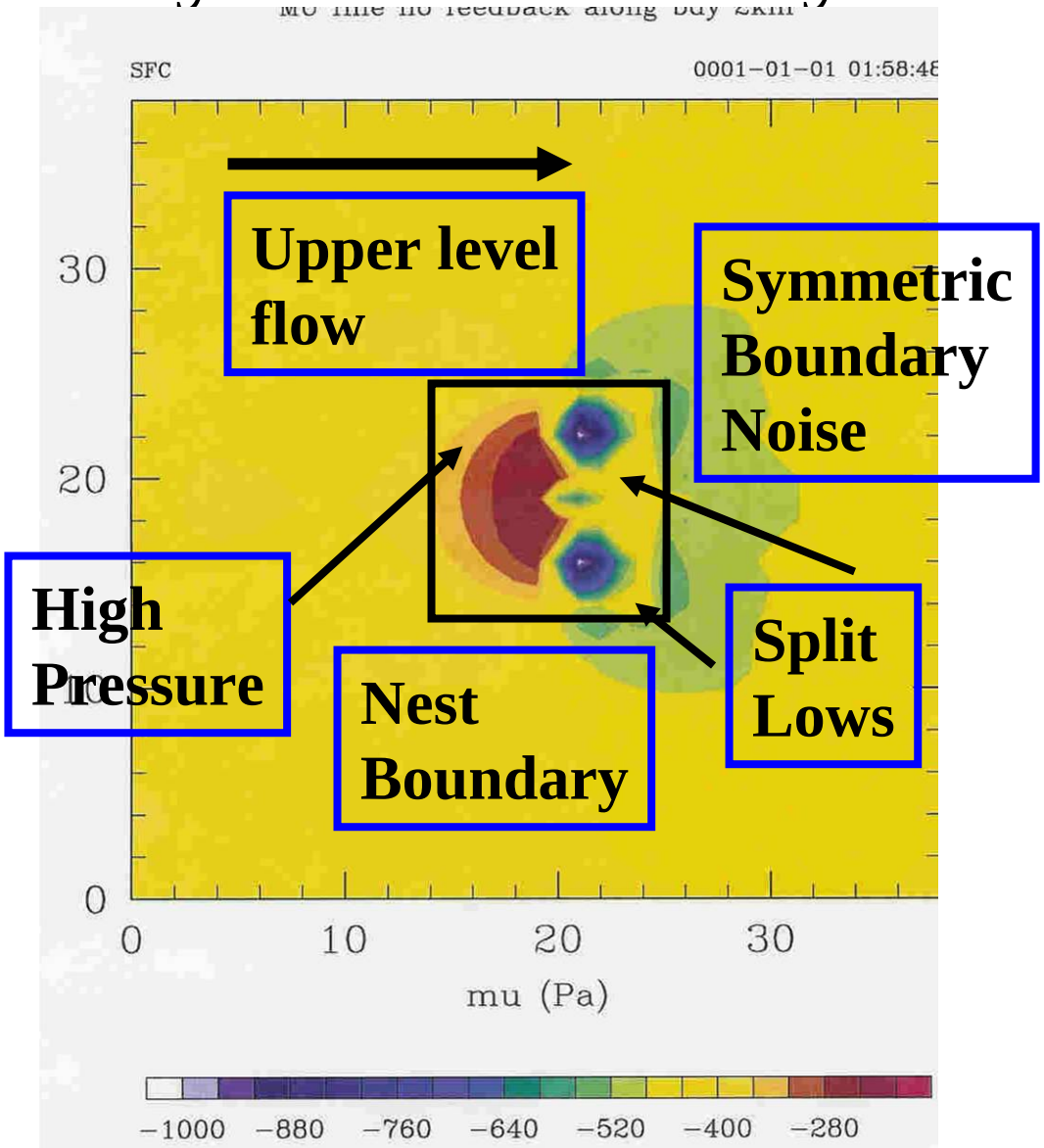
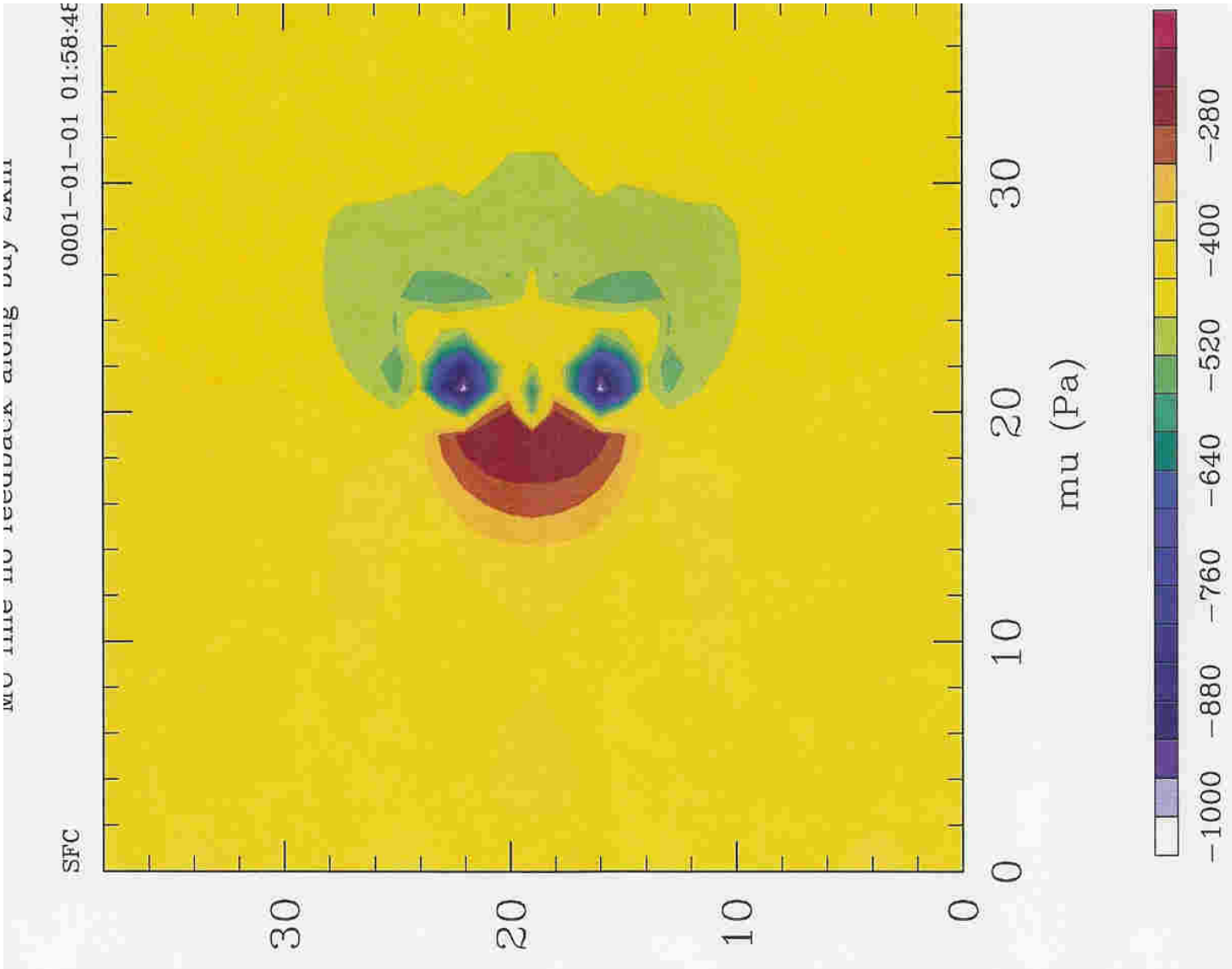
WRF ARW Nesting
One-Way and Two-Way

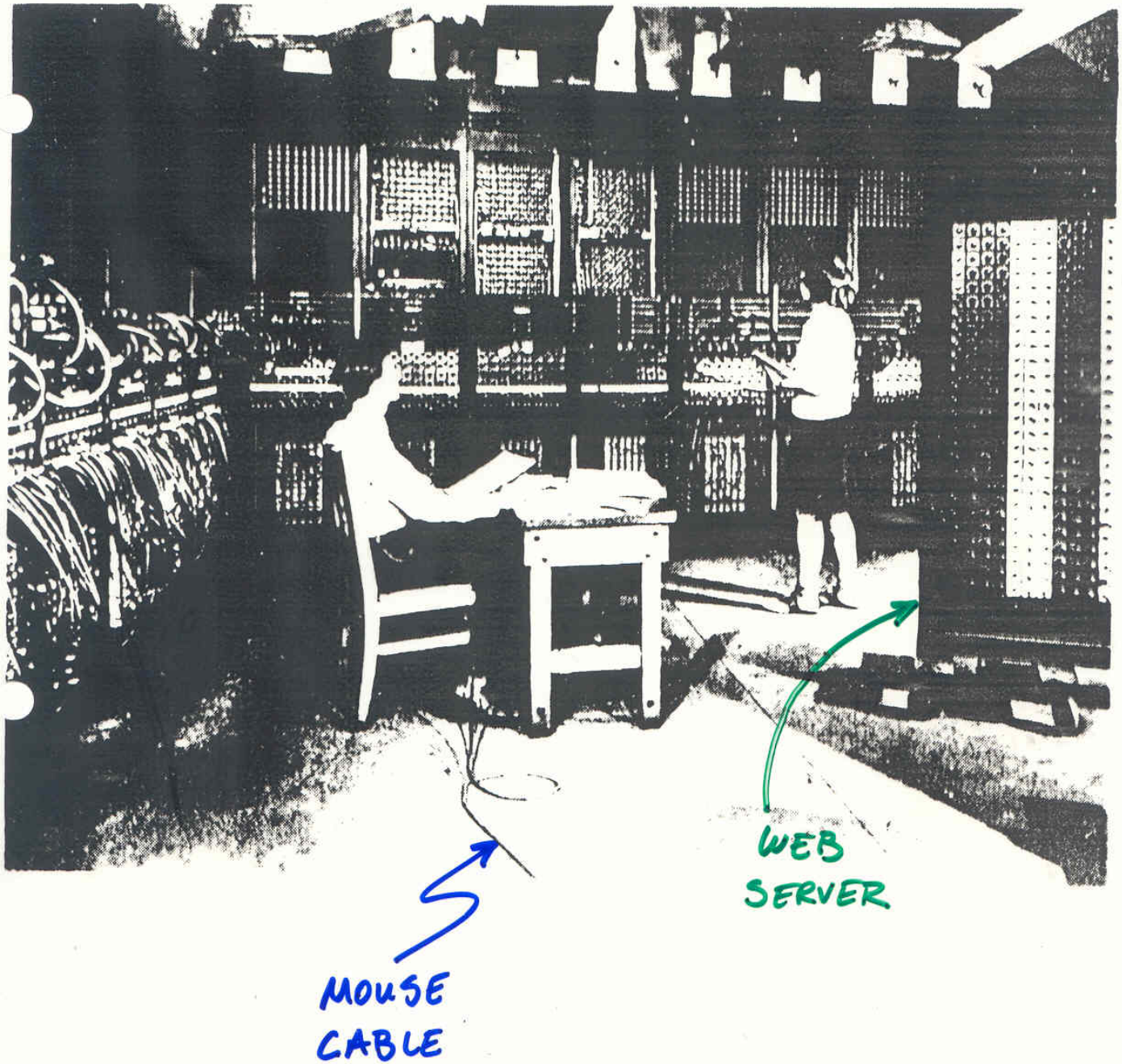
Dave Gill
gill@ucar.edu

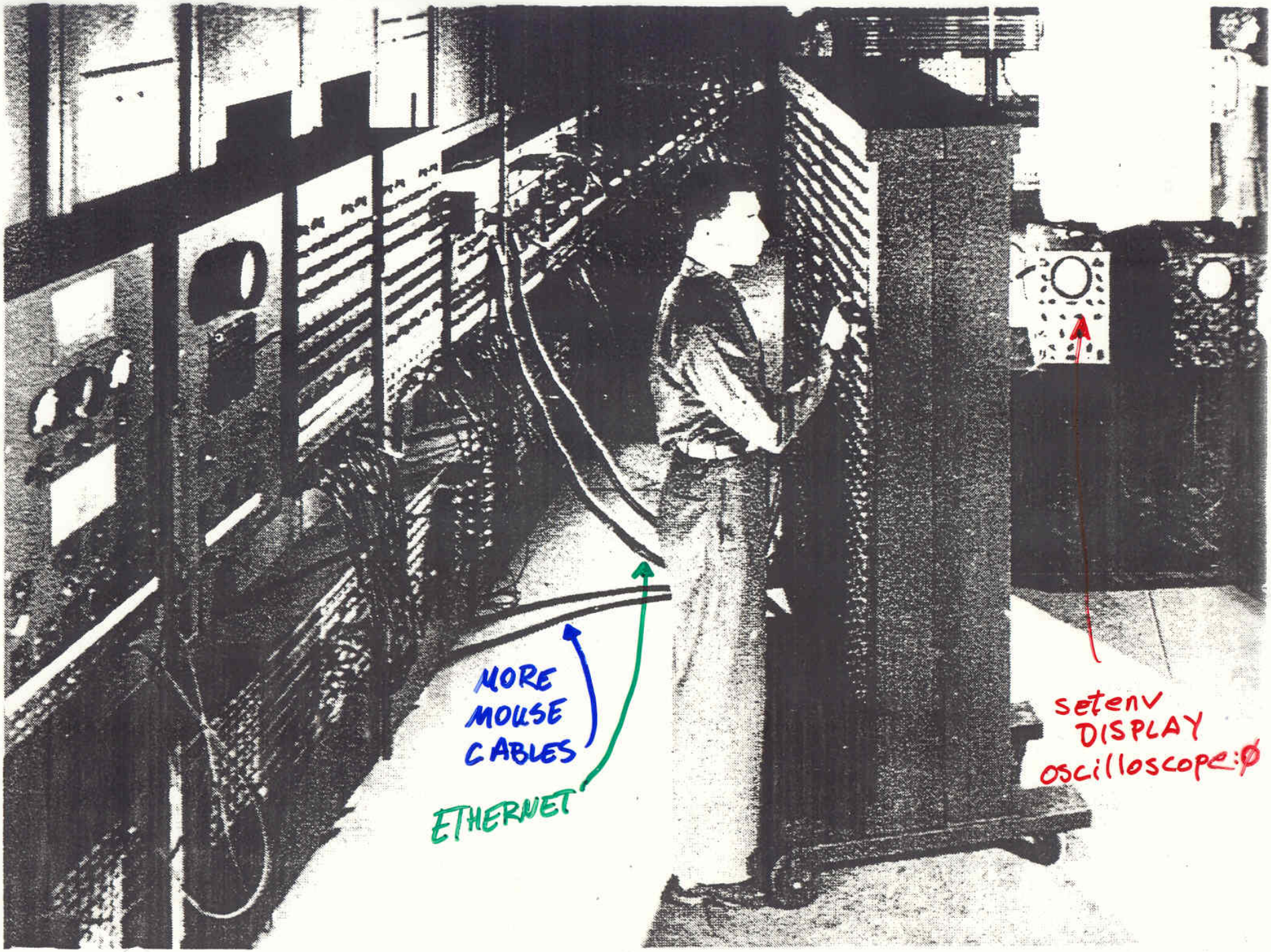


WRF ARW Nesting
One-Way and Two-Way

Dave Gill
gill@ucar.edu




MOUSE
CABLE
WEB
SERVER


MORE
MOUSE
CABLES
ETHERNET
setenv
DISPLAY
oscilloscope:0

Early Unix Interface to WRF model

Early Unix Interface to WRF model

Early Unix Interface to WRF model

Note usage of lower case “L” for the pipe character on keypuch machine

Configure WRF Code for Core, Machine, and Parallel Option

- Run configuration script, detects available options based on **`uname -a`**
- Guesses made for netcdf location, an environment variable can be set (csh)

  `setenv NETCDF /usr/local/netcdf`
- Do not ***accidently*** get the system configure command

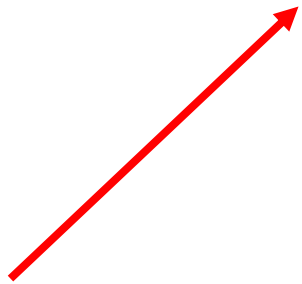
  `./configure`

What does **`./configure`** do?

- The configuration script assigns specific lines to the **`Makefile`** that are associated with the current architecture
- The user selects a certain type of build: serial, OpenMP, or MPI (if available for that architecture)
- ***Some*** settings support nesting
- Two different communication layers are usually possible: RSL and RSL_LITE
- Linux IA32 supports Intel, PGI, g95 compilers
- The direct result of running the configuration script is the generation of the file **`configure.wrf`** in the top-level WRF directory

Configuration File Enumerated Options

- A list of available compiler, parallel, and nesting options is given to the screen
- A numerical selection is made, usually #1 is serial and typically does not support nesting
- The classroom machines require a serial PGI build – but a nesting option exists



NOPE, you can do MPI

In the interest of clarity, only the PGI options are shown for the IA32 Linux (classroom test machines).

Note the "allows nesting" and "no nesting" options.

```
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)

Enter selection [1-6] :
```

Configuration File: Hmmm, I Want Another Option

- **IF** you want to modify an option that was selected through the configuration file
  - Different compiler
  - Serial, OpenMP, MPI
  - RSL *vs* RSL_LITE
  - Add the nesting capability (you can run a single domain with the nest option enabled)

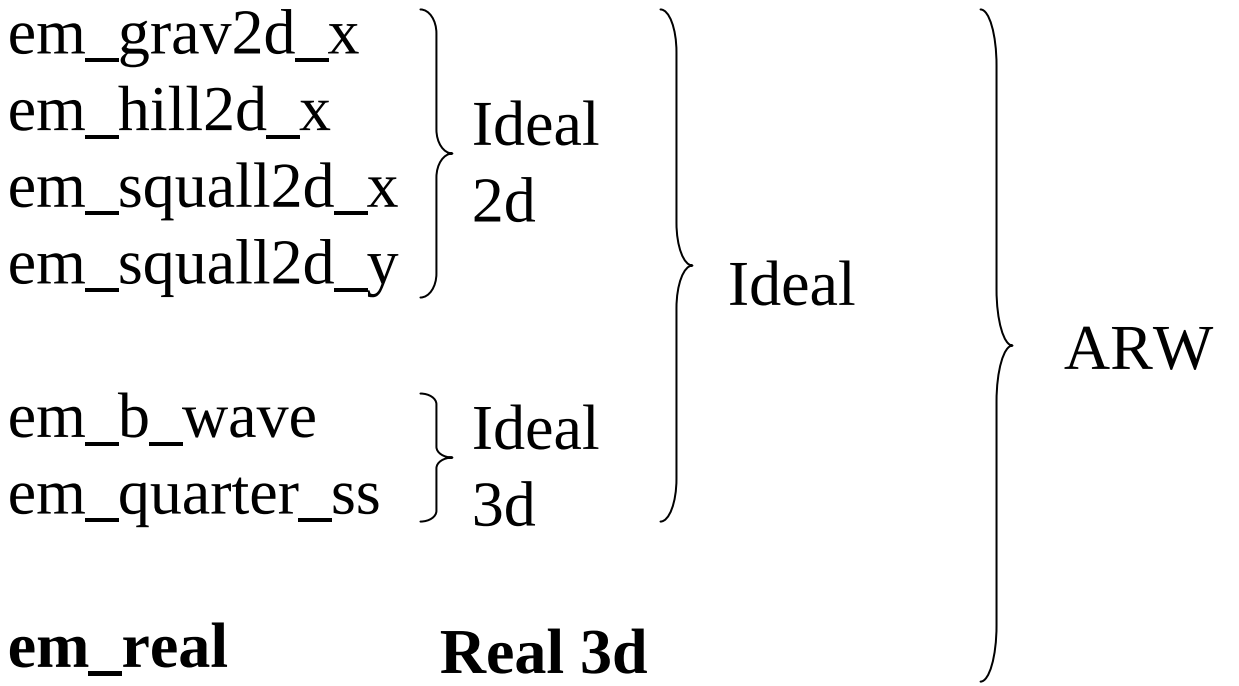
  THEN you have to start over the build and wipe out all of the generated files: include files, *.f90, object files, libraries, and executables

```
clean -a
```

Using the configuration file with **`./compile`**

Available compile targets are the names of the directories under **`./WRFV2/test`**, i.e.

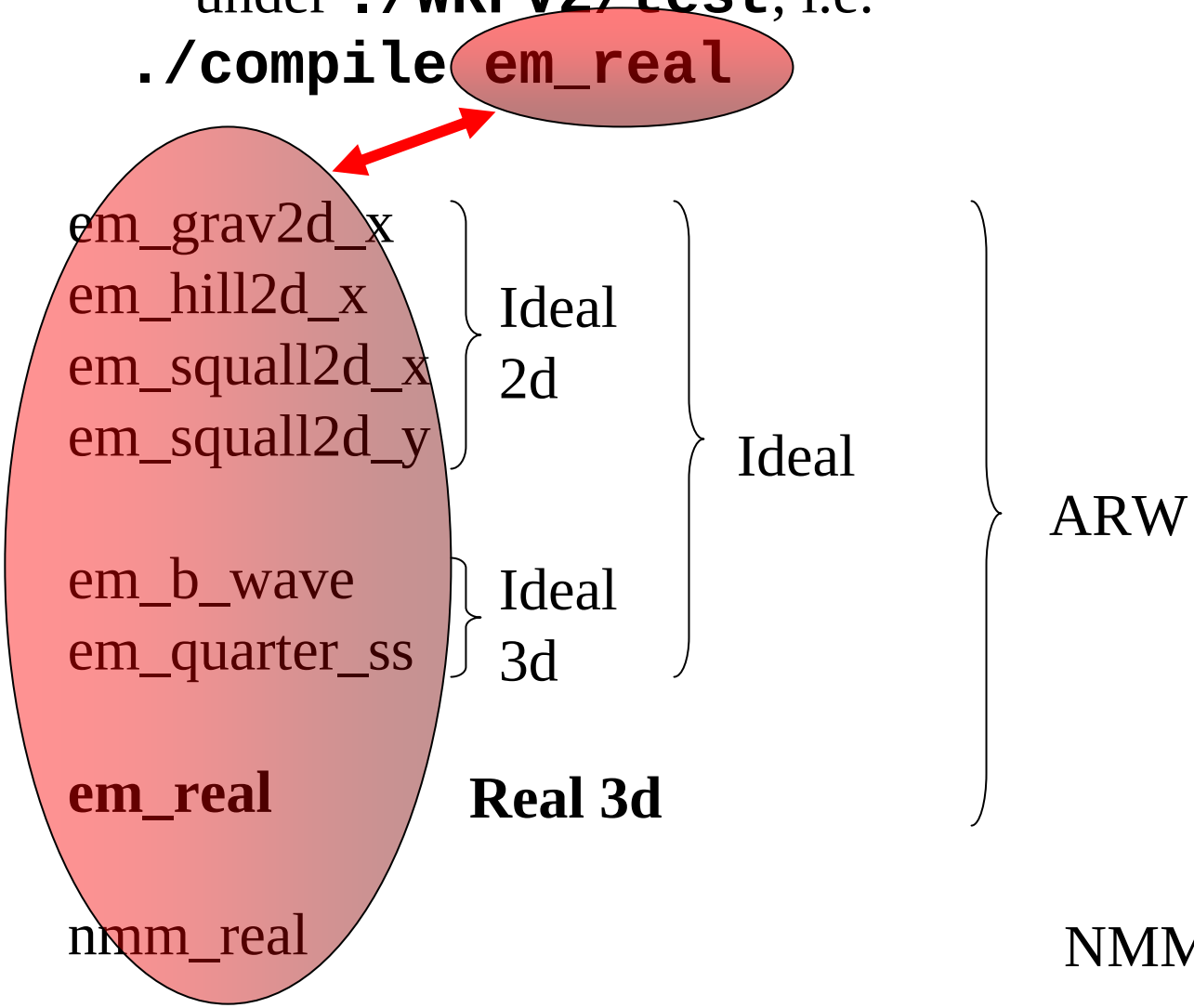
`./compile em_real`



Using the configuration file with `./compile`

Available compile targets are the names of the directories under **`./WRFV2/test`**, i.e.

`./compile em_real`



WRF Executables: Names and Locations

- The WRF executable programs are built in the **`./WRFV2/main`** directory
- The executables are linked both into the **`./WRFV2/run`** directory and to the **`./WRFV2/test/<test-case-name>`** directory

- The executables for real-data cases:

  `ndown.exe (`and **`nup.exe)`**

  `real.exe`

  `wrf.exe`

- The executables for ideal-data cases

  `ideal.exe`

  `wrf.exe`

Running WRF Executables - Preparations

- Real data cases require additional input files from the WPS package (SI is OK, too) to be in (or linked into) the run-time directory

- There are several physics related input files that are automatically linked into the run-time directories (such as various look-up tables)

- Edit the run-time configurable options in the **`namelist.input`** file located in the **`./WRFV2/test/<test-case-name>`**

  where **`<test-case-name>`** is **`em_real`**, **`em_quarter_ss`**, **`em_b_wave`**, etc.

Running WRF Executables – Preparations Real Data Case

- **`cd ./WRFV2/test/em_real`**
- **`ln –s ../../../WPS/met_em* .`**
- One WPS file required for each of the time periods for the outer-most grid
- Minimum of two WPS files (i.e. times) required for the outer-most grid for a real-data forecast
- All of the model domains can be processed through **`real.exe`** with the same **`namelist.input`** file that will be used for the WRF model
- If the model is to be run with multiple input domains ***(i.e., we are nesting)*** those should all be processed within **`real.exe`** during a single pre-processing run to reduce chances for errors

Running WRF Executables

- If the code was built Serial (one processor, csh syntax)

  `real.exe >& foo`

  `wrf.exe >& foo`

- If the build option was for OpenMP

  `setenv OMP_NUM_THREADS` ***`n`***

  `real.exe >& foo` (no timing benefit)

  `wrf.exe >& foo`

Running WRF Executables (not for class)

- Distributed memory, Message Passing (MPI)
  `mpirun –np` ***`n`*** **`real.exe`** (no large timing benefit)
  `mpirun –np` ***`n`*** **`wrf.exe`**

- On an IBM, batch mode (LSF)
  `mpirun.lsf real.exe` (no large timing benefit)
  `mpirun.lsf wrf.exe`

- On an IBM, interactively (csh syntax)
  `setenv MP_RMPOOL 1`
  `setenv MP_PROCS` ***`n`***
  `real.exe` (no large timing benefit)
  `wrf.exe`

Running WRF Executables (not for class)

- On an IBM, batch mode (LoadLeveler)
  `poe real.exe` (no large timing benefit)
  `poe wrf.exe`

- On an IBM, interactively
  `setenv MP_RMPOOL 1`
  `setenv MP_PROCS` ***`n`***
  `real.exe` (no large timing benefit)
  `wrf.exe`

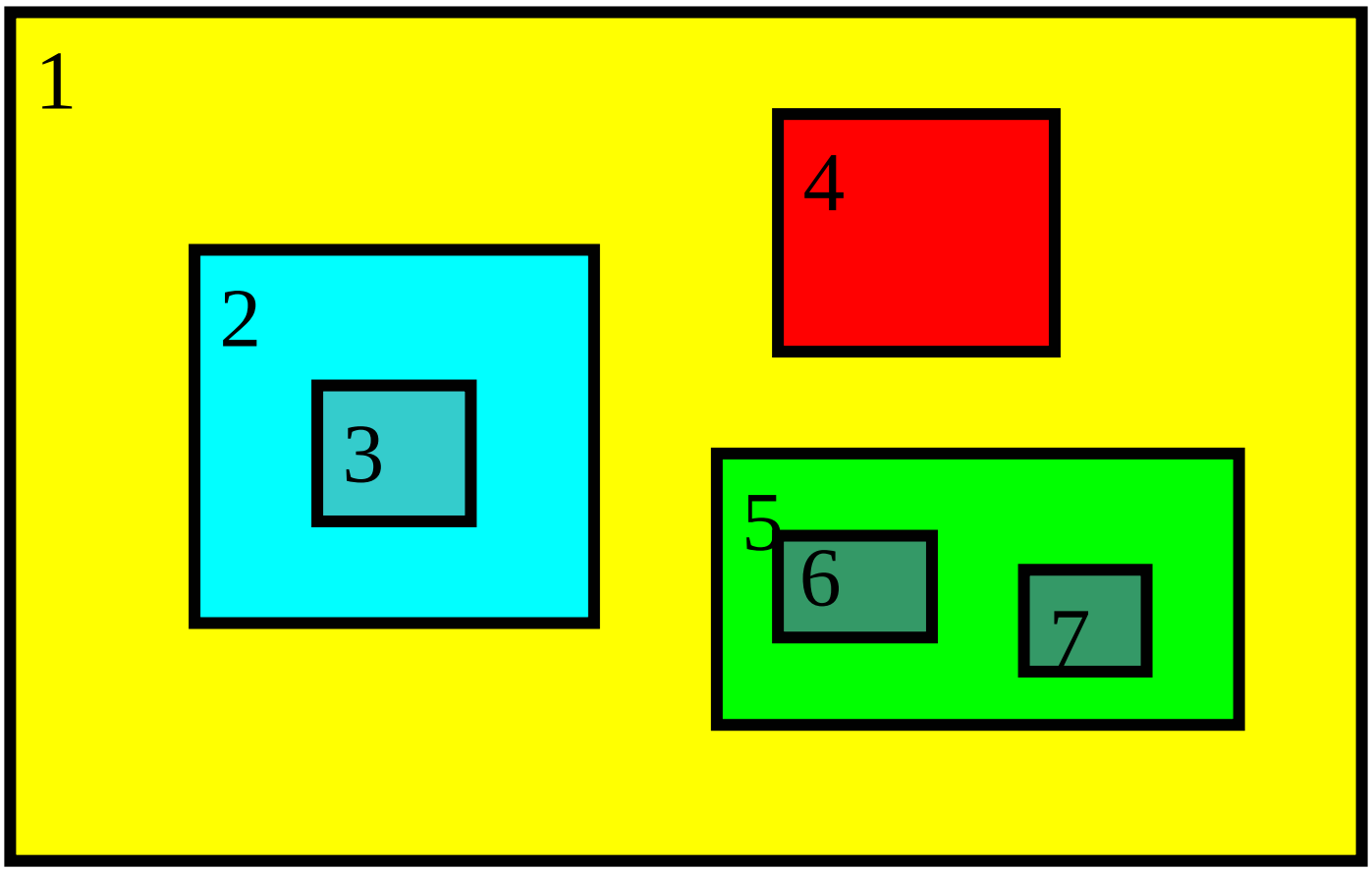
LoadLeveler leaves NCAR 31 Jan 2007

Some Nesting Hints

- Allowable domain specifications
- Defining a starting point
- Illegal domain specifications
- 1-way *vs* 2-way nesting

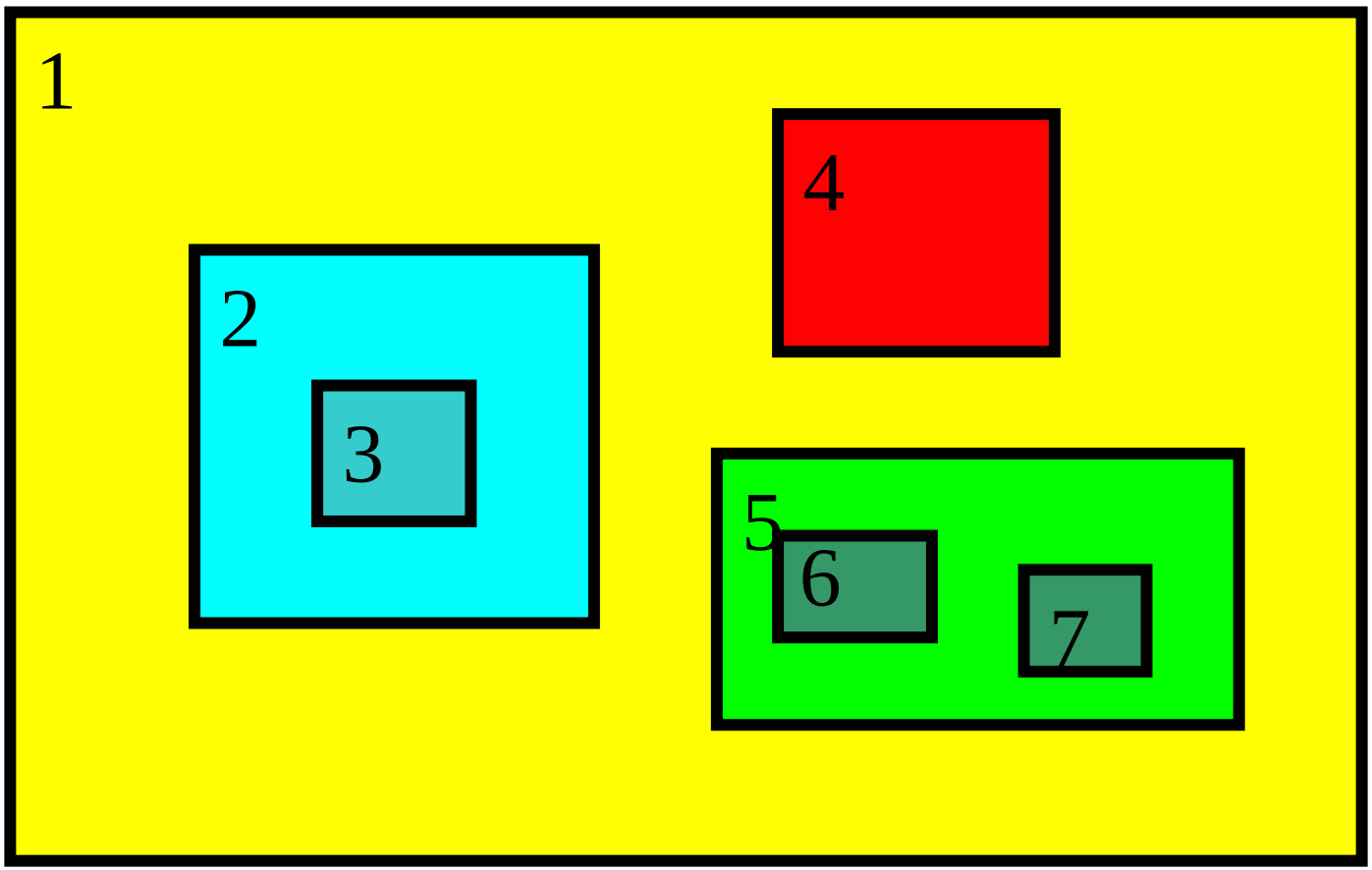
These are all OK

Telescoped to any depth
Any number of siblings



These are all OK

Telescoped to any depth
Any number of siblings



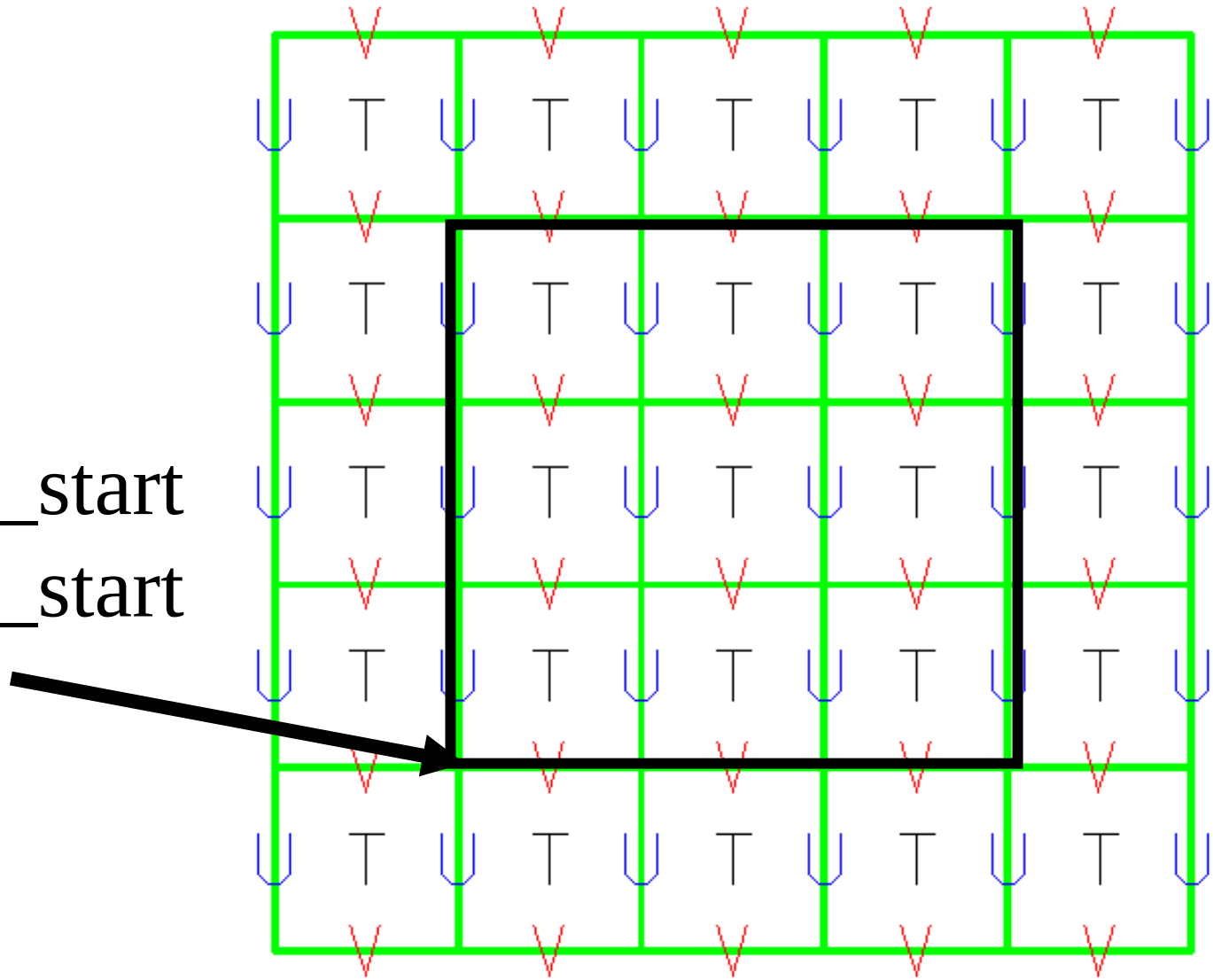
Some Nesting Hints

- Allowable domain specifications
- Defining a starting point
- Illegal domain specifications
- 1-way *vs* 2-way nesting

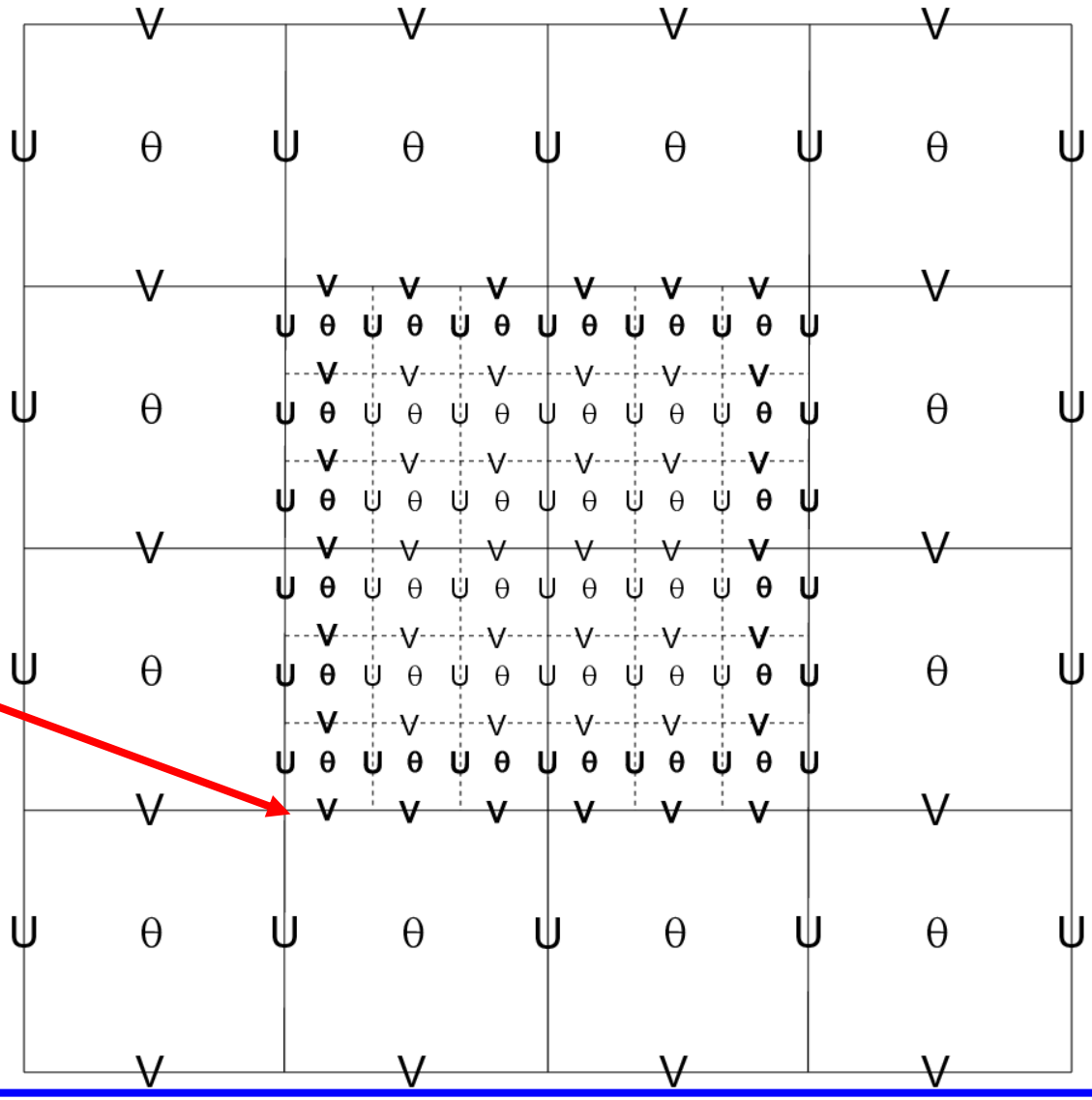
Coarse Grid Staggering

i_parent_start

j_parent_start



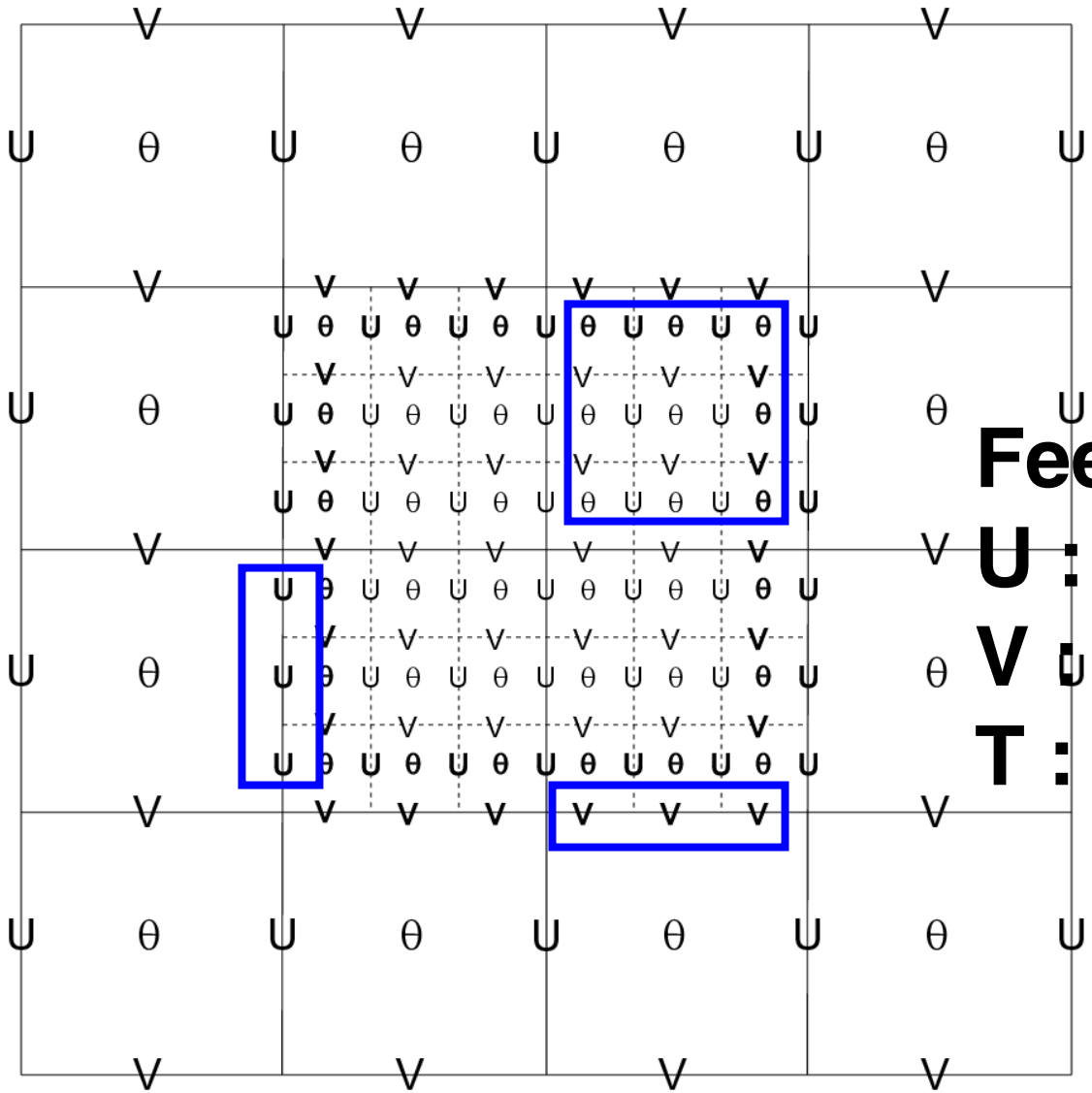
Coarse Grid Staggering 3:1 Ratio



Starting Location I = 31

CG … 30 31 32 33 34

Coarse Grid Staggering 3:1 Ratio



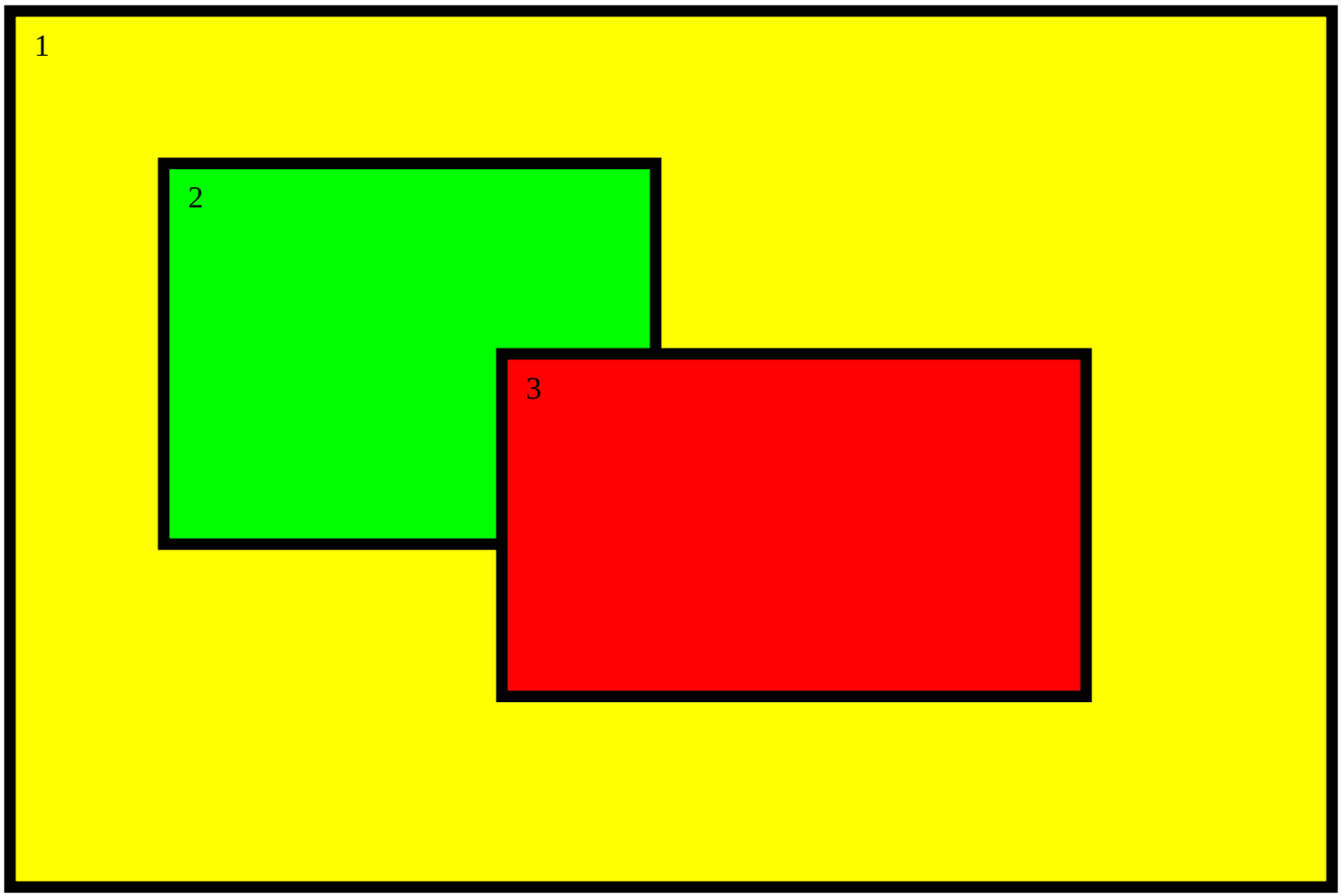
Feedback:
U : column
V : row
T : cell

Some Nesting Hints

- Allowable domain specifications
- Defining a starting point
- Illegal domain specifications
- 1-way *vs* 2-way nesting

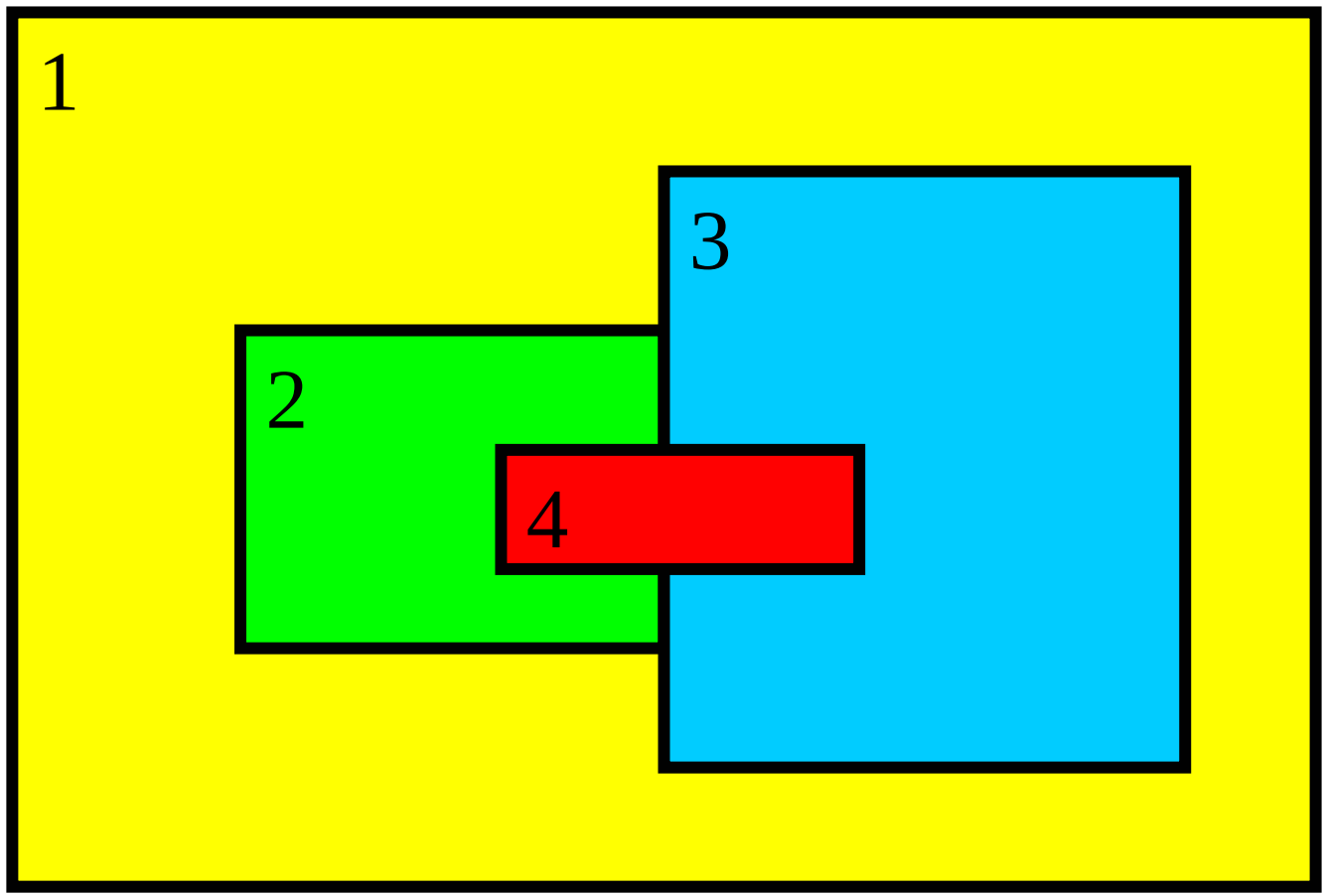
Not OK

Child domains may not have overlapping points in the parent domain (1-way nesting excluded).



Not OK either

Domains have one, and only one, parent -
(this is NOT acceptable even with 1-way nesting)



Some Nesting Hints

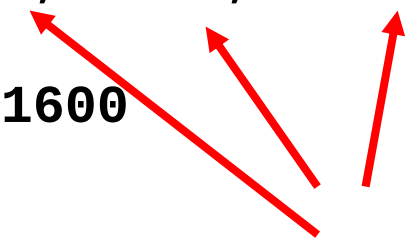
- Allowable domain specifications
- Defining a starting point
- Illegal domain specifications
- 1-way *vs* 2-way nesting

Namelist: **&time_control**

```
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
```

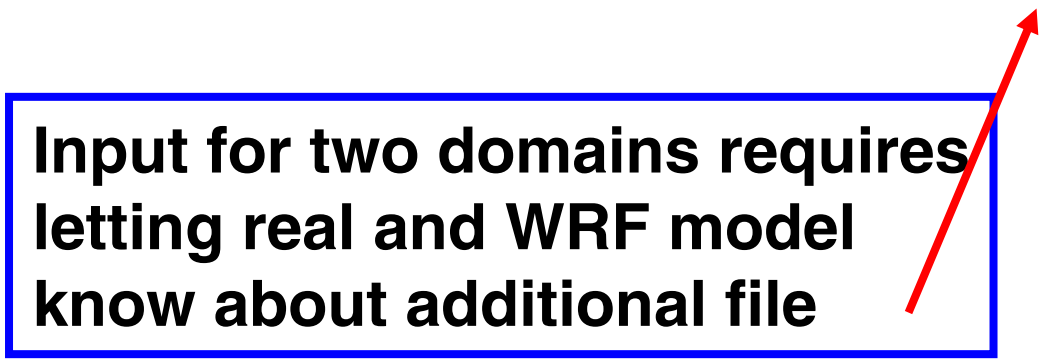


Multiple columns usually refer to different domains

Namelist: `&time_control`

```
run_days                            = 0,
run_hours                           = 12,
run_minutes                         = 0,
run_seconds                         = 0,

input_from_file                     = T,F,F,
```



Namelist: **&domains**

CG only

```
time_step                           = 180,

max_dom                             = 1,

e_we                                = 74,     112,    94,
e_sn                                = 61,     97,     91,
e_vert                              = 28,     28,     28,

dx                                  = 30000, 10000, 3333,
dy                                  = 30000, 10000, 3333,
```

Total number of all domains

This is a biggy for nesting!

Size in staggered units: u, v, w dimensions

Please keep these ratios the same as the parent grid ratio

Namelist: **&domains**

```
grid_id                             = 1,     2,     3,
parent_id                           = 0,     1,     2,

i_parent_start                      = 0,     31,    30,
j_parent_start                      = 0,     17,    30,

parent_grid_ratio                   = 1,     3,     3,
parent_time_step_ratio              = 1,     3,     3,

feedback                            = 1,
smooth_option                       = 0
```

Keep the same until you have specific needs

1-way nesting – easy way is shut off feedback

Namelist: **&domains**

```
num_moves                           = 3
move_id                             = 2 , 2 , 2
move_interval                       = 3 , 6 , 9
move_cd_x                           = 1 , 1 , 1
move_cd_y                           = 1 , 1 , 1
```

- add **-DMOVE_NESTS** to ARCHFLAGS in configure.wrf BEFORE ./compile command
- Keep number of moves large and increments: {-1, 0, 1}

Namelist: **&domains**

- add **-DMOVE_NESTS -DVORTEX_CENTER** to ARCHFLAGS in configure.wrf BEFORE ./compile command

Namelist: **&bdy_control**

```
spec_bdy_width                    = 5,
spec_zone                         = 1,
relax_zone                        = 4,
```

Reasonable defaults

```
specified                         = T , F , F
nested                            = F , T , T
```

LEAVE THESE ALONE!

1-way *vs.* 2-way Nesting

- wrf integrates 1 domain at a time
- CG forces FG through lateral boundaries
- No FG to CG feedback
- ndown run between CG wrf and FG wrf (or shut off feedback)

- wrf integrates 2 domains at a time
- CG forces FG at every FG timestep
- FG to CG feedback at every CG timestep
- ndown not required

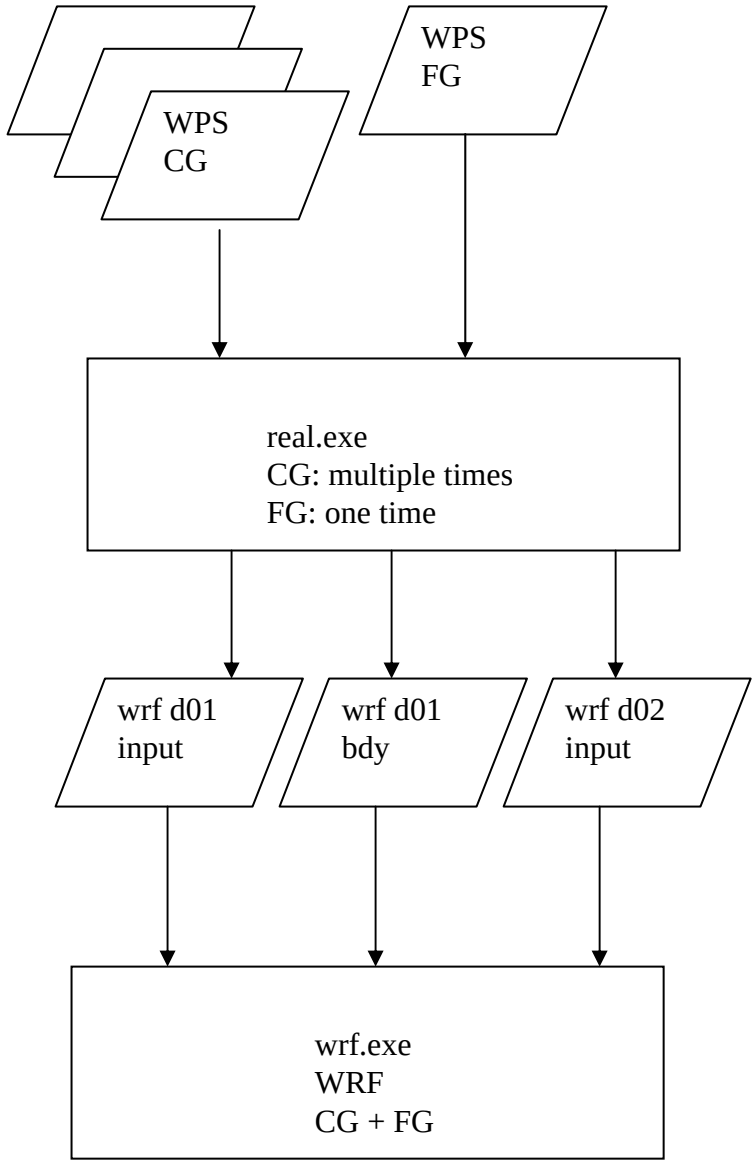
2-Way Nest with 2 Inputs

Coarse and fine grid domains must start at the same time, fine domain may end at any time

Feedback may be shut off to produce a 1-way nest (cell face and cell average)

Any integer ratio for coarse to fine is permitted, odd is usually chosen for real-data cases

Options are available to ingest only the static fields from the fine grid, with the coarse grid data horizontally interpolated to the nest



2-Way Nest with 2 Inputs

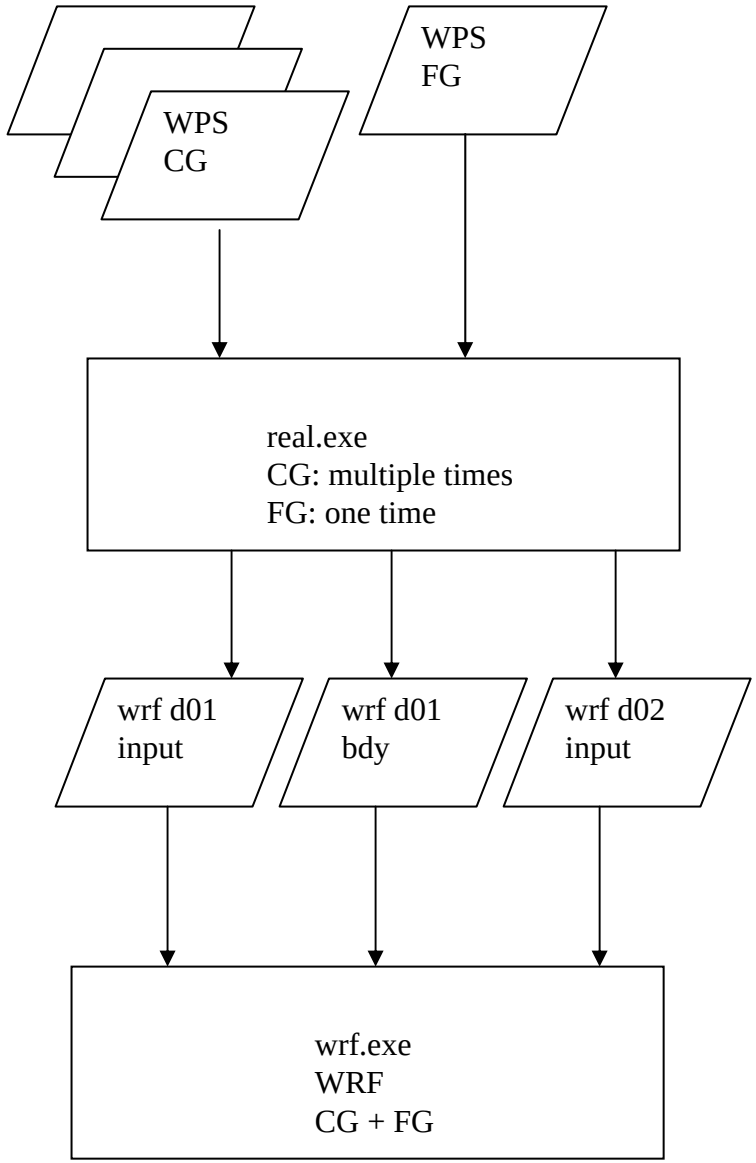
No vertical nesting

Usually the same physics are run on all of the domains (excepting cumulus)

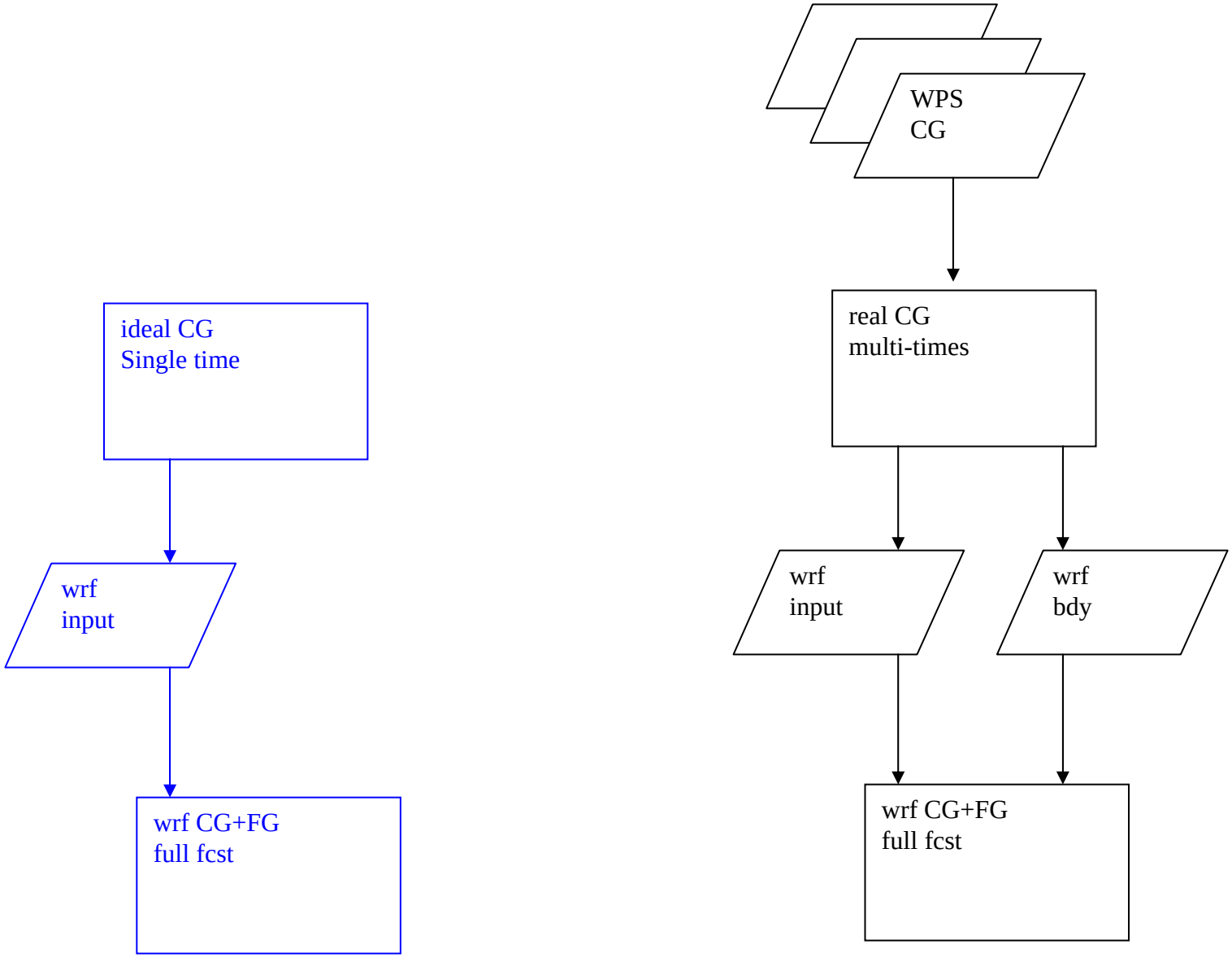
The grid distance ratio is not strictly tied to the time step ratio

Topography smoothly ramps from coarse grid to the fine grid along the interface along the nest boundary

All fine grids must use the nested lateral boundary condition



2-Way Nest with 1 Input



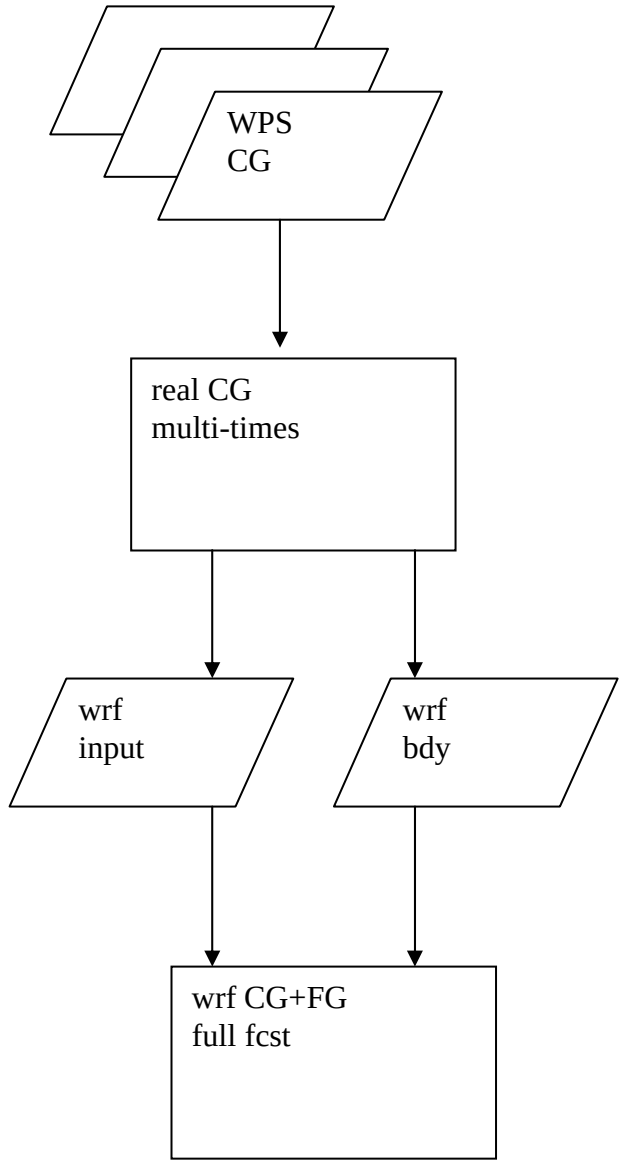
2-Way Nest with 1 Input

A single namelist column entry is tied to each domain

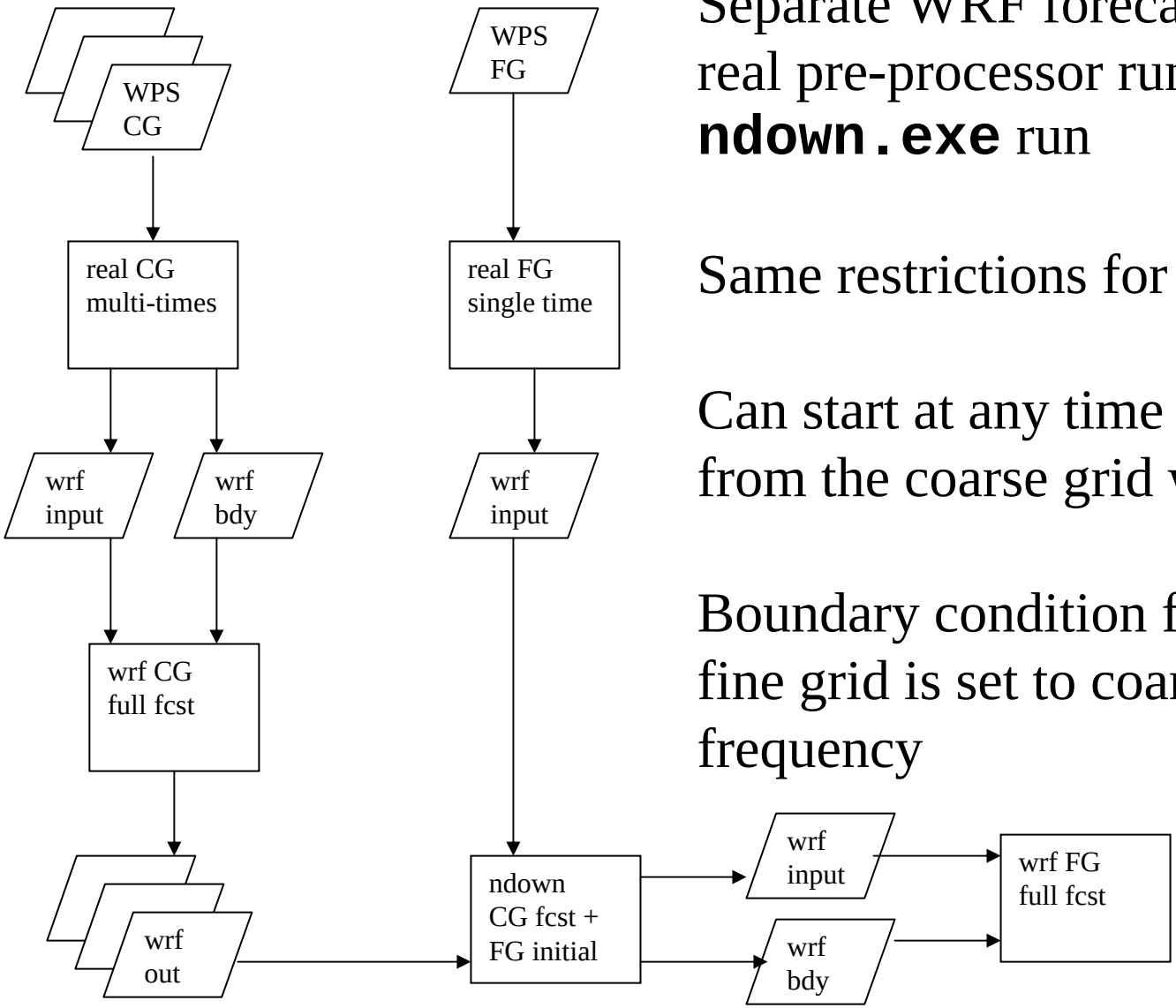
The horizontal interpolation method, feedback, and smoothing are largely controlled through the Registry file

For a 3:1 time step ratio, after the coarse grid is advanced, the lateral boundaries for the fine grid are computed, the fine grid is advanced three time steps, then the fine grid is fed back to the coarse grid (recursively, depth first)

Helpful run*.tar files are located in the **`./WRFV2/test/em_real`** directory



1-Way Nest with 2 Inputs



Separate WRF forecast runs, separate real pre-processor runs, intervening **`ndown.exe`** run

Same restrictions for nest ratios

Can start at any time that an output time from the coarse grid was created

Boundary condition frequency for the fine grid is set to coarse grid output frequency

Checking Output: Standard Printout

- The WRF pre-processors and the model generate print out for user checking, and can be dramatically increased via namelist settings
- For serial and OpenMP jobs, the printout is either on the screen or redirected to a log file
- The distributed memory jobs generate files of the form: **`rsl.out.0000`** and **`rsl.error.0000`** (from 0000 to ***n-1***, where ***n*** = the number of processors used)
- The last line is a (usually grammatically incorrect) status message:

```
real_em: SUCCESS COMPLETE REAL_EM INIT
```

Checking Output: Standard Printout

- For WRF, the time steps are of interest:

```
 WRF NUMBER OF TILES =   1
Timing for main: time 2000-01-24_12:03:00 on domain   1:   13.95000 elapsed seconds.
Timing for main: time 2000-01-24_12:06:00 on domain   1:    2.53000 elapsed seconds.
Timing for main: time 2000-01-24_12:09:00 on domain   1:    2.54000 elapsed seconds.
Timing for main: time 2000-01-24_12:12:00 on domain   1:    2.54000 elapsed seconds.
Timing for main: time 2000-01-24_12:15:00 on domain   1:    2.56000 elapsed seconds.
Timing for main: time 2000-01-24_12:18:00 on domain   1:    2.55000 elapsed seconds.
Timing for main: time 2000-01-24_12:21:00 on domain   1:    2.56000 elapsed seconds.
Timing for main: time 2000-01-24_12:24:00 on domain   1:    2.56000 elapsed seconds.
Timing for main: time 2000-01-24_12:27:00 on domain   1:    2.56000 elapsed seconds.
Timing for main: time 2000-01-24_12:30:00 on domain   1:   13.29000 elapsed seconds.
Timing for Writing wrfout_d01_2000-01-24_12:30:00 for domain        1:    0.29000
    elapsed seconds.
 wrf: SUCCESS COMPLETE WRF
```

Sample Data Sets

- There are sample data sets to make WRF porting easier
- In each of the ideal directories, there are several namelist.input files set up to test different grid distance, boundary condition, dynamics, and physics suites
- In the **`./WRFV2/test/em_real`** directory, there are several template namelists
- There is both a January 2000 and a June 2001 namelist, each associated with downloadable data sets from the WRF download page

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

- The SIMPLE test case for classroom example uses Jan 2000 file