https://open-cobol.sourceforge.io/

Future Steps

I hope to write a document similar that describes how to set up a MinGW GnuCOBOL development environment in Windows. It should mention additional tool options and describe how to redistribute or install GnuCOBOL programs and runtime libraries on other Windows computers without installing the GnuCOBOL compiler.

I hope to add instructions for building a normal Windows "setup.exe" file using INNO. This would be most useful when building a GnuCOBOL application to be installed on a different PC without installing the compiler.

The free INNO installer program can be found at this site:
http://www.jrsoftware.org/isinfo.php