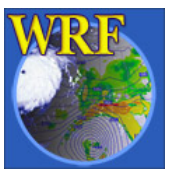
Set Up and Run WRF-ARW
(*ideal.exe / real.exe & wrf.exe*)

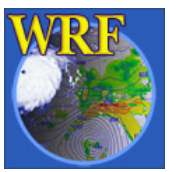
Wei Wang



Outline

- Download, compile and run WRF code
  - single domain here
  - nest talk, tomorrow
- Input and output files
- Check output

(Nesting and Nudging talks later)



Download WRF Source Code

- Download WRF source code from

  http://www.mmm.ucar.edu/wrf/users/downloads.html

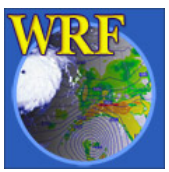
  Click ‘WRF’ on the side menu, then

  > ‘New Users’, register and download, or

  > ‘Returning Users’, your email and download

- What you get is the latest release:

  `WRFV2.2.TAR.gz`



Unzip and Untar WRF File

- Uncompress the file:

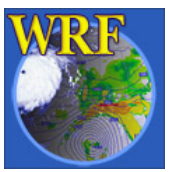
  `gunzip WRFV2.2.TAR.gz`

  `tar -xf WRFV2.2.TAR`

- After unzip and untar, you should see a directory **`WRFV2/`**
- Go to **`WRFV2/`** directory,

  `cd WRFV2`

  and you should see…



WRFV2 Directory

```
Makefile
README
README_test_cases
clean
compile
configure
Registry/
arch/
dyn_em/
dyn_exp/
external/
frame/
inc/
main/
phys/
share/
tools/
run/
test/
```

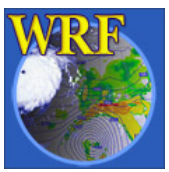
compile scripts

data dictionary

compile rules

source code directories

run directories



How to Compile?

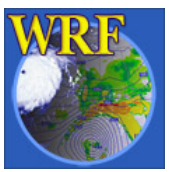
There are two steps:

1) Create a configuration file for your computer

   `./configure`

2) Compile the code

   `./compile` ***`test_case`***



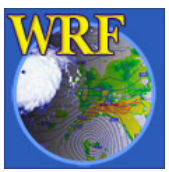
Create configuration file

Step 1: type

`./configure`

This is a script that checks the system hardware and software (mostly *netcdf*), and then offers a user to choose how one wants the code compiled:

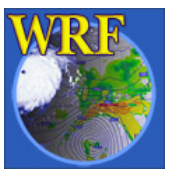
- Serial, OpenMP, or MPI
- RSL or RSL_LITE (interface to MPI)
- Nesting or no nesting



Running configuration script

```
checking for perl5... no
checking for perl... found /usr/bin/perl (perl)
Will use NETCDF in dir: /usr/local/netcdf-pgi
PHDF5 not set in environment. Will configure WRF for use without.
$JASPERLIB or $JASPERINC not found in environment, configuring to build without grib2 I/O...
------------------------------------------------------------------------
Please select from among the following supported platforms.

   1.  PC Linux i486 i586 i686, PGI compiler  (Single-threaded, no nesting)
   2.  PC Linux i486 i586 i686, PGI compiler  (single threaded, allows nesting using RSL without
    MPI)
   3.  PC Linux i486 i586 i686, PGI compiler  SM-Parallel (OpenMP, no nesting)
   4.  PC Linux i486 i586 i686, PGI compiler  SM-Parallel (OpenMP, allows nesting using RSL
    without MPI)
   5.  PC Linux i486 i586 i686, PGI compiler  DM-Parallel (RSL, MPICH, Allows nesting)
   6.  PC Linux i486 i586 i686, PGI compiler  DM-Parallel (RSL_LITE, MPICH, Allows nesting)
```

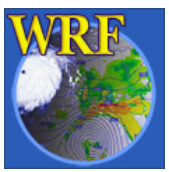


Create a configuration file

The result of running the **`configure`** script is the generation of a file called:

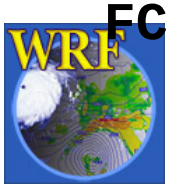
`configure.wrf`

This file contains compilation options, rules etc. specific to your computer.



Sample of what is inside a **`configure.wrf`** file

```
FC              =       pgf90
LD              =       pgf90
CC              =       gcc -DFSEEKO64_OK
SCC             =       $(CC)
RWORDSIZE       =       $(NATIVE_RWORDSIZE)
SFC             =       $(FC)
CFLAGS          =
FCOPTIM         =       -O2 # -fast
FCDEBUG         =       #-g
FCBASEOPTS      =       -w -byteswapio -Mfree
                        -tp p6 $(FCDEBUG)
FCFLAGS         =       $(FCOPTIM) $(FCBASEOPTS)
```



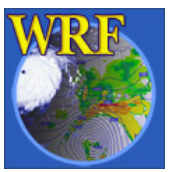
What consists of a **`configure.wrf`** file

The **`configure.wrf`** file is built from three pieces from arch/ directory:

1) **`preamble:`** uniform requirement for the code, such as maximum number of domains, word size, etc.

2) **`configure.defaults:`** selection of compiler, parallel, runtime system library (RSLs)

3) **`postamble:`** standard make rules and dependencies



How to Compile?

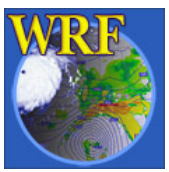
Step 2: type

./compile *test_case* or

./compile test_case >& compile.log

where ***test_case*** is one of the following:

em_real	3d real
em_quarter_ss	3d ideal
em_b_wave	3d ideal
em_hill2d_x	2d ideal
em_squall2d_x	2d ideal
em_squall2d_y	2d ideal
em_grav2d_x	2d ideal

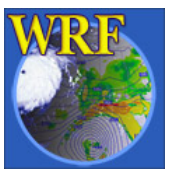


Make change for your system

- If netCDF is not in **`/usr/local`**, you can use environment variable NETCDF to set the path to netCDF before typing **`'configure'`**. e.g. on a Linux with PGI-compiled netCDF:

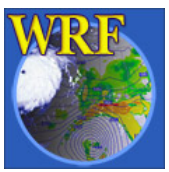
  `setenv NETCDF /usr/local/netcdf-pgi`

- If you use a Linux, a number of compiler may be available (PGI, Intel, g95). As a general rule, make sure your netCDF and MPI libraries are installed using the same compiler you use to compile WRF.



Make change for your system

- One may edit **`configure.wrf`** to make changes for your system
- If option for your system is not available, add one to **`arch/configure.defaults`**



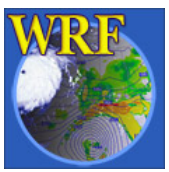
WRF executables: names and locations

If compile is successful, you should find these executables created in WRFV2/main/, if you compile for real data case:

- **`wrf.exe`** - model executable
- **`real.exe`** - real data initialization
- **`ndown.exe`** - one-way nesting
- **`nup.exe`** (not used much)

If you compile a ideal case, you should have:

- **`wrf.exe`** - model executable
- **`ideal.exe`** - ideal case initialization



WRF executables: names and locations

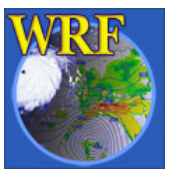
These executables will be linked to:

`WRFV2/run`

and

`WRFV2/test/em_`*`test_case`*

➔ One can go to either directory to run.



WRFV2/run directory

LANDUSE.TBL
ETAMPNEW_DATA
RRTM_DATA
SOILPARM.TBL
VEGPARM.TBL
urban_param.tbl
tr49t67
tr49t85
tr67t85
gribmap.txt
grib2map.tbl

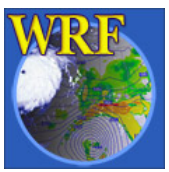
these files are for model physics use, and reside in this directory

namelist.input **-> ../test/*test_case*/*namelist.input***
real.exe -> ../main/real.exe
wrf.exe -> ../main/wrf.exe
ndown.exe -> ../main/ndown.exe

…. (a few more)

WRFV2/test/em_*real* directory

```
LANDUSE.TBL -> ../../run/LANDUSE.TBL
ETAMPNEW_DATA -> ../../run/ETAMPNEW_DATA
RRTM_DATA -> ../../run/RRTM_DATA
SOILPARM.TBL -> ../../run/SOILPARM.TBL
VEGPARM.TBL -> ../../run/VEGPARM.TBL
urban_param.tbl -> ../../run/urban_param.tbl
tr49t67 -> ../../run/tr49t67
tr49t85 -> ../../run/tr49t85
tr67t85 -> ../../run/tr67t85
gribmap.txt -> ../../run/gribmap.txt
grib2map.tbl -> ../../run/grib2map.tbl
namelist.input          - require editing
real.exe -> ../../main/real.exe
wrf.exe -> ../../main/wrf.exe
ndown.exe -> ../../main/ndown.exe
```



…. (a few more)

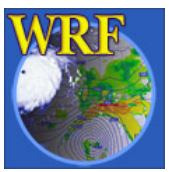
Running WRF executables - Preparation for *ideal* cases

Go to the desired ideal test case directory:

```
cd test/em_quarter_ss
```

If there is **'run_me_first.csh'** in the directory, run it first - this links physics data files to the currect directory:

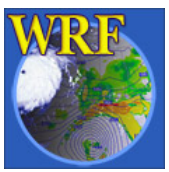
```
./run_me_first.csh
./ideal.exe
```



Running WRF executables - Preparation for *ideal* cases

- Edit **`namelist.input`** file to change options
- For your own case, you may provide a different sounding

- For 2D cases and baraclinic wave case, **`ideal.exe`** must be run serially
- For all 2D cases, **`wrf.exe`** must be run serially or with OpenMP

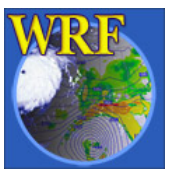


Running WRF executables - Preparation for *real data* cases

- One must successfully run the pre-processing system WPS, and create **`met_em.*`** file for more than one time period

- Link or copy WPS output files to the run directory:

```
cd test/em_real
ln -s ../../../WPS/met_em.* .
```



Running WRF executables - Preparation for *real data* cases

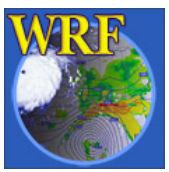
- Edit **`namelist.input`** file for runtime options **(`&time_control`** and **`&domains`,** at minimum)
- A more extensive list of namelist and their explanations can be found in

  `WRFV2/run/README.namelist`,

  and in the User's Guide (p5-22 – 5-35)
- Also see '*Namelist*' and '*One-way and Two-way Nesting*' lectures on Tuesday

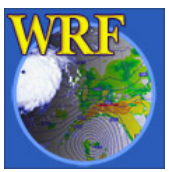
&time_control

```
run_days                            = 0,
 run_hours                          = 24,
 run_minutes                        = 0,
 run_seconds                        = 0,
 start_year                         = 2000, 2000, 2000,
 start_month                        = 11,   01,   01,
 start_day                          = 25,   24,   24,
 start_hour                         = 00,   12,   12,
 start_minute                       = 00,   00,   00,
 start_second                       = 00,   00,   00,
 end_year                           = 2000, 2000, 2000,
 end_month                          = 11,   01,   01,
 end_day                            = 26,   25,   25,
 end_hour                           = 00,   12,   12,
 end_minute                         = 00,   00,   00,
 end_second                         = 00,   00,   00,
 interval_seconds                   = 21600
 history_interval                   = 60,   60,   60,
```



&domains

```
time_step                           = 180
 time_step_fract_num                = 0,
 time_step_fract_den                = 1,
 max_dom                            = 1,
 s_we                               = 1,     1,     1,
 e_we                               = 74,    112,   94,
 s_sn                               = 1,     1,     1,
 e_sn                               = 61,    97,    91,
 s_vert                             = 1,     1,     1,
 e_vert                             = 27,    28,    28,
 num_metgrid_levels                 = 21
 dx                                 = 30000, 10000,  3333,
 dy                                 = 30000, 10000,  3333,
```



Running initialization executables

To run on single or OpenMP systems, type

`ideal.exe`

for a idealized case, and

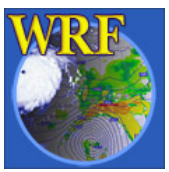
`real.exe`

for real-data cases.

To run on multi-processor system, typically type

`mpirun –np N real.exe`

where *N* is the number of processors



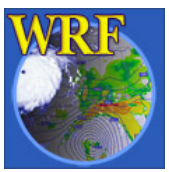
Running model executable

To run on single or OpenMP systems, type

`wrf.exe` or **`wrf.exe >& wrf.out`**

To run on multi-processor systems, type

`mpirun -np N wrf.exe`



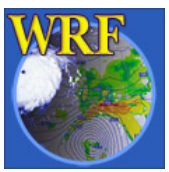
Output from a multi-processor run

The standard out and error will go to the following files for MPI runs:

`show_domain_0000:` domain-decomposition info from RSL

`rsl.out.0000`	**`rsl.error.0000`**
`rsl.out.0001`	**`rsl.error.0001`**
`rsl.out.0002`	**`rsl.error.0002`**
`rsl.out.0003`	**`rsl.error.0003`**

There is one pair of files for each processor requested



What is in a rsl file?

- A print of namelist options
- Time taken to compute one model step

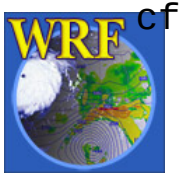
```
Timing for main: time 2000-01-24_12:03:00 on domain   1:    3.25000 elapsed seconds.
Timing for main: time 2000-01-24_12:06:00 on domain   1:    1.50000 elapsed seconds.
Timing for main: time 2000-01-24_12:09:00 on domain   1:    1.50000 elapsed seconds.
Timing for main: time 2000-01-24_12:12:00 on domain   1:    1.55000 elapsed seconds.
```

- Time taken to write history and restart file

```
Timing for Writing wrfout_d01_2000-01-24_18:00:00 for domain   1:   0.14000 elapsed
      seconds.
```

- Any model error prints:

```
5 points exceeded cfl=2 in domain 1 at time 4.200000 MAX AT i,j,k: 123 48 3
      cfl,w,d(eta)= 4.165821
```



WRF Files

Input to **`real.exe`**: multiple files from WPS

`met_em.d01.2000-01-24_12:00:00`

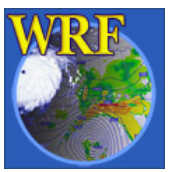
`met_em.d01.2000-01-24_18:00:00`

…

Output from **`real.exe`**:

`wrfinput_d01` ← single time level data

`wrfbdy_d01` ← BC data for multiple times



WRF Files

Output from wrf.exe (by default):

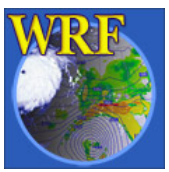
`wrfout_d01_ yyyy-mm-dd_hh:00:00`

➔ Usually contains multiple times

If restart file is written:

`wrfrst_d01_yyyy-mm-dd_hh:00:00`

➔ single time level date at the requested restart time



Check Output

Check run log file:

`wrf: SUCCESS COMPLETE WRF`

Use **`ncdump`**

`ncdump -v Times wrfout_d01_*`

to check output times. Or

`ncdump -v U wrfout_d01_*`

to check a particular variable (U)

Use **`read_wrf_nc.f`** (see lecture on “Tools”)