How to Compile WRF: The Complete Process

This page is meant to provide guidance through the steps of compiling WRF. It will take a beginning user through the processes of ensuring the computer environment is set up correctly, to testing the components and their compatibility with each other, then to installing WRFV3 and WPS, and finally to some guidance for preparing to run WPS and then WRFV3.

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 step.

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/compatibility verified through testing. We cannot be responsible or provide assistance for the installation of Linux, Linux utilities, or the compilers.

- We are attempting to walk you through the steps for building necessary libraries (netCDF, MPICH, JasPer, Libpng, and Zlib); however, if you experience errors, we cannot be responsible for helping to correct the errors, as these are related to your particular system, and are not supported by our wrfhelp group. You will need to contact someone in your systems administation office, or go to the library websites to contact someone in their support group for assistance.

- All of the examples given here are in tcsh. If you are very familiar with another shell (e.g., bash), and feel comfortable making the necessary alterations to the commands, then feel free to use your other shell. If not, however, we recommend using tcsh.

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 gfortran version 4.4.0 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. 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
   ```

   There are 7 tests available, so start at the top and run through them, one at a time.

   Test #1: Fixed Format Fortran Test: `TEST_1_fortran_only_fixed.f`

   Type the following in the command line:

   ```
   gfortran TEST_1_fortran_only_fixed.f
   ```

   Now type:

   ```
   ./a.out
   ```

   The following should print out to the screen:

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

   Test #2: Free Format Fortran: `TEST_2_fortran_only_free.f90`

   Type the following in the command line:

   ```
   gfortran TEST_2_fortran_only_free.f90
   ```

   and then type:

   ```
   ./a.out
   ```

   The following should print out to the screen:

   ```
   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`

   Type the following in the command line:

   ```
   gcc TEST_3_c_only.c
   ```

   and then type:

   ```
   ./a.out
   ```

   The following should print out to the screen:

   ```
   SUCCESS test 3 c only
   ```

   Test #4: Fortran Calling a C Function (our gcc and gfortran have different defaults, so we force both to always use 64 bit [-m64] when combining them): `TEST_4_fortran+c_c.c`, and `TEST_4_fortran+x_f.f90`

   Type the following in the command line:

   ```
   gcc -c -m64 TEST_4_fortran+c_c.c
   ```

   and then type:

   ```
   gfortran -c -m64 TEST_4_fortran+c_f.f90
   ```

   and then:

   ```
   gfortran -m64 TEST_4_fortran+c_f.o
   TEST_4_fortran+c_c.o
   ```

   and then issue:

   ```
   ./a.out
   ```

   The following should print out to the screen:

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

4. In addition to the compilers required to manufacture the WRF executables, the WRF build system has scripts as the top level for the user interface. The WRF scripting system uses, and therefore having the following is necessary:

   - csh
   - perl
   - sh

   To test whether these scripting languages are working properly on the system, there are 3 tests to run. These tests were included in the "Fortran and C Tests Tar File".

   Test #5:csh *In the command line, type:*

   ```
   ./TEST_csh.csh
   ```

   The result should be:

   ```
   SUCCESS csh test
   ```

   Test #6:perl *In the command line, type:*

   ```
   ./TEST_perl.pl
   ```

   The result should be:

   ```
   SUCCESS perl test
   ```

   Test #7:sh *In the command line, type:*

   ```
   ./TEST_sh.sh
   ```

   The result should be:

   ```
   SUCCESS sh test
   ```

5. Finally, inside the scripts are quite a few UNIX commands that are available regardless of which shell is used. The following standard UNIX commands are mandatory:

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

Go to top of page

<

Building Libraries

- Before getting started, you need to make another directory. Go inside your `Build_WRF` directory:

  ```
  cd Build_WRF
  ```

  and then make a directory called "LIBRARIES"

  ```
  mkdir LIBRARIES
  ```

- Depending on the type of run you wish to make, there are various libraries that should be installed. Below are 5 libraries. Download all 5 tar files and place them in the `LIBRARIES` directory.

  mpich-3.0.4
  netcdf-4.1.3
  Jasper-1.900.1
  libpng-1.2.50
  zlib-1.2.7

- **It is important to note that these libraries must all be installed with the same compilers as will be used to install WRFV3 and WPS.**

1. **NetCDF**: This library is always necessary!

   ```
   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

   tar xzvf netcdf-4.1.3.tar.gz      #or just .tar if no .gz
   present
   cd netcdf-4.1.3
   ./configure --prefix=$DIR/netcdf --disable-dap \
        --disable-netcdf-4 --disable-shared
   make
   make install
   setenv PATH $DIR/netcdf/bin:$PATH
   setenv NETCDF $DIR/netcdf
   cd ..
   ```

2. **MPICH**: This library is necessary if you are planning to build WRF in parallel. If your machine does not have more than 1 processor, or if you have no need to run WRF with multiple processors, you can skip installing MPICH.

   In principle, any implementation of the MPI-2 standard should work with WRF; however, we have the most experience with MPICH, and therefore, that is what will be described here.

   Assuming all the 'setenv' commands were already issued while setting up NetCDF, you can continue on to install MPICH, issuing each of the following commands:

   ```
   tar xzvf mpich-3.0.4.tar.gz      #or just .tar if no .gz
   present
   cd mpich-3.0.4
   ./configure --prefix=$DIR/mpich
   make
   make install
   setenv PATH $DIR/mpich/bin:$PATH
   cd ..
   ```

3. **zlib**: This is a compression library necessary for compiling WPS (specifically ungrib) with GRIB2 capability
   Assuming all the "setenv" commands from the NetCDF install are already set, you can move on to the commands to install zlib.

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

   tar xzvf zlib-1.2.7.tar.gz      #or just .tar if no .gz
   present
   cd zlib-1.2.7
   ./configure --prefix=$DIR/grib2
   make
   make install
   cd ..
   ```

4. **libpng**: This is a compression library necessary for compiling WPS (specifically ungrib) with GRIB2 capability
   Assuming all the "setenv" commands from the NetCDF install are already set, you can move on to the commands to install zlib.

   ```
   tar xzvf libpng-1.2.50.tar.gz      #or just .tar if no .gz
   present
   cd libpng-1.2.50
   ./configure --prefix=$DIR/grib2
   make
   make install
   cd ..
   ```

5. **JasPer**: This is a compression library necessary for compiling WPS (specifically ungrib) with GRIB2 capability
   Assuming all the "setenv" commands from the NetCDF install are already set, you can move on to the commands to install zlib.

   ```
   tar xzvf jasper-1.900.1.tar.gz      #or just .tar if no .gz
   present
   cd jasper-1.900.1
   ./configure --prefix=$DIR/grib2
   make
   make install
   cd ..
   ```

Go to top of page

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. 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 .
     ```

     Compile the Fortran and C codes for the purpose of this test (the -c option says to not try to build an executable). Type the following commands:

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

     The following should be displayed on your screen:

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

  2. **Test #2**: Fortran + C + NetCDF + MPI

     The NetCDF+MPI test requires include files from both of these packages be in this directory, but the MPI scripts automatically make the `mpif.h` file available without assistance, so no need to copy that one. Copy the NetCDF include file here:

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

     Note that the MPI executables `mpif90` and `mpicc` are used below when compiling. Issue the following commands:

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

     The following should be displayed on your screen:

     ```
     C function called by Fortran
     Values are xx = 2.00 and ii = 1
     status = 2
     SUCCESS test 2 fortran + c + netcdf + mpi
     ```

Go to top of page

Building WRFV3

- After ensuring that all libraries are compatible with the compilers, you can now prepare to build WRFV3. If you do not already have a WRFV3 tar file, you can find it below. Download that file and unpack it in the `Build_WRF` directory.

  WRFV3.9.1.1

  ```
  gunzip WRFV3.9.1.1.TAR.gz
  tar -xf WRFV3.9.1.1.TAR
  ```

- Go into the WRFV3 directory:

  ```
  cd WRFV3
  ```

- Create a configuration file for your computer and compiler:

  ```
  ./configure
  ```

  You will see various options. Choose the option that lists the compiler you are using and the way you wish to build WRFV3 (i.e., serially or in parallel). Although there are 3 different types of parallel (smpar, dmpar, and dm+sm), we have the most experience with dmpar and typically recommend choosing this option.

- Once your configuration is complete, you should have a `configure.wrf` file, and you are ready to compile. To compile WRFV3, you will need to decide which type of case you wish to compile. The options are listed below:

  `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)

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

  where *`case_name`* is one of the options listed above

  Compilation should take about 20-30 minutes.

- Once the compilation completes, to check whether it was successful, you need to look for executables in the `WRFV3/main` directory:

  ```
  ls -ls main/*.exe
  ```

  If you compiled a real case, you should see:

  `wrf.exe` (model executable)
  `real.exe` (real data initialization)
  `ndown.exe` (one-way nesting)
  `tc.exe` (for tc bogusing--serial only)

  If you compiled an idealized case, you should see:

  `wrf.exe` (model executable)
  `ideal.exe` (ideal case initialization)

  These executables are linked to 2 different directories:

  ```
  WRFV3/run
  WRFV3/test/em_real
  ```

  You can choose to run WRF from either directory.

Go to top of page

Building WPS

- After the WRF model is built, the next step is building the WPS program (if you plan to run real cases, as opposed to idealized cases). The WRF model MUST be properly built prior to trying to build the WPS programs. Below is a tar file containing the WPS source code. Download that file and unpack it in the `Build_WRF` directory:

  WPSV3.9.1

  ```
  gunzip WPSV3.9.1.TAR.gz
  tar -xf WPSV3.9.1.TAR
  ```

- Go into the WPS directory:

  ```
  cd WPS
  ```

- Similar to the WRF model, make sure the WPS directory is clean, by issuing:

  ```
  ./clean
  ```

- The next step is to configure WPS, however, you first need to set some paths for the ungrib libraries:

  ```
  setenv JASPERLIB $DIR/grib2/lib

  setenv JASPERINC $DIR/grib2/include
  ```

  and then you can configure:

  ```
  ./configure
  ```

  You should be given a list of various options for compiler types, whether to compile in serial or parallel, and whether to compile ungrib with GRIB2 capability. Unless you plan to create extremely large domains, it is recommended to compile WPS in serial mode, regardless of whether you compiled WRFV3 in parallel. It is also recommended that you choose a GRIB2 option (make sure you do not choose one that states "NO_GRIB2"). You may choose a non-grib2 option, but most data is now in grib2 format, so it is best to choose this option. You can still run grib1 data when you have built with grib2.

  Choose the option that lists a compiler to match what you used to compile WRFV3, serial, and grib2. ****Note: The option number will likely be different than the number you chose to compile WRFV3**

- the `metgrid.exe` and `geogrid.exe` programs rely on the WRF model's I/O libraries. There is a line in the `configure.wps` file that directs the WPS build system to the location of the I/O libraries from the WRF model:

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

  Above is the default setting. As long as the name of the WRF model's top-level directory is "WRFV3" and the WPS and WRFV3 directories are at the same level (which they should be if you have followed exactly as instructed on this page so far), then the existing default setting is correct and there is no need to change it. If it is not correct, you must modify the configure file and then save the changes before compiling.

- You can now compile WPS:

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

  Compilation should only take a few minutes.

- If the compilation is successful, there should be 3 main executables in the WPS top-level directory:

  ```
  geogrid.exe
  ungrib.exe
  metgrid.exe
  ```

  Verify that they are not zero-sized. To see file size, you can type:

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

Go to top of page

Static Geography Data

- The WRF modeling system is able to create idealized simulations, though most users are interested in the real-data cases. To initiate a real-data case, the domain's physical location on the globe and the static information for that location must be created. This requires a data set that includes such fields as topography and land use catergories. These data are available from the WRF download page (http://www2.mmm.ucar.edu/wrf/users/download/get_sources_wps_geog.html).

- Download the file and place it in the `Build_WRF` directory. Keep in mind that if you are downloading the complete dataset, the file is very large. If you are sharing space on a cluster, you may want to consider placing this in a central location so that everyone can use it, and it's not necessary to download for each person. Uncompress and un-tar the file:

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

- When you untar the file, it will be called "geog." Rename the file to "WPS_GEOG."

  ```
  mv geog WPS_GEOG
  ```

- The directory infomation is given to the geogrid program in the `namelist.wps` file in the `&geogrid` section:

  ```
  geog_data_path = ´path_to_directory/Build_WRF/WPS_GEOG´
  ```

- The data expands to approximately 10 GB. This data allows a user to run the `geogrid.exe` program.

Go to top of page

Real-time Data

- For real-data cases, the WRF model requires up-to-date meteorological information for both an initial condition and also for lateral boundary conditions. This meteorological data is traditionally a Grib file that is provided by a previously run external model or analysis. For a semi-operational set-up, the meteorological data is usually sourced from a global model, which permits locating the WRF model's domains anywhere on the globe.

- The National Centers for Environmental Prediction (NCEP) run the Global Forecast System (GFS) model four times daily (initializations valid for 0000, 0600, 1200, and 1800 UTC). This is a global, isobaric, 0.5 degree latitude/longitude, forecast data set that is freely available, and is usually accessible +4h after the initialization time period.

- A single data file needs to be acquired for each requested time period. For example, if we would like hours 0, 6, and 12 of a forecast that is initialized 2014 Jan 31 at 0000 UTC, we need the following times:

  2014013100 – 0 h
  2014013106 – 6 h
  2014013112 – 12 h

  These translate to the following file names to access:

  gfs.2014013100/gfs.t00z.pgrb2.0p50.f000
  gfs.2014013100/gfs.t00z.pgrb2.0p50.f006
  gfs.2014013100/gfs.t00z.pgrb2.0p50.f012

  Note that the initialization data and time (gfs.2014013100) remains the same, and that the forecast cycle remains the same (t00z). What is incremented is the forecast hour (f00, f06, f12).

- Before obtaining the data, creat a directory in `Build_WRF`, called "DATA", and then go into that directory:

  ```
  mkdir DATA

  cd DATA
  ```

- A simple set of interactive commands to grab these files from the NCEP servers in real-time would look like (****Note** that this is just an example time/date. Typically on the NCEP data servers, only the most recent 2-3 days are available at any given time. To use up-to-date real-time data, you will need to adjust the commands to reflect current date and time information):

  ```
  curl -s --disable-epsv --connect-timeout 30 -m 60 -u anonymous:USER_ID@INSTITUTION -o GFS_00h
  \
  ftp://ftpprd.ncep.noaa.gov/pub/data/nccf/com/gfs/prod/gfs.2014013100/gfs.t00z.pgrb2.0p50.f000

  curl -s --disable-epsv --connect-timeout 30 -m 60 -u anonymous:USER_ID@INSTITUTION -o GFS_06h
  \
  ftp://ftpprd.ncep.noaa.gov/pub/data/nccf/com/gfs/prod/gfs.2014013100/gfs.t00z.pgrb2.0p50.f006

  curl -s --disable-epsv --connect-timeout 30 -m 60 -u anonymous:USER_ID@INSTITUTION -o GFS_12h
  \
  ftp://ftpprd.ncep.noaa.gov/pub/data/nccf/com/gfs/prod/gfs.2014013100/gfs.t00z.pgrb2.0p50.f012
  ```

  Typically these commands return a complete file within a few seconds. The files returned from these commands (GFS_00h, GFS_06h, GFS_12h) are Grib Edition 2 files, able to be directly used by the ungrib program.

- You need to fill in the anonymous login information (which is not private, so there are no security concerns about leaving these scripts around). You will probably end up writing a short script to automatically increment the initialization time.

</Go to top of page

Run WPS and WRFV3

****Below are basic instructions for running WPS and WRFV3. For more detailed information, please see the WRF-ARW Online Tutorial**

Running WPS

- You are now ready to begin running WPS and WRFV3. Start by going to the WPS directory:

  ```
  cd WPS
  ```

- Make any changes to the `namelist.wps` file, to reflect information for your particular run

- Before running geogrid, make sure that you have your `geog_data_path` set to the location where you put your geography static data. Once that is set, you can run geogrid.

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

- If you successfully created a `geo_em*` file for each domain, then you are ready to prepare to run ungrib. Start by linking in the input GFS data:

  ```
  ./link_grib.csh path_where_you_placed_GFS_files
  ```

  Then link to the correct Vtable (GFS, for this case):

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

  Then run the ungrib executable:

  ```
  ./ungrib.exe
  ```

  You should now have files with the prefix "FILE" (or if you named them something else, they should have that prefix)

- You are now ready to run metgrid:

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

  You should now have files with the prefix `met_em*` for each of the time periods for which you are running.

Running WRFV3

- You are now ready to run WRFV3. Move into the `WRFV3` directory, and then into either the `run/` directory, or the `test/em_real/`
- directory:

  ```
  cd ../WRFV3/run
  ```

  or

  ```
  cd ../WRFV3/test/em_real
  ```

- Before running the "real" program, you need to make all necessary changes to reflect your particular case to the `namelist.input` file. Once that is complete, you need to copy or link your `met_em*` files into the working directory:

  `ln -sf ../../../WPS/met_em* .` (from the `test/em_real` directory), or
  `ln -sf ../../WPS/met_em* .` (from the `run/` directory).

  or, if you would rather copy the files in, instead of linking them, you can use the `cp` command, instead of the `ln -sf` command.
- You can now run the "real" program. The command for running this may vary depending on your system and the number of processors you have available, but it should be something similar to:

  ```
  mpirun -np 1 ./real.exe
  ```

  Check the end of your "rsl" files to make sure the run was successful:

  ```
  tail rsl.error.0000
  ```

  If you see a "SUCCESS" in there, and you see a `wrfbdy_d01` file, and `wrfinput_d0*` files for each of your domains, then the run was successful.
- To run WRFV3, type something similar to:

  ```
  mpirun -np 8 ./wrf.exe
  ```

  Again, check your "rsl" file for "SUCCESS", and make sure you have all the `wrfout*` files you anticipated having. If so, the run was successful, and you are ready to do analysis for your project.

Go to top of page