Chapter 2: Software Installation

Table of Contents

Introduction

The WRF modeling system (https://www2.mmm.ucar.edu/wrf/users/) software installation (https://www2.mmm.ucar.edu/wrf/users/download/get_source.html) is fairly straightforward on the ported platforms listed below. The model-component portion of the package is mostly self-contained. The WRF model contains the source code to a Fortran interface to ESMF and the source to FFTPACK. Contained within the WRF system is the WRFDA component, which has several external libraries that the user must install (for various observation types and linear algebra solvers). Similarly, the WPS package, separate from the WRF source code, has additional external libraries that must be built (in support of Grib2 processing). All of the systems require the netCDF library, which is one of the supported I/O API packages. netCDF libraries and source code are available from the Unidata homepage at http://www.unidata.ucar.edu (select DOWNLOADS, registration required).

The WRF model has been successfully ported to a number of Unix-based machines. We do not have access to all of them and must rely on outside users and vendors to supply the required configuration information for the compiler and loader options. Below is a list of the supported combinations of hardware and software for WRF, WRFDA, and WPS.

Vendor	Hardware	OS	Compiler
Cray	XC30 Intel	Linux	Intel
Cray	XE AMD	Linux	Intel

IBM	Power Series	AIX	vendor
IBM	Intel	Linux	Intel / PGI / gfortran
SGI	IA64 / Opteron	Linux	Intel
COTS*	IA32	Linux	Intel / PGI / gfortran / g95 / PathScale
COTS	IA64 / Opteron	Linux	Intel / PGI / gfortran / PathScale
Mac	Power Series	Darwin	xlf / g95 / PGI / Intel
Mac	Intel	Darwin	gfortran / PGI / Intel
NEC	NEC	Linux	vendor
Fujitsu	FX10 Intel	Linux	vendor

* Commercial Off-The-Shelf systems

The WRF model may be built to run on a single-processor machine, a shared-memory machine (that uses the OpenMP API), a distributed memory machine (with the appropriate MPI libraries), or on a distributed cluster (utilizing both OpenMP and MPI).

Required Compilers and Scripting Languages

The majority of the WRF model, WPS, and WRFDA codes are written in Fortran 90. The software layer, RSL, which sits between WRF and WRFDA, and the MPI interface is written in C. WPS makes direct calls to the MPI libraries for distributed memory message passing. There are also ancillary programs, written in C, to perform file parsing and file construction, which are required for default building of the WRF modeling code. Additionally, the WRF build mechanism uses several scripting languages: including perl, Cshell and Bourne shell. The traditional UNIX text/file processing utilities are used: make, m4, sed, and awk. See Chapter 8: WRF Software (Required Software) for a more detailed listing of the necessary pieces for the WRF build.

Required/Optional Libraries to Download

The only library that is *always* required is the netCDF package from Unidata (login > Downloads > NetCDF). Most of the WRF post-processing packages assume that the data from the WRF model, the WPS package, or the WRFDA program are using the netCDF libraries. One may also need to add '/path-to-netcdf/netcdf/bin' to their path so that they may execute netCDF utility commands, such as **ncdump**. Use a netCDF version that is

3.6.1 or later. To utilize the compression capabilities, use netCDF 4.0 or later. Note that compression will require the use of HDF5.

Note 1: The entire step-by-step recipe for building the WRF and WPS packages is available at: http://www2.mmm.ucar.edu/wrf/OnLineTutorial/compilation_tutorial.php This page includes complete turn-key directions, from tests of your machines's utilities all the way up through where to download real-time data.

Note 2: If one wants to compile WRF system components on a Linux or Darwin system that has access to multiple compilers, link the correct external libraries. For example, do not link the libraries built with PathScale when compiling the WRF components with gfortran. The same options used to build the netCDF libraries must be used when building the WRF code (32 *vs* 64 bit, assumptions about underscores in the symbol names, etc.).

Note 3: If netCDF-4 is used, be sure that it is installed without activating parallel I/O based on HDF5. The WRF modeling system is able to use either the classic data model from netCDF-3 or the compression options supported in netCDF-4.

If you are going to run distributed memory WRF jobs, you will need an MPI library (e.g., mpich [https://www.mpich.org/]). It may be easiest to have a systems administrator at your institution install this library. A working installation of MPI is required prior to a build of WRF using distributed memory. Either MPI-1 or MPI-2 are acceptable. If you already have an MPI library lying around, try

```
which mpif90
which mpicc
which mpirun
```

If these are all defined executables in your path, you are probably OK. Make sure your paths are set up to point to the MPI **`lib`**, **`include`**, and **`bin`** directories. As with the netCDF libraries, you must build MPI consistently with the WRF source code.

Note that to output WRF model data in Grib1 format, Todd Hutchinson (https://www.ibm.com/weather) has provided a complete source library that is included with the software release. However, when trying to link the WPS, the WRF model, and the WRFDA data streams together, always use the netCDF format.

Post-Processing Utilities

The more widely used *(and therefore supported)* WRF post-processing utilities are:

- NCL (http://www.ncl.ucar.edu/)

  - NCAR Command Language written by NCAR's Computer Information Systems Laboratory
  - NCL scripts written and maintained by WRF support
  - many template scripts are provided that are tailored for specific real-data and idealized cases (http://www2.mmm.ucar.edu/wrf/OnLineTutorial/Graphics/NCL/NCL_examples.php)
  - raw WRF output can be input with the NCL scripts
  - interactive or command-file driven

- GrADS (http://grads.iges.org/grads/grads.html)
  - download GrADS executable, build format converter
  - program *ARWpost* is available to convert the WRF output into an input format suitable for GrADS
  - simple to generate publication quality
  - interactive or command-file driven

- RIP4 (https://www2.mmm.ucar.edu/wrf/users/docs/ripug.htm)
  - RIP4 written and maintained by Mark Stoelinga, UW
  - interpolation to various surfaces, trajectories, hundreds of diagnostic calculations
  - Fortran source provided
  - based on the NCAR Graphics package
  - pre-processor converts WRF, WPS, and WRFDA data to RIP input format
  - table driven

UNIX Environment Settings

There are only a few environmental settings that are WRF system related. Most of these are not required, but when things start acting badly, test some out. In Cshell syntax:

- **`setenv WRF_EM_CORE 1`**
  - explicitly defines which model core to build
- **`setenv WRF_NMM_CORE 0`**
  - explicitly defines which model core NOT to build
- **`setenv WRF_DA_CORE 0`**
  - explicitly defines no data assimilation
- **`setenv NETCDF /usr/local/netcdf`** (or wherever you have it stored)
  - all of the WRF components want both the lib and the include directories
- **`setenv OMP_NUM_THREADS n`** (where ***n*** is the number of procs to use)
  - if you have OpenMP on your system, this is how to specify the number of threads
- **`setenv MP_STACK_SIZE 64000000`**
  - OpenMP blows through the stack size, set it large

- However, if the model still crashes, it may be a problem of over-specifying stack size. Set stack size sufficiently large, but not unlimited.
- On some systems, the equivalent parameter could be KMP_STACKSIZE, or OMP_STACKSIZE
- Note: There are many other issues that can contribute to model failure that may not be due to stack size.

- **`unlimit`**
  - especially if you are on a small system

Building the WRF Code

The WRF code's build mechanism tries to determine the architecture of your computing system, and then presents options to select the preferred build method. For example, if you are on a Linux machine, it determines whether you are using a 32 or 64 bit machine, and then prompts for the desired usage of processors (such as serial, shared memory, or distributed memory). From the available compiling options in the build mechanism, only select an option for a compiler you know is installed on your system (e.g., do not choose a PGI build if you do not have PGI compilers installed on your system).

An instructional web site describes the sequence of steps required to build the WRF and WPS codes (though the instructions are specifically given for tcsh and GNU compilers).

http://www2.mmm.ucar.edu/wrf/OnLineTutorial/compilation_tutorial.php

- Get the WRF system code (including WRFDA and WRF-Chem) from
  - http://www2.mmm.ucar.edu/wrf/users/download/get_source.html
  - Always get the latest version if you are not continuing a long project, or duplicating previous work
- **`cd WRF`**
- **`./configure`**
  - **serial** means single processor
  - **smpar** means Symmetric Multi-Processing/Shared Memory Parallel (OpenMP) – this does not reliably work on most non-IBM machines
  - **dmpar** means Distributed Memory Parallel (MPI)
  - **dm+sm** means Distributed Memory with Shared Memory (for example, MPI across nodes with OpenMP within a node) – usually better performance is through **dmpar** only
  - The second option is for nesting: 0 = no nesting, 1 = standard static nesting, 2 = nesting with a prescribed set of moves, 3 = nesting that allows a domain to follow a vortex (typhoon tracking)
  - A typical option that may be included on the ./**`configure`** command is the flag "**`-d`**" (for debug). This option removes optimization, which is useful when running a debugger (such as gdb or dbx)

  - For bounds checking and some additional exception handling, the debugging flag "**-D**" may be selected. Only PGI, Intel, and gfortran have been set up to use this option.
- **./compile em_real** (or any of the directory names in **./WRF/test** directory)
- **ls -ls main/*.exe**
  - If you built a real-data case, you should see **ndown.exe**, **real.exe**, and **wrf.exe**
  - If you built an ideal-data case, you should see **ideal.exe** and **wrf.exe**

The WRF code supports a parallel *build* option, an option that compiles separate source code files in the WRF directories at the same time on separate processors (though those processors need to share memory) via a parallel make. The purpose of this option is to speed-up the time required to construct executables. In practice, users typically see approximately a 2x speed-up, a limit imposed by the various dependencies in the code due to modules and USE association. To enable the parallel build option, set an environment variable, J. For example, in csh, to utilize two processors, before the ./compile command, issue the following:

```
setenv J "-j 2"
```

Users may wish to only use a single processor for the build. In which case:

```
setenv J "-j 1"
```

WRF chemistry code is packaged with the WRF code. Inside the WRF directory, is a directory named "chem" which contains the chemistry-specific code. To build the model for chemistry purposes, users will need to compile following guidance found here: https://ruc.noaa.gov/wrf/wrf-chem/.

Building the WPS Code

Building WPS requires that WRF be already built.

If you plan to use Grib2 data, additional libraries for zlib, png, and jasper are required. Please see details in Chapter 3.

- Get the WPS code from
  - http://www2.mmm.ucar.edu/wrf/users/download/get_source.html
- Download the geographical datasets (https://www2.mmm.ucar.edu/wrf/users/download/get_sources_wps_geog.html).
  * Note there are newer data sets for land cover for North America (NLCD), and high-resolution urban data sets for select North American cities.
- **cd WPS**
- **./configure**

- Choose an option that corresponds with your installed compiler.
- Serial builds are always recommended unless you plan to use a very large domain (1000's x 1000's of grid cells), even if you chose a non-serial option for WRF. The WPS executables run quickly.
- WPS requires that you build for the appropriate Grib decoding. Select an option that is suitable for the data you will use with the ungrib program (the Grib2 option will work for either Grib1 or Grib2 data)
- If you select a Grib2 option, you must have those libraries prepared and built in advance (see the chapter on WPS for the location of these compression libraries). Add the paths to these libraries and include files using variables COMPRESSION_LIBS and COMPRESSION_INC in **`configure.wps`**. Inside the **`configure.wps`** file is the location of the built WRF directory, which may need to be modified, depending where you build WRF and what you named the directory. The easiest solution is to simply set the WRF_DIR environment variable (**`setenv WRF_DIR`** ***`path-to-wrf-directory/WRF-directory-name`***). This is how the WPS picks up all of the required IO pieces to build the geogrid.exe and metgrid.exe files.

- **`./compile`**
- **`ls -ls *.exe`**
  - You should see **`geogrid.exe`**, **`ungrib.exe`**, and **`metgrid.exe`** (if you are missing both **`geogrid.exe`** and **`metgrid.exe`**, you likely need to correct the path to WRF in the **`configure.wps`** file (or set the WRF_DIR variable as indicated above); if you are missing **`ungrib.exe`**, the problem is likely that your COMPRESSION_LIB and COMPRESSION_INC variables are not set correctly in **`configure.wps`**, or that the Grib2 libraries are not linked or built correctly.
- **`ls -ls util/*.exe`**
  - You should see a number of utility executables and NCL scripts: **`avg_tsfc.exe`**, **`calc_ecmwf_p.exe`**, **`g1print.exe`**, **`g2print.exe`**, **`height_ukmo.exe`**, **`int2nc.exe`**, **`mod_levs.exe`**, **`rd_intermediate.exe`**, **`gfs.ncl`**, **`plotfmt.ncl`**, **`plotfmt_nc.ncl`**, and **`plotgrids.ncl`** (NCL files require an installation of NCAR Graphics, which comes with the NCL package [https://www.ncl.ucar.edu/])