

Building DSDv1.8.4

This document would not be possible without the hard work done by **Louis-Erig HERVE**. We thank him for making his work publically available.

These are detailed instructions for installing Cygwin and the dependencies it needs to run correctly and compile the 3 parts needed for DSD v1.8.4. These parts are DSD and the Cygwin.dlls support itpp and the mbelibl

NOTE: These steps are for both 32 and 64 bit OS

NOTE: The images show the 32 files selected however the text lists the 64bit files needed (in pink) in addition to the 32 bit files.

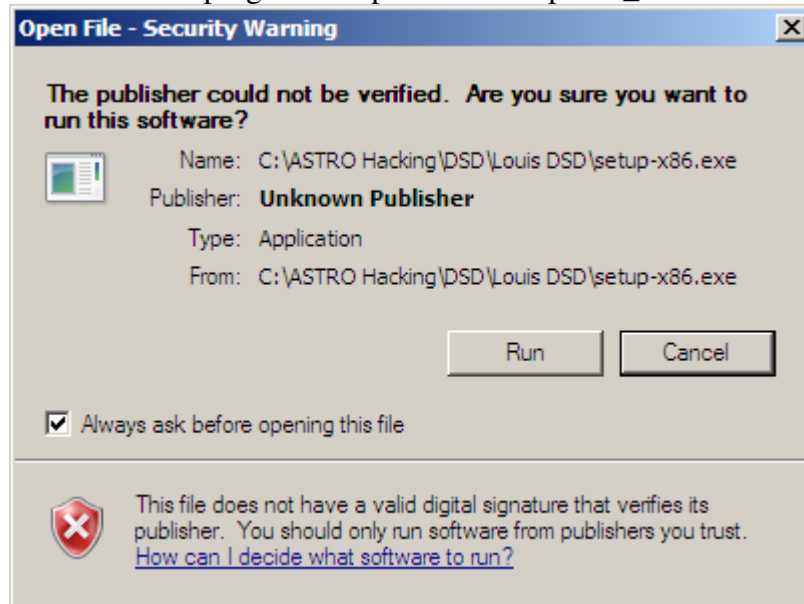
NOTE: If you are running the 64 bit OS you need the 32 bit files.

Step 1: installing Cygwin

Go to <https://www.cygwin.com/setup-x86.exe> (32bit) or https://www.cygwin.com/setup-x86_64.exe (64bit) and download the installer.

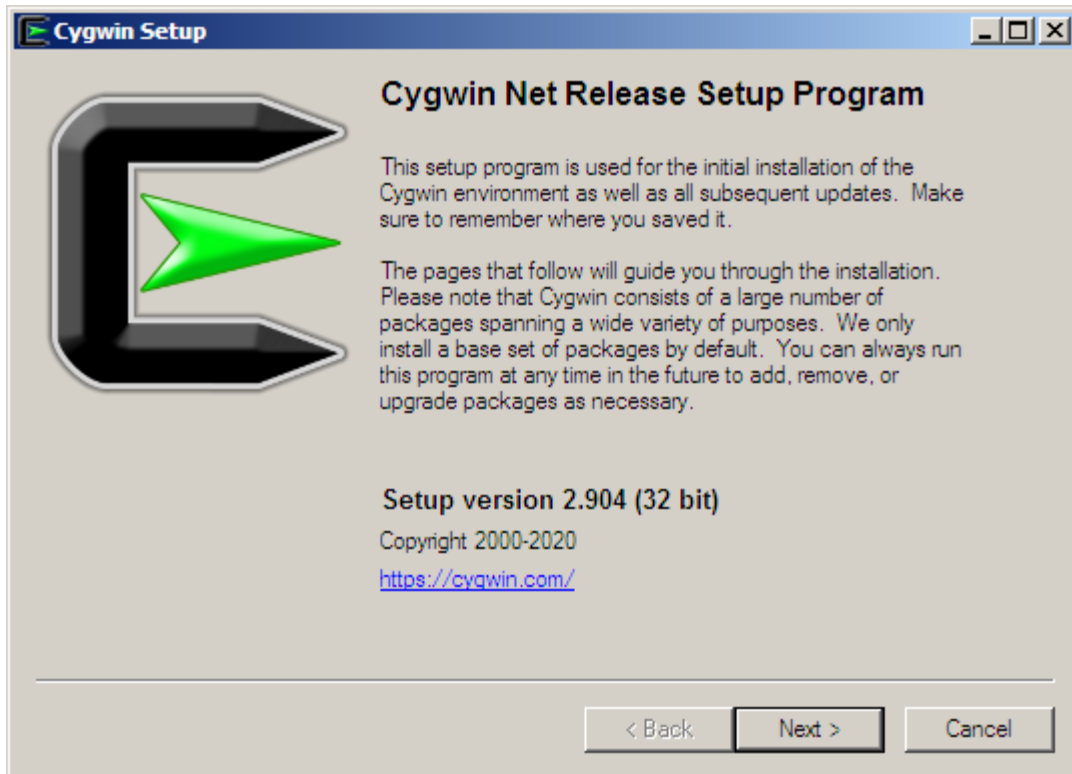
NOTE: Due to how Cygwin functions some antivirus programs don't like it. They may try and block the install, popup virus found messages etc. We recommend temporarily disabling the virus program while you install and run Cygwin. AVG is especially touchy.

Run the install program setup-x86. or setup-x86_64 and select Run.

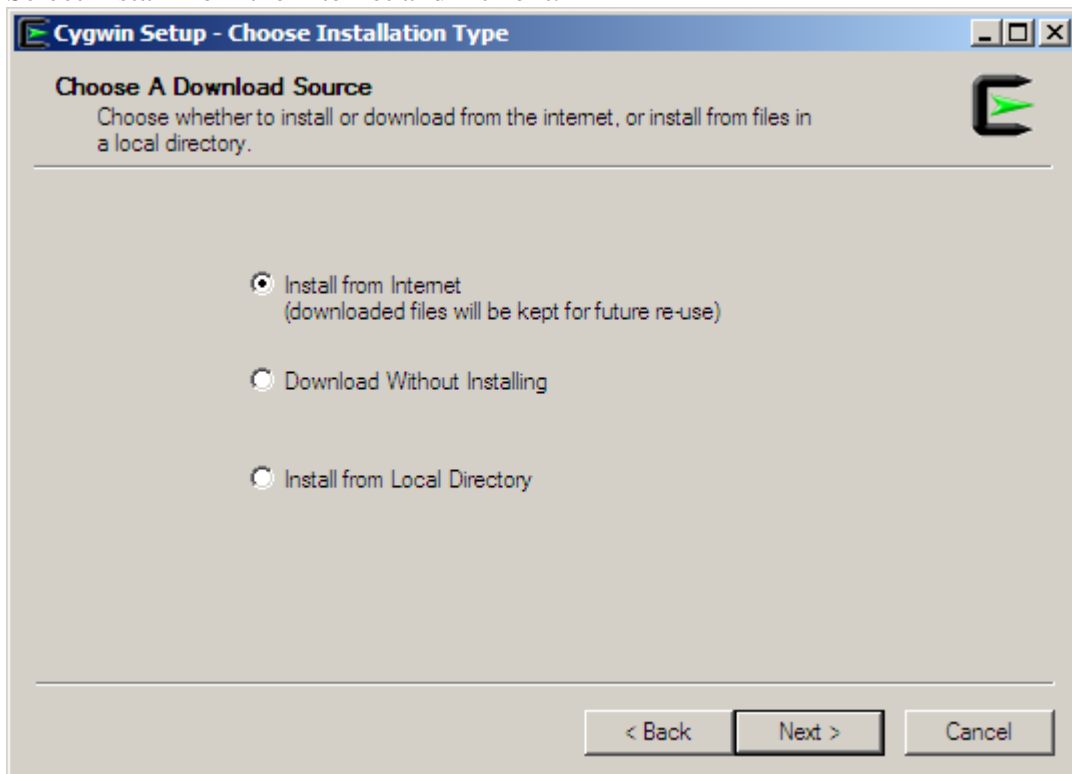


If prompted by a pop up about trusted sources select yes.

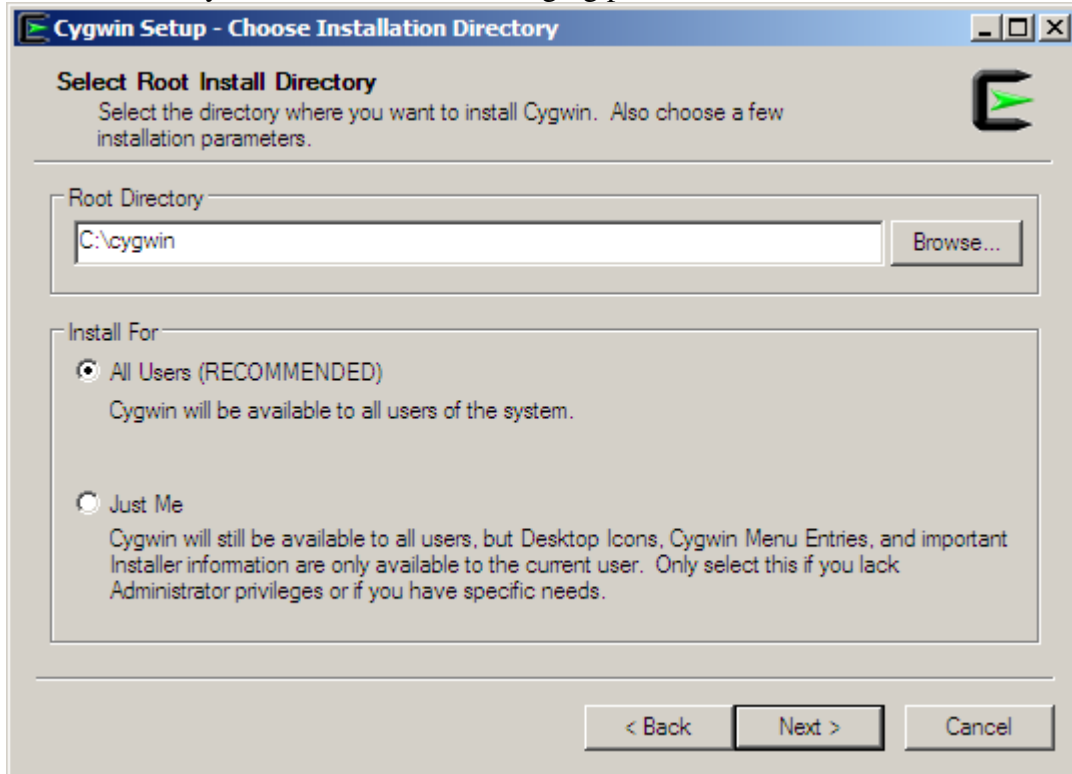
From the main entry screen select next.



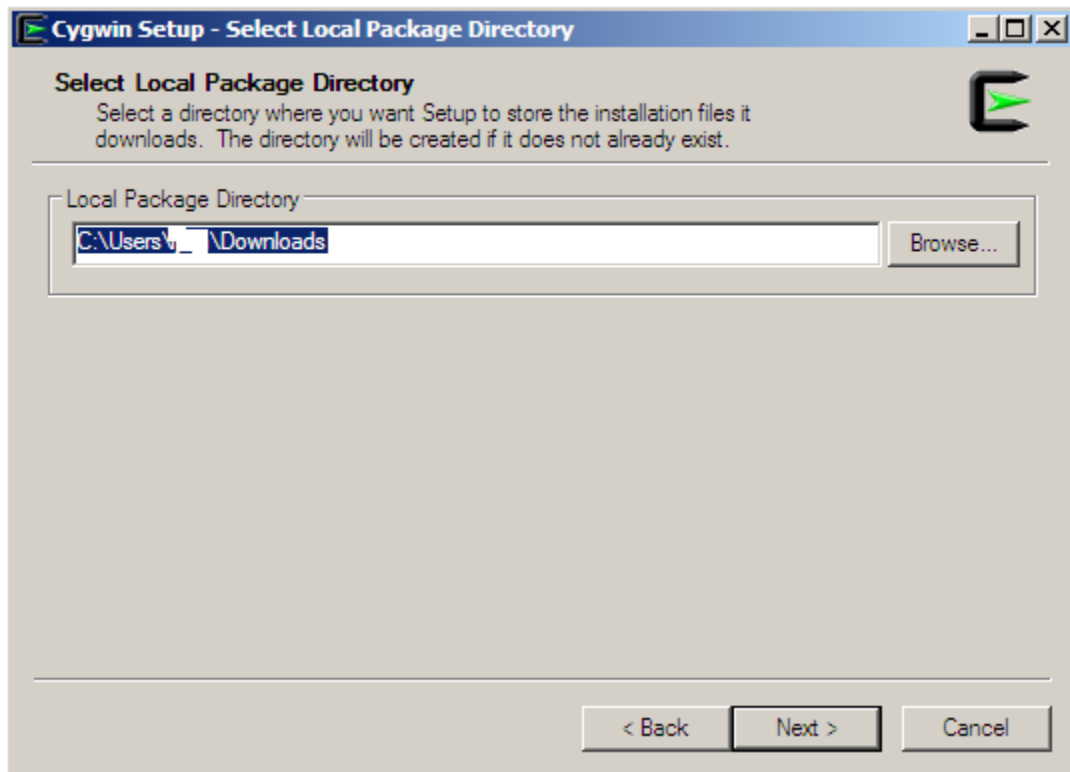
Select install from the internet and hit next.



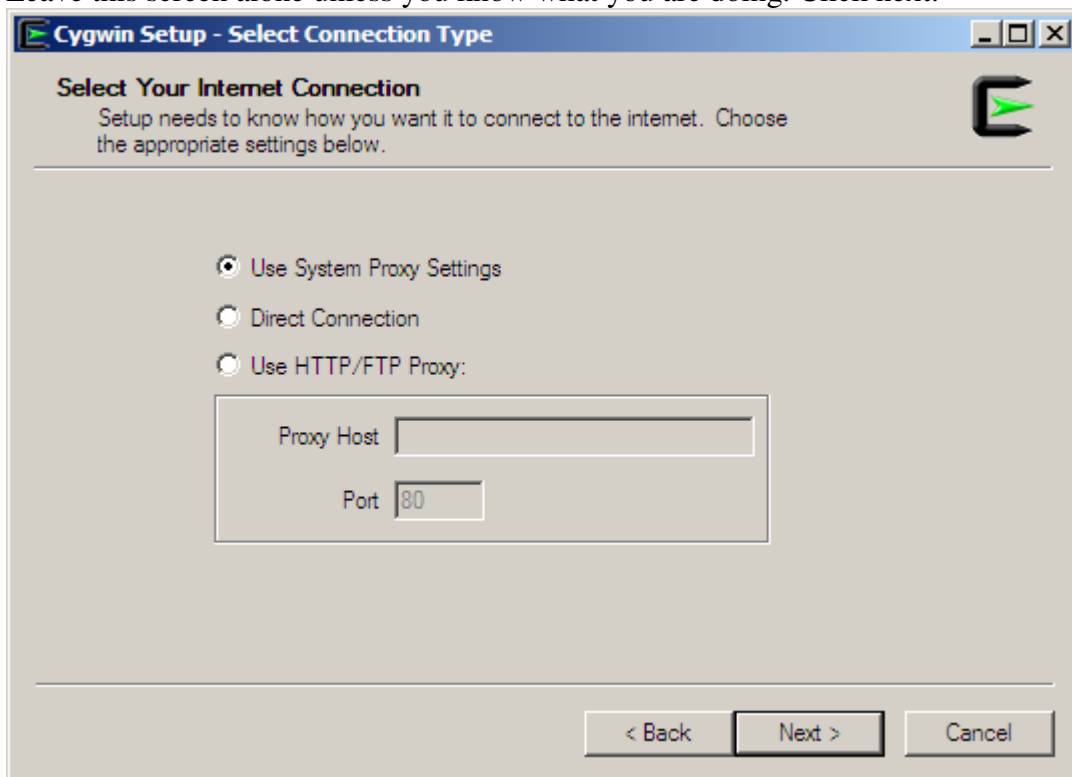
You will be asked to set up the default folder for Cygwin. We suggest you use the default location unless you are familiar with changing paths and directories. Click on next.



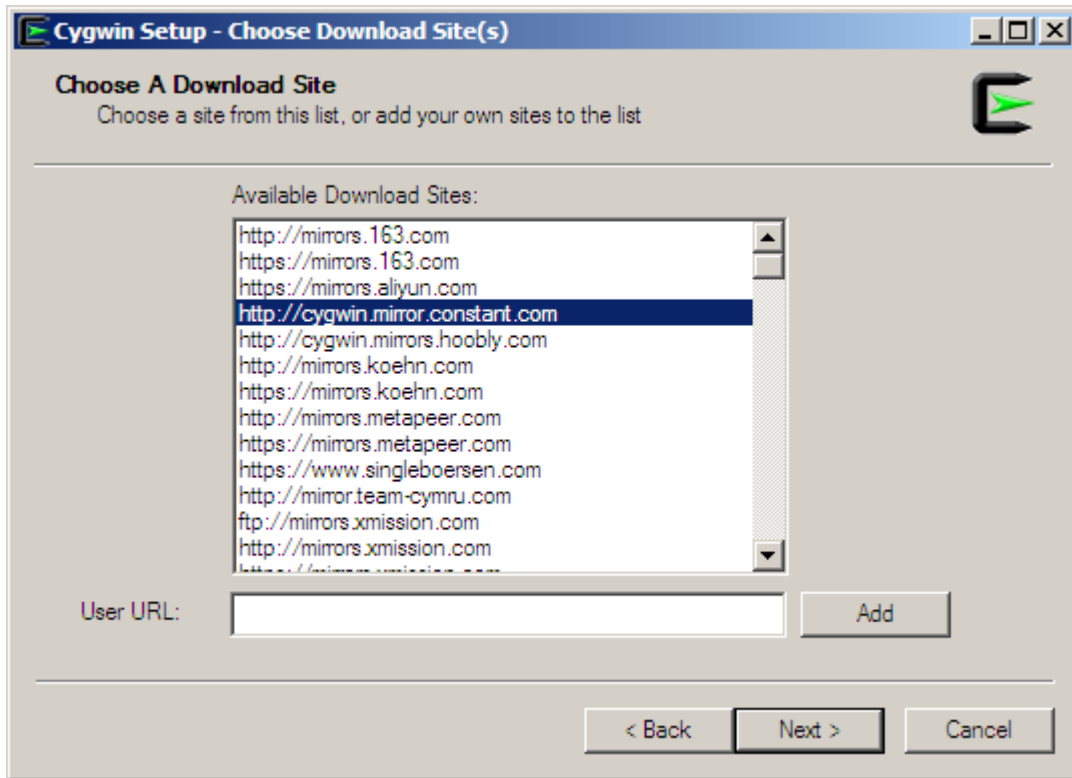
The same goes for the install file temp directory. Again click on next.
NOTE: for privacy reasons we redacted the user name.



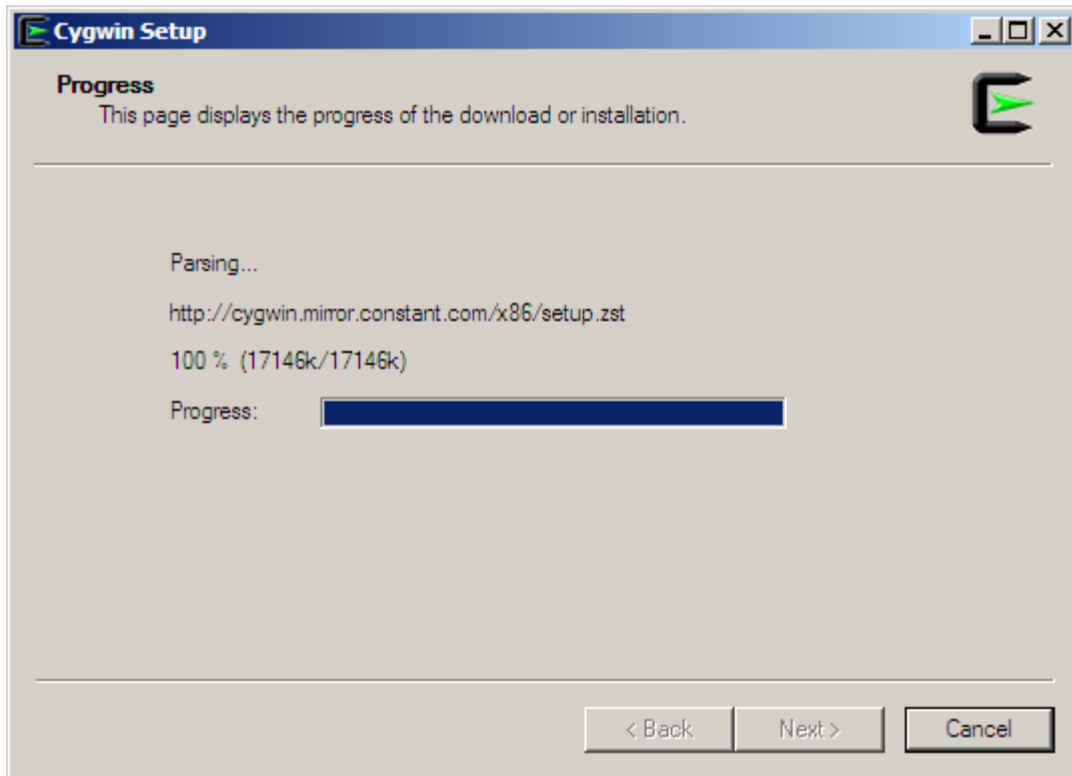
Leave this screen alone unless you know what you are doing. Click next.



You will be taken to a download site page. There are many fine sources for the files on this page. We have chosen the highlighted at random. Make your selection and click on next.



The program will connect to the server and download the basic Cygwin files.



You will be prompted to place a icon on the desk top we recommend you do so. If you stop now you will only install Cygwin. Like many Linux programs there are a pile of dependencies that will also need to be downloaded. We recommend that you install Cygwin first then exit the install program. For the next step you will restart the install program and it will automatically navigate to the steps below.

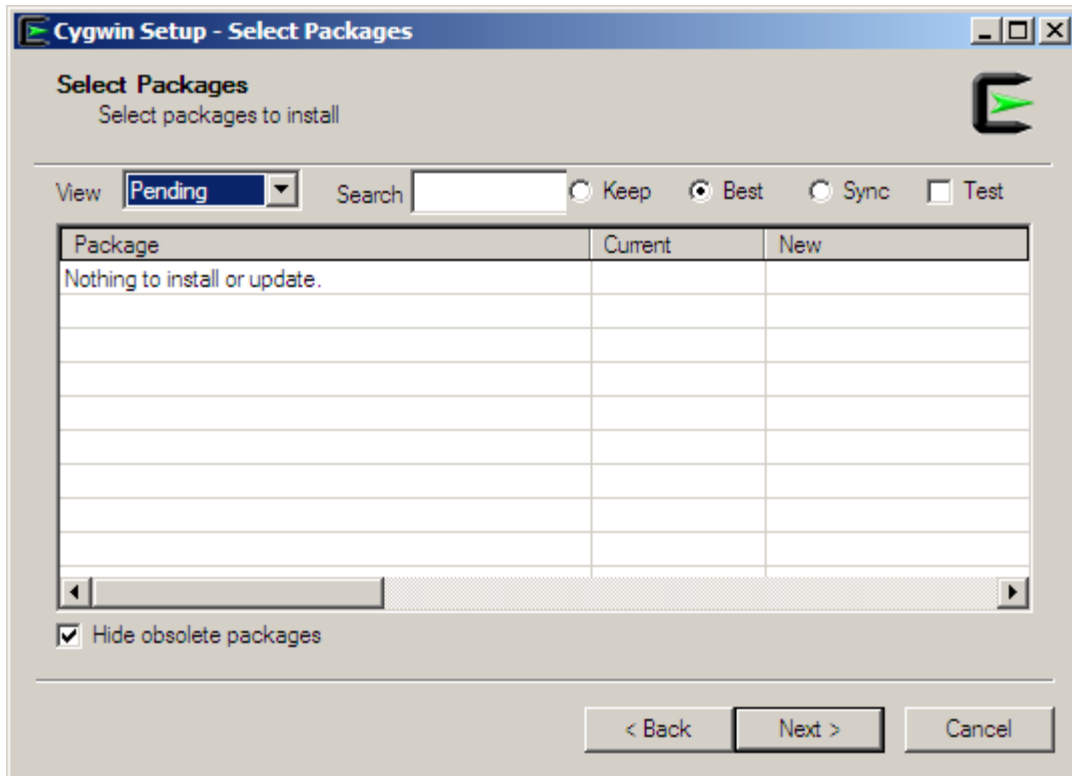
Step 2: Installing the dependencies.

DSDv1.8.3 requires a pile of dependencies to be installed so that it can properly open and build the program and necessary support files. Below are the steps needed to download and install these dependencies? The dependences needed are gcc, make, sndfile, git, wget, zip, lepack, fft and libas. Each will be discussed below.

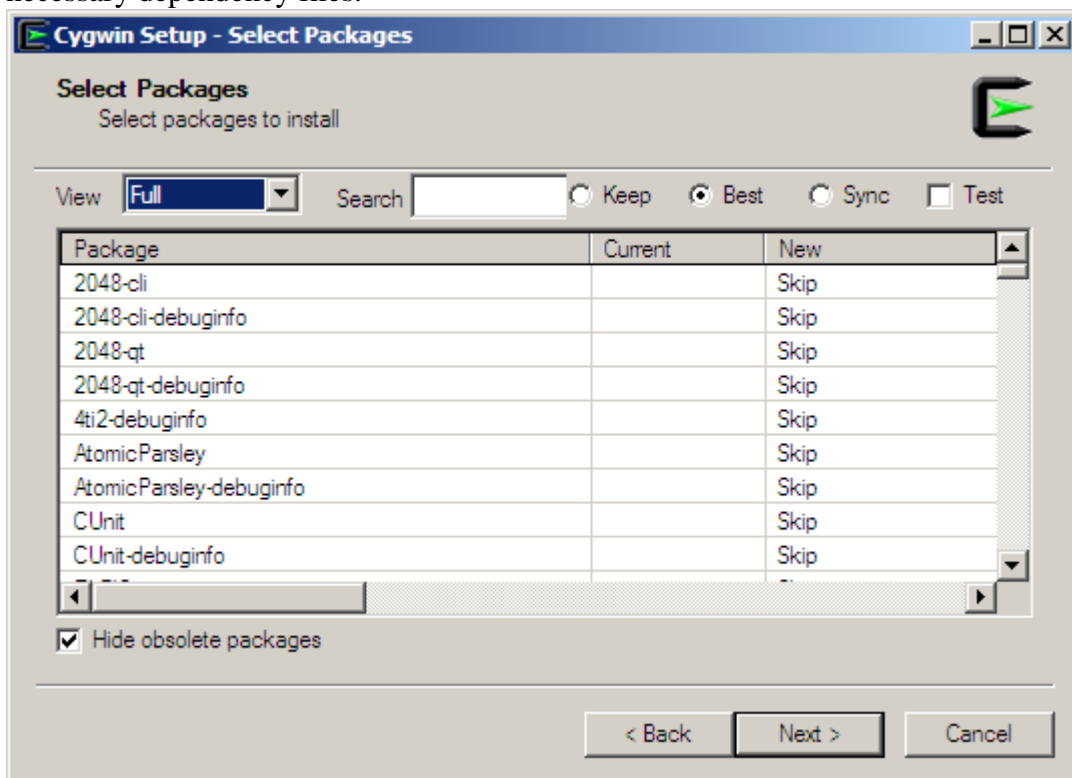
NOTE: The gcc files will be discussed first and in detail. The other files are all selected by identical steps.

NOTE: It is assumed you have completed the step1 section above.

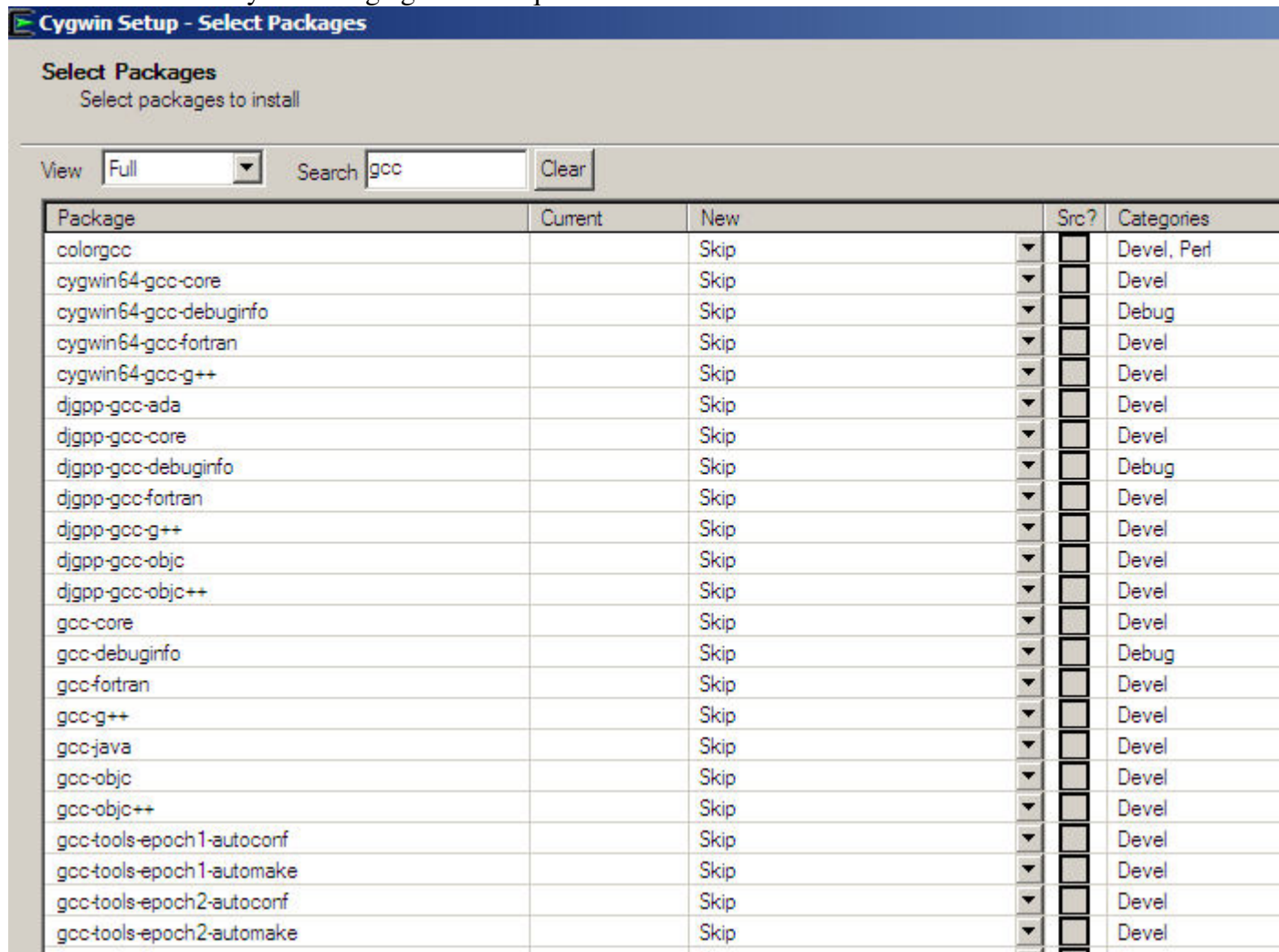
NOTE: it is assumed that you have restarted the install program and are now looking at a screen similar to the one below.



The first thing to do is to change the View dropdown from “Pending” to “Full”. You can ignore all the file packages that appear. We are now ready to search for and select the necessary dependency files.



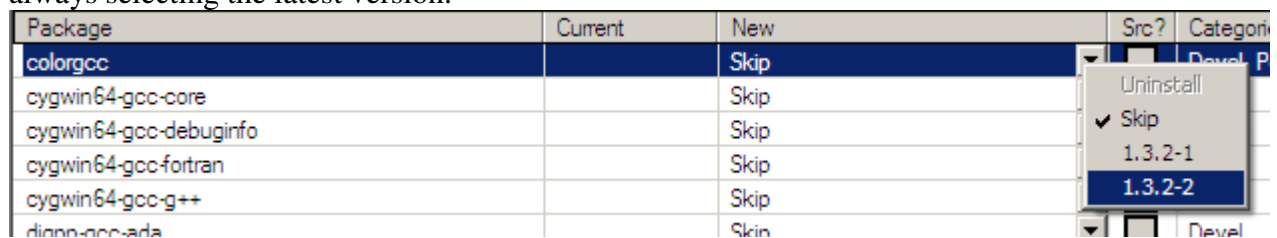
To locate the dependency files we will be using the search window. We will also change the view to full screen so you can see the information better. The first set of dependencies we want are found by searching “gcc” – No quotes.



The files we want are “colorgcc, gcc-core, gcc-g++, libgcc1 and libgccpp1. [mingw64-i686-gcc-core](#), [mingw64-i686-gcc-g++](#), [mingw64-x86_64-gcc-core](#), [mingw64-x86_64-gcc-g++](#)

NOTE: The pink file names are for 64bit OS.

To select the files select the small dropdown arrow and the end of the “skip” area. This will open up a dropdown with one or more version numbers for the file. We recommend always selecting the latest version.



Once the version is selected the “skip” line will show the version number selected.

Package	Current	New	Src?	Category
colorgcc		1.3.2-2	☐	Devel. F
cygwin64-gcc-core		Skip	☐	Devel
cygwin64-gcc-debuginfo		Skip	☐	Devel

Repeat the process for the other files listed above. The more observant will have noticed that the file libgcc1 already has a version number this means that this particular file was already installed.

NOTE: Different computers will have some files already there from other applications. Leave them there do not remove them.

NOTE: If a file already exists you can use the dropdown to update it if you want to. If in doubt leave it alone.

NOTE: If you are building DSD v1.8.3 or earlier you need to select a gcc.core, and gcc++ Earlier than v10.2.0.1 or you will have build issues.

Package	Current	New
colorgcc		1.3.2-2
cygwin64-gcc-core		Skip
cygwin64-gcc-debuginfo		Skip
cygwin64-gcc-fortran		Skip
cygwin64-gcc-g++		Skip
djgpp-gcc-ada		Skip
djgpp-gcc-core		Skip
djgpp-gcc-debuginfo		Skip
djgpp-gcc-fortran		Skip
djgpp-gcc-g++		Skip
djgpp-gcc-objc		Skip
djgpp-gcc-objc++		Skip
gcc-core		10.2.0-1
gcc-debuginfo		Skip
gcc-fortran		Skip
gcc-g++		10.2.0-1
gcc-java		Skip
gcc-objc		Skip
gcc-objc++		Skip
gcc-tools-epoch1-autoconf		Skip
gcc-tools-epoch1-automake		Skip
gcc-tools-epoch2-autoconf		Skip
gcc-tools-epoch2-automake		Skip
gcomakedep		Skip
libgcc1	10.2.0-1	Keep
libgccpp1		8.0.4-1
mingw64-i686-gcc-core		Skip
mingw64-i686-gcc-debuginfo		Skip

This concludes the gcc file selection we now continue on with the other files. Change the search from gcc to make. Select the following files automake, automake1.16, cmake and make. **There are no 64bit OS versions.**

Package	Current	New
WindowMaker-debuginfo		Skip
automake		11-1
automake 1.10		Skip
automake 1.11		Skip
automake 1.12		Skip
automake 1.13		Skip
automake 1.14		Skip
automake 1.15		Skip
automake 1.16		1.16.1-1
automake 1.4		Skip
automake 1.5		Skip
automake 1.6		Skip
automake 1.7		Skip
automake 1.8		Skip
automake 1.9		Skip
cmake	3.17.3-2	Keep
cmake-debuginfo		Skip
cmake-doc	3.17.3-2	Keep
cmake-gui	3.17.3-2	Keep
emacs-cmake		Skip

Change the search from make to sndfile. Select the following files libsndfile-devel, libsndfile-utils and libsndfile1. [mingw64-i686-lsndfile](#) and [mingw64-x86_64-lsndfile](#)

Package	Current	New
libsndfile-debuginfo		Skip
libsndfile-devel		1.0.28-2
libsndfile-utils		1.0.28-2
libsndfile1		1.0.28-2
mingw64-i686-lsndfile		Skip
mingw64-x86_64-lsndfile		Skip

Change the search from sndfile to git. Select git. [There are no 64bit OS versions.](#)

Package	Current	New
cgit		Skip
cgit-debuginfo		Skip
engauge-digitizer		Skip
engauge-digitizer-debuginfo		Skip
geany-plugins-git-changebar		Skip
girepository-Ggit 1.0		Skip
git		2.28.0-1
git-archive-all		Skip
git-clann-format		Skip

Change the search from git to wget. Select wget. [There are no 64bit OS versions.](#)

Package	Current	New
pwget		Skip
wget		1.19.1-2
wget-debuginfo		Skip

Change the search from wget to zip. Select bzip2, gzip, zip and unzip. **There are no 64bit OS versions.**

Package	Current	New
bzip2	1.0.8-1	Keep
bzip2-debuginfo		Skip
cygwin64-minizip		Skip
forackzip		Skip
gzip	1.8-1	Keep
gzip-debuginfo		Skip
rzip		Skip
unzip		6.0-17
unzip-debuginfo		Skip
zip		3.0-12
zip-debuginfo		Skip

Change the search from zip to lapack. Select liblapack-devel, liblapack-doc and liblapack0. **mingw64-i686-lapack and mingw64-x86_64-lapack.**

Package	Current	New
lapack-debuginfo		Skip
liblapack-devel		3.9.0-2
liblapack-doc		3.9.0-2
liblapack0		3.9.0-2
mingw64-i686-lapack		Skip
mingw64-x86_64-lapack		Skip

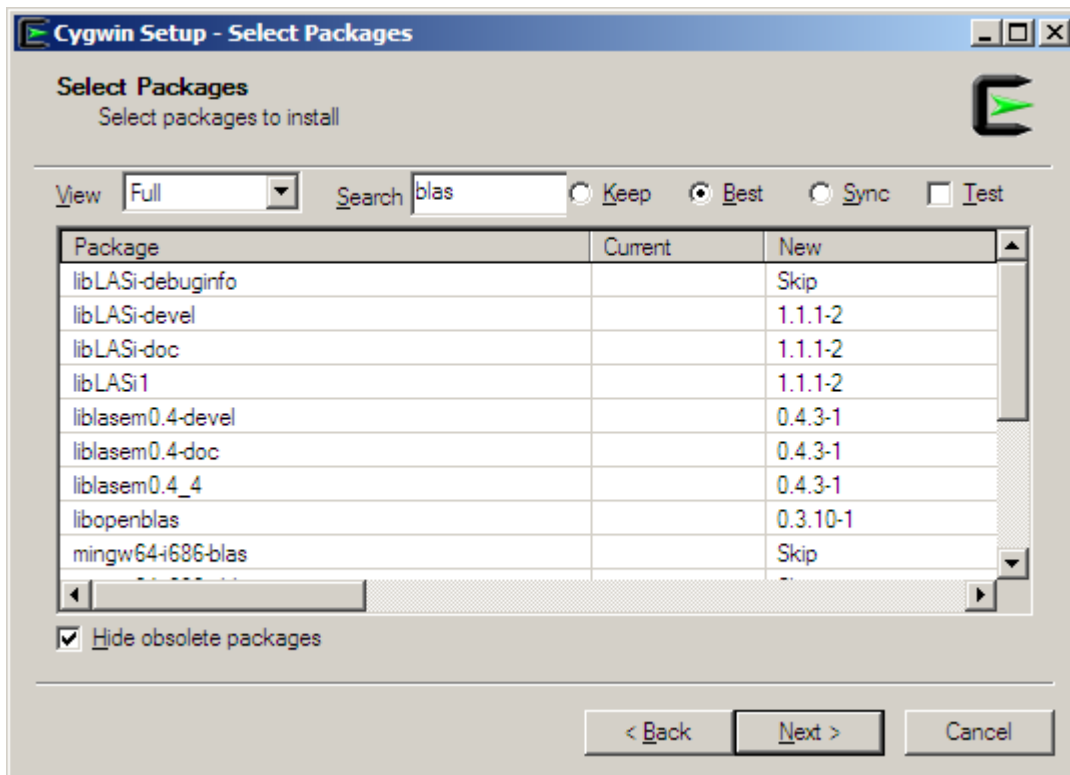
Change the search from lapack to fft. Select fftw3, fftw3-doc, libfftw3-devel, libfftw3-omp3 and libfftw3_3. **mingw64-i686-fftw3 and mingw64-x86_64-fftw3.**

Package	Current	New
fftw3		3.3.8-1
fftw3-debuginfo		Skip
fftw3-doc		3.3.8-1
libfftw3-devel		3.3.8-1
libfftw3-omp3		3.3.8-1
libfftw3_3		3.3.8-1
mingw64-i686-fftw3		Skip
mingw64-x86_64-fftw3		Skip

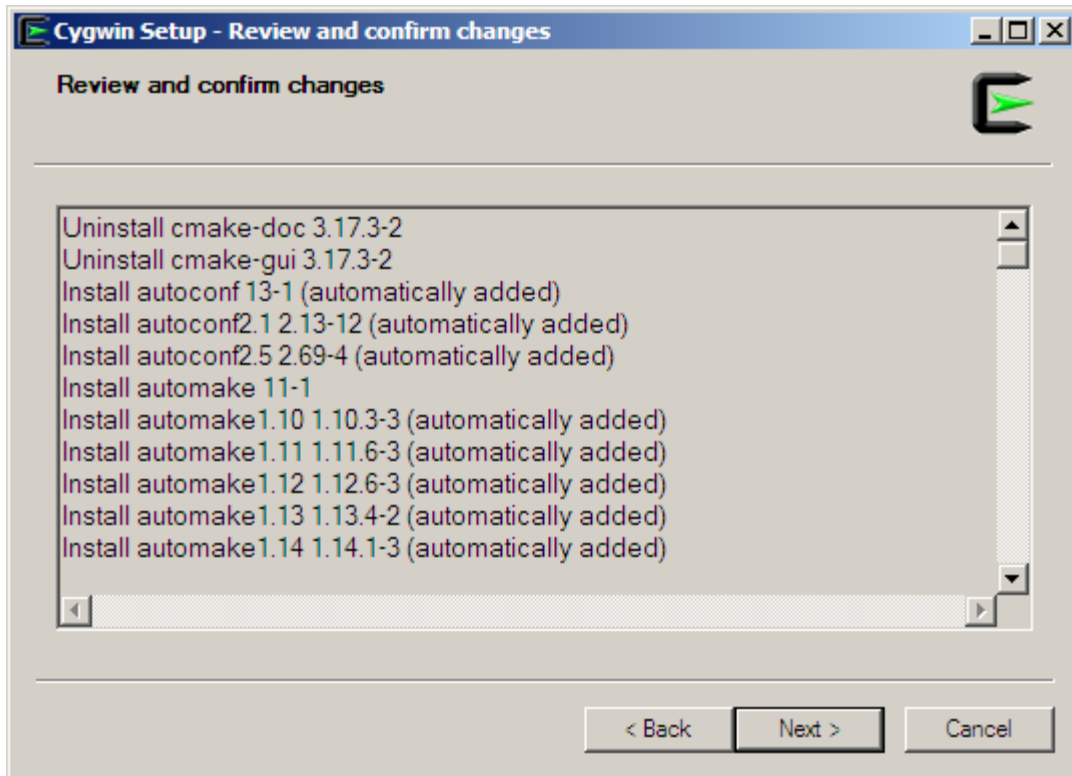
Change the search from fft to blas. Select libLASi-devel, libLASi-doc, liblasl1, liblasem0.4-devel, liblasem0.4-doc, liblasem0.4_4, libopenblas and openblas.doc. **mingw64-i686-libblas, mingw64-i686-libcblas, mingw64-i686-libLASi and mingw64-x86_64-blas, mingw64-x86_64-cblas, mingw64-x86_64-libLASi.**

Package	Current	New
libLASi-debuginfo		Skip
libLASi-devel		1.1.1-2
libLASi-doc		1.1.1-2
libLASi1		1.1.1-2
liblasem0.4-devel		0.4.3-1
liblasem0.4-doc		0.4.3-1
liblasem0.4_4		0.4.3-1
libopenblas		0.3.10-1
mingw64-i686-blas		Skip
mingw64-i686-cblas		Skip
mingw64-i686-libLASi		Skip
mingw64-x86_64-blas		Skip
mingw64-x86_64-cblas		Skip
mingw64-x86_64-libLASi		Skip
openblas-debuginfo		Skip
openblas-doc		0.3.10-1

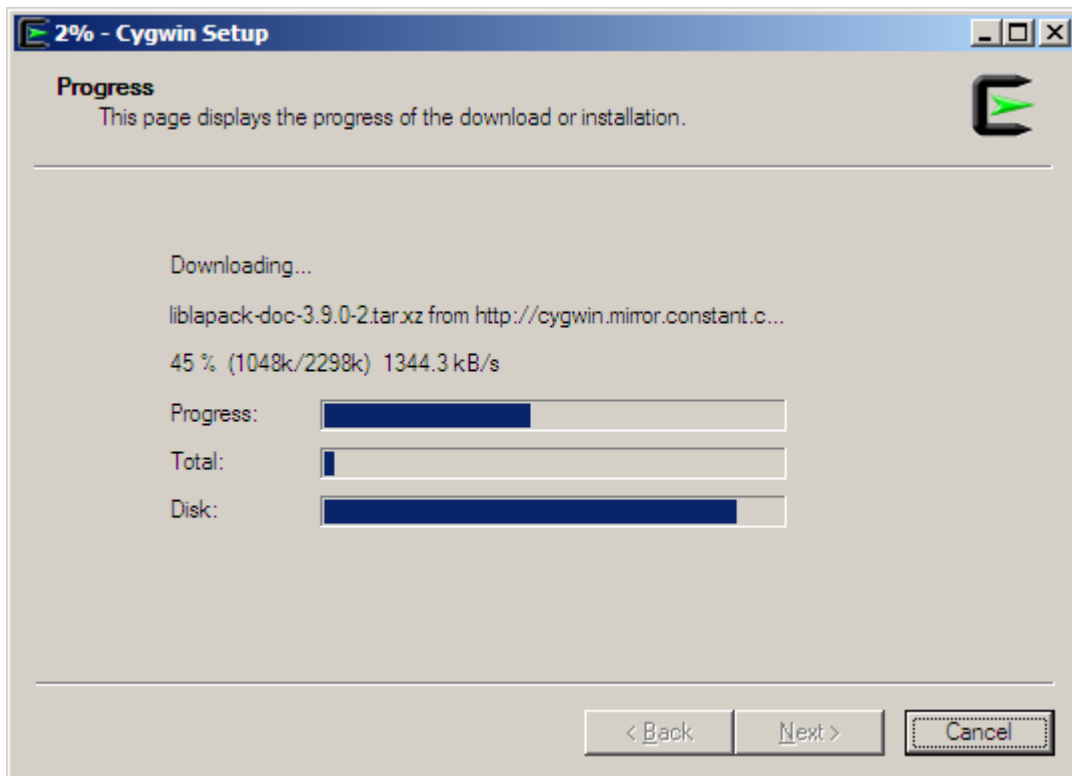
We have now selected are all the dependencies needed. We now need to download and install them. At the bottom of the window there is a next button click on it.



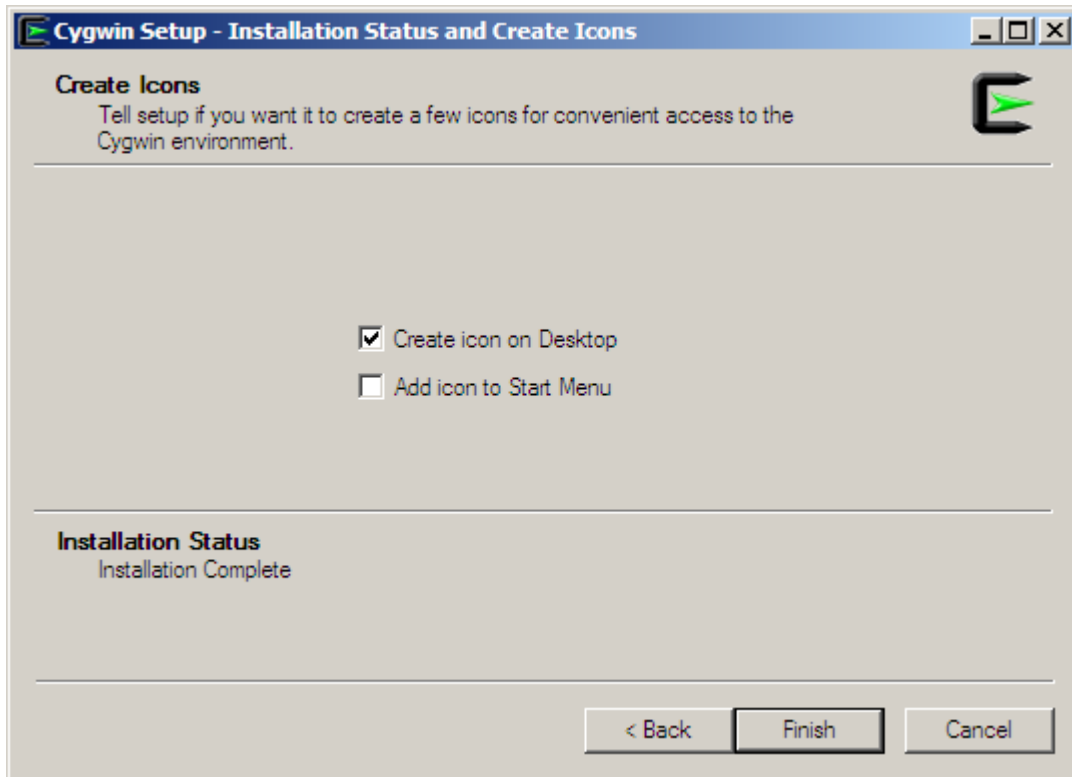
This opens a window asking you to confirm the changes. Click next and the program will start downloading and installing the dependencies selected.



Click next and the program will start downloading and installing the dependencies selected.



When all the files have downloaded and installed you will see this. We suggest you select making the desktop icon as it makes opening and running the Cygwin commands a lot easier.



Select finish and you are done downloading and installing the dependencies.

Step 3: Building Itt and mbelib .dlls

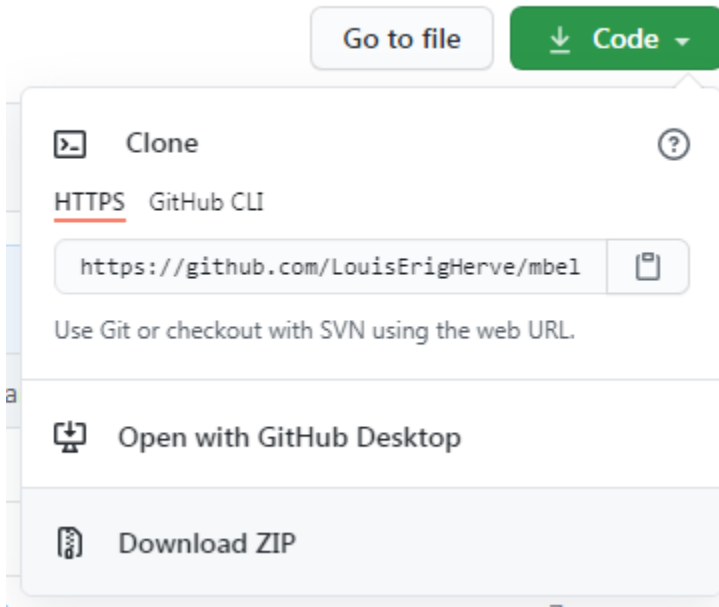
NOTE: Cygwin does not like folders with too many dashes, spaces in the names. Try to use single word folder names.

Go to <https://sourceforge.net/projects/itpp/> and download the itt install files. Place the files in a known folder and unzip them. Use WinRar to decompress the .bz2 file directly. Check that the files all extracted to a single folder directly and not an nested folder pair like itpp-4.3.1/itpp-4.3.1/**files are here**.

Name ^	Date modified
itpp-4.3.1	9/23/2020 12:35 PM

Name ^	Date modified	Type	Size
cmake	9/23/2020 12:34 PM	File folder	
doc	9/23/2020 12:34 PM	File folder	
extras	9/23/2020 12:34 PM	File folder	
gtests	9/23/2020 12:34 PM	File folder	
itpp	9/23/2020 12:35 PM	File folder	
m4	9/23/2020 12:35 PM	File folder	
tests	9/23/2020 12:35 PM	File folder	
win32	9/23/2020 12:35 PM	File folder	
AUTHORS	7/6/2013 5:11 AM	File	2 KB
autogen.sh	7/6/2013 5:11 AM	SH File	2 KB
ChangeLog	7/6/2013 5:11 AM	File	12 KB
ChangeLog-2005	7/6/2013 5:11 AM	File	21 KB

Do the same for the mbelib files. They are at <https://github.com/LouisErigHerve/mbelib>. click on the green code and select download as zip.



For ease of compiling place the unzipped files in the same master folder as the .itt files above. The mbelib files will extract to a folder called mbelib-master. This folder contains a second folder called mbelib-master. For ease of compiling cut and paste all the files from this second folder back into the first folder named mbelib-master. Then rename the folder mbelib.

Name ^	Date modified
itpp-4.3.1	9/23/2020 12:35 PM
mbelib-master	9/23/2020 10:57 AM

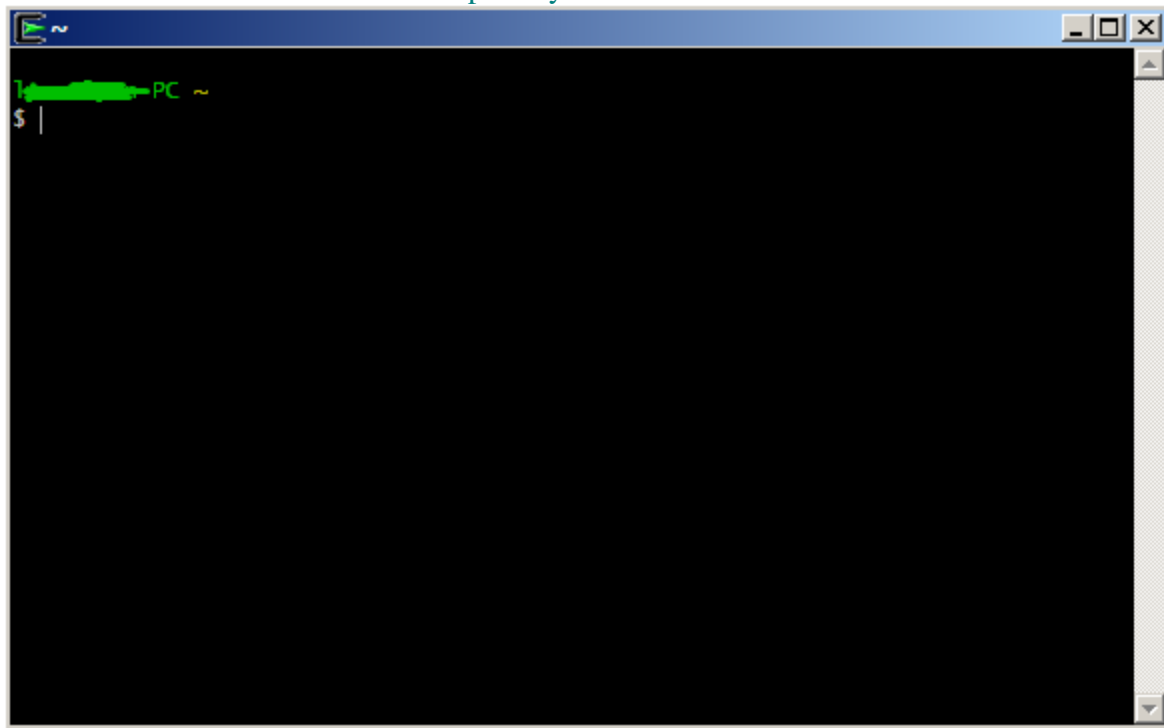
Name ^	Date modified	
itpp-4.3.1	9/23/2020 12:35 PM	
mbelib	9/23/2020 10:57 AM	
Name ^	Date modified	Type
debian	9/23/2020 10:56 AM	File folder
test	9/23/2020 10:56 AM	File folder
.gitignore	9/23/2020 10:56 AM	GITIGNORE File
.travis.yml	9/23/2020 10:56 AM	YML File
ambe3600x2400.c	9/23/2020 10:56 AM	C Source file

We are now ready to compile the files.

NOTE: The itt files will be compiled first and in detail. The mbelib files are compiled by identical steps with just the paths being changed to represent the different file.

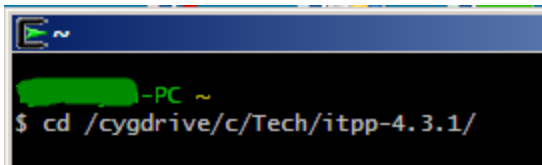
Run Cygwin this is a command prompt type application so you will be typing in commands and then hitting enter.

NOTE: the PC name is redacted for privacy.



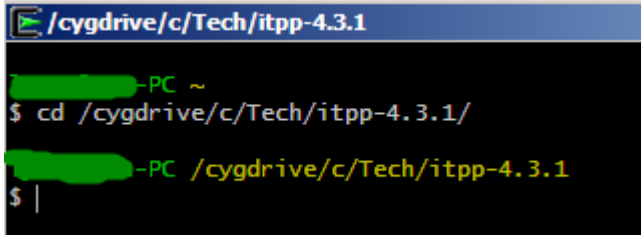
NOTE: On all Cygwin commands watch out for spaces, upper/lower case, oddball characters and spelling.

The first command to enter is. `cd /cygdrive/c/your folder name with itt and mbelib files from above/itt4.3.1`. We are using Tech as our folder name so we will type in `cd /cygdrive/c/Tech/itpp-4.3.1`.



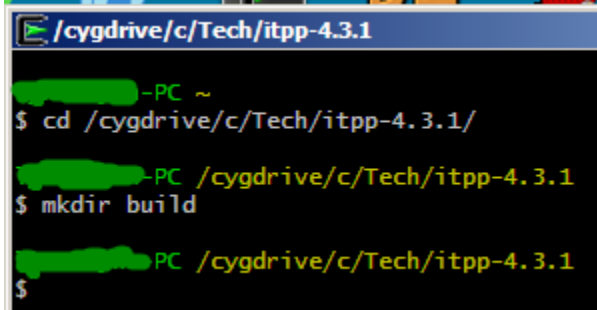
```
~  
-PC ~  
$ cd /cygdrive/c/Tech/itpp-4.3.1/
```

Hit enter and the command is processed and Cygwin is ready for the next command. You will notice Cygwin is now looking at the itpp folder and its contents.



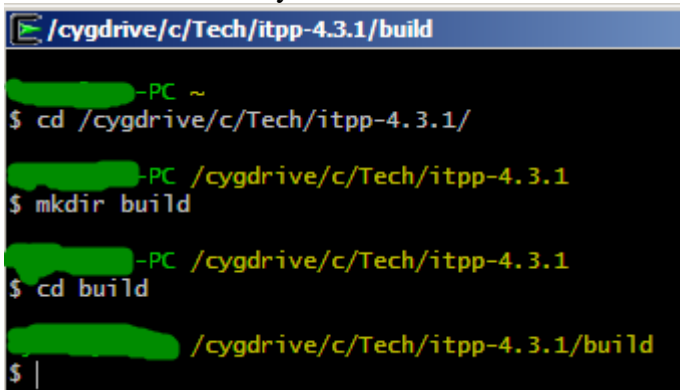
```
/cygdrive/c/Tech/itpp-4.3.1  
-PC ~  
$ cd /cygdrive/c/Tech/itpp-4.3.1/  
-PC /cygdrive/c/Tech/itpp-4.3.1  
$ |
```

Next we create the folder the compiled files will be placed into. This is done by using the command `mkdir build` and hit enter to process it. This says make directory called build.



```
/cygdrive/c/Tech/itpp-4.3.1  
-PC ~  
$ cd /cygdrive/c/Tech/itpp-4.3.1/  
-PC /cygdrive/c/Tech/itpp-4.3.1  
$ mkdir build  
-PC /cygdrive/c/Tech/itpp-4.3.1  
$
```

Next we need to navigate to the build folder just created. This is done with the `cd build` command followed by enter.



```
/cygdrive/c/Tech/itpp-4.3.1/build  
-PC ~  
$ cd /cygdrive/c/Tech/itpp-4.3.1/  
-PC /cygdrive/c/Tech/itpp-4.3.1  
$ mkdir build  
-PC /cygdrive/c/Tech/itpp-4.3.1  
$ cd build  
-PC /cygdrive/c/Tech/itpp-4.3.1/build  
$ |
```

The next command is `cmake ..` followed by enter this tells Cygwin to start compiling the files.

```
/cygdrive/c/Tech/itpp-4.3.1/build
[redacted]-PC ~
$ cd /cygdrive/c/Tech/itpp-4.3.1/

[redacted]-PC /cygdrive/c/Tech/itpp-4.3.1
$ mkdir build

[redacted]-PC /cygdrive/c/Tech/itpp-4.3.1
$ cd build

[redacted]-PC /cygdrive/c/Tech/itpp-4.3.1/build
$ cmake ..
```

Once you hit enter you will see Cygwin processing the command.

NOTE: If you get an error like this 1 [main] cmake 2054 child_info_fork::abort: \\?\\C:\\Cygwin\\bin\\cygcrypto-1.1.dll: Loaded to different address: parent(0xD10000) != child(0xA30000). The solution found is to remove (delete it) the Cygwing folder from C:\\ drive. Turn off your antivirus and run the setup to reinstall Cygwin. Let it download and install everything including dependency files. Keep the antivirus turned off until everything in this document is done then turn it back on.

```
/cygdrive/c/Tech/itpp-4.3.1/build
[redacted]-PC ~
$ cd /cygdrive/c/Tech/itpp-4.3.1/

[redacted]-PC /cygdrive/c/Tech/itpp-4.3.1
$ mkdir build

[redacted]-PC /cygdrive/c/Tech/itpp-4.3.1
$ cd build

[redacted]-PC /cygdrive/c/Tech/itpp-4.3.1/build
$ cmake ..
-- The C compiler identification is GNU 10.2.0
-- The CXX compiler identification is GNU 10.2.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc - works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++.exe
-- Check for working CXX compiler: /usr/bin/c++.exe - works
-- Detecting CXX compiler ABI info
-- BLAS vendor: All
-- Configuring done
-- Generating done
-- Build files have been written to: /cygdrive/c/Tech/itpp-4.3.1/build

lynn@lynn-PC /cygdrive/c/Tech/itpp-4.3.1/build
$
```

Next use the make -j4 command followed by enter. Again you will see Cygwin processing the command.

```
/cygdrive/c/Tech/itpp-4.3.1/build
$ make -j4
```

```
/cygdrive/c/Tech/itpp-4.3.1/build
$ make -j4
Scanning dependencies of target itpp
[ 0%] Building CXX object itpp/CMakeFiles/itpp.dir/base/algebra/cholesky.cpp.o
[ 1%] Building CXX object itpp/CMakeFiles/itpp.dir/base/algebra/det.cpp.o
[ 2%] Building CXX object itpp/CMakeFiles/itpp.dir/base/algebra/eigen.cpp.o
[ 3%] Building CXX object itpp/CMakeFiles/itpp.dir/base/algebra/inv.cpp.o
[ 4%] Building CXX object itpp/CMakeFiles/itpp.dir/base/algebra/ls_solve.cpp.o
[ 5%] Building CXX object itpp/CMakeFiles/itpp.dir/base/algebra/lu.cpp.o
[ 5%] Building CXX object itpp/CMakeFiles/itpp.dir/base/algebra/qr.cpp.o
[ 6%] Building CXX object itpp/CMakeFiles/itpp.dir/base/algebra/schur.cpp.o
[ 9%] Building CXX object itpp/CMakeFiles/itpp.dir/base/bessel/airy.cpp.o
[ 9%] Building CXX object itpp/CMakeFiles/itpp.dir/base/algebra/svd.cpp.o
[ 9%] Building CXX object itpp/CMakeFiles/itpp.dir/base/bessel.cpp.o
[10%] Building CXX object itpp/CMakeFiles/itpp.dir/base/bessel/chbev1.cpp.o
[11%] Building CXX object itpp/CMakeFiles/itpp.dir/base/bessel/gamma.cpp.o
[11%] Building CXX object itpp/CMakeFiles/itpp.dir/base/bessel/hyperg.cpp.o
```

```
/cygdrive/c/Tech/itpp-4.3.1/build
[ 83%] Building CXX object itpp/CMakeFiles/itpp.dir/signal/freq_filt.cpp.o
[ 84%] Building CXX object itpp/CMakeFiles/itpp.dir/signal/poly.cpp.o
[ 85%] Building CXX object itpp/CMakeFiles/itpp.dir/signal/resampling.cpp.o
[ 86%] Building CXX object itpp/CMakeFiles/itpp.dir/signal/sigfun.cpp.o
[ 87%] Building CXX object itpp/CMakeFiles/itpp.dir/signal/source.cpp.o
[ 88%] Building CXX object itpp/CMakeFiles/itpp.dir/signal/transforms.cpp.o
[ 88%] Building CXX object itpp/CMakeFiles/itpp.dir/signal/window.cpp.o
[ 89%] Building CXX object itpp/CMakeFiles/itpp.dir/srccode/audiofile.cpp.o
[ 90%] Building CXX object itpp/CMakeFiles/itpp.dir/srccode/g711.cpp.o
[ 91%] Building CXX object itpp/CMakeFiles/itpp.dir/srccode/gmm.cpp.o
[ 92%] Building CXX object itpp/CMakeFiles/itpp.dir/srccode/lpcfunc.cpp.o
[ 93%] Building CXX object itpp/CMakeFiles/itpp.dir/srccode/pnm.cpp.o
[ 94%] Building CXX object itpp/CMakeFiles/itpp.dir/srccode/vq.cpp.o
[ 94%] Building CXX object itpp/CMakeFiles/itpp.dir/srccode/vqtrain.cpp.o
[ 95%] Building CXX object itpp/CMakeFiles/itpp.dir/stat/misc_stat.cpp.o
[ 96%] Building CXX object itpp/CMakeFiles/itpp.dir/stat/mog_diag.cpp.o
[ 97%] Building CXX object itpp/CMakeFiles/itpp.dir/stat/mog_diag_em.cpp.o
[ 98%] Building CXX object itpp/CMakeFiles/itpp.dir/stat/mog_diag_kmeans.cpp.o
[ 99%] Building CXX object itpp/CMakeFiles/itpp.dir/stat/mog_generic.cpp.o
[100%] Linking CXX shared library cygitpp-8.dll
[100%] Built target itpp

lynn@lynn-PC /cygdrive/c/Tech/itpp-4.3.1/build
$
```

When completed the cygotpp-8.dll will have been created. The final step will be to install the.dll so that it can be used. This is done with the make install command followed by enter. As expected Cygwin will process the command.

```
[100%] Linking CXX shared library cygitpp-8.dll
[100%] Built target itpp

-PC /cygdrive/c/Tech/itpp-4.3.1/build
$ make install
```

```
/cygdrive/c/Tech/itpp-4.3.1/build
-- Installing: /usr/local/include/itpp/base/mat.h
-- Installing: /usr/local/include/itpp/base/matfunc.h
-- Installing: /usr/local/include/itpp/base/math
-- Installing: /usr/local/include/itpp/base/math/elem_math.h
-- Installing: /usr/local/include/itpp/base/math/error.h
-- Installing: /usr/local/include/itpp/base/math/integration.h
-- Installing: /usr/local/include/itpp/base/math/log_exp.h
-- Installing: /usr/local/include/itpp/base/math/min_max.h
-- Installing: /usr/local/include/itpp/base/math/misc.h
-- Installing: /usr/local/include/itpp/base/math/trig_hyp.h
-- Installing: /usr/local/include/itpp/base/operators.h
-- Installing: /usr/local/include/itpp/base/parser.h
-- Installing: /usr/local/include/itpp/base/random.h
-- Installing: /usr/local/include/itpp/base/random_dsfmt.h
-- Installing: /usr/local/include/itpp/base/smat.h
-- Installing: /usr/local/include/itpp/base/sort.h
-- Installing: /usr/local/include/itpp/base/specmat.h
-- Installing: /usr/local/include/itpp/base/stack.h
-- Installing: /usr/local/include/itpp/base/svec.h
-- Installing: /usr/local/include/itpp/base/timing.h
-- Installing: /usr/local/include/itpp/base/vec.h
-- Installing: /usr/local/include/itpp/comm
-- Installing: /usr/local/include/itpp/comm/bch.h
```

This concludes the compiling and installation of ittp. Next is the installation of mbelib. The commands used are identical with the exception of the first. The difference being the file name is changed to mbelib from itpp-4.3.1.

NOTE: you can continue these steps right after the itpp install. There is no need to close and restart Cygwin.

`cd /cygdrive/c/Tech/mbelib/`

```
-PC /cygdrive/c/Tech/itpp-4.3.1/build
$ cd /cygdrive/c/Tech/mbelib/

-PC /cygdrive/c/Tech/mbelib
$
```

`mkdir build`

```
-PC /cygdrive/c/Tech/itpp-4.3.1/build
$ cd /cygdrive/c/Tech/mbelib/

-PC /cygdrive/c/Tech/mbelib
$ mkdir build

-PC /cygdrive/c/Tech/mbelib
$ |
```

`cd build`

```

[redacted]-PC /cygdrive/c/Tech/itpp-4.3.1/build
$ cd /cygdrive/c/Tech/mbelib/

[redacted]-PC /cygdrive/c/Tech/mbelib
$ mkdir build

[redacted]-PC /cygdrive/c/Tech/mbelib
$ cd build

[redacted]-PC /cygdrive/c/Tech/mbelib/build
$ |

```

cmake ..

```

[redacted]-PC /cygdrive/c/Tech/itpp-4.3.1/build
$ cd /cygdrive/c/Tech/mbelib/

[redacted]-PC /cygdrive/c/Tech/mbelib
$ mkdir build

[redacted]-PC /cygdrive/c/Tech/mbelib
$ cd build

[redacted]-PC /cygdrive/c/Tech/mbelib/build
$ cmake ..|

```

```

/cygdrive/c/Tech/mbelib/build
ge):
CMake no longer defines WIN32 on Cygwin!

(1) If you are just trying to build this project, ignore this warning or
quiet it by setting CMAKE_LEGACY_CYGWIN_WIN32=0 in your environment or in
the CMake cache. If later configuration or build errors occur then this
project may have been written under the assumption that Cygwin is WIN32.
In that case, set CMAKE_LEGACY_CYGWIN_WIN32=1 instead.

(2) If you are developing this project, add the line

    set(CMAKE_LEGACY_CYGWIN_WIN32 0) # Remove when CMake >= 2.8.4 is required

at the top of your top-level CMakeLists.txt file or set the minimum
required version of CMake to 2.8.4 or higher. Then teach your project to
build on Cygwin without WIN32.
Call Stack (most recent call first):
  /usr/share/cmake-3.17.3/Modules/CMakeSystemSpecificInformation.cmake:26 (inclu
de)
  CMakeLists.txt:1 (project)

-- Check for working C compiler: /usr/bin/cc

-- Generating done
-- Build files have been written to: /cygdrive/c/Tech/mbelib/build

[redacted]-PC /cygdrive/c/Tech/mbelib/build
$ |

```

make -j4 install

NOTE: this is a combined command. You can use the `$ make -j4` command and when completed use the `make install` command.

```
-PC /cygdrive/c/Tech/mbelib/build
$ make -j4 install

/cygdrive/c/Tech/mbelib/build

-PC /cygdrive/c/Tech/mbelib/build
$ make -j4 install
Scanning dependencies of target mbe-shared
Scanning dependencies of target mbe-static
Scanning dependencies of target gmock
Scanning dependencies of target gtest
[ 3%] Building C object CMakeFiles/mbe-shared.dir/ambe3600x2400.c.o
[ 7%] Building C object CMakeFiles/mbe-static.dir/ambe3600x2400.c.o
[ 11%] Building CXX object test/gtest/CMakeFiles/gtest.dir/src/gtest-all.cc.o
[ 15%] Building CXX object test/gmock/CMakeFiles/gmock.dir/src/gmock-all.cc.o
[ 19%] Building C object CMakeFiles/mbe-shared.dir/ambe3600x2450.c.o
[ 23%] Building C object CMakeFiles/mbe-static.dir/ambe3600x2450.c.o
Install the project...
-- Install configuration: ""
-- Installing: /usr/local/lib/pkgconfig/libmbe.pc
-- Installing: /usr/local/lib/libmbe.a
-- Installing: /usr/local/include/mbelib.h
-- Installing: /usr/local/lib/libmbe.dll.a
-- Up-to-date: /usr/local/include/mbelib.h

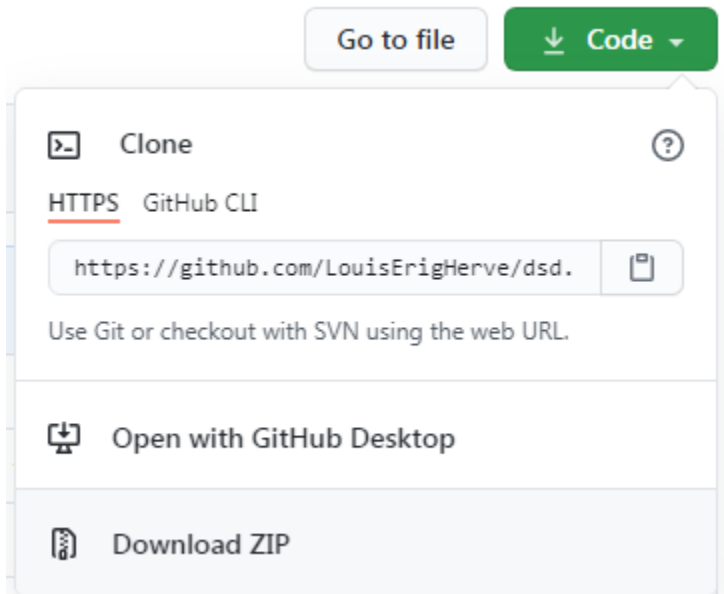
-PC /cygdrive/c/Tech/mbelib/build
$
```

When completed the `cygmbe-1.dll` will have been created and installed. This completes the building and installation of `mbelib`.

NOTE: you do not need to shut down Cygwin if you wish to continue with the building of DSD

Step 4: Building DSD v1.8.4.

Get the files for v1.8.4. They are at <https://github.com/LouisErigHerve/dsd>. Click on the green code and select download as zip.



Like in the steps above unzip and place the files in the same folder you used. Again making sure you don't have the nested file folders.

Name ^	Date modified		
dsd	9/23/2020 4:04 PM		
itpp-4.3.1	9/23/2020 3:13 PM		
mbelib	9/23/2020 3:36 PM		
Name ^	Date modified	Type	Size
cmake	9/23/2020 4:03 PM	File folder	
debian	9/23/2020 4:03 PM	File folder	
include	9/23/2020 4:03 PM	File folder	
src	9/23/2020 4:03 PM	File folder	
test	9/23/2020 4:03 PM	File folder	
.gitignore	9/23/2020 4:03 PM	GITIGNORE File	1 KB
.travis.yml	9/23/2020 4:03 PM	YML File	1 KB
CHANGELOG	9/23/2020 4:03 PM	File	5 KB

If you did not shut down Cygwin you can now continue to enter in the commands. If you did shut Cygwin restart it. With Cygwin running type in this command `cd`

`/cygdrive/c/Tech/dsd/`

NOTE: remember to edit it to go to the folder you have created for the install files in step3 above.

NOTE: We did not shut down Cygwin so the image below still shows the previous folder information in yellow.

`cd /cygdrive/c/Tech/dsd/`

```
██████████-PC /cygdrive/c/Tech/mbelib/build
$ cd /cygdrive/c/Tech/dsd/

██████████-PC /cygdrive/c/Tech/dsd
$ |
```

mkdir build

```
██████████-PC /cygdrive/c/Tech/mbelib/build
$ cd /cygdrive/c/Tech/dsd/

██████████-PC /cygdrive/c/Tech/dsd
$ mkdir build

██████████-PC /cygdrive/c/Tech/dsd
$
```

cd build

```
██████████-PC /cygdrive/c/Tech/mbelib/build
$ cd /cygdrive/c/Tech/dsd/

██████████-PC /cygdrive/c/Tech/dsd
$ mkdir build

██████████-PC /cygdrive/c/Tech/dsd
$ cd build

██████████-PC /cygdrive/c/Tech/dsd/build
$ |
```

cmake ..

```
██████████-PC /cygdrive/c/Tech/mbelib/build
$ cd /cygdrive/c/Tech/dsd/

██████████-PC /cygdrive/c/Tech/dsd
$ mkdir build

██████████-PC /cygdrive/c/Tech/dsd
$ cd build

██████████-PC /cygdrive/c/Tech/dsd/build
$ cmake ..|
```

```
/cygdrive/c/Tech/dsd/build
ge):
CMake no longer defines WIN32 on Cygwin!

(1) If you are just trying to build this project, ignore this warning or
quiet it by setting CMAKE_LEGACY_CYGWIN_WIN32=0 in your environment or in
the CMake cache. If later configuration or build errors occur then this
project may have been written under the assumption that Cygwin is WIN32.
In that case, set CMAKE_LEGACY_CYGWIN_WIN32=1 instead.

(2) If you are developing this project, add the line

    set(CMAKE_LEGACY_CYGWIN_WIN32 0) # Remove when CMake >= 2.8.4 is required

at the top of your top-level CMakeLists.txt file or set the minimum
required version of CMake to 2.8.4 or higher. Then teach your project to
build on Cygwin without WIN32.
Call Stack (most recent call first):
  /usr/share/cmake-3.17.3/Modules/CMakeSystemSpecificInformation.cmake:26 (inclu
de)
  CMakeLists.txt:1 (project)

-- Check for working C compiler: /usr/bin/cc

-- Configuring done
-- Generating done
-- Build files have been written to: /cygdrive/c/Tech/dsd/build
[redacted]-PC /cygdrive/c/Tech/dsd/build
$
```

[make](#)

NOTE: the command is make and not make -j4

```
-- Configuring done
-- Generating done
-- Build files have been written to: /cygdrive/c/Tech/dsd/build
[redacted]-PC /cygdrive/c/Tech/dsd/build
$ make
```

```
/cygdrive/c/Tech/dsd/build
lynn@lynn-PC /cygdrive/c/Tech/dsd/build
$ make
Scanning dependencies of target dsd
[ 1%] Building CXX object CMakeFiles/dsd.dir/src/Hamming.cpp.o
[ 3%] Building C object CMakeFiles/dsd.dir/src/bptc.c.o
[ 4%] Building C object CMakeFiles/dsd.dir/src/dmr_data.c.o
[ 6%] Building C object CMakeFiles/dsd.dir/src/dmr_encryption.c.o
[ 7%] Building C object CMakeFiles/dsd.dir/src/dmr_sync.c.o
[ 9%] Building C object CMakeFiles/dsd.dir/src/dmr_voice.c.o
[10%] Building C object CMakeFiles/dsd.dir/src/dpmr_data.c.o
[12%] Building C object CMakeFiles/dsd.dir/src/dpmr_voice.c.o
[14%] Building C object CMakeFiles/dsd.dir/src/dsd_audio.c.o
[15%] Building C object CMakeFiles/dsd.dir/src/dsd_dibit.c.o
[17%] Building C object CMakeFiles/dsd.dir/src/dsd_file.c.o
/cygdrive/c/Tech/dsd/src/dsd_file.c: In function 'closeMbeOutFile':
/cygdrive/c/Tech/dsd/src/dsd_file.c:179:7: warning: implicit declaration of function 'strptime'; did you mean 'strftime'? [-Wimplicit-function-declaration]
 179 |         strptime (opts->mbe_out_file, "%s.imb", &timep);
      |         ^~~~~~
      |         strftime
[18%] Building C object CMakeFiles/dsd.dir/src/dsd_filters.c.o
[20%] Building C object CMakeFiles/dsd.dir/src/dsd_frame.c.o
c.o
[100%] Linking CXX static library libgmock_main.a
[100%] Built target gmock_main
-PC /cygdrive/c/Tech/dsd/build
$
```

make install

```
[100%] Linking CXX static library libgmock_main.a
[100%] Built target gmock_main
-PC /cygdrive/c/Tech/dsd/build
$ make install
-PC /cygdrive/c/Tech/dsd/build
$ make install
[ 76%] Built target dsd
[ 79%] Built target gmock
[ 82%] Built target gtest
[ 92%] Built target dsdtest
[ 95%] Built target gtest_main
[100%] Built target gmock_main
Install the project...
-- Install configuration: ""
-- Installing: /usr/local/bin/dsd.exe
-PC /cygdrive/c/Tech/dsd/build
$
```

This concludes the build instructions for DSD v1.8.4. The program can be run either from the build file or the files can be copied from the build folder to another folder more convenient.