Compiling WRF model: An applied example.

[adaptation from the original document published by the University Corporation for Atmospheric Research]

P. Iskra Mejía Estrada

Hydrology Group

School of Geographical Sciences

University of BRISTOL

January 2016

Contents

Preface

The present manual aims at constituting a first approach for beginner users to the Weather Research and Forecasting Model. This constitutes one of the meso-scale Numerical Weather Prediction models used to analyse the physics behind and consequences of a meteorological event. The present manual is based on that originally published online by the National Center for Atmospheric Research (NCAR, [no date]) and the easiness with which it can be read as well as the crucial order to execute its instructions have been kept, hence the high similarity between both documents. This resemblance should be considered only for comparing purposes as this document intends by no means to replace or disregard the original version.

An example case study is included along with the required input information so that the installing process of the WRF model could be as straightforward as possible. Throughout the installing and running steps, the reader will be likely to consult related literature so that an acceptable understanding of the numerical process can be achieved in the end.

To run the commands hereby displayed, the user should enter the text that comes after the symbol `$` which serves as a prompt character in the command shell. **It is strictly mandatory that the directory structure stated described here remains unchanged** so that there are no problems when configuring the model.

It is expected that the reader has minimum training in programming language and sufficient knowledge of LINUX OS, considering that moving between directories, displaying their content, searching and linking files are common functions to be used. Some problems encountered while installing the model are mentioned as well as their solution. If any other difficulty appears, the reader is encouraged to search for other sources of information such as WRF user manuals, forums, user guides, and even contacting the WRF Help team at wrfhelp@ucar.edu.

The author of this manual would like to thank Prof. Paul Bates, Head of School of Geographical Sciences, Dr. Miguel Rico Ramírez, Senior Lecturer at the Department of Civil Engineering, Prof. Dawei Han, Professor of Hydroinformatics from the Water and Environmental Engineering Group and the IT Service Desk, all from the University of Bristol, for the invaluable guidance provided.

1 Directory Structure

In any given location, create a directory named `Build_WRF` which will serve as the parent directory to the rest of folders that will be used to store data needed to install and run the WRF model:

```
$ mkdir Build_WRF
```

Go into that directory and create the following directories:

```
$ cd Build_WRF
$ mkdir LIBRARIES TESTS WPS_GEOG DATA
```

2 System Environment Tests

1. There are some specific compilers that need to be on the system to run the WRF model, specifically gfortran, gcc and cpp. To verify that they can be used, type:

```
$ which gfortran
$ which cpp
$ which gcc
```

And a path to each of their locations will be displayed.

Also, make sure that the current gfortran version 4.4.0 or later. To know which version is currently installed, type:

```
$ gcc --version
```

If the current version is older than 4.4.0, go to Annex B.1 to check the availability of a newer version, in which case update the compiler to run the rest of the tests.

2. The following tests will be run to verify that Fortran compiler is built properly and that is compatible with C compiler. Go into the `TESTS` directory

```
$ cd TESTS
```

Download the following tar file and place it in the current directory:

Fortran_and_C__Tests.tar

Untar the file:

```
$ tar -xf Fortran_C_tests.tar
```

Test #1 Fixed Format Fortran Test

```
$ gfortran TEST_1_fortran_only_fixed.f
$ ./a.out
```

The following lines should be displayed in the terminal:

```
     SUCCESS test 1 fortran only fixed format
```

Test #2 Free Format Fortran

```
$ gfortran TEST_2_fortran_only_free.f90
$ ./a.out
```

The following lines should be displayed in the terminal:

```
Assume Fortran 2003: has FLUSH, ALLOCATABLE, derived type, and ISO C
Binding
SUCCESS test 2 fortran only free format
```

Test #3: C: TEST_3_c_only.c

```
$ gcc TEST_3_c_only.c
$ ./a.out
```

The following line should be displayed in the terminal:

```
SUCCESS test 3 c only
```

Test #4: Fortran Calling a C Function.

As stated in the Tutorial, this test will be performed to force gcc and gfortran to use a 64 bit configuration when used together.

```
$ gcc -c -m64 TEST_4_fortran+c_c.c
$ gfortran -c -m64 TEST_4_fortran+c_f.f90
$ gfortran -m64 TEST_4_fortran+c_f.o TEST_4_fortran+c_c.o
$ ./a.out
```

The following lines should be displayed in the terminal:

```
C function called by Fortran
Values are xx = 2.00 and ii = 1
SUCCESS test 4 fortran calling c
```

Test #5: C SHELL scripting language.

```
$ ./TEST_csh.csh
```

The following line should be shown:

```
SUCCESS csh test
```

Test #6: PERL scripting language.

```
$ ./TEST_perl.pl
```

The following line should be shown:

```
SUCCESS perl test
```

Test #7: SHELL scripting language

```
$ ./TEST_sh.sh
```

The following line should be shown:

```
SUCCESS sh test
```

Finally, make sure that the following commands are available regardless the current shell:

```
ar, awk, cat, cd, cp, cut, expr, file, grep, gzip, head, hostname, ln, ls,
make, mkdir, m4, mv, nm, printf, rm, sed, sleep, sort, tar, touch, tr,
uname, wc, which.
```

by typing `which` before each one so that the path to its location is shown. For example

```
$ which ar
$ which awk.
$ which cat
```

And so on.

Finally, exit the `TESTS` directory.

```
$ cd ..
```

3 Building Libraries

Go into the `LIBRARIES` directory

```
$ cd LIBRARIES
```

Download and place the following tar files in it:

netcdf-4.1.3

mpich-3.0.4

zlib-1.2.7

libpng-1.2.50

Jasper-1.900.1

The NetCDF (Network Common Data Form) is a mandatory library required to run the WRF model. MPICH library is needed to build WRF in parallel when multiple processors are available, therefore installing MPICH is optional but recommended. The zlib, libpng and JasPer libraries will enable WRF to support GRIB2 format files and are required to compile the ungrib module from the WRF Preprocessing System, henceforth WPS.

The variable environment needs to be set to install these libraries. To do so, first type

```
$ echo $SHELL
```

To display the path of the current shell: `/bin/csh` stands for C Shell, and `/bin/bash` stands for Bash Shell. The type of shell will determine the syntax to be used, so type the following commands accordingly (see Annex A.1 to know what to write instead of *path/to/directory*):

a) If using C Shell

```
$ setenv DIR path/to/directory/Build_WRF/LIBRARIES
$ setenv CC gcc
$ setenv CXX g++
$ setenv FC gfortran
$ setenv FCFLAGS -m64
$ setenv F77 gfortran
$ setenv FFLAGS -m64
```

b) If using Bash Shell

```
$ export DIR=path/to/directory/Build_WRF/LIBRARIES
$ export CC=gcc
$ export CXX=g++
$ export FC=gfortran
$ export FCFLAGS=-m64
$ export F77=gfortran
$ export FFLAGS=-m64
```

Then proceed to install the libraries:

Library #1 NetCDF

```
$ tar xzvf netcdf-4.1.3.tar.gz
$ cd netcdf-4.1.3
```

Write the following lines separated by a single space:

```
$ ./configure
  --prefix=$DIR/netcdf
  --disable-dap
  \--disable-netcdf-4
  --disable-shared
$ make
$ make install
$ make check
```

a) If using C Shell

```
$ setenv PATH $DIR/netcdf/bin:$PATH
$ setenv NETCDF $DIR/netcdf
```

b) If using Bash Shell

```
$ export PATH=$DIR/netcdf/bin:$PATH
$ export NETCDF=$DIR/netcdf
```

Exit the directory:

```
$ cd ..
```

Library #2 MPICH

```
$ tar xzvf mpich-3.0.4.tar.gz
$ cd mpich-3.0.4
$ ./configure --prefix=$DIR/mpich
$ make
$ make install
```

Set the PATH variable:

a) If using C Shell

```
$ setenv PATH $DIR/mpich/bin:$PATH
```

b) If using Bash Shell

```
$ export PATH=$DIR/mpich/bin:$PATH
```

Exit the directory:

```
$ cd ..
```

Library #3 zlib

Again, type the two commands that correspond to the current type of shell:

a) If using C Shell

```
$ setenv LDFLAGS -L$DIR/grib2/lib
$ setenv CPPFLAGS -I$DIR/grib2/include
```

b) If using Bash Shell

```
$ export LDFLAGS=-L$DIR/grib2/lib
$ export CPPFLAGS=-I$DIR/grib2/include
```

And then type:

```
$ tar xzvf zlib-1.2.7.tar.gz
$ cd zlib-1.2.7
$ ./configure --prefix=$DIR/grib2
$ make
$ make install
```

Exit the directory:

```
$ cd ..
```

Library #4 libpng

```
$ tar xzvf libpng-1.2.50.tar.gz
$ cd libpng-1.2.50
$ ./configure --prefix=$DIR/grib2
$ make
$ make install
```

Exit the directory:

```
$ cd ..
```

Library #5 JasPer

```
$ tar xzvf jasper-1.900.1.tar.gz
$ cd jasper-1.900.1
$ ./configure --prefix=$DIR/grib2
$ make
$ make install
```

Exit the directory by going up two levels:

```
$ cd ../..
```

4 Library Compatibility Tests

To verify that that libraries installed can be used with the compilers that set up WRF and WPS, two additional tests need to be undergone. Download the tar file that contains said tests from

Fortran_C_NETCDF_MPI_tests.tar

Move to the `TESTS` directory:

```
$ cd TESTS
```

Place the tar file here and untar it by issuing:

```
$ tar –xf Fortran_C_NETCDF_MPI_tests.tar
```

Test #1

Set the NetCDF environment variable:

a) If using C Shell

```
$ setenv NETCDF /path/to/directory/Build_WRF/LIBRARIES/netcdf
```

b) If using Bash Shell

```
$ export NETCDF=/path/to/directory/Build_WRF/LIBRARIES/netcdf
```

Copy the include file from the NetCDF package here:

```
$ cp ${NETCDF}/include/netcdf.inc .
```

(Note that there is a space between the path of the include file and the final dot. The latter means “current directory”).

And compile the Fortran and C codes:

```
$ gfortran -c 01_fortran+c+netcdf_f.f
$ gcc -c 01_fortran+c+netcdf_c.c
```

Write the following lines separated by a single space:

```
$ gfortran 01_fortran+c+netcdf_f.o
 01_fortran+c+netcdf_c.o
 \-L${NETCDF}/lib
 -lnetcdff
 -lnetcdf
```

And then:

```
$ ./a.out
```

The output should be:

```
     C function called by Fortran
     Values are xx = 2.00 and ii = 1
     SUCCESS test 1 fortran + c + netcdf
```

Test #2

Copy the include file from the NetCDF package here:

```
$ cp ${NETCDF}/include/netcdf.inc .
$ mpif90 -c 02_fortran+c+netcdf+mpi_f.f
```

If an error message appears stating "`mpif90 not found; modules not loading`", then have a look at Annex B.2. If not or after the problem is solved, continue the test:

```
$ mpicc -c 02_fortran+c+netcdf+mpi_c.c
```

Write the following lines separated by a single space:

```
$ mpif90
  02_fortran+c+netcdf+mpi_f.o
  \02_fortran+c+netcdf+mpi_c.o
  \-L${NETCDF}/lib
  -lnetcdff
  -lnetcdf
$ mpirun ./a.out
```

The result should display:

```
     Values are xx = 2.00 and ii = 1
     status = 2
     SUCCESS test 2 fortran + c + netcdf + mpi
```

Finally, exit the TESTS directory

```
$ cd ..
```

5 Building WRFV3

Download the following tar file and place it in the current (`Build_WRF`) directory:

WRFV3.7

Untar it:

```
$ gunzip WRFV3.7.TAR.gz
$ tar -xf WRFV3.7.TAR
```

Go into the `WRFV3` directory

```
$ cd WRFV3
```

Before continuing, refer to Annex A.2, steps 1 and 2 to know how to select the correct compiler, and step 3 to see an applied example on how to configure WRF.

Run the following command to configure the WRF model.

```
$ ./configure
```

After selecting the appropriate option a warning could appear:

```
********************** W A R N I N G ****************************
There are some Fortran 20003 features in WRF that your compiler does not
recognize
The IEEE signaling call has been removed. That may not be enough.
*****************************************************************
```

This warning does not represent a problem to run the model. The message that must appear is the one stating that Fortran C compiler was successfully configured. If any other warnings appear, reconfigure (i. e. type again `./configure`) and select another option.

The next stage is to select which type of case will be compiled among the following list, where "em" stands for Eulerian mass solver developed at the National Centre for Atmospheric Research:

- em_real (3D real case)
- em_quarter_ss (3D ideal case)
- em_b_wave (3D ideal case)
- em_les (3D ideal case)
- em_heldsuarez (3d ideal case)
- em_tropical_cyclone (3D ideal case)
- em_hill2d_x (2D ideal case)
- em_squall2d_x (2D ideal case)
- em_squall2d_y (2D ideal case)
- em_grav2d_x (2D ideal case)
- em_seabreeze2d_x (2D ideal case)
- em_scm_xy (1D ideal case)

Since this example (and any other model runs) are real data cases, em_real is the most suitable option. Type:

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

If another option were to be run, then substitute em_real for the correspondent case.

After compilation is completed, then look into the `main` directory (nested under `WRFV3`) and the following files should appear:

- wrf.exe (executable file of the model)
- real.exe (initializes real data)
- ndown.exe (enables one-way nesting)
- tc.exe (tropical cyclone bogusing method, serial only)

If an idealized case was compiled, then the files in the main directory should be:

- wrf.exe (executable file of the model)
- ideal.exe (initializes ideal data)

In either case, all files are linked to the directories `WRFV3/run` and `WRFV/test/em_real`. The WRF model can be run from any of them.

Finally, exit the `WRFV3` directory:

```
$ cd ..
```

6 Building WPS

The WRF Preprocessing System (WPS) comprises three programs that prepare the input data to be used to run a simulation. These are *geogrid* (sets grids and static geographical data), *ungrib* (processes meteorological data) and *metgrid* (horizontally interpolates the information used by *ungrib* into the domain defined by *geogrid*).

Download the following tar file and place it in the current (`Build_WRF`) directory:

WPSV3.7

Untar it:

```
$ gunzip WPSV3.7.TAR.gz
$ tar -xf WPSV3.7.TAR
```

Go into the WPS directory and make sure it is clean:

```
$ cd WPS
$ ./clean
```

Set the paths for the ungrib libraries:

a) If using C Shell

```
$ setenv JASPERLIB $DIR/grib2/lib
$ setenv JASPERINC $DIR/grib2/include
```

b) If using Bash Shell

```
$ export JASPERLIB=$DIR/grib2/lib
$ export JASPERINC=$DIR/grib2/include
```

And start the configuration:

```
$ ./configure
```

Similarly to setting up WRF, a list of options will be displayed. Go to Annex A.3 to see an example of the options displayed and the most appropriate configuration selected.

Open the `configure.wps` file (see Annex A.4 to edit textfiles in the shell) and make sure that the WPS directory is at the same level as directory WRFV3 by checking that the following line in contained in the file:

```
WRF_DIR = ../WRFV3
```

If so, then type:

```
$ ./compile >& log.compile
```

And check that three links were created in the current (WPS) directory, which should be linked to their respective executable file in their correspondent directory under WPS. Run

```
$ ls -ls *.exe
```

And the following should display:

```
geogrid.exe -> geogrid/src/geogrid.exe
ungrib.exe -> ungrib/src/ungrib.exe
```

```
metgrid.exe -> metgrid/src/metgrid.exe
```

If geogrid.exe and metgrid.exe are not created, refer to Annex B.3.

Finally, exit the WPS directory

```
$ cd ..
```

7 Static Geographic Data

Aside from the particular atmospheric input data needed to run the WRF model, a set of static geographical parameters must be implemented. This includes the location and extent of the single or nested grids, topography, and land use, to name a few.

Go to the `WPS_GEOG` directory:

```
$ cd WPS_GEOG
```

Download the following tar file:

Geographic Static Data

Place it in the `WPS_GEOG` directory and untar it:

```
$ gunzip geog.tar.gz
$ tar -xf geog.tar
```

The extracted file is the folder `geog` that contains the geographical parameters. Move them from the extracted folder to the `WPS_GEOG` directory:

```
$ mv ./geog/* .
```

A couple of folders that contains information regarding variance of subgrid-scale orography (varsso) are still missing. Go to this link and download the varsso_5m, varsso_2m, lai_modis_10m, lake_depth files (they are compressed .tar.bz2 files). Place them in the `WPS_GEOG` directory and untar them:

```
$ tar -xjvf varsso_2m.tar.bz2
$ tar -xjvf varsso_5m.tar.bz2
$ tar –xjvf lai_modis_10m.tar.bz2
$ tar –xjvf lake_depth.tar.bz2
```

Now all the geographic static data needed to run WRF is contained within the `WPS_GEOG` directory. Go up one level:

```
$ cd ..
```

8 Atmospheric Input Data

Lateral boundary conditions and an initial meteorological settings are needed to run the WRF model. This information can be downloaded from the NCAR/UCAR database, which includes information from analyses, reanalyses and forecasts. Subsets of the ECMWF databases can be downloaded as well. The selection of a set of parameters and its source depends on the specific needs of every run. In that regard, recommendations on each one are beyond the scope of this manual.

In this particular case, the event to be modelled is Hurricane Katrina (NCAR, 2012). This event, which occurred in August 2005 and caused the larger amount of damages in New Orleans, is considered one of the deadliest meteorological events ever occurred to the United States. The importance of understanding hurricane dynamics enables the study of their development and hence improve databases, future projections and as ultimate goal, population safety. A subset of atmospheric information to drive a Hurricane Katrina case study can be downloaded from this link.

Place it in the `DATA` directory and untar it:

```
$ gunzip Katrina.tar.gz
$ tar -xf Katrina.tar
```

The tar file contains several avn* files, which were named after Aviation when the numerical model was named Global Forecast System. They are in GRIB1 Format and available in 6 hour periods.

Exit the current directory and go up one level:

```
$ cd ..
```

9 Running WPS

To set the particular conditions for each run, the parameters of the pre-processing stage must be modified. Go into the WPS directory:

```
$ cd WPS
```

And edit the information of the `namelist.wps` file (see instructions in Annex A.4 to do so) to match the one stated below:

```
&share
 wrf_core = 'ARW'
 max_dom = 1
 start_date = '2005-08-28_00:00:00'
 end_date   = '2005-08-29_00:00:00'
 interval_seconds = 21600
 io_form_geogrid = 2
/

&geogrid
 parent_id         =   1
 parent_grid_ratio =   1
 i_parent_start    =   1
 j_parent_start    =   1
 e_we              =  140
 e_sn              =  94
 geog_data_res     = '10m'
 dx = 45000
 dy = 45000
 map_proj = 'mercator'
 ref_lat   =  20.0
 ref_lon   = -75.0
 truelat1  =  20.0
 truelat2  =  20.0
 stand_lon = -75.0
 geog_data_path = '/path/to/directory/Build_WRF/WPS_GEOG/'
  /

&ungrib
 out_format = 'WPS'
 prefix = 'FILE'
/

&metgrid
```

```
     fg_name = 'FILE'
     io_form_metgrid = 2
    /
```

The previous parameters mean that there is a single domain that has its center in 75.0W, 20.0N. It is comprised by 140 cells in the horizontal direction and 94 cells in the vertical direction, with a cell size of 45 000 m. The topography resolution is 10 arc-minutes (approximately 16 km). A complete list of the variables in the namelist.wps file can be found in the User's Guide issued by NCAR (2015a). When editing this file, a time saver is knowing that `max_dom` will only read the first column of all the parameters, so that there is no need to erase all the information already contained in the namelist.wps file.

Then run geogrid:

```
$ ./geogrid.exe >& log.geogrid
```

If the run was successful, a log.geogrid file must be created with the final lines:

```
     !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
     !      Successful completion of geogrid    !
     !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

And a `geo_em.d01.nc` in the current directory, which is a NetCDF file created per nested domain (see `max_dom` value in the `namelist.wps` file). Is multiple domains had been set, then geogrid would have produced a `geo_em.d**.nc` file per domain.

Link the input atmospheric information contained in the DATA directory:

```
$ ./link_grib.csh /path/to/directory/Build_WRF/DATA/avn*
```

This will create a `GRIBFILE` link for every `avn*` file inside `DATA`. It is crucial that the previous command line ends with the prefix of the files (in this case, `avn`) and an asterisk that serves as a wildcard to select all the files within `DATA`.

Link to the correct Vtable. This is a Variable Table that contains columns with information on how the information is coded and how it will be read by the model. Some Vtables were included during the configuration of WPS and they can be found at `/path/to/directory/Build_WRF/WPS/ungrib/Variable_Tables`. For this study, the correspondent Vtable is the one issued by the GFS, so run the following command:

```
$ ln -sf ungrib/Variable_Tables/Vtable.GFS Vtable
```

And run ungrib:

```
$ ./ungrib.exe >& log.geogrid
```

If the run was successful, then the shell should show:

```
      !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
      !     Successful completion of ungrib     !
      !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

And several files with the prefix "FILE" (the prefix stated in the `&ungrib` section of the `namelist.wps` file) should appear in the current directory.

Finally, run metgrid:

```
$ ./metgrid.exe >& log.metgrid
```

If the run was successful, then the shell should show:

```
      !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
      !     Successful completion of metgrid    !
      !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

As well as met_em* files, one every `interval_seconds` stated in the `namelist.wps` file.

Finally, exit the current directory:

```
$ cd ..
```

10 Running WRFV3

Go into `/path/to/directory/Build_WRF/WRFV3/run` or `/WRFV3/test/em_real`:

```
$ cd WRFV3/run
```

Or:

```
$ cd WRFV3/test/em_real
```

Edit the `namelist.input` file to match the following parameters by opening the file or in the shell, as instructed in Annex A.4:

```
     &time_control
      run_days                            = 1,
      run_hours                           = 0,
      run_minutes                         = 0,
      run_seconds                         = 0,
      start_year                          = 2005,
      start_month                         = 08,
      start_day                           = 28,
      start_hour                          = 00,
      start_minute                        = 00,
      start_second                        = 00,
      end_year                            = 2005,
      end_month                           = 08,
      end_day                             = 29,
      end_hour                            = 00,
      end_minute                          = 00,
      end_second                          = 00,
      interval_seconds                    = 21600
      input_from_file                     = .true.,
      history_interval                    = 180,
      frames_per_outfile                  = 1,
      restart                             = .false.,
      restart_interval                    = 1440,
      io_form_history                      = 2,
      io_form_restart                      = 2,
      io_form_input                        = 2,
      io_form_boundary                     = 2,
      debug_level                          = 0,
      /

      &domains
      time_step                           = 60,
      max_dom                             = 1,
      e_we                                = 140,
      e_sn                                = 94,
      e_vert                              = 57,
      p_top_requested                     = 1000,
      num_metgrid_levels                  = 27,
      num_metgrid_soil_levels             = 4,
      dx                                  = 45000,
      dy                                  = 45000,
```

The rest of the parameters in the `namelist.input` file can remain the way they are. A more extensive explanation of the parameters can be found NCAR (2015b).

Link the met_em files into the current directory:

a) If current directory is `/WRFV3/test/em_real`, type:

```
$ ln -sf ../../../WPS/met_em* .
```

b) If current directory is `/WRFV3/run`, type:

```
$ ln -sf ../../WPS/met_em* .
```

Now the “real” program can be run. Type:

```
$ mpirun –np 1 ./real.exe
```

If an error message appears stating “`mpirun: command not found`”, run the commands and load the module stated in Annex B.2 “Library Compatibility tests”.

A `SUCCESS` message should display in the shell. Furthermore, a `wrfbdy_d01` and a `wrfinput_d01` files should be created. Note than when an idealized case is run, `wrfbdy` is not created.

Finally, to run the WRF model simply type:

```
$ mpirun –np 8 ./wrf.exe
```

And wrfout* files will be created, which are the final outputs of the model.

References

National Center for Atmospheric Research, Mesoscale & Microscale Meteorology Laboratory. [No date]. *Compiling WRF*. [Online]. [Accessed 15 December 2015]. Available from: http://www2.mmm.ucar.edu/wrf/OnLineTutorial/compilation_tutorial.php

National Center for Atmospheric Research, Mesoscale & Microscale Meteorology Laboratory. 2012. Compiling WRF. [Online]. [Accessed 9 December 2015]. Available from http://www2.mmm.ucar.edu/wrf/OnLineTutorial/Class/cases/var_start_case.htm

National Center for Atmospheric Research, Mesoscale & Microscale Meteorology Laboratory. 2015a. *ARW Version 3 Modeling System User's Guide*. Chapter 3: WRF Preprocessing System (WPS). [Online]. [Accessed 18 December 2015]. Available from http://www2.mmm.ucar.edu/wrf/users/docs/user_guide_V3/ARWUsersGuideV3.6.1.pdf

National Center for Atmospheric Research, Mesoscale & Microscale Meteorology Laboratory. 2015b. *ARW Version 3 Modeling System User's Guide*. Chapter 6: WRF Data Assimilation. [Online]. [Accessed 18 December 2015]. Available from http://www2.mmm.ucar.edu/wrf/users/docs/user_guide_V3/ARWUsersGuideV3.6.1.pdf

Annex A. Choosing the correct settings

A.1 What does *path/to/directory* mean?

It stands for "absolute path to the directory". To know exactly what to write instead of *path/to/directory*/Build_WRF, simply type:

$ find ~ -type d -name Build_WRF

Which searches within the home directory (~) a certain type of file (d: directory) by its exact name (meaning that the search is case sensitive) and the search term is Build_WRF. Therefore, */path/to/directory/*Build_WRF must be replaced by the complete output of the command. For example:

```
/home/ab12345/projects/Build_WRF
```

A.2 How to select the correct WRF configuration?

The current compiler and collection available must be known to install the WRF model.

1) To know which the current compiler collection is, run:

```
$ cc -v
```

And check the output:

```
Using built-in specs.
COLLECT_GCC=cc
COLLECT_LTO_WRAPPER=/usr/libexec/gcc/x86_64-redhat-linux/4.8.5/lto-
wrapper
Target: x86_64-redhat-linux
Configured with: ../configure --prefix=/usr --mandir=/usr/share/man --
infodir=/u     sr/share/info --with-
bugurl=http://bugzilla.redhat.com/bugzilla --enable-bootstr     ap --
enable-shared --enable-threads=posix --enable-checking=release --with-
syste     m-zlib --enable-__cxa_atexit --disable-libunwind-exceptions
--enable-gnu-unique-     object --enable-linker-build-id --with-
linker-hash-style=gnu --enable-languages=     c,c++,objc,obj-
c++,java,fortran,ada,go,lto --enable-plugin --enable-initfini-arr
ay --disable-libgcj --with-isl=/builddir/build/BUILD/gcc-4.8.5-
20150702/obj-x86_     64-redhat-linux/isl-install --with-
cloog=/builddir/build/BUILD/gcc-4.8.5-2015070     2/obj-x86_64-redhat-
linux/cloog-install --enable-gnu-indirect-function --with-tu
ne=generic --with-arch_32=x86-64 --build=x86_64-redhat-linux
Thread model: posix
gcc version 4.8.5 20150623 (Red Hat 4.8.5-4) (GCC)
```

The last line states the current compiler system implemented. In this example: GCC (GNU Compiler Collection).

2) To determine the current compiler, type:

```
$ yum list installed | grep -i --color gcc
```

Which would display a list of compilers associated to GCC highlighted in colour, for example:

```
Skipping unreadable repository '/etc/yum.repos.d/uob-zd-centos-
private.repo'
Skipping unreadable repository '/etc/yum.repos.d/uob-zd-centos6-
private.repo'
compat-gcc-44.x86_64           4.4.7-8.el7  @centos7
compat-gcc-44-c++.x86_64       4.4.7-8.el7  @centos7
compat-gcc-44-gfortran.x86_64  4.4.7-8.el7  @centos7
```

```
     gcc.x86_64                       4.8.5-4.el7  @centos7
     gcc-c++.x86_64                   4.8.5-4.el7  @centos7
     gcc-gfortran.x86_64              4.8.5-4.el7  @centos7
     libgcc.i686                      4.8.5-4.el7  @centos7
     libgcc.x86_64                    4.8.5-4.el7  @centos7
     opt-gcc-4.4.7.x86_64             1:4.4.7-1    @uob-zd-centos6
     opt-gcc-4.4.7-modules.x86_64     1:4.4.7-1    @uob-zd-centos6
     opt-gcc-4.7.1.x86_64             1:4.7.1-1    @uob-zd-centos6
     opt-gcc-4.7.1-modules.x86_64     1:4.7.1-1    @uob-zd-centos6
     opt-gcc-4.9.1.x86_64             1:4.9.1-1    @uob-zd-centos6
     opt-gcc-4.9.1-modules.x86_64     1:4.9.1-1    @uob-zd-centos6
     opt-gcc-5.2.0.x86_64             1:5.2.0-1    @uob-zd-centos6
     opt-gcc-5.2.0-modules.x86_64     1:5.2.0-1    @uob-zd-centos6
```

Identify the compilers linked to the collection determined in step 1 (GCC). In this example: GCC/C++ and GCC/gfortran.

3) When configuring the model as stated in section 5 “Building WRFV3” by typing:

```
$ ./configure
```

A list such as the following will be displayed:

```
     1. (serial)  2. (smpar)  3. (dmpar)  4. (dm+sm)  PGI (pgf90/gcc)
     5. (serial)  6. (smpar)  7. (dmpar)  8. (dm+sm)  PGI (pgf90/pgcc): SGI
                                                       MPT
     9. (serial)  10. (smpar) 11. (dmpar) 12. (dm+sm) PGI (pgf90/gcc): PGI
                                                       accelerator
     13. (serial) 14. (smpar) 15. (dmpar) 16. (dm+sm) INTEL (ifort/icc)
                                          17. (dm+sm) INTEL (ifort/icc):
                                                       Xeon Phi (MIC
                                                       architecture)
     18. (serial) 19. (smpar) 20. (dmpar) 21. (dm+sm) INTEL (ifort/icc):
                                                       Xeon (SNB with AVX
                                                       mods)
     22. (serial) 23. (smpar) 24. (dmpar) 25. (dm+sm) INTEL (ifort/icc):
                                                       SGI MPT
     26. (serial) 27. (smpar) 28. (dmpar) 29. (dm+sm)  INTEL (ifort/icc):
                                                       IBM POE
     30. (serial)             31. (dmpar)              PATHSCALE
                                                       (pathf90/pathcc)
     32. (serial) 33. (smpar) 34. (dmpar) 35. (dm+sm)  GNU (gfortran/gcc)
     36. (serial) 37. (smpar) 38. (dmpar) 39. (dm+sm)  IBM (xlf90_r/cc_r)
     40. (serial) 41. (smpar) 42. (dmpar) 43. (dm+sm)  PGI (ftn/gcc): Cray
                                                       XC CLE
     44. (serial) 45. (smpar) 46. (dmpar) 47. (dm+sm)  CRAY CCE (ftn/gcc):
                                                       Cray XE and XC
     48. (serial) 49. (smpar) 50. (dmpar) 51. (dm+sm)  INTEL (ftn/icc):
                                                       Cray XC
     52. (serial) 53. (smpar) 54. (dmpar) 55. (dm+sm)  PGI (pgf90/pgcc)
     56. (serial) 57. (smpar) 58. (dmpar) 59. (dm+sm)  PGI (pgf90/gcc): -
                                                       f90=pgf90
     60. (serial) 61. (smpar) 62. (dmpar) 63. (dm+sm)  PGI (pgf90/pgcc): -
                                                       f90=pgf90
```

The compiler and the collection determined in steps 1 and 2 must be selected. In this example, the correct option would be 32, 33, 34 or 35: GNU (gfortran/gcc).The WRF model can be run serially or in parallel (`smpar`: shared-memory parallelism, `dmpar`: distributed-memory parallelism, and `dm+sm`: distributed and shared memory) being the most recommended option to select `dmpar` (in this example: option 34).

After this option is selected, some compilations require a second decision regarding the type of nesting, namely: no nesting (0), basic (1), pre-set moves (2), and vortex following (3). When running idealized cases, use the `no nesting` option. This number will not be available when selecting a `dmpar` or `dm+sm` compilation type. The `basic` option could be a fitting first approach for simple cases.

A.3 How to select the correct WPS configuration?

To select the correct option, the user must have in mind the compiler selected when configuring WRF (see step 2 of Annex A.2)

An example of the list of options when configuring WPS is as follows:

```
1.  Linux x86_64, gfortran    (serial)
2.  Linux x86_64, gfortran    (serial_NO_GRIB2)
3.  Linux x86_64, gfortran    (dmpar)
4.  Linux x86_64, gfortran    (dmpar_NO_GRIB2)
5.  Linux x86_64, PGI compiler   (serial)
6.  Linux x86_64, PGI compiler   (serial_NO_GRIB2)
7.  Linux x86_64, PGI compiler   (dmpar)
8.  Linux x86_64, PGI compiler   (dmpar_NO_GRIB2)
9.  Linux x86_64, PGI compiler, SGI MPT   (serial)
10.  Linux x86_64, PGI compiler, SGI MPT   (serial_NO_GRIB2)
11.  Linux x86_64, PGI compiler, SGI MPT   (dmpar)
12.  Linux x86_64, PGI compiler, SGI MPT   (dmpar_NO_GRIB2)
13.  Linux x86_64, IA64 and Opteron    (serial)
14.  Linux x86_64, IA64 and Opteron    (serial_NO_GRIB2)
15.  Linux x86_64, IA64 and Opteron    (dmpar)
16.  Linux x86_64, IA64 and Opteron    (dmpar_NO_GRIB2)
17.  Linux x86_64, Intel compiler    (serial)
18.  Linux x86_64, Intel compiler    (serial_NO_GRIB2)
19.  Linux x86_64, Intel compiler    (dmpar)
20.  Linux x86_64, Intel compiler    (dmpar_NO_GRIB2)
21.  Linux x86_64, Intel compiler, SGI MPT    (serial)
22.  Linux x86_64, Intel compiler, SGI MPT    (serial_NO_GRIB2)
23.  Linux x86_64, Intel compiler, SGI MPT    (dmpar)
24.  Linux x86_64, Intel compiler, SGI MPT    (dmpar_NO_GRIB2)
25.  Linux x86_64, Intel compiler, IBM POE    (serial)
26.  Linux x86_64, Intel compiler, IBM POE    (serial_NO_GRIB2)
27.  Linux x86_64, Intel compiler, IBM POE    (dmpar)
28.  Linux x86_64, Intel compiler, IBM POE    (dmpar_NO_GRIB2)
29.  Linux x86_64 g95 compiler     (serial)
30.  Linux x86_64 g95 compiler     (serial_NO_GRIB2)
31.  Linux x86_64 g95 compiler     (dmpar)
32.  Linux x86_64 g95 compiler     (dmpar_NO_GRIB2)
33.  Cray XE/XC CLE/Linux x86_64, Cray compiler   (serial)
34.  Cray XE/XC CLE/Linux x86_64, Cray compiler   (serial_NO_GRIB2)
35.  Cray XE/XC CLE/Linux x86_64, Cray compiler   (dmpar)
36.  Cray XE/XC CLE/Linux x86_64, Cray compiler   (dmpar_NO_GRIB2)
37.  Cray XC CLE/Linux x86_64, Intel compiler   (serial)
38.  Cray XC CLE/Linux x86_64, Intel compiler   (serial_NO_GRIB2)
39.  Cray XC CLE/Linux x86_64, Intel compiler   (dmpar)
40.  Cray XC CLE/Linux x86_64, Intel compiler   (dmpar_NO_GRIB2)
```

The WPS can be compiled in serial or in parallel and it could include or not support for GRIB2 files. Select the “serial” option unless the domain area is extremely large. Additionally, support for GRIB2 files is suggested because the input data for the model may be found in this format. Once knowing the working compiler by running the commands in Annex A.2, and following the recommendations on the type of compilation and file support, then the most suitable option in this example would be number 1.

A.4 How to edit text files in the current shell?

A simply way to edit a file that contains readable information is to issue the command:

```
$ vi name_of_file.extension
```

And the shell will shift to the content of the file. To start editing simply type `i` (lowercase). To exit the editor, press the ESC key and a colon prompt will appear. Type

```
: q to quit
: q! to quit without saving changes
: wq to overwrite the file and quit.
```

As it is not necessary for the file to have a .txt extension, `configure.wps`, `namelist.wps` and `namelist.input` can be edited using the `vi` command.

Annex B. Difficulties found and overcome while installing WPS and WRF

B.1 System Environment Tests.

If the current gcc version is older than 4.4.0, type the following to display a list of the available installed modules:

```
$ module avail
```

The list should look like this:

```
emacs-24.4-x86_64            intel/fc/10.1.015              petsc-3.3-p6-x86_64
fcm-2015.02.0-x86_64         intel/idb/10.1.015             python-2.7-ipython-3.0.0-x86_64
ferret-6.9-x86_64            intel-composer-xe-2015-x86_64  python-2.7-spyder-2.3.3-x86_64
gcc-4.4.7-x86_64             libharu-2.2.1-x86_64           python-3.4-x86_64
gcc-4.7.1-x86_64             matlab-R2013a-x86_64           saga-2.1.0-x86_64
gcc-4.9.1-x86_64             matlab-R2014a-x86_64           tau-2.22.2-mpi-x86_64
gcc-5.2.0-x86_64             matlab-R2015a-x86_64           tau-2.22.2-openmp-x86_64
geotrans-3.3-x86_64          opari2-1.0.7-x86_64            tau-2.22.2-singlethreaded-x86_64
hpctoolkit-20130530-x86_64   openmpi-1.4.5-x86_64           visit-parallel-2.9.0-x86_64
hpcviewer-5.3.2-x86_64       openmpi-1.6.4-ifort-x86_64     visit-serial-2.9.0-x86_64
idl/8.1                      papi-5.1.1-x86_64              wxWidgets-2.9.4-x86_64
idl-8.2-x86_64               pdt-3.19-x86_64
idl-8.4-x86_64               perfexpert-2.1.2-x86_64
```

Load any gcc version equal or more recent than the one required by the WRF (version 4.4.0) by issuing the `module load` command, for example:

```
$ module load gcc-4.7.1-x86_64
```

To be sure that it was loaded successfully, type the following to show the full path to its location:

```
$ which gcc
```

And the output should be similar to:

```
/usr/bin/gcc
```

B.2 Library Compatibility Tests. Test #2

If after typing:

```
$ mpif90 -c 02_fortran+c+netcdf+mpi_f.f
```

The error “mpif90:command not found” appears, run:

```
$ module avail
```

To display the list of available commands:

```
emacs-24.4-x86_64             intel/fc/10.1.015              petsc-3.3-p6-x86_64
fcm-2015.02.0-x86_64          intel/idb/10.1.015             python-2.7-ipython-3.0.0-x86_64
ferret-6.9-x86_64             intel-composer-xe-2015-x86_64  python-2.7-spyder-2.3.3-x86_64
gcc-4.4.7-x86_64              libharu-2.2.1-x86_64           python-3.4-x86_64
gcc-4.7.1-x86_64              matlab-R2013a-x86_64           saga-2.1.0-x86_64
gcc-4.9.1-x86_64              matlab-R2014a-x86_64           tau-2.22.2-mpi-x86_64
gcc-5.2.0-x86_64              matlab-R2015a-x86_64           tau-2.22.2-openmp-x86_64
geotrans-3.3-x86_64           opari2-1.0.7-x86_64            tau-2.22.2-singlethreaded-x86_64
hpctoolkit-20130530-x86_64    openmpi-1.4.5-x86_64           visit-parallel-2.9.0-x86_64
hpcviewer-5.3.2-x86_64        openmpi-1.6.4-ifort-x86_64     visit-serial-2.9.0-x86_64
idl/8.1                       papi-5.1.1-x86_64              wxWidgets-2.9.4-x86_64
idl-8.2-x86_64                pdt-3.19-x86_64
idl-8.4-x86_64                perfexpert-2.1.2-x86_64
```

The module needed is listed as `mpi/openmpi-x86_64`, so load it:

```
$ module load mpi/openmpi-x86_64
```

To be sure that it is now available, type the following to show the full path to its location:

```
$ which mpif90
```

And the output should be similar to:

```
/usr/lib64/openmpi/bin/mpif90
```

B.3 geogrid.exe and metgrid.exe not created

When compiling WPS, the most popular explanation for the absence of these two files is that the path to the correspondent WRF files is incorrect. That is to say, in the configure.wps file WFR_DIR must point to the WRFV3 directory, whether using a relative or an absolute path.

```
WRF_DIR = ../WRFV3
```

If after double checking this parameter, geogrid and metgrid are still not created, then experience has proven useful to start the installing of the model from the top, since the process can be done again and at this stage it is not time consuming.