Set Up and Run WRF

(Ideal and real data)

Wei Wang
NCAR/MMM
July 2015
WRF System Flowchart

WPS -> ideal.exe -> real.exe -> wrf.exe

WRFV3

Programs to create IC/BC

Program to integrate model
Outline

• Running WRF code
  – Things to check before you run..
  – Running real-data case
  – Running idealized case
• Basic runtime options for a single domain run (namelist)
• Check output
• Simple trouble shooting
• Running a nested case: later
Before You Run ..

- Make sure appropriate executables are created in WRFV3/main/ directory:
  - ideal.exe – executable to create idealized IC
  - real.exe – executable to create IC/BC
  - wrf.exe – executable for model integration
  - ndown.exe – utility
  - tc.exe – utility routine for TC bogusing

- If you are working with real data, be sure that files for a few time periods from WPS are correctly generated:
  - met_em.d01.*
WRF test case directories

You have these choices in `WRFV3/test/` (made at compile time):

```
em_real  }  3-dimensional real-data – real.exe
em_quarter_ss
em_b_wave
em_les
em_tropical_cyclone
em_heldsuarez
em_hill2d_x
em_squall2d_x
em_squall2d_y
em_grav2d_x
em_seabreeze2d_x
em_scm_xy
```

```
em_b_wave
em_les
em_tropical_cyclone
em_heldsuarez
em_hill2d_x
em_squall2d_x
em_squall2d_y
em_grav2d_x
em_seabreeze2d_x
em_scm_xy
```

3d ideal

```
em_b_wave
em_les
em_tropical_cyclone
em_heldsuarez
em_hill2d_x
em_squall2d_x
em_squall2d_y
em_grav2d_x
em_seabreeze2d_x
em_scm_xy
```

ideal.exe

2d ideal

```
em_b_wave
em_les
em_tropical_cyclone
em_heldsuarez
em_hill2d_x
em_squall2d_x
em_squall2d_y
em_grav2d_x
em_seabreeze2d_x
em_scm_xy
```

1d ideal
Steps to Run

1. cd to \textit{run/} or one of the \textit{test case} directories
2. Move or link WPS output files to the directory for real-data cases
3. Edit \textit{namelist.input} file for the appropriate grid and times of the case
4. Run a initialization program (\textit{ideal.exe} or \textit{real.exe})
5. Run model executable, \textit{wrf.exe}
WRFV3/run directory

- README.namelist
- LANDUSE.TBL
- GENPARM.TBL
- SOILPARM.TBL
- VEGPARM.TBL
- URBPARM.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
- gribmap.txt
- grib2map.tbl

These are model physics data files: they are used to either initialize physics variables, or make physics computation faster.

- for grib IO

(description of namelists)
An example after `em_real` case compile

- namelist.input - copied from `../test/em_real/namelist.input`
- real.exe -> `../main/real.exe`
- wrf.exe -> `../main/wrf.exe`
- ndown.exe -> `../main/ndown.exe`
- ... (a few more)
Running a Real-Data Case
Running a 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`
  - `WRFV3/test/em_real`

⇒ One can go to either directory to run.
WRFV3/test/em_real directory

LANDUSE.TBL -> ../../../run/LANDUSE.TBL
GENPARM.TBL -> ../../../run/GENPARM.TBL
SOILPARM.TBL -> ../../../run/SOILPARM.TBL
VEGPARM.TBL -> ../../../run/VEGPARM.TBL
URBPARM.TBL -> ../../../run/URBPARM.TBL
RRTM_DATA -> ../../../run/RRTM_DATA
RRTMG_SW_DATA -> ../../../run/RRTMG_SW_DATA
RRTMG_LW_DATA -> ../../../run/RRTMG_LW_DATA
ETAMPNEW_DATA -> ../../../run/ETAMPNEW_DATA
tr49t67 -> ../../../run/tr49t67
tr49t85 -> ../../../run/tr49t85
tr67t85 -> ../../../run/tr67t85
...
	namelist.input - editing required
real.exe -> ../../../main/real.exe
wrf.exe -> ../../../main/wrf.exe
ndown.exe -> ../../../main/ndown.exe
.... (a few more)
Running a Real-data Case

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

- Move or link WPS output files to the run directory:
  
  ```
  cd test/em_real
  ln -s ../../../WPS/met_em.d01.* .
  ```
Running a Real-data Case

• 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 a MPI job
Running a Real-data Case

• 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 for lateral boundaries, and only for domain 1

  N: the number of time periods processed

  ncdump -v Times wrfbdy_d01
Running a Real-data Case

• Typing ‘ncdump -v Times wrfbdy_d01’ will give you, for a 24 hour period, 6 hourly data interval:
  .. a bunch of prints and at the end:

  data:

  Times =
  "2005-08-28_00:00:00",
  "2005-08-28_06:00:00",
  "2005-08-28_12:00:00",
  "2005-08-28_18:00:00" ;
Running a Real-data Case

- 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 set in namelist
- and a restart file if restart interval is set to a time within the range of the forecast time:
  ```
  wrfrst_d01_2005-08-28_12:00:00
  ```
  Exact time at a restart
Running a Real Data Case

\[ \text{wrfout\_d01\_2005-08-28\_00:00:00} \]

Based on start date set in namelist

\[
\begin{align*}
\text{start\_year} &= 2008, 2008, 2008, \\
\text{start\_month} &= 08, 08, 08, \\
\text{start\_day} &= 28, 28, 28, \\
\text{start\_hour} &= 00, 00, 00, \\
\text{start\_minute} &= 00, 00, 00, \\
\text{start\_second} &= 00, 00, 00, \\
\text{end\_year} &= 2008, 2008, 2008, \\
\text{end\_month} &= 08, 08, 08, \\
\text{end\_day} &= 29, 29, 29, \\
\text{end\_hour} &= 00, 00, 00, \\
\text{end\_minute} &= 00, 00, 00, \\
\text{end\_second} &= 00, 00, 00,
\end{align*}
\]
Running an Idealized Case
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`
  - `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.
\[
\text{cd test/em\_quarter\_ss}
\]

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 2D input (e.g. `input_jet` in `em_b_wave` case).

Running `ideal.exe only` creates WRF initial condition file: `wrfinput_d01`
Running an *Idealized* Case

Note that wrfbdy file is not needed for idealized cases.

Instead, the boundary conditions are set in the `namelist.input` file. For example, these are for options in east-west, or x direction:

```plaintext
periodic_x    = .false., .false., .false.,
symmetric_xs  = .false., .false., .false.,
symmetric_xe  = .false., .false., .false.,
open_xs       = .true., .false., .false.,
open_xe       = .true., .false., .false.,
```
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 set in namelist*
Running an *Idealized* Case

\[ \text{wrfout}_d01_{-0001-01-01_00:00:00} \]

Based on start date set in namelist

\[
\begin{align*}
\text{start\_year} &= 0001, 0001, 0001, \\
\text{start\_month} &= 01, 01, 01, \\
\text{start\_day} &= 01, 01, 01, \\
\text{start\_hour} &= 00, 00, 00, \\
\text{start\_minute} &= 00, 00, 00, \\
\text{start\_second} &= 00, 00, 00, \\
\text{end\_year} &= 0001, 0001, 0001, \\
\text{end\_month} &= 01, 01, 01, \\
\text{end\_day} &= 01, 01, 01, \\
\text{end\_hour} &= 00, 00, 00, \\
\text{end\_minute} &= 120, 120, 120, \\
\text{end\_second} &= 00, 00, 00,
\end{align*}
\]
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. (*more in talk on Thur.*)

**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
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

A namelist file may contain a number of records.
namelist record **&time_control**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>run_days</td>
<td>0</td>
</tr>
<tr>
<td>run_hours</td>
<td>24</td>
</tr>
<tr>
<td>run_minutes</td>
<td>0</td>
</tr>
<tr>
<td>run_seconds</td>
<td>0</td>
</tr>
<tr>
<td>start_year</td>
<td>2000</td>
</tr>
<tr>
<td>start_month</td>
<td>01</td>
</tr>
<tr>
<td>start_day</td>
<td>24</td>
</tr>
<tr>
<td>start_hour</td>
<td>12</td>
</tr>
<tr>
<td>start_minute</td>
<td>00</td>
</tr>
<tr>
<td>start_second</td>
<td>00</td>
</tr>
<tr>
<td>end_year</td>
<td>2000</td>
</tr>
<tr>
<td>end_month</td>
<td>01</td>
</tr>
<tr>
<td>end_day</td>
<td>25</td>
</tr>
<tr>
<td>end_hour</td>
<td>12</td>
</tr>
<tr>
<td>end_minute</td>
<td>00</td>
</tr>
<tr>
<td>end_second</td>
<td>00</td>
</tr>
<tr>
<td>interval_seconds</td>
<td>21600</td>
</tr>
<tr>
<td>history_interval</td>
<td>180</td>
</tr>
<tr>
<td>frames_per_outfile</td>
<td>1000, 1000, 1000,</td>
</tr>
<tr>
<td>restart_interval</td>
<td>360</td>
</tr>
<tr>
<td>restart</td>
<td>.true.</td>
</tr>
</tbody>
</table>

**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 lateral BC (and lower BC) 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 \texttt{\&time\_control}

- \textit{frames\_per\_outfile}:
  - Number of history times written to one file.

- \textit{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 \texttt{valid time} is in its file name, e.g. a restart file for domain 1 valid for 0000 UTC Jan 25 2000 is \texttt{wrfrst\_d01\_2000-01-25\_00:00:00}

- \textit{restart}:
  - whether this is a restart run
Notes on \texttt{&time\_control}

Example 1: all output times are in a single file

\begin{verbatim}
history_interval  = 180, 60, 60,
frames_per_outfile = 1000, 1000, 1000,
\texttt{wrfout\_d01\_2000-01-24\_12:00:00}
\end{verbatim}

Example 2: each output file only contains a single time

\begin{verbatim}
history_interval  = 180, 60, 60,
frames_per_outfile = 1, 1, 1,
\texttt{wrfout\_d01\_2000-01-24\_12:00:00}
\texttt{wrfout\_d01\_2000-01-24\_15:00:00}
\texttt{wrfout\_d01\_2000-01-24\_18:00:00}
\end{verbatim}
Notes on \textit{restart}

• What is a \textit{restart} run?
  – A restart run is a continuation of a model run.

• How to do a \textit{restart} run:
  – In the first run, set \textit{restart\_interval} to a value that is within the model integration time.
  – A restart file will be created. e.g. \texttt{wrfrst\_d01\_2000-01-25\_00:00:00}

• When doing a restart run:
  – Set \textit{restart} = .true.,
  – Set start time to restart time in namelist
&time_control

io_form_history = 2,
io_form_restart = 2,
io_form_input = 2,
io_form_boundary = 2,

IO format options:
= 1, binary
= 2, netcdf (most common)
= 4, PHDF5
= 5, Grib 1
=10, Grib 2
=11, pnetCDF

For large files:
io_form_restart = 102:
write output in patch sizes: fast for large grid and useful for restart file
namelist record &domains

time_step = 180

time_step_fract_num = 0,
time_step_fract_den = 1,
max_dom = 1,
e_we = 74, 112, 94,
e_sn = 61, 97, 91,
e_vert = 28, 28, 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,
Notes on &domains

- **time_step, time_step_fраст_num, time_step_fраст_den**:  
  - Time step for model integration in seconds.  
  - Fractional time step specified in separate integers of numerator and denominator.  
  - Typically 5 to 6xDX (DX is grid distance in km)

- **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  
  Found by typing `ncdump -h met_em.d01.<date> | more`

- **dx, dy**:  
  - grid distance: in meters
Notes on &domains

• **p_top_requested**:  
  – Pressure value at the model top.  
  – Constrained by the available data from WPS.  
  – Default is 5000 Pa (recommended as lowest $P_{top}$)

• **eta_levels**:  
  – Specify your own model levels from 1.0 to 0.0.  
  – If not specified, program *real* will calculate a set of levels  
  – Use a minimum of 30 levels with 5000 Pa model top to limit vertical grid distance < 1 km. Use more vertical levels when decreasing horizontal grid sizes.
namelist record &bdy_control

* typical *

- spec_bdy_width  = 5, (10)
- spec_zone  = 1, (1)
- relax_zone  = 4, (9)
- specified  = .true., .false., .false.,
- nested  = .false., .true., .true.,

* optional *

May change relax_zone and spec_bdy_width

(spec_zone + relax_zone = spec_bdy_width)

* Wider boundary zone may work better for coarser driving data
Where do I start?

- Always start with a namelist template provided in a test case directory, whether it is an ideal case, or a 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
Where do I start?

• For different applications, please refer to p5-25 to 5-27 of the ARW User’s Guide:
  – 2 or 4 km convection-permitting runs
  – 20 – 30 km, 2 – 3 day runs
  – Antarctic region
  – Tropical storm forecasting
  – Regional climate
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_COMMON` and `registry.io_boilerplate` (for IO options) (look for character string ‘`namelist`’)

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 `rsl.*` files
- Model history file(s):
  `wrfout_d01_<date>`
- Model restart file(s), optional
  `wrfrst_d01_<date>`
Output from a multi-processor run

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

```bash
mpirun -np 4 ./wrf.exe
```

- `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 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` (great tool!)

• Use post-processing tools (see talks later)
What is in a *wrf.out* or *rsl* file?

- **Model version, decomposition info:**
  
  \[
  \begin{align*}
  \text{Ntasks in X} & : 2, \\
  \text{Ntasks in Y} & : 4
  \end{align*}
  \]

- **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 \( cf1=2 \) in domain 1 at time 4.200000 MAX AT \( i,j,k: 123 \ 48 \ 3 \)
  \( cf1,w,d(\text{eta})= 4.165821 \)

  → 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
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, and on a small computer

- 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 User’s Guide, Chapter 5
- Also see ‘Nesting Setup and Run’ and ‘Other Runtime Options’ talks.