COMPILING WRF & WPS

Kelly Werner, *NCAR/MMM*

PRESENTATION OUTLINE

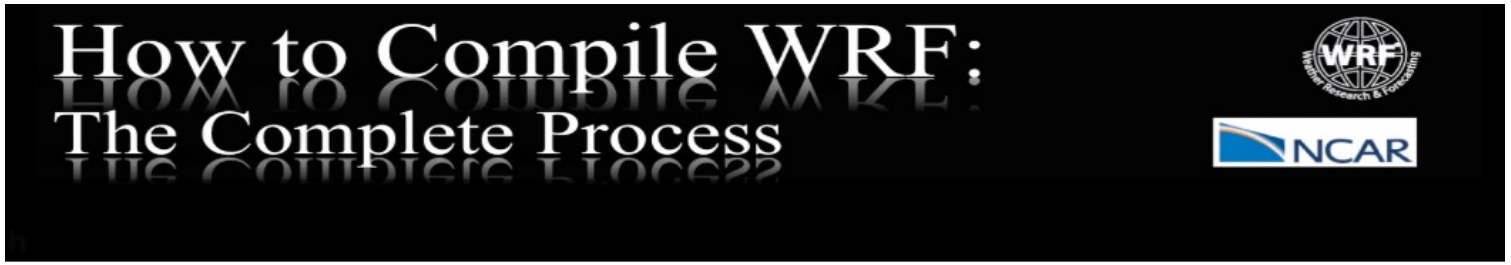
1. Check system requirements
2. Install libraries
3. Obtain source code
4. Compile WRF
5. Compile WPS
6. Troubleshooting

ACCEPTABLE SYSTEMS FOR INSTALLING WRF

- Generally any 32- or 64-bit hardware, running a UNIX-like operating system
- Dual-booting into a UNIX-like OS (e.g., Windows with Linux parallel build) is also okay

Examples

- Laptops, desktops, and clusters running Linux
- Laptops and desktops running MacOS
- Clusters running Unix-like OS: Linux, AIX



This page is meant to provide guidance through the steps of compiling WRF. It will take a beginning user through the processes of e
testing the components and their compatibility with each other, then to installing WRF and WPS, and finally to some guidance for pr

Click on a tab below for quick navigation. If you are a beginner, it is recommended to start at the beginning and follow through each

System Environment Tests | Building Libraries | Library Compatibility Tests | Building WRFV3

Building WPS | Static Geography Data | Real-time Data | Run WPS and WRFV3

**IMPORTANT NOTES: PLEASE READ BEFORE CONTINUING!

- In order to use personal machines, you must have all the pre-required programs and compilers built, as well as their functionality,

responsible or provide assistance for the installation of Linux, Linux utilities, or the compilers.

WRF/WPS Compiling Website

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

CHECK SYSTEM REQUIREMENTS

Mandatory Requirements

- Fortran compiler (e.g., gfortran)
- C compiler
- cpp

Check if they exist on your system

```
> which gfortran
> which cpp
> which gcc
```

If installed, you will be given a path for each

NOTE: You cannot compile newer versions of WRF with older gcc, or older versions of WRF with newer gcc.
Check the version:

```
> gcc -version
```

System Environment Tests

1. First and foremost, it is very important to have a gfortran compiler, as well as gcc and cpp. To test whether these exist on the system, type the following:

    - `which gfortran`
    - `which cpp`
    - `which gcc`

    If you have these installed, you should be given a path for the location of each.

    We recommend using a Fortran compiler that supports Fortran2003 standard (version 4.6 or later). To determine the version of gfortran you have, type:

    ```
    gcc --version
    ```

2. Create a new, clean directory called `Build_WRF`, and another one called `TESTS`.

3. There are a few simple tests that can be run to verify that the fortran compiler is built properly, and that it is compatible with the C compiler.
   NOTE: If any of these tests fail, you will need to contact the systems administrator at your institution for help, as these are specific to your particular environment, and we do not have the resources to support these types of errors.

   Below is a tar file that contains the tests. Download the tar file and place it in the `TESTS` directory.

   Fortran and C Tests Tar File

   To unpack the tar file, type:

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

Tests are available for checking that your fortran compiler is built properly, and is compatible with the C compiler.

Scripting Language & Unix Command Requirements

Scripting Languages

(testing available in test package)

csh
perl
sh

Unix Commands

ar	awk	head	sed	hostname	sleep
cat	ls	sort	tar	cd	cp
make	touch	mkdir	tr	expr	mv
wc	uname	grep	rm	file	printf
nm	which				

PRESENTATION OUTLINE

1. Check system requirements
2. Install libraries
3. Obtain source code
4. Compile WRF
5. Compile WPS
6. Troubleshooting

Required Libraries

NetCDF (for WRF & WPS)

- NetCDF Version 3 or 4
- If using netCDF4 capabilities, you must install HDF5
  - See *Notes on Building NetCDF4 for WRF*

Libraries for GRIB2 Meteorological Data

- JasPer (JPEG 2000 “lossy” compression library)
- PNG (“lossless” compression library)
- Zlib (compression library used by PNG)

Optional MPI Library

- E.g., MPICH2 or OpenMPI
- In principle any implementation of the MPI standard should work, but we have the most experience with MPICH.

Important Information For Library Installation

- These libraries (MPICH2, NetCDF, JasPer, zlib, & libpng) are NOT included in the WPS & WRF installation packages.

- Compilation website includes library files for download, and includes
  - Installation instructions
  - Library compatibility tests

VERY IMPORTANT!
Make sure libraries are installed using the same compilers (and versions) that will be used to install WRF & WPS

Before Installing Libraries: Set Environment Variables

***Example for Installing GNU**

```
> export DIR=directory-where-you-are-installing-libs
> export CC=gcc
> export CXX=g++
> export FC=gfortran
> export FCFLAGS=-m64
> export F77=gfortran
> export FFLAGS=-m64
> export JASPERLIB=$DIR/grib2/lib
> export JASPERINC=$DIR/grib2/include
> export LDFLAGS="-L$DIR/grib2/lib"
> export CPPFLAGS="-I$DIR/grib2/include"
```

NOTE

These can be set either in the command line, or in the environment shell script so the settings will be saved and initiated each time a terminal window is opened.

Installing NetCDF

```
> tar xzvf netcdf-4.1.3.tar.gz     # no '.gz' if downloaded to most Macs
> cd netcdf-4.1.3
> ./configure --prefix=$DIR/netcdf --disable-dap --dissable-netcdf-4 /
     --disable-shared
> make
> make install
> setenv PATH $DIR/netcdf/bin:$PATH
> setenv NETCDF $DIR/netcdf
> cd ..
```

- Step-by-step instructions for installing remaining libraries (MPICH2, zlib, libpng, jasper) available from How to Compile WRF
- Order of installation is important for some libraries.
  - Install NetCDF first
  - When installing GRIB2 compression libraries, install zlib before libpng

Set Paths in Environment Script

Environment Scripts: e.g., .cshrc, .bash, .tcshrc

```
> export DIR=directory-where-your-tar-files-are
> export PATH=$DIR/mpich/bin:$DIR/netcdf/bin:$PATH
> export NETCDF=$DIR/netcdf
> export LD_LIBRARY_PATH=$DIR/grib2/lib
> export JASPERLIB=$DIR/grib2/lib
> export JASPERINC=$DIR/grib2/include
```

Ensures settings are in place every time you open a new terminal window!

Library Compatibility Checks

Make sure libraries are compatible with compilers

Test 1

Fortran + C + netCDF

Test 2

Fortran + C + netCDF + MPI

Library Compatibility Tests

- Once the target machine is able to make small Fortran and C executables (what was verified in the System Environment Tests section), and after the NetCDF and MPI libraries are constructed (two of the libraries from the Building Libraries section), to emulate the WRF code's behavior, two additional small tests are required. We need to verify that the libraries are able to work with the compilers that are to be used for the WPS and WRF builds.
  NOTE: If any of these tests fail, you will need to contact the systems administrator at your institution for help, as these are specific to your particular environment, and we do not have the resources to support these types of errors.

  Below is a tar file that contans these tests. Download this tar file and place it in the `TESTS` directory, and then "cd" into the `TESTS` directory:

  Fortran_C_NETCDF_MPI_tests.tar

  To unpack the tar file, type:

  ```
  tar -xf Fortran_C_NETCDF_MPI_tests.tar
  ```

- There are 2 tests:

  1. **Test #1**: Fortran + C + NetCDF

     The NetCDF-only test requires the include file from the NETCDF package be in this directory. Copy the file here:

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

PRESENTATION OUTLINE

1. Check system requirements
2. Install libraries
3. Obtain source code
4. Compile WRF
5. Compile WPS
6. Troubleshooting

Obtain WRF/WPS Source Code

WRF Source Code Registration and Download

Beginning with V4.0 of the WRF/WRFDA/WRF-Chem/WPS code, all release downloads and corresponding information will be available from our public WRF-Model GitHub page. **For code downloads prior to V4.0, click here.**

There are 2 methods to obtain the WRF-Modeling System source code:

1. The recommended method is to clone the code from our public GitHub repository. This can be done in the command-line. This options requires an installation of git (which most modern systems likely already have – you can check with the command (csh e.g.): which git). This method provides more flexibility to update the version and facilitates the most direct method for contributing development back into the WRF-Model code base.

   WRF Model Source Code (includes WRF, WRFDA, & WRF-Chem):
   `git clone https://github.com/wrf-model/WRF`

   WRF Preprocessing System Source Code :
   `git clone https://github.com/wrf-model/WPS`

   See the archives page for all **release notes.**

   Since V4.0, WRFDA/WRFPlus code is now fully-integrated into the WRF code. See the WRFDA V4.0 Update Summary and chapter 6 of the Users Guide for additional information.

2. The second method is to aquire the code through the archive file on GitHub. The disadvantage to this method is the lack of flexibility with the ability to troubleshoot with version control. Archive files are provided in both zip and tar.gz formats. Each release provides an archive file, and users should download the archive file for the most relevant released version.

   WRF Model Archive File (includes WRF, WRFDA, WRF-Chem)

   WRF Preprocessing System (WPS) Model Archive File

Obtain Source Code from:

WRF Source Codes and Graphics Software

1. Choose ‘New User,’ and then register, or
2. Click ‘Returning User,’ enter your email, and go to the download page.

All code is stored in a GitHub repository, and can be obtained by:

- Cloning from GitHub
- Downloading archived tar file from GitHub

Cloning Source Code from GitHub

Clone WRF from GitHub repository *"wrf-model"*

```
cheyenne:/glade/scratch/kkeene>git clone --recurse-submodule https://git@github.com/wrf-model/WRF
Cloning into 'WRF'...
remote: Enumerating objects: 62656, done.
remote: Counting objects: 100% (250/250), done.
remote: Compressing objects: 100% (134/134), done.
remote: Total 62656 (delta 130), reused 189 (delta 116), pack-reused 62406
Receiving objects: 100% (62656/62656), 266.71 MiB | 22.74 MiB/s, done.
Resolving deltas: 100% (48594/48594), done.
Updating files: 100% (4758/4758), done.
Submodule 'phys/noahmp' (https://github.com/NCAR/noahmp) registered for path 'phys/noahmp'
Cloning into '/glade/scratch/kkeene/WRF/phys/noahmp'...
remote: Enumerating objects: 1149, done.
remote: Counting objects: 100% (285/285), done.
remote: Compressing objects: 100% (216/216), done.
remote: Total 1149 (delta 76), reused 245 (delta 62), pack-reused 864
Receiving objects: 100% (1149/1149), 7.23 MiB | 21.72 MiB/s, done.
Resolving deltas: 100% (377/377), done.
Submodule path 'phys/noahmp': checked out '3be0b2860dab167006a0b3c4822e234ca253c3df'
cheyenne:/glade/scratch/kkeene>
```

**Must have 'git' installed on your system!

PRESENTATION OUTLINE

1. Check system requirements
2. Install libraries
3. Obtain source code
4. Compile WRF
5. Compile WPS
6. Troubleshooting

Step 1: Configure for WRF

In the WRF directory, issue:

./configure

Configuration Output

configure.wrf

```
$JASPERLIB or $JASPERINC not found in environment, configuring to build without grib2 I/O...
------------------------------------------------------------------------
Please select from among the following Linux x86_64 options:

  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 $(NOOMP)/cc): 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
 64. (serial)  65. (smpar)  66. (dmpar)  67. (dm+sm)   INTEL (ifort/icc): HSW/BDW
 68. (serial)  69. (smpar)  70. (dmpar)  71. (dm+sm)   INTEL (ifort/icc): KNL MIC
 72. (serial)  73. (smpar)  74. (dmpar)  75. (dm+sm)   FUJITSU (frtpx/fccpx): FX10/FX100 SPARC64 IXfx/Xlfx

Enter selection [1-75] : 34
------------------------------------------------------------------------
Compile for nesting? (1=basic, 2=preset moves, 3=vortex following) [default 1]:
```

Additional Configuration Options

```
> ./configure -d
```

- No optimization
- Extra debugging

```
> ./configure -D
```

- No optimization
- Checks uninitialized variables

```
> ./configure -r8
```

- Double-precision
- *Works for GNU, Intel, and PGI compilers*

Compiling With Multiple Processors

To build WRF with multiple processors, before or after configure, set (bash e.g.):

```
export J="-j 6"
```

# of Processors	Time to Compile
1	38 Mins 43 Secs
2	25 Mins 4 Secs
3	21 Mins 48 Secs
4	20 Mins 14 Secs
5	19 Mins 16 Secs
6	19 Mins 10 Secs

*Compiled with GNU V10.1.0

Step 2: Compile WRF

In the WRF/ directory, type

```
./compile em_case >& compile.log
```

where **em_case** *is one of the following (type* **./compile** *to see all options):*

Real-data Case

em_real

3D Ideal Cases

em_quarter_ss
em_b_wave
em_les
em_heldsuarez
em_tropical_cyclone
em_convrad

2D Ideal Cases

em_hill2d_x
em_squall2d_x
em_squall2d_y
em_grav2d_x
em_seabreeze2d_x

1D Ideal Case

em_scm_xy

Compilation will take ~10-50 mins

Successful Compile

If compiling is successful, this message appears at the end of the compile log:

```
--->      Executables successfully built                  <---

-rwxr-xr-x 1 wrfhelp ncar 54128088 Jul 22 15:54 main/ndown.exe
-rwxr-xr-x 1 wrfhelp ncar 54180032 Jul 22 15:54 main/real.exe
-rwxr-xr-x 1 wrfhelp ncar 53464304 Jul 22 15:54 main/tc.exe
-rwxr-xr-x 1 wrfhelp ncar 60686816 Jul 22 15:52 main/wrf.exe
```

Real data case

wrf.exe - model executable
real.exe – real data initialization
ndown.exe – separate one-way nesting
tc.exe – for tropical cyclone bogusing

```
--->      Executables successfully built                  <---

-rwxr-xr-x 1 wrfhelp ncar 45136368 Nov  2 17:39 main/ideal.exe
-rwxr-xr-x 1 wrfhelp ncar 53253904 Nov  2 17:39 main/wrf.exe
```

Ideal case

wrf.exe - model executable
ideal.exe – ideal data initialization

**Note: Each ideal case compile creates a different ideal.exe executable, but with the same name*

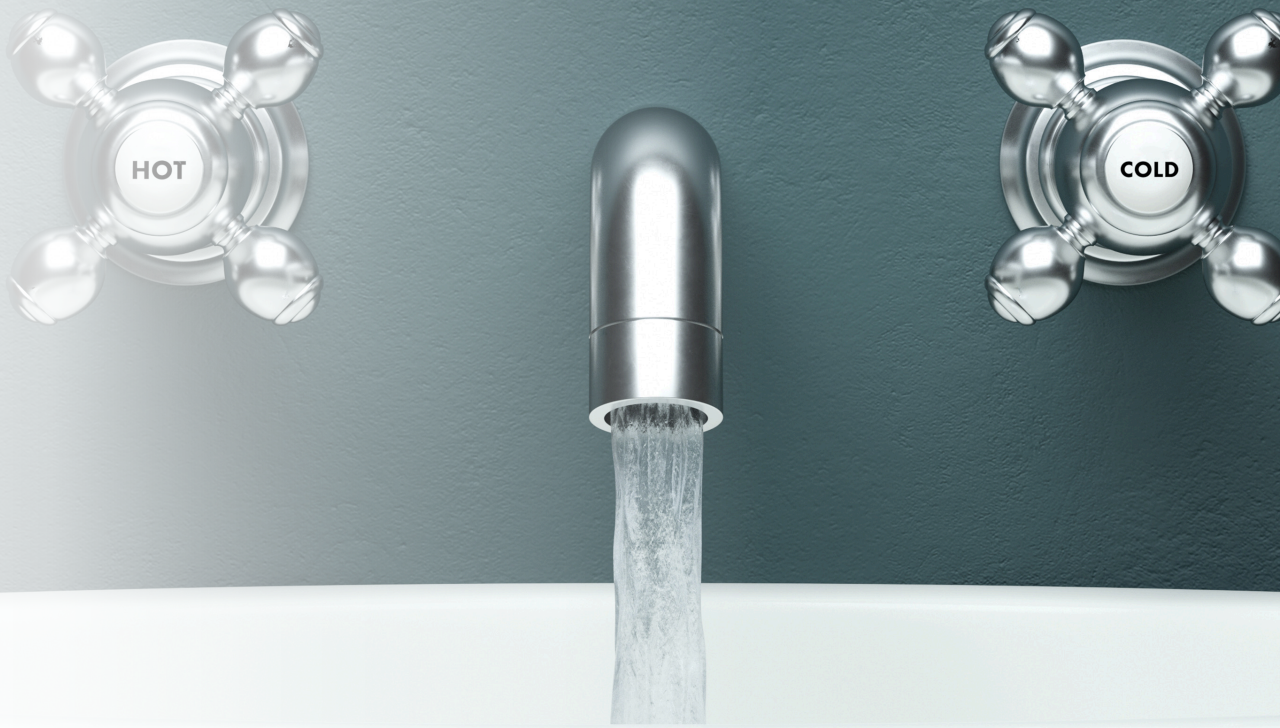
These executables are linked to 2 different directories. You can go to either place to run WRF.

- WRF/run
- WRF/test/em_*case*

Using "*clean –a*"

- The './clean –a' command should be used when modifications have been made to the configure.wrf(wps) file, or any changes to the registry. If so, issue 'clean –a' prior to recompiling.

- Modifications to subroutines within the code will require a recompile, but DO NOT require a 'clean –a', nor a reconfigure. Simply recompile. This compilation should be much faster than a clean compile.



Choosing the Right Compiler

Compile

WRF V4.0
dmpar/basic nesting
4 processors

Compiler	Compile Time	Run Time
GNU 6.3.0 **FREE**	25.01 Mins	3.92 Mins
Intel 17.0.1	46.77 Mins	2.20 Min

Run

Single domain
Small domain (75x70), 30km resolution
12 hours
8 processors

PRESENTATION OUTLINE

1. Check system requirements
2. Install libraries
3. Obtain source code
4. Compile WRF
5. Compile WPS
6. Troubleshooting

Step 1: Configure for WPS

Inside the WPS/ directory, type

```
export WRF_DIR=path-to-top-level-WRF/WRF
./configure
```

```
$JASPERLIB or $JASPERINC not found in environment. Using default values for library paths...
------------------------------------------------------------------------
Please select from among the following supported platforms.

   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)
```

Configuration Output

configure.wps

- Always choose a **serial** compile for WPS (even if you compile WRF with a parallel option)
  - * Exception: You are using a VERY large domain (1000's x 1000's)
  - NOTE: if you do compile WPS in parallel, ungrib.exe must run serially

Step 2: Compile WPS

In the WPS/ directory, type

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

*Compilation should be quick.

If successful, these executables should be in your WPS/ directory (linked from their source code directories):

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

PRESENTATION OUTLINE

1. Check system requirements
2. Install libraries
3. Obtain source code
4. Compile WRF
5. Compile WPS
6. Troubleshooting

Failed WRF Compile

Search for errors in the *compile.log*

- Search for ‘Error’ with a capital ‘E’
- Typically the first ‘Error’ in the file is the culprit

Visit the WRF & MPAS-A Support Forum

- See Frequently Asked Questions (FAQ)
- Search the forum to see if your issue has already been addressed – if not, post a new topic

Before Recompiling

- Issue the *./clean –a* command
- Reconfigure
  - If you need to make changes to the *configure.wrf* file, do that after issuing *./configure*, and then save the edited file
- Recompile

Failed WPS Compile:
No geogrid or metgrid

- Did WRF compile successfully?

  WPS uses the external I/O libraries from WRF, which are built when WRF is installed

- Check that you are using the same compiler (& version) that was used to compile WRF

- Check that you are using the same NetCDF (& version) that was used to compile WRF

Failed WPS Compile:
No ungrib

- Make sure jasper, zlib, and libpng libraries are correctly installed

- Make sure you are using the correct path format for these lines in *configure.wps*

```
COMPRESSION_LIBS = -L/${DIR}/UNGRIB_LIBRARIES/lib -ljasper -lpng -lz
COMPRESSION_INC = -I/${DIR}/UNGRIB_LIBRARIES/include
```

THANK YOU!