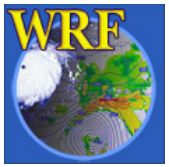
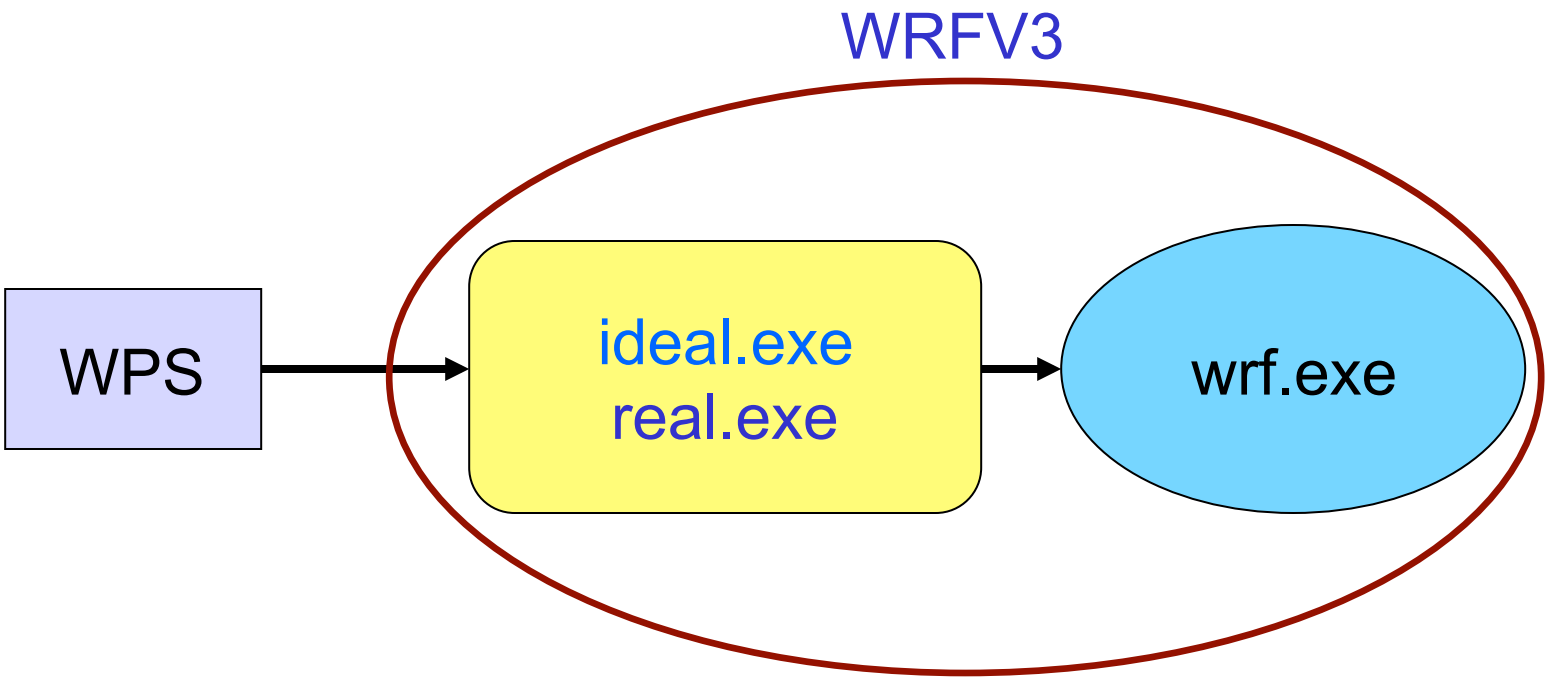
Set Up and Run WRF

(real data)

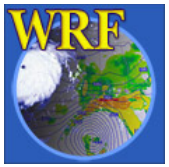


WRF System Flowchart



Outline

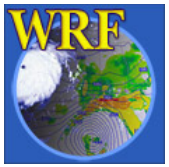
- Running WRF code
  - Before you run..
  - Running idealized case
  - Running ARW real-data case
- Basic runtime options for a ***single*** domain run (*namelist*)
- Check output
- Simple trouble shooting
- Running a nested case: later



Before You Run ..

- Check and make sure appropriate executables are created in **`WRFV3/main/`** directory:
  - **`ideal.exe`**
  - **`real.exe`**
  - **`wrf.exe`**
  - **`ndown.exe`**
  - **`tc.exe`**

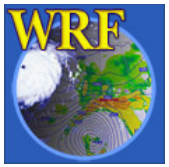
- If you are running a real-data case, be sure that files for ***a few time periods*** from WPS are correctly generated:
  - **`met_em.d01.*`**



WRF test case directories

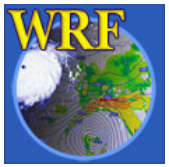
You have these choices in **`WRFV3/test/`**
(made at compile time):

`em_real`	3d real-data
`em_quarter_ss`	3d ideal
`em_b_wave`	
`em_les`	
`em_tropical_cyclone`	
`em_heldsuarez`	
`em_hill2d_x`	2d ideal
`em_squall2d_x`	
`em_squall2d_y`	
`em_grav2d_x`	
`em_seabreeze2d_x`	
`em_scm_xy`	1d ideal



Steps to Run

1. cd to *run/* or one of the *test case* directories
2. Move or link 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* and *real.exe*)
5. Run model executable, *wrf.exe*



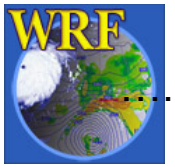
WRFV3/run directory

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
... (a few more)

description of namelists

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

for grib IO

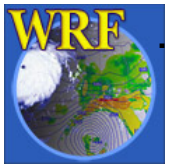


WRFV3/run directory after compile

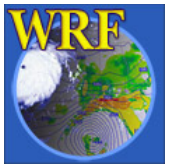
```
LANDUSE.TBL
SOILPARM.TBL
VEGPARM.TBL
GENPARM.TBL
URBPARM.TBL
RRTM_DATA
RRTMG_SW_DATA
RRTMG_LW_DATA
ETAMPNEW_DATA
tr49t67
tr49t85
tr67t85
…
namelist.input   -> ../test/em_real/namelist.input
real.exe -> ../main/real.exe
wrf.exe -> ../main/wrf.exe
ndown.exe -> ../main/ndown.exe
```

… (a few more)

An example after em_real case compile



Running 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

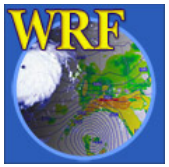
  tc.exe - program for TC bogusing

- These executables are linked to:

  WRFV3/run

  and

  WRFV3/test/*em_real*

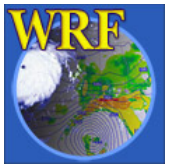


➔ One can go to either directory to run.

WRFV3/test/em_*real* directory

```
LANDUSE.TBL -> ../../run/LANDUSE.TBL
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)

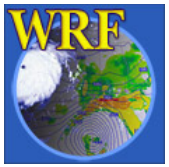


Running WRF ARW Real-data Cases

- 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 WRF ARW Real-data Cases

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

- Run the real-data initialization program:

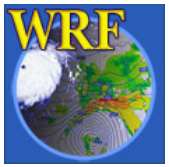
  `./real.exe`, if compiled serially / SMP, or

  `mpirun -np N ./real.exe`, or

  `mpirun -machinefile file -np N ./real.exe`

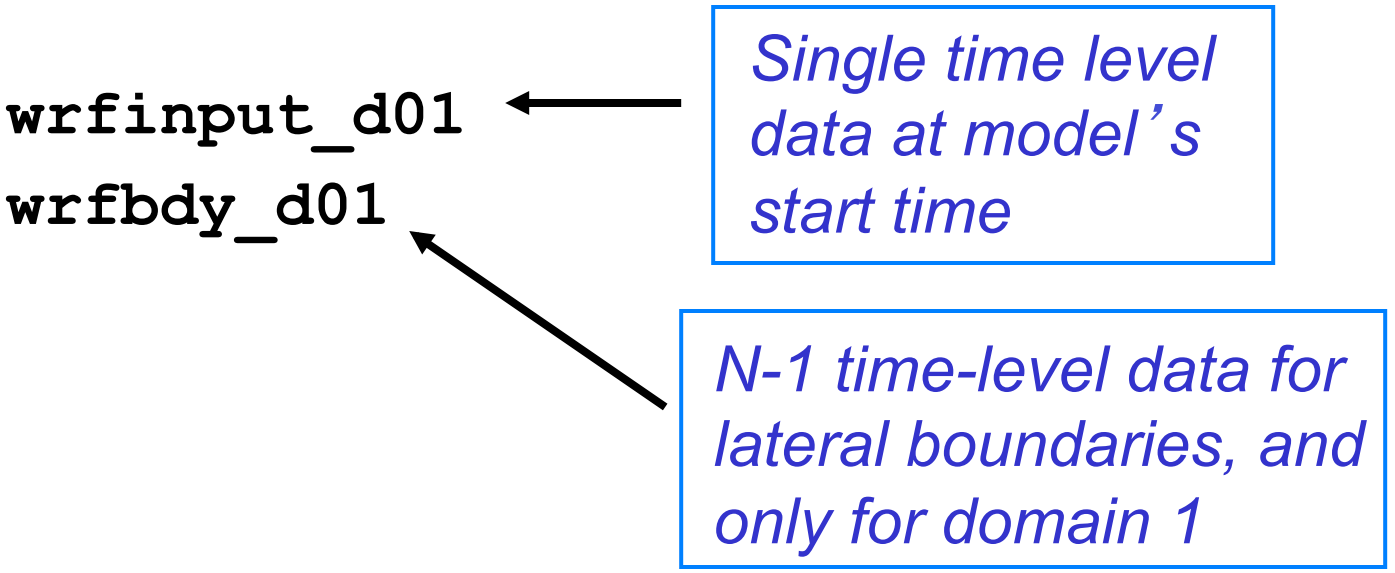
  for a MPI job

  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:



N: the number of time periods processed

```
ncdump -v Times wrfbdy_d01
```

Running WRF ARW Real-data Cases

- Run the model executable by typing:

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

  or

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

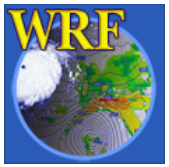
- Successfully running the model will a create model *history* file:

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

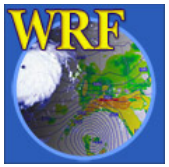
  Based on start date defined in namelist

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

  `wrfrst_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:

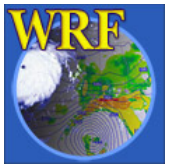
  `&namelist-record` - start

  `/` - end

- As a general rule:

  Multiple columns: domain dependent

  Single column: value valid for all domains

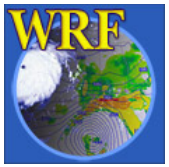


&time_control

```
run_days                            = 0,
run_hours                           = 24,
run_minutes                         = 0,
run_seconds                         = 0,
start_year                          = 2000, 2000, 2000,
start_month                         = 01,   01,   01,
start_day                           = 24,   24,   24,
start_hour                          = 12,   12,   12,
start_minute                        = 00,   00,   00,
start_second                        = 00,   00,   00,
end_year                            = 2000, 2000, 2000,
end_month                           = 01,   01,   01,
end_day                             = 25,   25,   25,
end_hour                            = 12,   12,   12,
end_minute                          = 00,   00,   00,
end_second                          = 00,   00,   00,
interval_seconds                    = 21600
history_interval                    = 180,  60,   60,
frame_per_outfile                   = 1000, 1000, 1000,
restart_interval                    = 360,
restart                             = .true.,
```

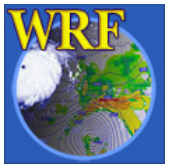
domain 1 option

nest options



Notes on `&time_control`

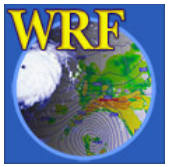
- *run_** time variables:
  - Model simulation length: *wrf.exe* and domain 1 only
- *start_** and *end_** time variables:
  - Program *real* will use WPS output between these times to produce lateral (and lower) boundary file
  - They can also be used to specify the start and end of simulation times 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

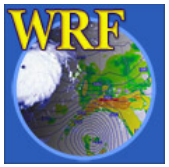
    `wrfout_d01_2000-01-24_12:00:00`



Notes on `&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

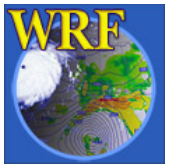
    `wrfrst_d01_2000-01-25_00:00:00`
- *restart:*
  - whether this is a restart run



Notes on *restart*

- What is a *restart* run?
  - A restart run is a continuation of a model run.
- How to do a *restart* run:
  - In the first run, set *restart_interval* to a value that is within the model integration time.
  - A restart file will be created. e.g.

    `wrfrst_d01_2000-01-25_00:00:00`
- When doing a restart run:
  - Set *restart* = .true.,
  - Set start times to restart times in namelist



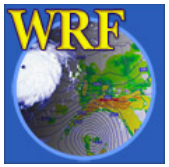
&time_control

```
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

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

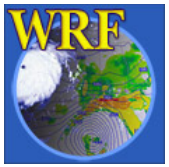
Debug print control: Increasing values give more prints.



&domains

```
time_step                  = 180
time_step_fract_num        = 0,
time_step_fract_den        = 1,
max_dom                    = 1,
e_we                       = 74,    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,
```

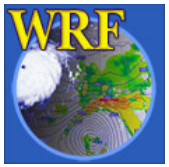
nest options



Notes on **&domains**

- *time_step, time_step_fract_num, time_step_frac_den*:
  - Time step for model integration in seconds.
  - Fractional time step specified in separate integers of numerator and denominator.
  - 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

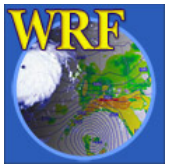
  Found by typing **`ncdump -h met_*.d01.<date> | more`**
- *dx, dy*:
  - grid distances: in meters



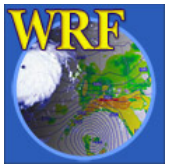
Notes on `&domains`

- *p_top_requested*:
  - Pressure value at the model top.
  - Constrained by the available data from WPS.
  - Default is 5000 Pa (recommended as lowest Ptop)

- *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.



Check Output



Output After a Model Run

- Standard out/error files:

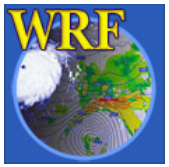
  `wrf.out`, or **`rsl`**.* files

- Model history file(s):

  `wrfout_d01_<date>`

- Model restart file(s), optional

  `wrfrst_d01_<date>`



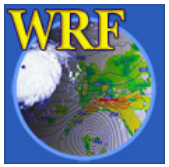
Output from a multi-processor run

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

```
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?

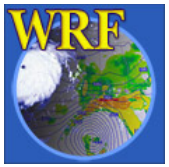
Check run log file by typing

`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**:

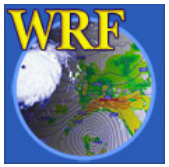
  ncdump -v Times wrfout_d01_<date>

  to check output times. Or

  ncdump -v U wrfout_d01_<date>

  to check a particular variable (U)

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



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

- Model version, decomposition info
- 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.
```

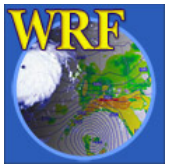
- Time taken to write history and restart file:

```
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



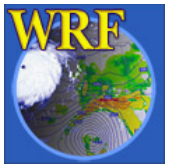
Often-seen runtime problems

- `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.