WRF Installation Workflow

Before compiling the WRF code on a computer, check to see if the netCDF library is installed. This is because one of the supported WRF I/O options is netCDF, and it is the one commonly used and supported by the post-processing programs. If the netCDF is installed in a directory other than `/usr/local/`, then find the path, and use the environment variable NETCDF to define where the path is. To do so, type

```
setenv NETCDF path-to-netcdf-library
```

Often the netCDF library and its include/ directory are collocated. If this is not the case, create a directory, link both netCDF lib and include directories in this directory, and use the environment variable to set the path to this directory. For example,

```
netcdf_links/lib -> /netcdf-lib-dir/lib
netcdf_links/include -> /where-include-dir-is/include

setenv NETCDF /directory-where-netcdf_links-is/netcdf_links
```

If the netCDF library is not available on the computer, it needs to be installed first. NetCDF source code or pre-built binary may be downloaded from, and installation instruction can be found on, the Unidata Web page at http://www.unidata.ucar.edu/.

Hint: for Linux users:

If PGI, Intel, gfortran or g95 compilers are are used on a Linux computer, make sure netCDF is installed using the same compiler. Use the NETCDF environment variable to point to the PGI/Intel/g95 compiled netCDF library.

Hint: If using netCDF-4, make sure that the new capabilities (such as parallel I/O based on HDF5) are not activated at the install time, unless you intend to use the compression capability from netCDF-4 (supported in V3.5. More info below).

The WRF source code tar file can be downloaded from http://www.mmm.ucar.edu/wrf/users/download/get_source.html. Once the tar file is unzipped (`gunzip WRFV3.TAR.gz`), and untared (`tar -xf WRFV3.TAR`), it will create a WRFV3/ directory. This contains:

`Makefile`	Top-level makefile
`README`	General information about the WRF/ARW core
`README_test_cases`	Explanation of the test cases
`README.NMM`	General information for the WRF/NMM core
`README.DA`	General information for WRFDA
`README.rsl_output`	Information for dealing with rsl files
`README.io_config`	Information for runtime IO

`README.windtrubine`	Information on using wind farm parameterization
`README.hydro`	Information on WRF-Hydro
`Registry/`	Directory for WRF Registry files
`arch/`	Directory where compile options are gathered
`clean`	script to clean created files and executables
`compile`	script for compiling the WRF code
`configure`	script to create the *configure.wrf* file for compiling
`chem/`	WRF chemistry, supported by NOAA/GSD
`dyn_em/`	Directory for ARW dynamics and numerics
`dyn_exp/`	Directory for a 'toy' dynamic core
`dyn_nmm/`	Directory for NMM dynamics and numerics, supported by DTC
`external/`	Directory that contains external packages, such as those for IO, time keeping and MPI
`frame/`	Directory that contains modules for the WRF framework
`inc/`	Directory that contains 'include' files
`main/`	Directory for main routines, such as wrf.F, and all executables after compilation
`phys/`	Directory for all physics modules
`run/`	Directory where one may run WRF
`share/`	Directory that contains mostly modules for the WRF mediation layer and WRF I/O
`test/`	Directory that contains test case directories, may be used to run WRF
`tools/`	Directory that contains tools for developers

The steps to compile and run the model are:

1. configure: generate a configuration file for compilation
2. compile: compile the code
3. run the model

Go to the WRFV3 (top) directory and type

```
./configure
```

and a list of choices for your computer should appear. These choices range from compiling for a single processor job (serial), to using OpenMP shared-memory (smpar) or distributed-memory parallelization (dmpar) options for multiple processors, or a combination of shared-memory and distributed-memory options (dm+sm). When a selection is made, a second choice for compiling nesting will appear. For example, on a Linux computer, the above steps may look like:

```
> setenv NETCDF /usr/local/netcdf-pgi
> ./configure

checking for perl5... no
```

```
checking for perl... found /usr/bin/perl (perl)
Will use NETCDF in dir: /usr/local/netcdf-pgi32
PHDF5 not set in environment. Will configure WRF for use without.
./configure:  WRF operating system set to "Linux" via environment variable $WRF_OS
Will use 'time' to report timing information
$JASPERLIB or $JASPERINC not found in environment, configuring to build without
grib2 I/O...
------------------------------------------------------------------------
Please select from among the following Linux ARCH options:
 1. (serial)   2. (smpar)   3. (dmpar)   4. (dm+sm)   NEC SX (sxf90/sxcc)
 5. (serial)   6. (smpar)   7. (dmpar)   8. (dm+sm)   GNU (gfortran/gcc)
 9. (serial)               10. (dmpar)                GNU (g95/gcc)
11. (serial)  12. (smpar)  13. (dmpar)  14. (dm+sm)   PGI (pgf90/gcc)
15. (serial)  16. (smpar)  17. (dmpar)  18. (dm+sm)   PGI (pgf90/pgcc): SGI MPT
19. (serial)  20. (smpar)  21. (dmpar)  22. (dm+sm)   PGI (pgf90/gcc): PGI accelerator
23. (serial)  24. (smpar)  25. (dmpar)  26. (dm+sm)   INTEL (ifort/icc)
                      27. (dm+sm)   INTEL (ifort/icc): Xeon Phi (MIC architecture)
28. (serial)  29. (smpar)  30. (dmpar)  31. (dm+sm)   INTEL (ifort/icc): Xeon (SNB with AVX
mods)
32. (serial)  33. (smpar)  34. (dmpar)  35. (dm+sm)   INTEL (ifort/icc): SGI MPT
36. (serial)  37. (smpar)  38. (dmpar)  39. (dm+sm)   INTEL (ifort/icc): IBM POE
40. (serial)  41. (smpar)  42. (dmpar)  43. (dm+sm)   INTEL (ifort/icc): ia64
44. (serial)  45. (smpar)  46. (dmpar)  47. (dm+sm)   INTEL (ifort/icc): SGI Altix
48. (serial)               49. (dmpar)                PATHSCALE (pathf90/pathcc)
50. (serial)  51. (smpar)  52. (dmpar)  53. (dm+sm)   GNU (gfortran/gcc)
54. (serial)  55. (smpar)  56. (dmpar)  57. (dm+sm)   IBM (xlf90_r/cc_r)
58. (serial)  59. (smpar)  60. (dmpar)  61. (dm+sm)   PGI (ftn/gcc): Cray XC CLE
62. (serial)  63. (smpar)  64. (dmpar)  65. (dm+sm)   CRAY CCE (ftn/gcc): Cray XE and XC
66. (serial)  67. (smpar)  68. (dmpar)  69. (dm+sm)   INTEL (ftn/icc): Cray XC
70. (serial)  71. (smpar)  72. (dmpar)  73. (dm+sm)   FUJITSU (frtpx/fccpx): FX10 SPARC64
IXfx
               74. (dmpar)                IBM (blrts_xlf90/blrts_xlc): ppc64 Blue Gene\L
        75. (smpar)  76. (dmpar)  77. (dm+sm)   IBM (mpixlf90_r/mpixlc_r): ppc64 Blue
Gene\P
               78. (dmpar)                IBM (xlf90_r/xlc_r): ppc64 IBM Blade
79. (serial)  80. (smpar)  81. (dmpar)  82. (dm+sm)   PGI (pgf90/pgcc)
83. (serial)  84. (smpar)  85. (dmpar)  86. (dm+sm)   PGI (pgf90/gcc): -f90=pgf90
87. (serial)  88. (smpar)  89. (dmpar)  90. (dm+sm)   PGI (pgf90/pgcc): -f90=pgf90
Enter selection [1-90] :
Compile for nesting? (0=no nesting, 1=basic, 2=preset moves, 3=vortex following)
[default 0]: 1
```

Enter the appropriate options that are best for your computer and application.

When the return key is hit, a `configure.wrf` file will be created. Edit compile options/paths, if necessary.

Hint: It is helpful to start with something simple, such as the serial build. If it is successful, move on to build smpar or dmpar code. Remember to type 'clean –a' between each build.

Hint: If you anticipate generating a netCDF file that is larger than 2Gb (whether it is a single- or multi-time period data [e.g. model history]) file), you may set the following environment variable to activate the large-file support option from netCDF (in c-shell):

```
setenv WRFIO_NCD_LARGE_FILE_SUPPORT 1
```

Hint: If you would like to use parallel netCDF (p-netCDF) developed by Argonne National Lab (http://trac.mcs.anl.gov/projects/parallel-netcdf), you will need to install p-netCDF separately, and use the environment variable PNETCDF to set the path:

```
setenv PNETCDF path-to-pnetcdf-library
```

Hint: Since V3.5, compilation may take a bit longer due to the addition of the CLM4 module. If you do not intend to use the CLM4 land-surface model option, you can modify your `configure.wrf` file by removing `-DWRF_USE_CLM` from `ARCH_LOCAL`.

To compile the code, type

```
./compile
```

and the following choices will appear:

```
Usage:

compile wrf           compile wrf in run dir (Note, no real.exe, ndown.exe or
ideal.exe generated)

or choose a test case (see README_test_cases for details):

    compile em_b_wave
    compile em_convrad (new in V3.7)

    compile em_esmf_exp (example only)
    compile em_grav2d_x

    compile em_heldsuarez
    compile em_hill2d_x

    compile em_les
    compile em_quarter_ss
    compile em_real
```

```
      compile em_seabreeze2d_x
      compile em_squall2d_x
      compile em_squall2d_y
      compile em_tropical_cyclone
      compile em_tropical_cyclone

      compile exp_real  (example of a toy solver)

      compile nmm_real  (NMM solver)

  compile -h              help message
```

where **em** stands for the Advanced Research WRF dynamic solver (which currently is the '**E**ulerian **m**ass-coordinate' solver). Type one of the above to compile. When you switch from one test case to another, you must type one of the above to recompile. The recompile is necessary to create a new initialization executable (i.e. `real.exe`, and `ideal.exe` - there is a different `ideal.exe` for each of the idealized test cases), while `wrf.exe` is the same for all test cases.

If you want to remove all object files (except those in the `external/` directory) and executables, type `'clean'`.

Type `'clean -a'` to remove built files in ALL directories, including `configure.wrf` (the original `configure.wrf` will be saved to `configure.wrf.backup`). This is recommended if you make any mistake during the process, or if you have edited the `configure.wrf` or `Registry` files.

Hint: If you have trouble compiling routines, like solve_em.F, you can try to run the *`configure`* script with the optional argument '`-s`', i.e.

```
./configure -s
```

This will configure to compile solve_em.F and a few other routines with reduced optimization.

If you would like to turn off optimization for all the code, say during code development and debugging, you can run the *`configure`* script with option '*`-d`*':

```
./configure -d
```

Beginning with V3.5, the compression function in netCDF4 is supported. This option will typically reduce the file size by more than 50%. It will require netCDF4 to be installed with the option `--enable-netcdf-4`. Before compiling WRF, you will need to set the environment variable NETCDF4. In a C-shell environment, type `setenv NETCDF4 1`, followed by '`configure`' and '`compile`'.

For more detailed information, visit:
http://www.mmm.ucar.edu/wrf/users/wrfv3.5/building-netcdf4.html

a. Idealized case

For any 2D test case (labeled in the case names), serial or OpenMP (smpar) compile options must be used. Additionally, you must only choose the '0=no nesting' option when you configure. For all other cases, you may use serial or parallel (dmpar) and nesting. Suppose you would like to compile and run the 2-dimensional squall case, type

```
./compile em_squall2d_x >& compile.log
```

After a successful compilation, you should have two executables created in the `main/` directory: `ideal.exe` and `wrf.exe`. These two executables will be linked to the corresponding `test/case_name` and `run/` directories. `cd` to either directory to run the model.

It is a good practice to save the entire compile output to a file. When the executables are not present, this output is useful to help diagnose the compile errors.

b. Real-data case

For a real-data case, type

```
./compile em_real >& compile.log &
```

When the compile is successful, it will create three executables in the `main/`directory: `ndown.exe`, `real.exe` and `wrf.exe`.

`real.exe`: for WRF initialization of real data cases
`ndown.exe` : for one-way nesting
`wrf.exe` : WRF model integration

Like in the idealized cases, these executables will be linked to the `test/em_real` and `run/` directories. `cd` to one of these two directories to run the model.