



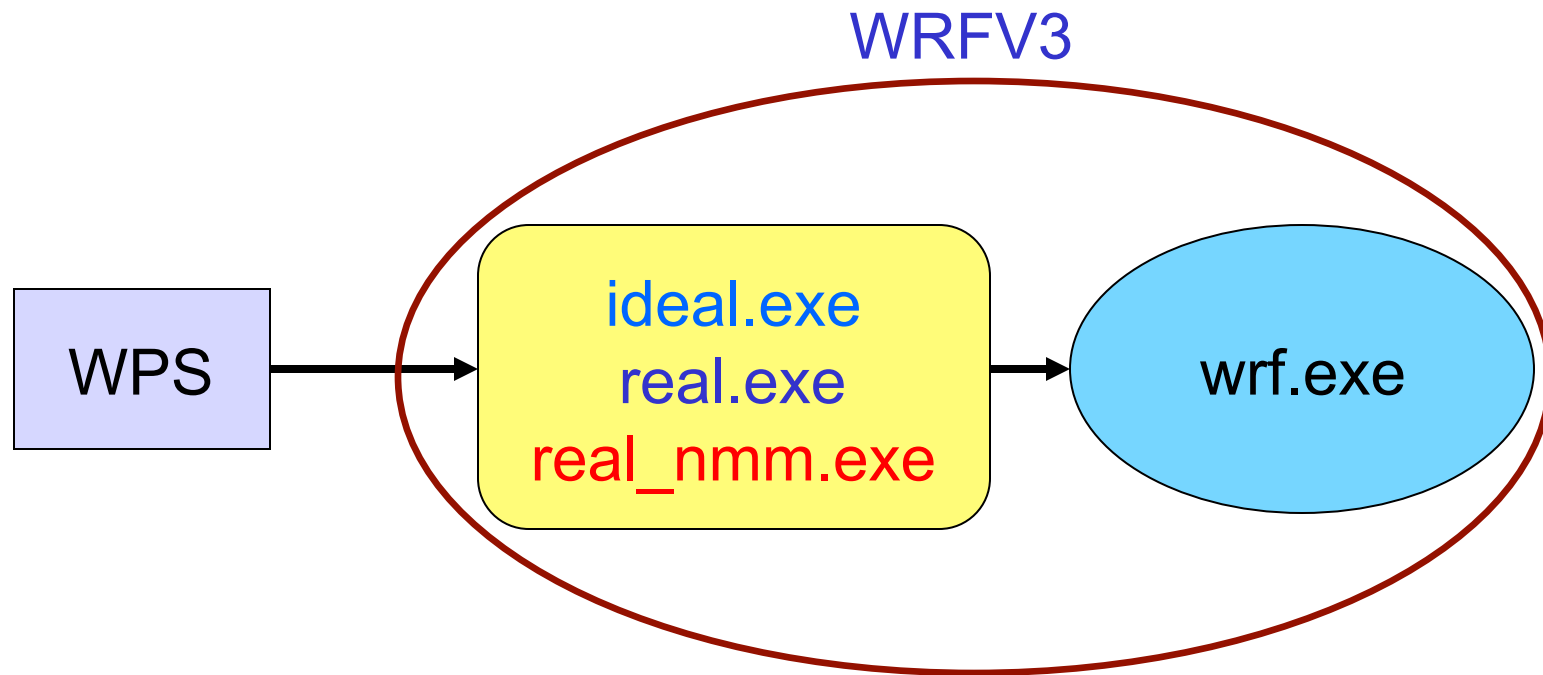
Set Up and Run WRF

(ARW-Ideal, ARW-real, and NMM)

Wei Wang
NCAR/NESL/MMM
Jan 2011



WRF System Flowchart



Outline

- Running WRF code
 - Before you run..
 - Running **idealized** case
 - Running **ARW real-data** case
 - Running **NMM real-data** case
- Basic runtime options for a single domain run (*namelist*)
- Check output
- Simple trouble shooting

This talk is complementary to 'Nesting' talk later.



Before You Run ..

- Check and make sure appropriate executables are created in **WRFV3/main/** directory:

For ARW:

- **ideal.exe**
- **real.exe**
- **wrf.exe**
- **ndown.exe**
- **tc.exe**

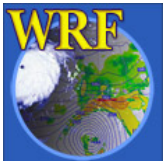
For NMM:

- **real_nmm.exe**
- **wrf.exe**

- If you are running a real-data case, be sure that files from WPS are correctly generated:

- **met_em.d01.***, for ARW or
- **met_nmm.d01.*** for NMM

- Prepare **namelist.input** for runtime options.



WRF test case directories

You have these choices in **WRFV3/test/**

(made at compile time):

nmm_real

em_real

em_quarter_ss

em_b_wave

em_les

em_heldsuarez

em_hill2d_x

em_squal12d_x

em_squal12d_y

em_grav2d_x

em_seabreeze2d_x

em_scm_xy

} 3d real-data

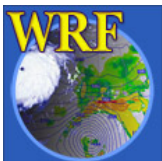
} 3d ideal

} 2d ideal

} 1d ideal

NMM

ARW



Steps to Run

1. cd to *run/* or one of the *test case* directories
2. Link or copy WPS output files to the directory for real-data cases
3. Edit *namelist.input* file for the appropriate grid and times of the case
4. Run initialization program (*ideal.exe*, *real.exe*, or *real_nmm.exe*)
5. Run model executable, *wrf.exe*



WRFV3/run directory

README.namelist

LANDUSE.TBL

GENPARM.TBL

SOILPARM.TBL

VEGPARM.TBL

URBAN_PARAM.TBL

RRTM_DATA

RRTMG_SW_DATA

RRTMG_LW_DATA

CAM_ABS_DATA

CAM_AEROPT_DATA

ozone.formatted

ozone_lat.formatted

ozone_plev.formatted

ETAMPNEW_DATA

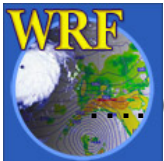
tr49t67

tr49t85

tr67t85

(a few more)

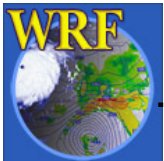
*these files are model
physics data files: they are
used to either initialize
physics variables, or make
physics computation more
efficient*



WRFV3/run directory after compile

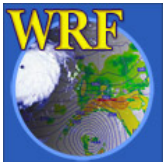
LANDUSE.TBL
ETAMPNEW_DATA
GENPARM.TBL
RRTM_DATA
RRTMG_SW_DATA
RRTMG_LW_DATA
SOILPARM.TBL
VEGPARM.TBL
URBAN_PARAM.TBL
tr49t67
tr49t85
tr67t85
gribmap.txt
grib2map.tbl
namelist.input -> ../test/em_real/*namelist.input*
real.exe -> ../main/real.exe
wrf.exe -> ../main/wrf.exe
ndown.exe -> ../main/ndown.exe
... (a few more)

*An example after
ARW real case
compile*



Running an Idealized Case

ARW only



Running an *Idealized* Case

- If you have compiled an ideal case, you should have:
`ideal.exe` - ideal case initialization program
`wrf.exe` - model executable
 - These executables are linked to:
`WRFV3/run`
and
`WRFV3/test/em_test-case`
- ➔ One can go to either directory to run.



Running an *Idealized* Case

Go to the desired *ideal* test case directory: e.g.

```
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 current directory:

```
./run_me_first.csh
```



Running an *Idealized* Case

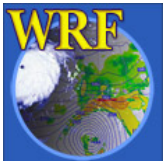
Then run the ideal initialization program:

`./ideal.exe`

The input to this program is typically a sounding file (file named *input_sounding*), or a pre-defined 3D input (e.g. *input_jet* in *em_b_wave* case).

Running *ideal.exe* creates WRF initial condition file:
wrfinput_d01

Note that wrfbdy file is not needed for idealized cases.



Running an *Idealized* Case

- To run the model interactively, type
`./wrf.exe >& wrf.out &`
for single processor (serial) or SMP run. Or
`mpirun -np N ./wrf.exe &`
for a MPI run (where *N* is the number of processors requested)
- Successful running of the model executable will create a model history file called `wrfout_d01_<date>`
e.g. `wrfout_d01_0001-01-01_00:00:00`

Based on start date defined in namelist

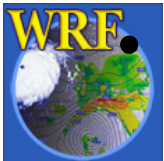


Running an *Idealized* Case

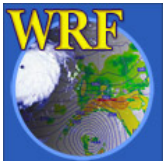
- Edit `namelist.input` file to change options.
- For your own case, you may provide a different sounding.
- You may also edit `dyn_em/module_initialize_<case>.F` to change other aspects of the initialization.

Note:

- For 2D cases and baroclinic wave case, `ideal.exe` must be run serially
- For all 2D cases, `wrf.exe` must be run serially or with SMP
- For the 1D case, compile and run serially



Running **ARW** Real-Data Case



Running **ARW** Real-Data Case

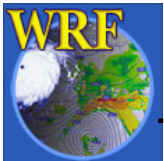
- If you have compiled the *em_real* case, you should have:
 - real.exe** - real data initialization program
 - wrf.exe** - model executable
 - ndown.exe** - program for doing one-way nesting
 - tc.exe** - program for TC bogusing
- These executables are linked to:
 - WRFV3/run**
 - and
 - WRFV3/test/*em_real***



➔ One can go to either directory to run.

WRFV3/test/em_real directory

```
LANDUSE.TBL -> ../../run/LANDUSE.TBL
ETAMPNEW_DATA -> ../../run/ETAMPNEW_DATA
GENPARM.TBL -> ../../run/GENPARM.TBL
RRTM_DATA -> ../../run/RRTM_DATA
RRTMG_SW_DATA -> ../../run/RRTMG_SW_DATA
RRTMG_LW_DATA -> ../../run/RRTMG_LW_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 *ARW* Real-data Cases

- One must successfully run WPS, and create *met_em.** file for more than one time period
- Link or copy WPS output files to the run directory:

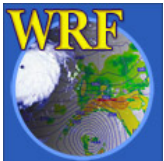
```
cd test/em_real
```

```
ln -s ../ ../WPS/met_em.d01.* .
```



Running WRF **ARW** Real-data Cases

- Edit **namelist.input** file for runtime options (at minimum, one must edit **&time_control** for start, end and integration times, and **&domains** for grid dimensions)
- Run the real-data initialization program:
 ./real.exe, if compiled serially / SMP, or
 mpirun -np N ./real.exe, or
 mpirun -machinefile file -np N ./real.exe
 for a MPI job
 where **N** is the number of processors requested, and
 file has a list of CPUs for the MPI job



Running WRF *ARW* Real-data Cases

- Successfully running this program will create model initial and boundary files:

`wrfinput_d01`

`wrfbdy_d01`

*Single time level
data at model's
start time*

*N-1 time-level data at
the lateral boundary,
and only for domain 1*

N: the number of time periods processed



Running WRF **ARW** Real-data Cases

- Run the model executable by typing:

```
./wrf.exe >& wrf.out &
```

or

```
mpirun -np N ./wrf.exe &
```

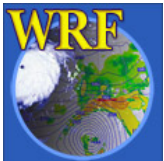
- Successfully running the model will create a model *history* file:

```
wrfout_d01_2005-08-28_00:00:00
```

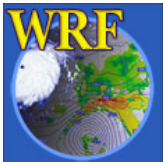
Based on start date defined in namelist

And *restart* file if **restart_interval** is set to a time within the range of the forecast time:

```
wrfirst_d01_2008-08-28_12:00:00
```

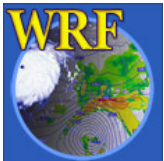


Running **NMM** Real-Data Case



Running **NMM** Real-Data Case

- If you have compiled the *nmm_real*, you should have:
real_nmm.exe - NMM real date initialization program
wrf.exe - NMM model executable
 - These executables are linked to:
WRFV3/run
and
WRFV3/test/*nmm_real*
- ➔ One can go to either directory to run.



WRFV3/test/*nmm_real* directory

LANDUSE.TBL -> ../../run/LANDUSE.TBL
ETAMPNEW_DATA -> ../../run/ETAMPNEW_DATA
GENPARM.TBL -> ../../run/GENPARM.TBL
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_nmm.exe -> ../../main/real_nmm.exe
wrf.exe -> ../../main/wrf.exe



Running **NMM** Real-data Cases

- One must successfully run WPS, and create **met_nmm.*** file for more than one time period
- Link or copy WPS output files to the run directory:

```
cd test/nmm_real
```

```
ln -s ../ ../WPS/met_nmm.d01.* .
```



Running **NMM** Real-data Cases

- Edit `namelist.input` file for runtime options (at minimum, one must edit `&time_control` for start, end and integration time, and `&domains` for grid dimensions)

- Run the real-data initialization program in MPI:

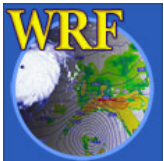
```
mpirun -np N ./real_nmm.exe
```

Or serially: `./real_nmm.exe >& real_nmm.out`

- Successfully running this program will create model initial and boundary files:

```
wrfinput_d01
```

```
wrfbdy_d01
```



Running **NMM** Real-data Cases

- Run the model executable by typing in MPI:

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

Or serially: **./wrf.exe >& wrf.out**

- Successfully running the model will create a model *history* file:

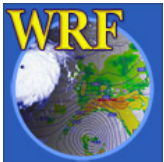
```
wrfout_d01_2005-08-28_00:00:00
```

And *restart* file if **restart_interval** is set to a time within the range of the forecast time:

```
wrfirst_d01_2008-08-28_12:00:00
```



Basic namelist Options



What is a namelist?

- A Fortran namelist contains a list of *runtime* options for the code to read in during its execution. Use of a namelist allows one to change runtime configuration without the need to recompile the source code.
- Fortran 90 namelist has very specific format, so edit with care:

```
&namelist-record - start  
/                      - end
```

- As a general rule:
Multiple columns: domain dependent
Single column: value valid for all domains



&time_control

```
run_days  
run_hours  
run_minutes  
run_seconds  
start_year  
start_month  
start_day  
start_hour  
start_minute  
start_second  
end_year  
end_month  
end_day  
end_hour  
end_minute  
end_second  
interval_seconds  
history_interval  
frame_per_outfile  
restart_interval  
restart
```

```
= 0,  
= 24,  
= 0,  
= 0,  
= 2000, 2000, 2000,  
= 01, 01, 01,  
= 24, 24, 24,  
= 12, 12, 12,  
= 00, 00, 00,  
= 00, 00, 00,  
= 2000, 2000, 2000,  
= 01, 01, 01,  
= 25, 25, 25,  
= 12, 12, 12,  
= 00, 00, 00,  
= 00, 00, 00,  
= 21600  
= 180, 60, 60  
= 1000, 1000, 1000,  
= 360,  
= .true.,
```

First column: domain 1 option

nest options



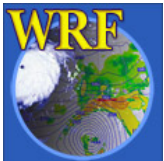
Notes on `&time_control`

- `run_*` time variables:
 - Model simulation length: `wrf.exe` and domain 1 only
- `start_*` and `end_*` time variables:
 - Program `real` will use WPS output between these times to produce lateral (and lower) boundary file
 - They can also be used to specify the start and end of simulation times for the coarse grid if `run_*` variables are not set (or set to 0).



Notes on `&time_control`

- *Interval_seconds*:
 - Time interval between WPS output times, and LBC update frequency
- *history_interval*:
 - Time interval in minutes when a history output is written
 - The time stamp in a history file name is the time when the history file is first written, and multiple time periods may be written in one file. e.g. a history file for domain 1 that is first written for 1200 UTC Jan 24 2000 is
`wrfout_d01_2000-01-24_12:00:00`



Notes on `&time_control`

- *frame_per_outfile*:
 - Number of history times written to one file.
- *restart_interval*:
 - Time interval in minutes when a restart file is written.
 - By default, restart file is not written at hour 0.
 - A restart file contains only one time level data, and its valid time is in its file name, e.g. a restart file for domain 1 that is valid for 0000 UTC Jan 25 2000 is
`wrfst_d01_2000-01-25_00:00:00`
- *restart*:
 - whether this is a restart run



&time_control

```
io_form_history      = 2,  
io_form_restart     = 2,  
io_form_input       = 2,  
io_form_boundary    = 2,  
debug_level        = 0,
```

IO format options:

- = 1, binary
- = 2, **netcdf** (most common)
- = 4, PHDF5
- = 5, Grib 1
- = 10, Grib 2
- = 11, pNetCDF

Useful alternative:

`io_form_restart = 102 :`
write output in patch
sizes: fast for large grids
and useful for restart file

Debug print control:
Increasing values give
more prints.

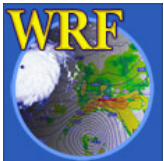


&domains

```
time_step           = 180
time_step_fract_num = 0,
time_step_fract_den = 1,
max_dom             = 1,
e_we                = 74,
e_sn                = 61,
e_vert              = 28,
num_metgrid_levels  = 21
num_metgrid_soil_levels = 4
dx                  = 30000, 10000, 3333,
dy                  = 30000, 10000, 3333,
eta_levels           = 1.0, 0.996, 0.99, 0.98, ... 0.0
p_top_requested      = 5000,
```

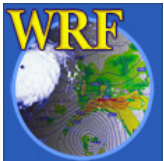
*Match the dimensions
defined in WPS*

nest
options



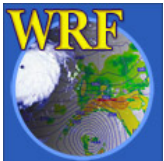
Notes on **&domains**

- *time_step, time_step_fract_num, time_step_fract_den*:
 - Time step for model integration in seconds.
 - Fractional time step specified in separate integers of numerator and denominator.
 - ARW: 6xDX; NMM: 2.25xDX (DX is grid distance in km)
 - May be divided by output intervals
- *e_we, e_sn, e_vert*:
 - Model grid dimensions (staggered) in X, Y and Z directions.
- *num_metgrid_levels*:
 - Number of *metgrid* (input) data levels.
- *num_metgrid_soil_levels*:
 - Number of soil data levels in the input data
- *dx, dy*:
 - grid distances: in meters for ARW; in degrees for NMM.



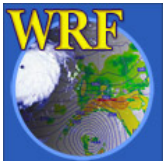
Notes on `&domains`

- *`p_top_requested`*:
 - Pressure value at the model top.
 - Constrained by the available data from WPS.
 - Default is 5000 Pa.
- *`eta_levels`*:
 - Specify your own model levels from 1.0 to 0.0.
 - If not specified, program *`real`* will calculate a set of levels.
- *`ptsgm` (NMM only)*:
 - Pressure level (Pa) at which the WRF-NMM hybrid coordinate transitions from sigma to pressure (default: 42000 Pa).



Where do I start?

- Always start with a *namelist* template provided in a test case directory, whether it is an ideal case, ARW or NMM real data case.
 - A number of namelist templates are provided in *test/test_<case>/* directories
 - For example: in *test/em_real/*, there are
 - namelist.input.4km ~ 4 km grid size
 - namelist.input.jun01 ~ 10 km grid size
 - namelist.input.jan00 ~ 30 km grid size
 - Examples of namelists for various applications, Chapter 5 of the User's Guide



Where do I start?

- Use document to guide the modification of the namelist values:
 - `run/README.namelist`
 - `test/em_real/examples.namelist`
 - User's Guide, Chapter 5 (online version has the latest)
 - Full list of namelists and their default values can be found in Registry files: [Registry.EM](#) (ARW), [Registry.NMM](#) and `registry.io_boilerplate` (IO options, shared)



To run a job in a different directory..

- Directories *run/* and *test_<case>/* are convenient places to run, but it does not have to be.
- Copy or link the content of these directories to another directory, including physics data files, wrf input and boundary files, wrf **namelist** and **executables**, and you should be able to run a job anywhere on your system.

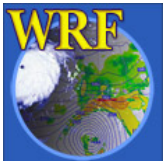


Check Output



Output After a Model Run

- Standard out/error files:
`wrf.out`, or `rs1.*` files
- Model history file(s):
`wrfout_d01_<date>`
- Model restart file(s), optional
`wrfirst_d01_<date>`



Output from a multi-processor run

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

```
mpirun -np 4 ./wrf.exe ➔
```

```
rs1.out.0000
```

```
rs1.error.0000
```

```
rs1.out.0001
```

```
rs1.error.0001
```

```
rs1.out.0002
```

```
rs1.error.0002
```

```
rs1.out.0003
```

```
rs1.error.0003
```

There is one pair of files for each processor requested



What to Look for in a standard out File?

Check run log file by typing

```
tail wrf.out, or  
tail rsl.out.0000
```

You should see the following if the job is successfully completed:

```
wrf: SUCCESS COMPLETE WRF
```



How to Check Model History File?

- Use **ncdump**:

`ncdump -v Times wrfout_d01_<date>`

to check output times. Or

`ncdump -v U wrfout_d01_<date>`

to check a particular variable (U)

- Use **ncview** or **ncBrowse** (great tools!)
- Use post-processing tools (see talks later)



What is in a *wrf.out* or *rsl* file?

- 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: (example from ARW run)

```
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
```

➔ An indication the model has become numerically unstable



Simple Trouble Shooting



Often-seen runtime problems

- `module_configure: initial_config: error reading namelist: &dynamics`
 - Typos or erroneous namelist variables exist in namelist record `&dynamics` in *namelist.input* file
- `input_wrf.F: SIZE MISMATCH: namelist ide,jde,num_metgrid_levels= 70 61 27 ; input data ide,jde,num_metgrid_levels= 74 61 27`
 - Grid dimensions in error
- More are added since V3.2 to guide one through namelist setup errors.



Often-seen runtime problems

- **Segmentation fault** (core dumped)
 - Often typing `'unlimit'` or `'ulimit -s unlimited'` or equivalent can help when this happens quickly in a run.
- If you do: `grep cfl rsl.error.*` and see
121 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
 - Model becomes unstable due to various reasons. If it happens soon after the start time, check input data, and/or reduce time step.



References

- Information on compiling and running WRF, and a more extensive list of namelist options and their definition / explanations can be found in the ARW and NMM [User's Guide, Chapter 5](#)
- Also see '[Nesting Setup and Run](#)' and '[Other Runtime Options](#)' talks.

