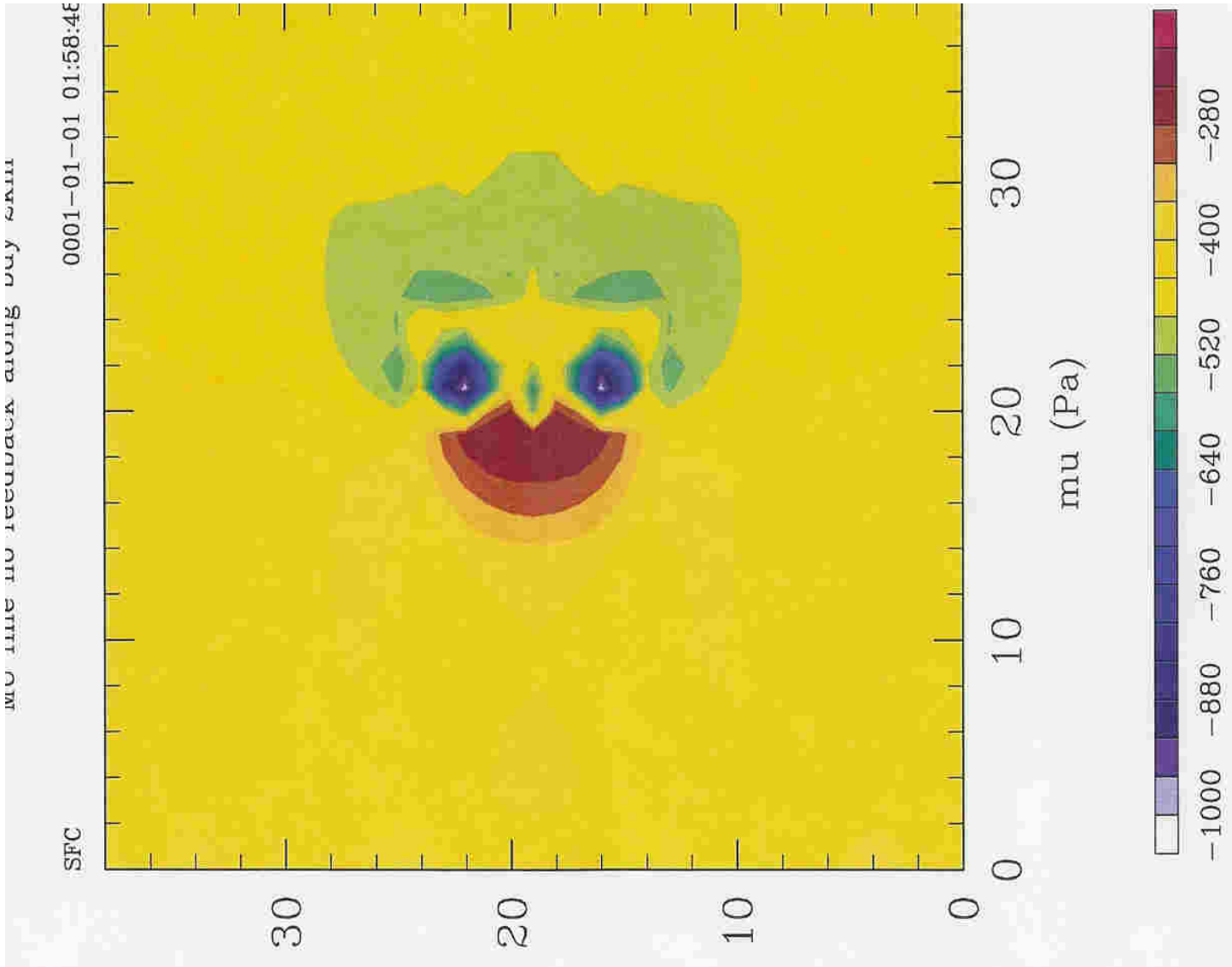
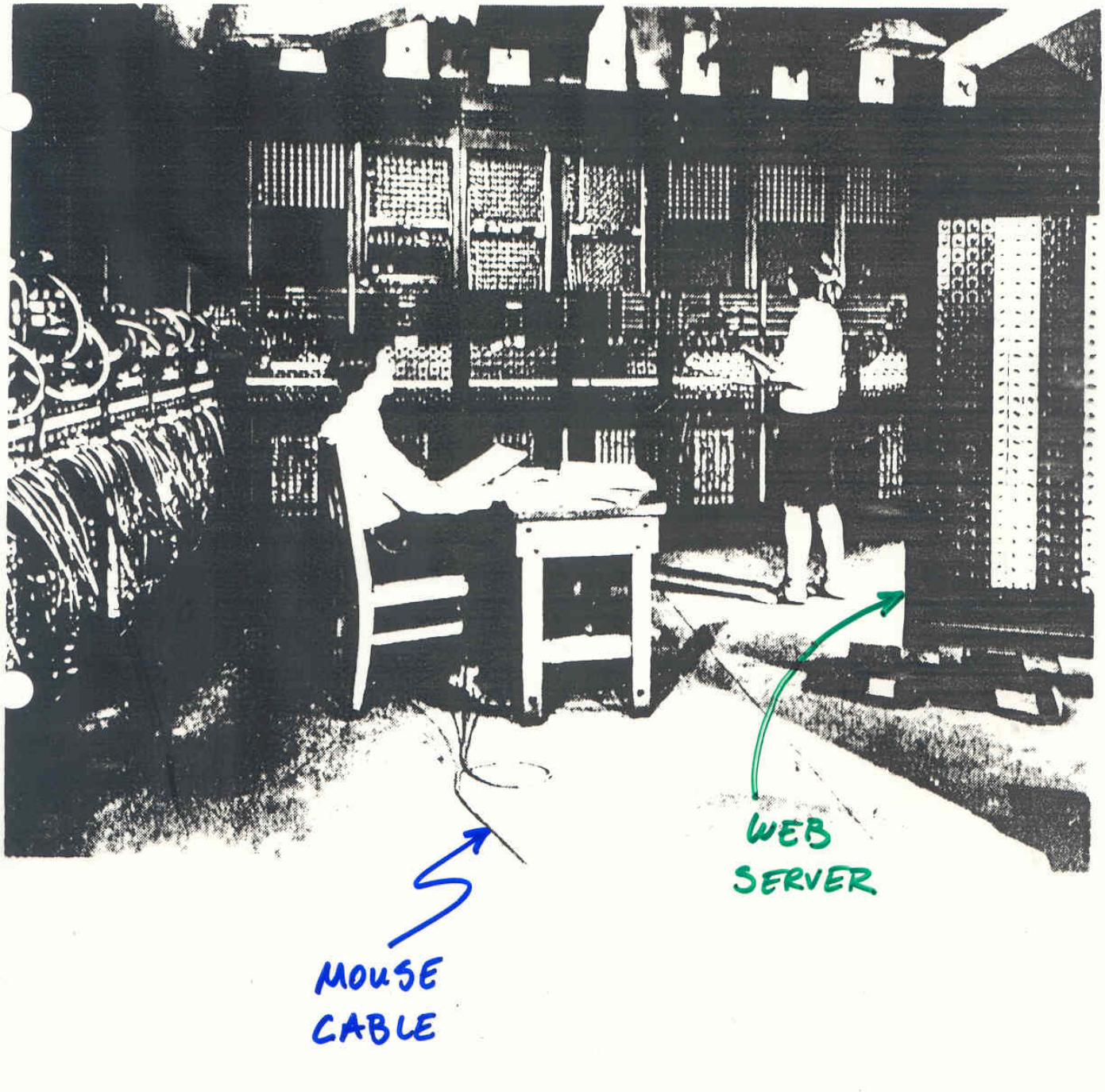
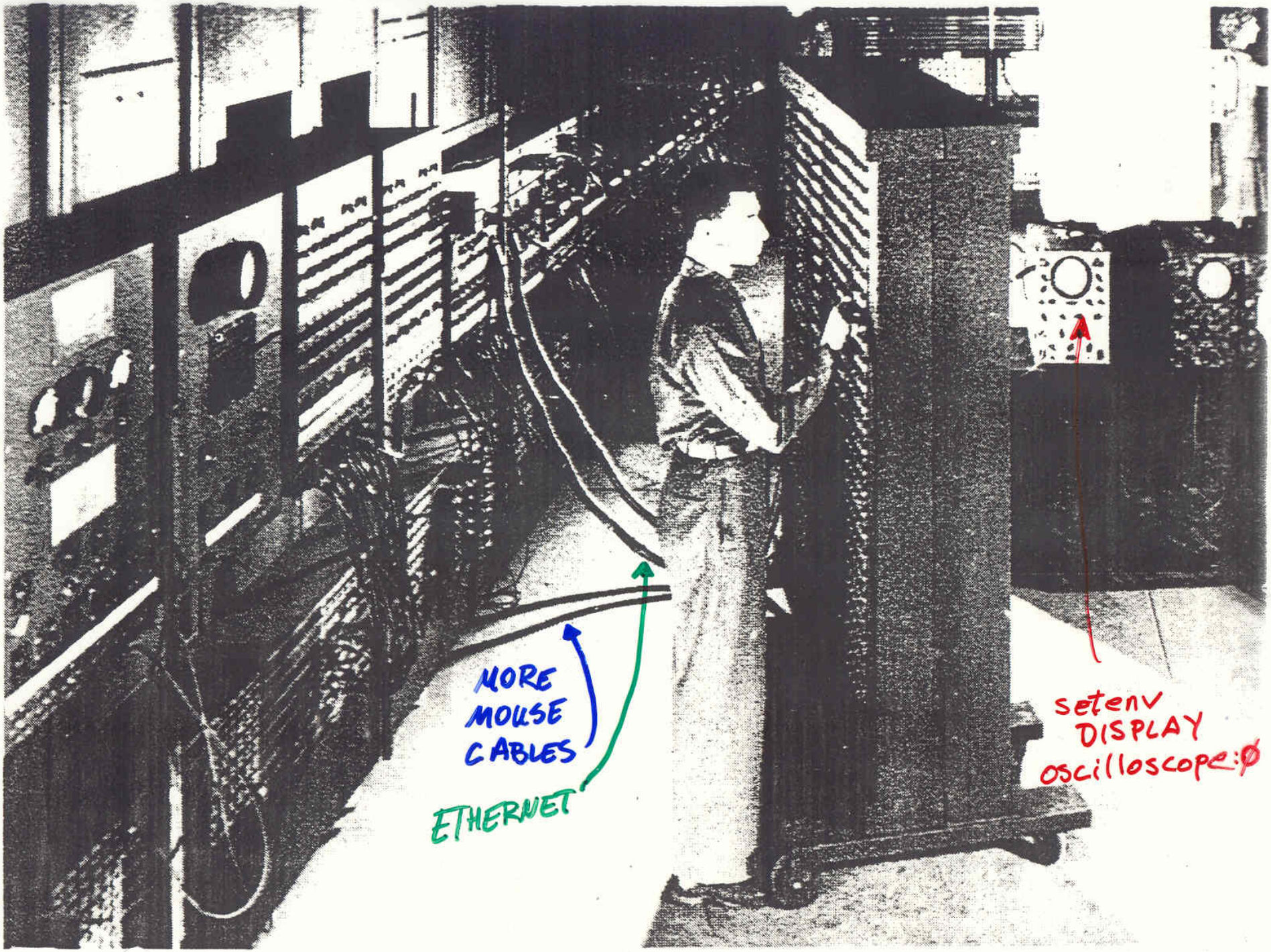
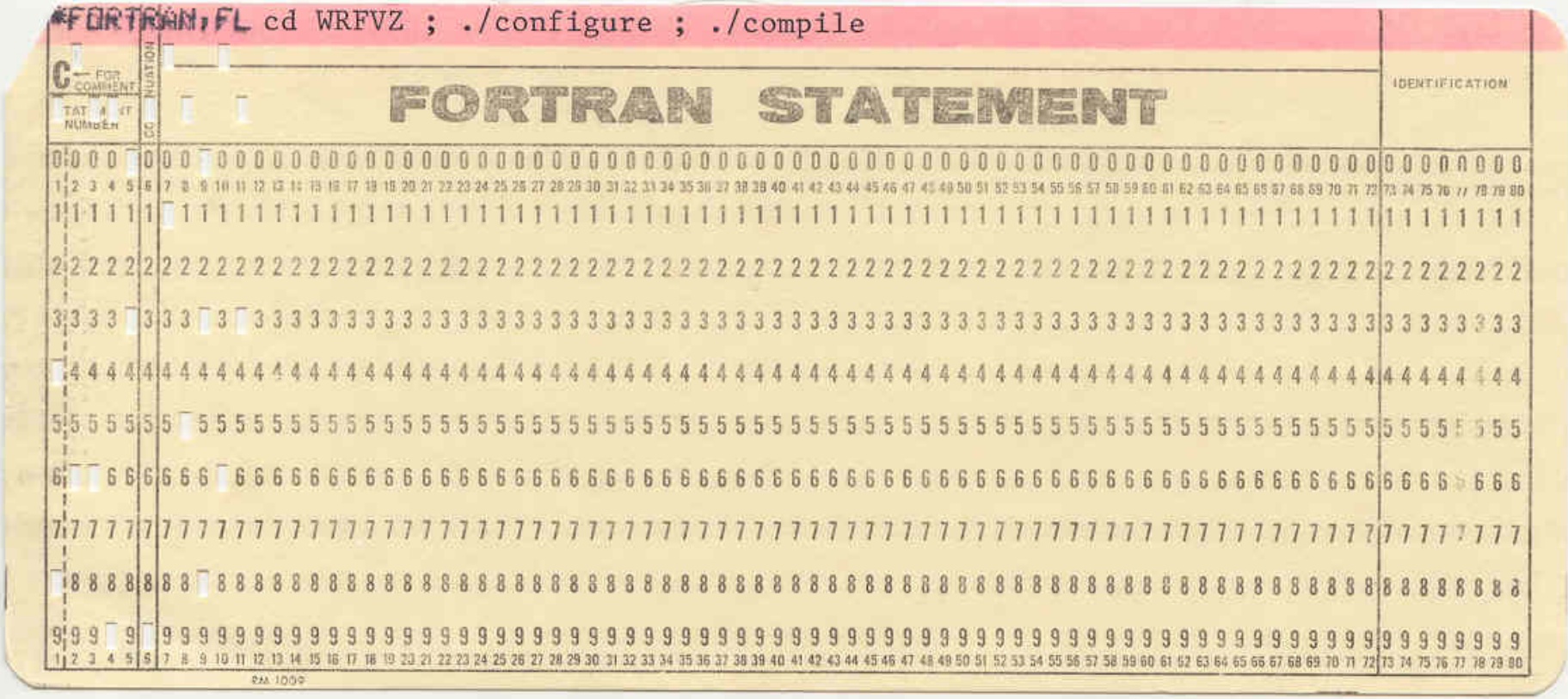
WRF ARW
How to Set Up and Run

Dave Gill
gill@ucar.edu




WEB
SERVER
MOUSE
CABLE


MORE
MOUSE
CABLES
ETHERNET
setenv
DISPLAY
oscilloscope:0


*FORTRAN,FL cd WRFVZ ; ./configure ; ./compile
C — FOR COMMENT
STATEMENT NUMBER
CONTINUATION
FORTRAN STATEMENT
IDENTIFICATION

```
cd WRFVZ ; ./configure ; ./compile
```

FORTRAN STATEMENT

WRF ARW
How to Set-up and Run

- Downloading the source
- The WRF top-level directory
- Getting friendly with your configure.wrf file
- compile: the WRF build command

WRF

- Executables: names, locations
- Run preparations for ideal, real, ndown, wrf
- Running on different architectures, processor counts
- A few nesting hints
- Checking output
- Sample data sets

ARW

Download WRF

- Download WRF (unified ARW + NMM) source code from

  http://www.mmm.ucar.edu/wrf/users/downloads.html
- Then select the Download tab, and choose WRF V2, where you are interrogated relentlessly
- What you will get is

  `WRFV2.1.2.TAR.gz`

For the practice sessions, get the file from our local disk

Download WRF

- Unreleased code is tied to WPS
- Download WRF (unified ARW + NMM) source code EVENTUALLY from

  http://www.mmm.ucar.edu/wrf/users/downloads.html
- Then select the Download tab, and choose WRF V2, where you are interrogated relentlessly
- What you will get is

  `WRFV2.2.TAR.gz`

Download WRF

- Unreleased code is tied to WPS for real-data cases
- **`WPS+WRFV2.tar.gz`**

  For the practice sessions, get the file from our local disk

Unzip and Untar Downloaded WRF File

```
cd to_where_you_want_the_source
```

- For the practice, this is also where the code will be run.

```
tar –xf WPS+WRFV2.tar
cd WPS+WRFV2
```

- After **`untar`** and **`cd`**, you should see two directories **`WPS/`** and **`WRFV2/`**

```
cd WRFV2
```

WRF
Top-Level
Directory

Unified
ARW/NMM

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

build scripts

CASE input files

machine build rules

source code directories

execution directories

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`

- Separation point for ARW/NMM, default is ARW

  `setenv WRF_NMM_CORE 1`

- 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 both Intel and PGI 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

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] :
```

What Is Inside the **`configure.wrf`** File

The **`configure.wrf`** file is built from three pieces

- **`preamble:`** uniform requirements for the code, such as maximum number of domains and moving nest instructions
- **`configure.defaults:`** selection of compiler, parallel, communication layer (users typically edit this part if they change the compilation options or library locations)
- **`postamble:`** standard make rules and dependencies

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)
```

This is not a good place to randomly experiment!

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

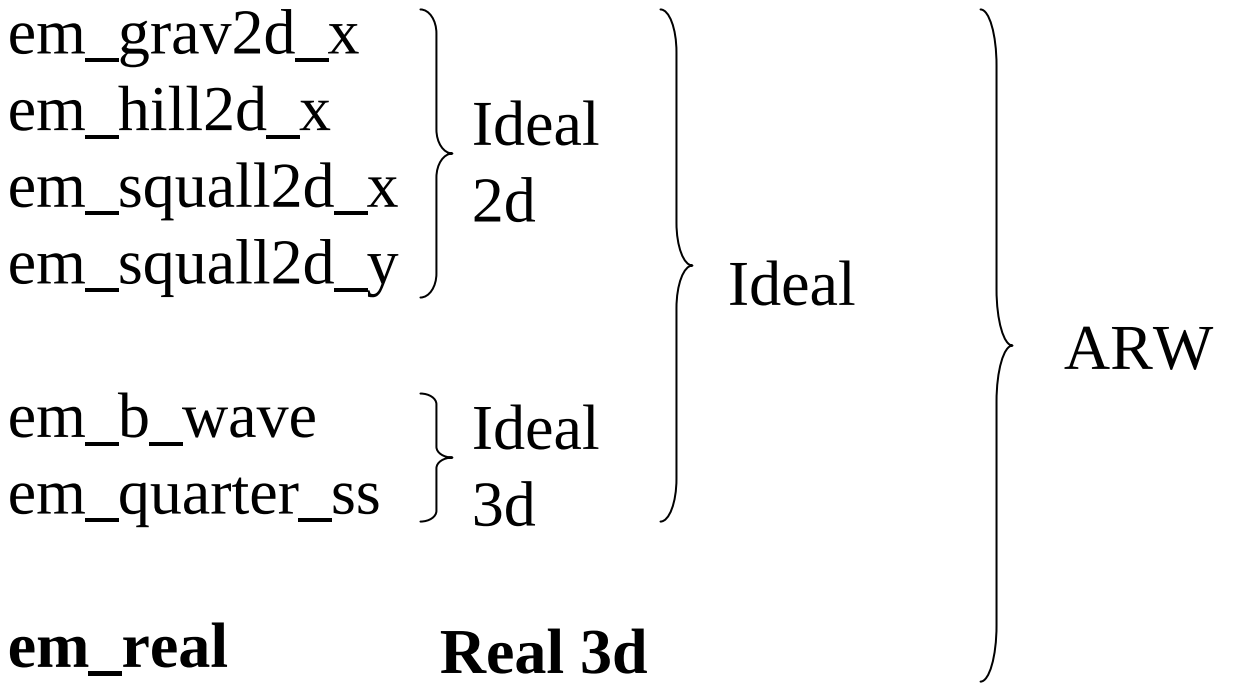
  THEN you have to start over the build and wipe out all of the generated files: include files, *.f, object files, executables

- **`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`



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

- For real data cases, the executables built are:

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

  `real.exe`

  `wrf.exe`

- For ideal data cases, the executables built are:

  `ideal.exe`

  `wrf.exe`

Running WRF Executables - Preparations

- Real data cases require additional input files from the WPS package 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 (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 Ideal Data Case

- **`cd ./WRFV2/test/em_b_wave`** (for example)
- No external input data is required except for a sounding
- No lateral boundary condition file is generated for any of the ideal data cases
- Only a single model input domain is generated by the **`ideal.exe`** program
- All sub-domains in the WRF model are internally generated via horizontal interpolation
- Some ideal data directories have a **`run_me_first.csh`** file that links in some required physics tables

Running WRF Executables – Preparations Ideal Data Case

- 2d cases and baroclinic wave case must run ideal.exe serially
- All 2d cases must run model either serially or with OMP

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 boundary times 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, those should all be processed within **`real.exe`** during a single pre-processing run to reduce chances for errors

Running WRF Executables

- Serial (one processor, csh syntax)

  `real.exe >&! foo`

  `wrf.exe >&! foo`

- OpenMP

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

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

  `wrf.exe >&! foo`

Running WRF Executables

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

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

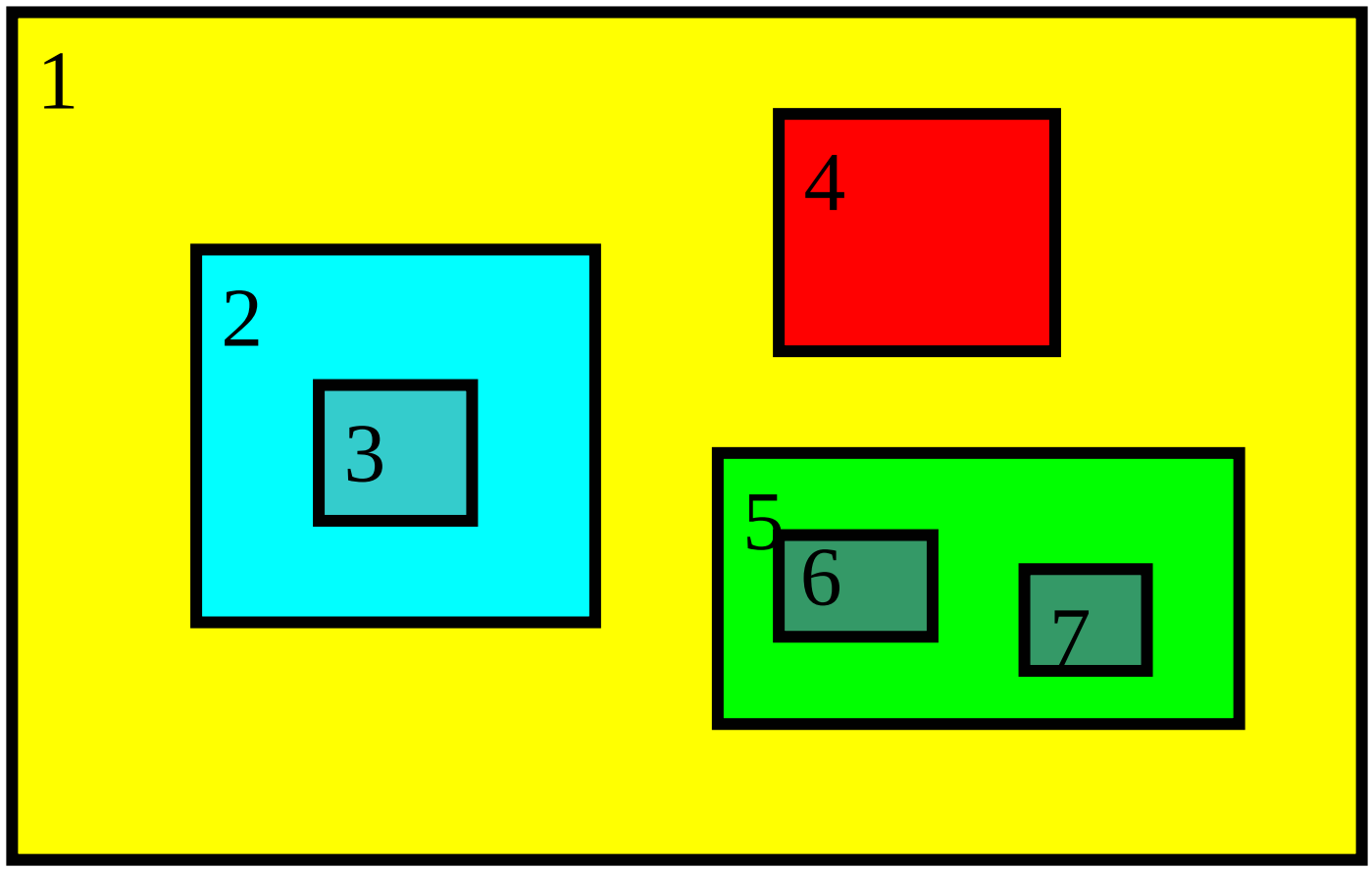
- On an IBM, interactively
  `setenv MP_RMPOOL 1`
  `setenv MP_PROCS` *`n`*
  `poe real.exe` (no large timing benefit)
  `poe wrf.exe`

Some Nesting Hints

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

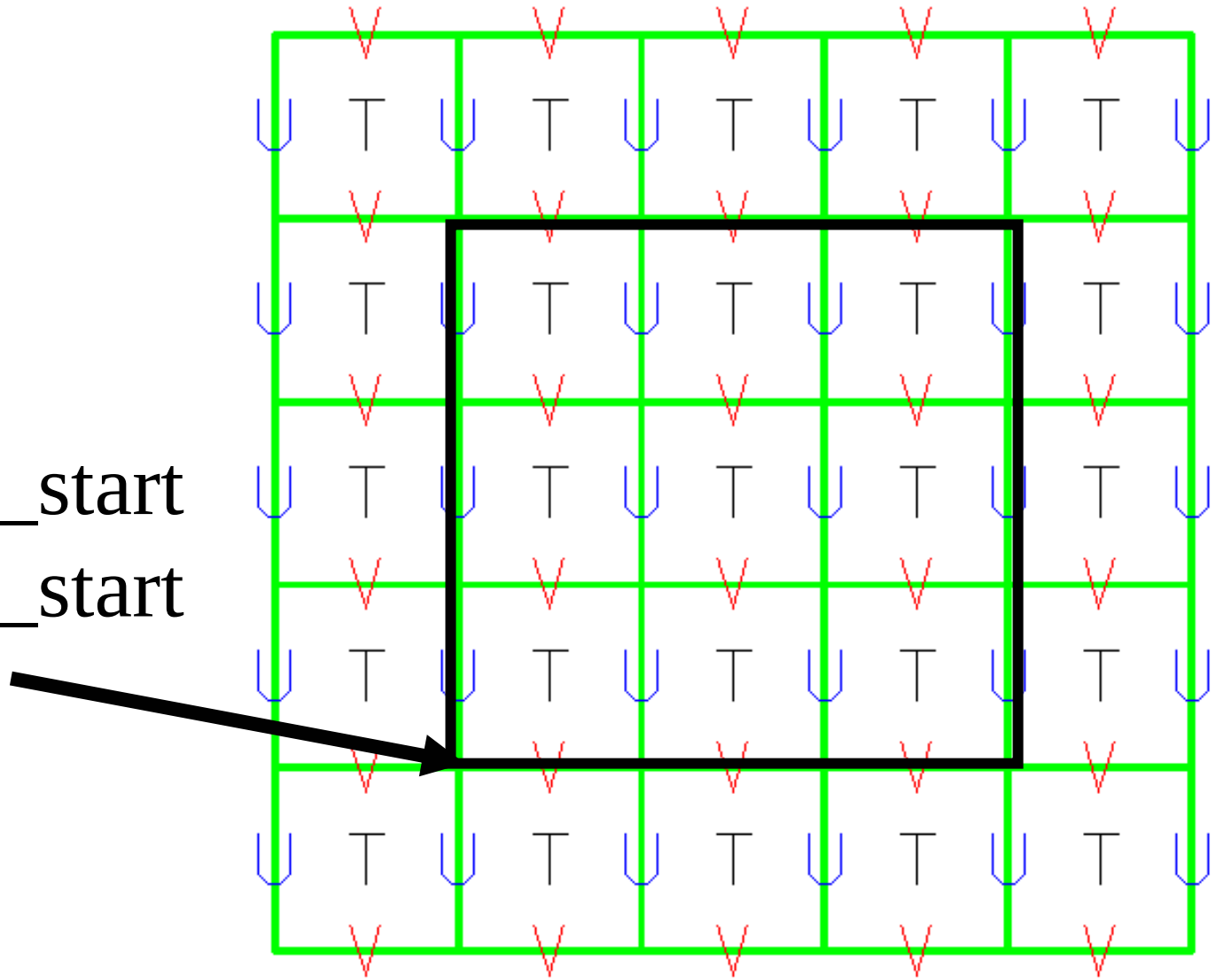
This is OK

Telescoped to any depth
Any number of siblings



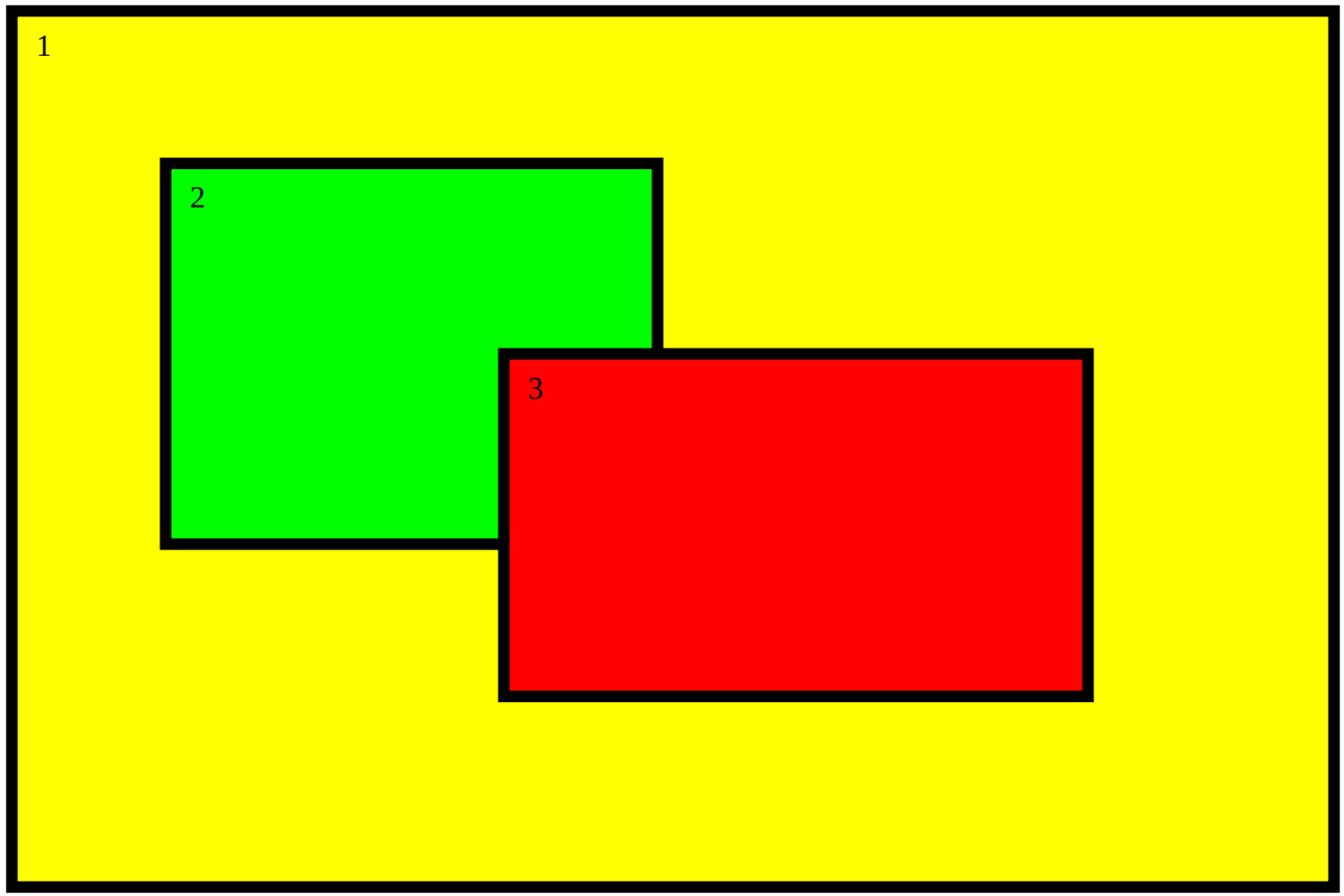
Coarse Grid Staggering

i_parent_start
j_parent_start



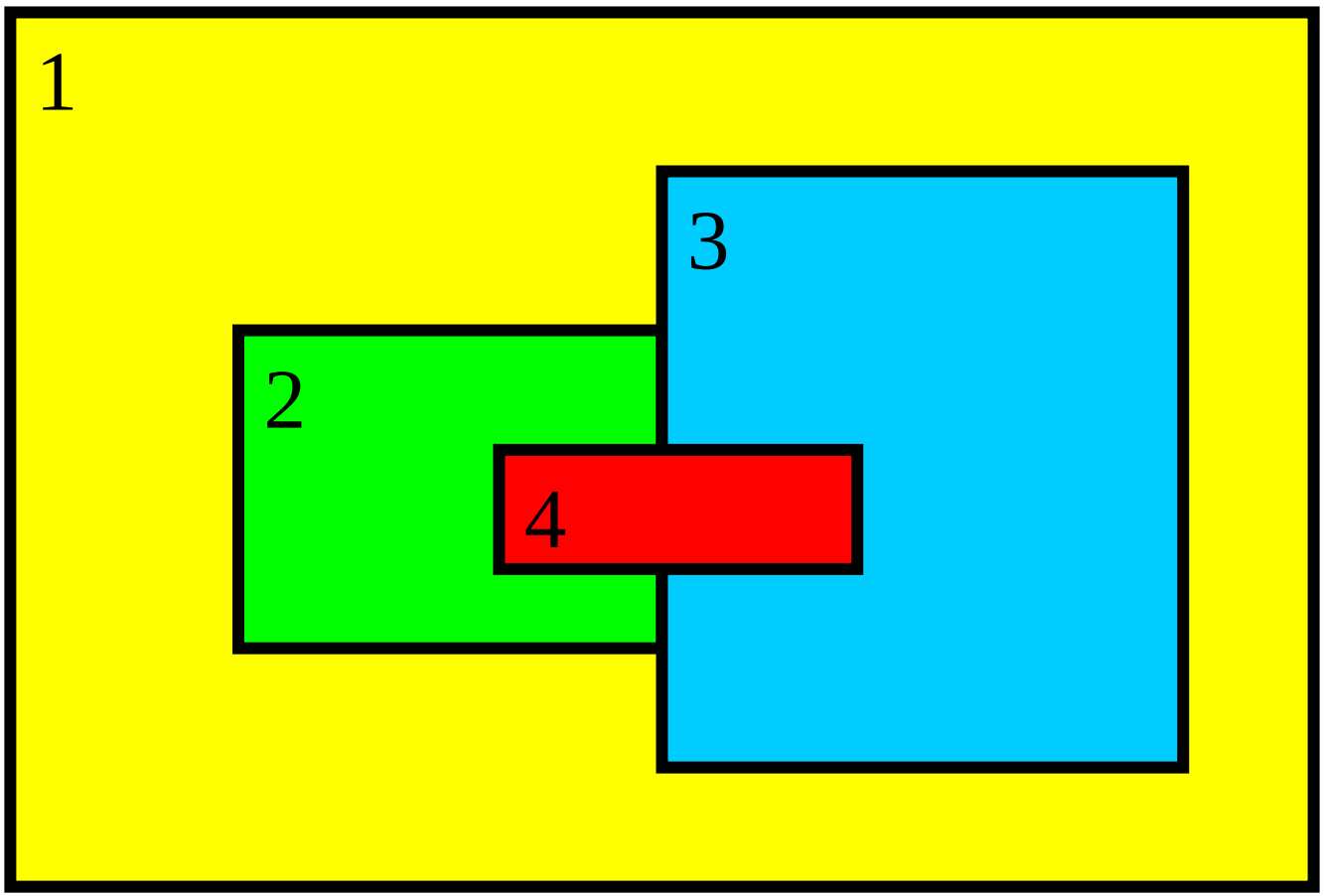
Not OK

Domains may not have overlapping points in the CG



Not OK either

Domains have one, and only one, parent



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

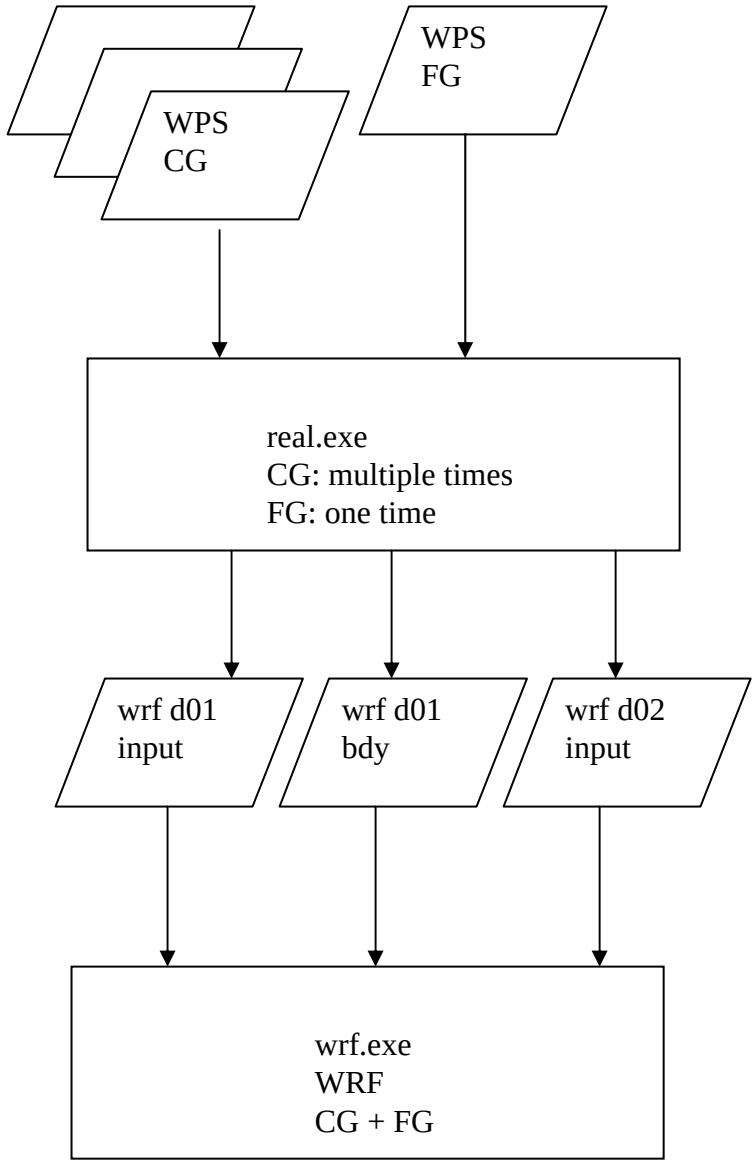
2-Way Nest with 2 Inputs

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

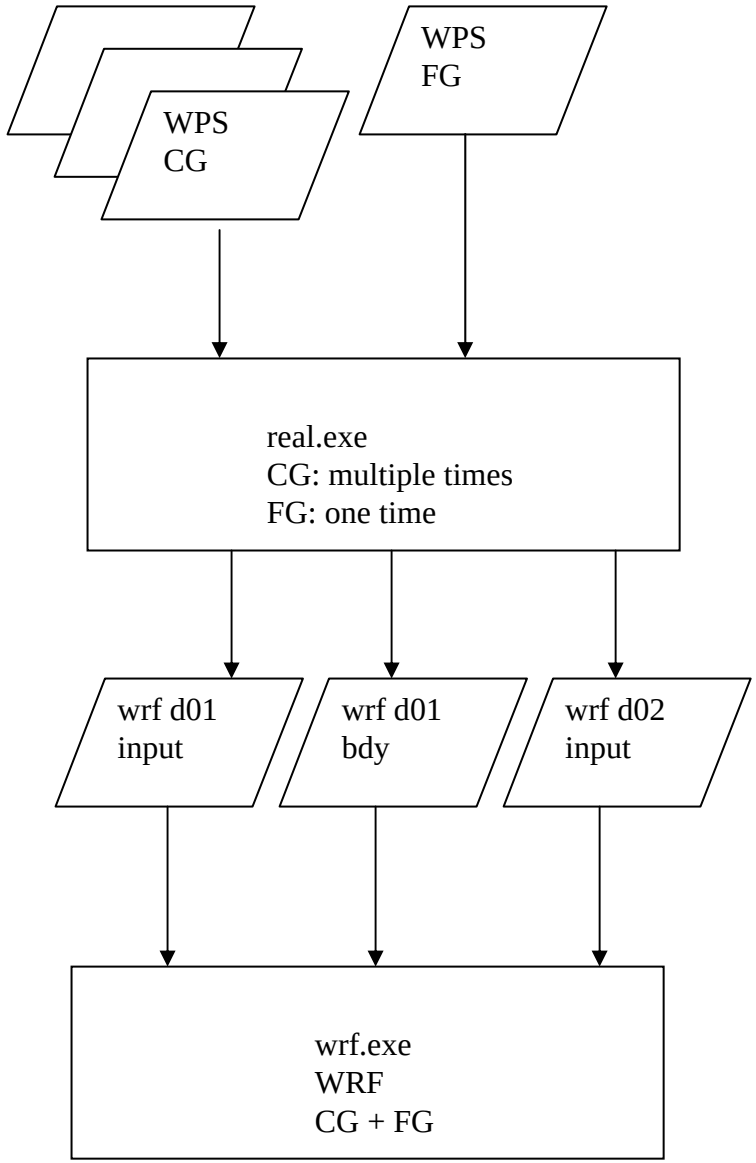
No vertical nesting

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

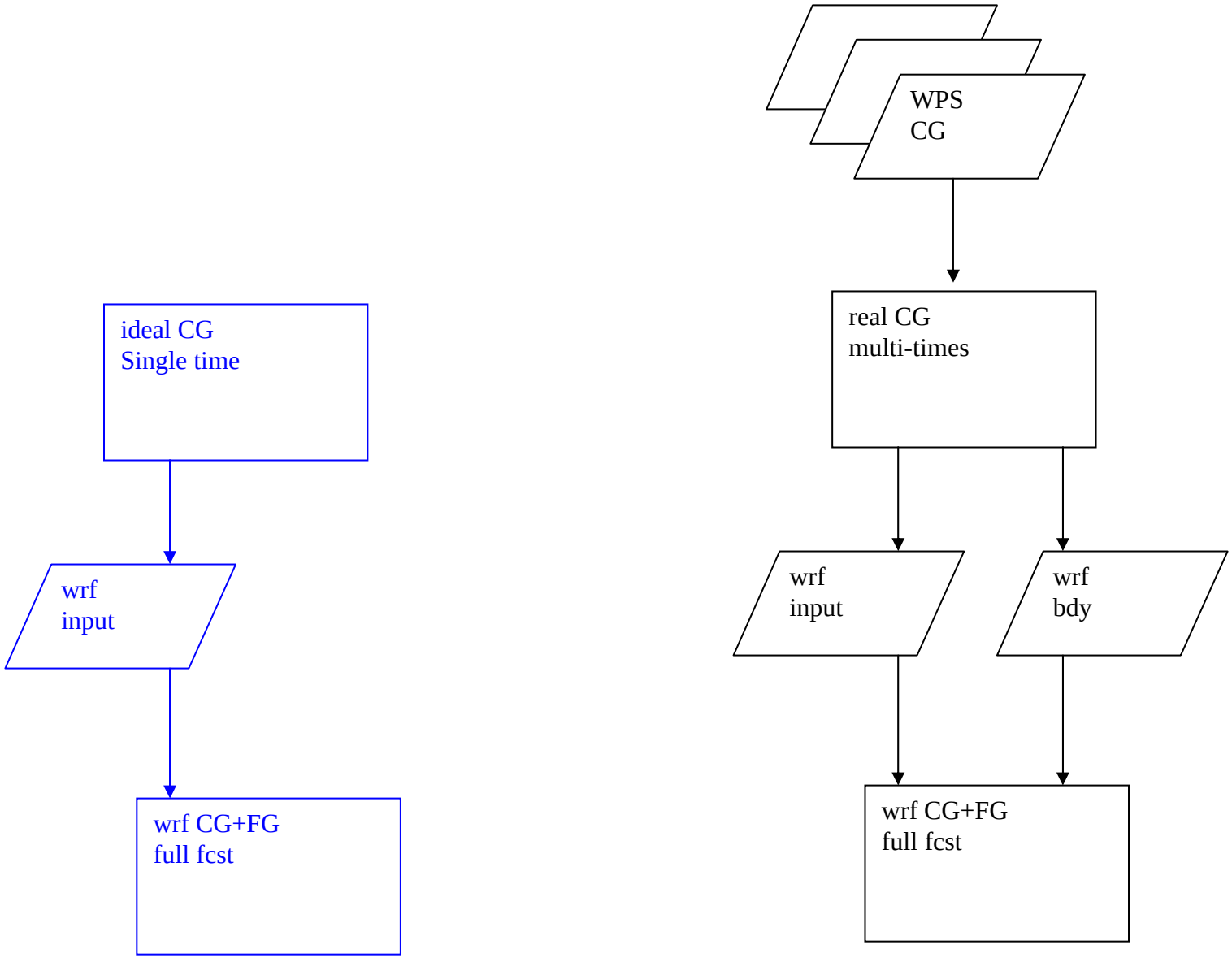
The grid distance ratio is not strictly tied to the time step ratio

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



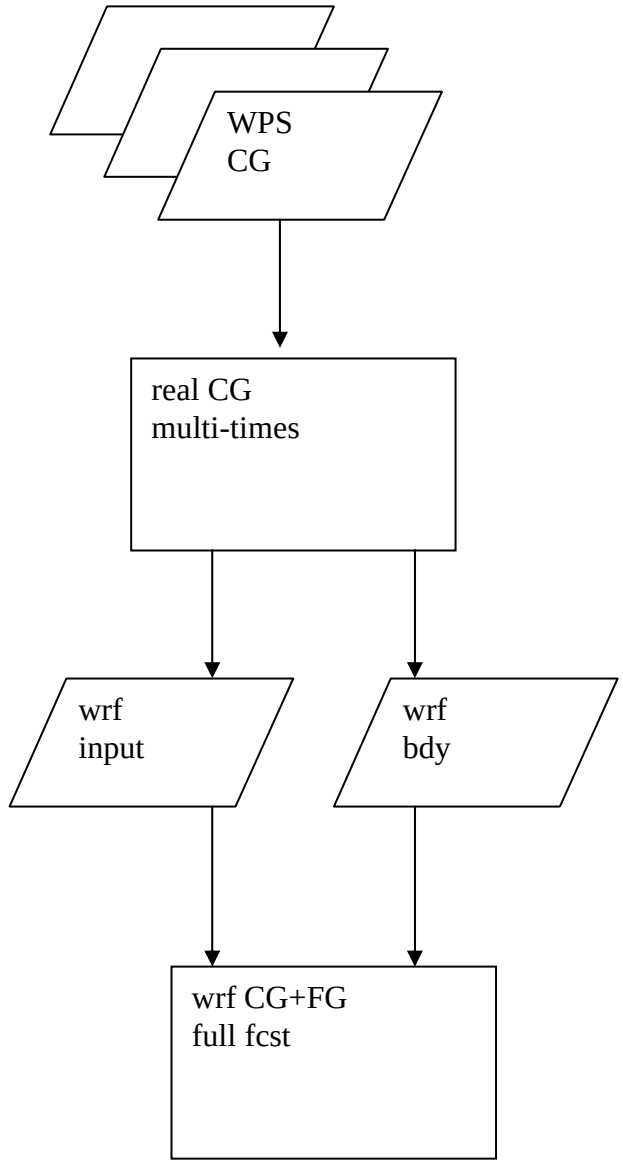
2-Way Nest with 1 Input

A single namelist column entry is tied to each domain

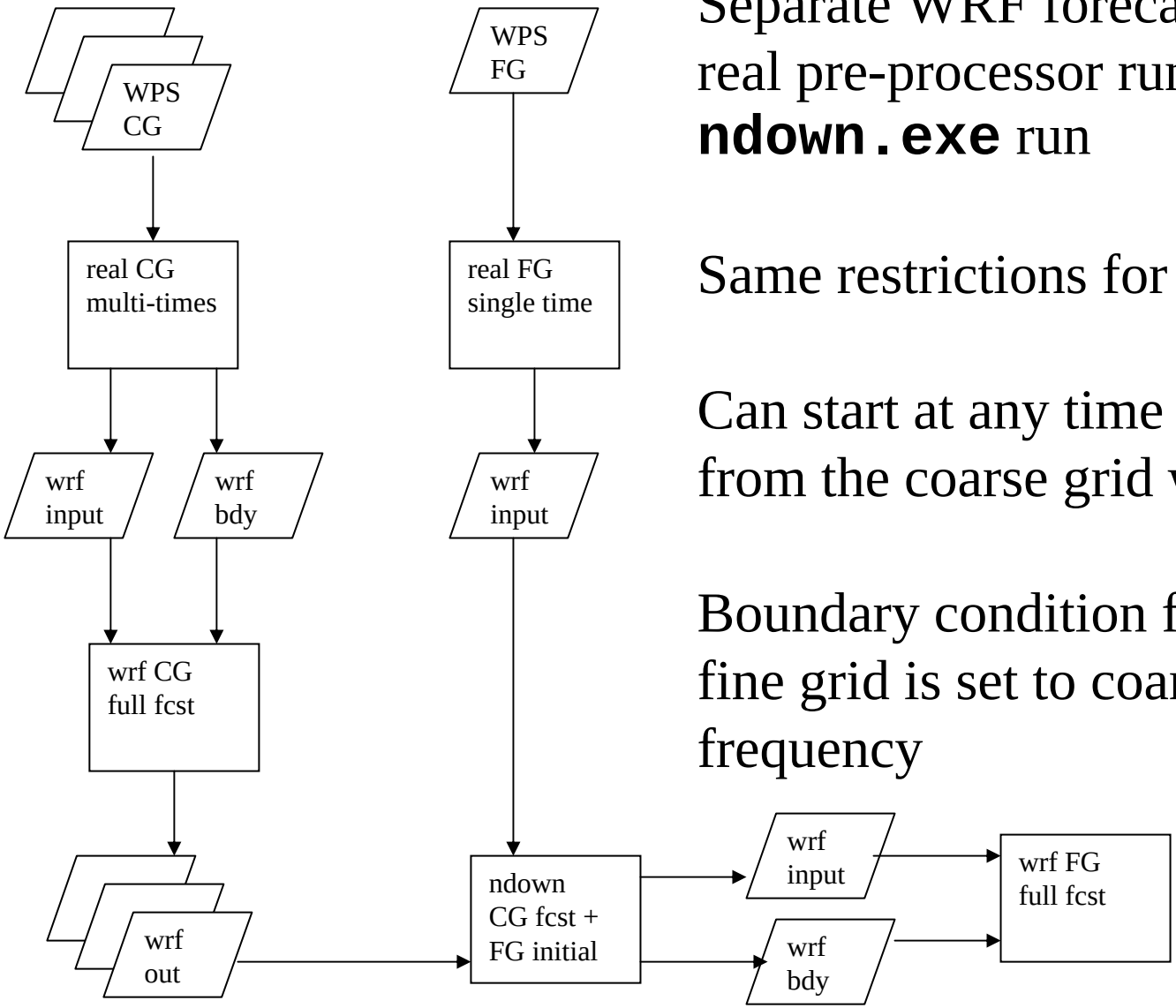
The horizontal interpolation method, feedback, and smoothing are largely controlled through the Registry file

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: **`ideal.exe`**

- The WRF pre-processor **`ideal.exe`** produces a single input file:
  `wrfinput_d01`

Checking Output: **`real.exe`**

- The WRF pre-processor real.exe produces an input file for each domain, and a lateral boundary file for the outer-most grid

  `wrfbdy_d01`

  `wrfinput_d01, wrfinput_d02`

- Optionally, a lower boundary file is generated containing the SST and sea ice for each domain (for long simulations)

  `wrfloinp_d01, wrfloinp_d02`

Checking Output: **`wrf.exe`**

- The WRF model produces an output file for each domain
  `wrfout_d01_2000-01-24_12:00:00`
  `wrfout_d02_2000-01-24_12:00:00`

- For restarts, a restart file is created for each domain, at each requested time
  `wrfrst_d01_2000-01-24_18:00:00`
  `wrfrst_d01_2000-01-24_21:00:00`
  `wrfrst_d02_2000-01-24_18:00:00`
  `wrfrst_d02_2000-01-24_21:00:00`

- Input for 3dvar may be generated for cycling purposes (user defined name), each time, each domain

Checking Output: Simple netCDF commands

- The netCDF data can be easily queried
- Get the header info (domain sizes, list of variables, various metacode)

  `ncdump –h wrfinput_d01`

- All of the data for a single variable

  `ncdump –v U wrfout_d01_2000_01_24-12:00:00`

  `ncdump –v Times wrfout_d01_2000_01_24-12:00:00`

- A portion of a variable

  `ncdump –v U –f f –b f wrfinput_d01 | grep “(1,1,1,1)”`

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 the number of used processors)
- The last line is usually a 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 two namelists: **`namelist.input.jan00`** and **`namelist.input.jun01`**
- Each is associated with a downloadable data set from the WRF download page

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

- Test case for classroom example uses Jan 2000 file