Simple Instructions for Installing LBLRTM in a Windows Environment Using MINGW

The follow set of instructions provides a list of basis steps that can be used to create a Microsoft Window's version of lblrtm using open source tools. It is by no means the only method for constructing a command line executable version of lblrtm, and is intended for use in Windows environment with little or no installed software development tools. Visual Studio (with an installed version of the Intel Fortran compiler) can also be used to create a command line version of lblrtm as well as lnfl. A prpject file is int included as part of the standard lblrtm distribution, but any of the Intel tags found in the makefile.common, make_lblrtm and male_lnfl can be used as a template for developing the appropriated project(s). A list of input source files is provided in section 3 of this document to assist in this process.

1 Steps for Building LBLRTM

The following are simple step-by-step instruction for building LBLRTM using MinGW. They include instructions for downloading and installing MinGW itself as well as instructions for distributing a pre-build version of LBLRTM using MinGW.

1) **Download LBLRTM**: Get latest version of LBLRTM from http://rtweb.aer.com
2) **Download MinGW:** Get the latest self-installer for MinGW (mingw-get-install-XXX.exe) from http://sourceforge.net/projects/mingw/
3) **Install:** Run mingw installer (usually located in My Documents\Downloads)
    a. **Select:** Fortran Compiler and MSYS Basic System at a minimum from "Select Component Menu"
4) **Open shell:** Launch a MinGW shell using Start -> All Programs-> MinGW -> MinGW Shell
5) **Copy/Move:** Copy aerlbl_v12.0_package.tar.gz to MinGW user home directory.**Example:**

    $ cp /c/*location of downloaded gzipped tarball*/aerlbl_v12.0_package.tar.gz .

6) **Untar:** Uncompress the required files

    $ tar –xzf aerlbl_v12.0_package.tar.gz
    $ tar –xzf aerlbl_v12.0.tar.gz

7) **Build:** Build the lblrtm executable

    $ cd lblrtm

```
$ cd build
$ make -f  make_lblrtm mingwGNUdbl
```
or
```
$ make -f  make_lblrtm mingwGNUsgl
```
The mingwGNUdbl tag will create a double precision version of lblrtm, and the mingwGNUdbl tag will compile and build a single precision version.

8) **Testing**
Mingw lblrtm can be tested from the base lblrtm directory using the command sequence listed below. This sequence tests the single precision version. The double precision version can be tested by substituting the sgl with dbl in the commands below.

```
$ cd run_examples/run_example_built_in_atm_upwelling
$ cp ../../lblrtm_v12.0_mingw_gnu_sgl .
$ lblrtm_v12.0_mingw_gnu_sgl
```

9) **Distribute:** If one desires to move the executable to a different machine or run this version of lblrtm from a standard windows command line, the following dlls will need to be place in the same directory as the resulting executable: An distributable zip file can be constructed from the base lblrtm directory using the following sequence of commands

```
$ cp /c/MinGW/bin/libgcc*.dll .
$ cp /c/MinGW/bin/libgfortran.dll .
$ zip lblrtmExecutable.zip libgcc*.dll lblrtm_v12.0_mingw_gnu_sgl
```
or
```
$ zip lblrtmExecutable.zip libgcc*.dll lblrtm_v12.0_mingw_gnu_dbl
```

2 Building LNFL

LNFL can also be built and distributed using a slightly modified version of steps 7 and 9 listed above. In this case, the source and makefiles are located in the LNFL tarball provided with the standard LBLRTM distribution. Untar the LNFL tarball, cd to the current LNFL directory and use a similar make command with –f make_lnfl. The only valid build directive for LNFL is mingwGNU_sgl. The double precision option does not apply to LNFL.

3 Notes on Using Visual Studio

Microsoft Visual Studios (with and installed version of the Intel FORTRAN Compiler) can also be used to create a command line version of LBLRTM/LNFL. The required input source files, located in the src directory, are as follows:

LBLRTM: lblrtm.f, oprop.f, contnm.f, xmerge.f, testmm.f, lblatm.f, lbllow.f, postsub.f, pltlbl.f, lbldum.f, solar.f, nonlte.f, (fftscn.f or fftscn_dbl.f) and util_linux_intel.f

LNFL: lnfl.f and util_linux_intel.f