
# MET3220C & MET6480
# Computational Statistics


## Programming – week #8

Dr. Mark Bourassa

TA: David Rock, LOV305A, 644-8577
Office Hours: Tue. 1-2PM & Wed. 2:30-3:30
drock@met.fsu.edu

No *new* assignment this week. Enjoy your vacation.

# IF THEN ELSE ENDIF

- There are many ways of setting up conditionals. For a simple example, we can convert information about humidity to classify conditions as dry, moderate or wet.

```
INTEGER :: rh_condition
REAL: rh
IF ( rh < 0.3 ) THEN
  rh_condition = 0  ! dry
ELSE
  IF ( rh < 0.7 ) THEN
     rh_condition = 1   ! moderate
  ELSE
     rh_condition = 2    ! Wet
  ENDIF
ENDIF
```


# IF THEN ELSE ENDIF


- Alternatively, we can make this logic easier to follow if we take advantage of the ELSE IF structure.

```
INTEGER :: rh_condition
REAL: rh
IF ( rh < 0.3 ) THEN
  rh_condition = 0  ! dry
ELSE IF ( rh < 0.7 ) THEN
   rh_condition = 1   ! moderate
ELSE
   rh_condition = 2    ! Wet
ENDIF
```

- This makes the code somewhat simpler, and a lot easier to read!


# IF THEN ELSE ENDIF & Efficiency


- IF statements are computationally expensive.
- The run time of code can be reduced (sometimes enormously) by cutting down on the number of times IF statements are evaluated.
- To do this, put make the most likely event first in a series of IFs.

```
INTEGER :: rh_condition
REAL: rh
IF ( rh >= 0.3 .AND. rh < 0.7 ) THEN
  rh_condition = 1     ! moderate
ELSE IF ( rh > 0.7 ) THEN
   rh_condition = 2    ! Wet
ELSE
   rh_condition = 0    ! Dry
ENDIF
```


# Manipulating Character Strings


- One of the great uses of STRING manipulation is determining file names on the fly.
- For example, there could be one file for each day of data, or one file for each satellite orbit, or one file for the time of each analysis in an operational weather product.
- Alternative approach: It is pretty easy to write all these names to a file
  - ls /path_to_data_files/* > filenames
  - Would write all the file names in the directory to the file names *filenames*.
  - The *filenames* file could then be opened, and the filenames could be stored in a character array.
- Alternatively, and better for more flexible applications, the filenames for the files of interest could be determined and used in the code.
  - The filename would be a variable of type CHARACTER, and a length appropriate to the application.


# String Manipulation


- Example: Files in one directory, with names of the following format are being opened and read.
  - The format is windsYYYYJJJ.dat
  - Where YYYY is the four digit year, and JJJ is the Julian day.
- The path could be specified in a variable *path*, and the file name would be a character string of length 16 (call this variable *filename*).
- The path and the filename can be spliced together with two slashes //.
  - file = path//filename


# Writing to a Character Variable


- It is possible to write to a character variable in a similar manner to writing to one line in a file.

```
year = 2005
DO i_days = 201, 210   ! loop through 10 days of data
 ! Determine file name of file to open
  if ( i_days .LT. 10 ) then
    write(filename,FMT="('winds',I4,'00',I1,'.dat')") year, i_days
  else if  ( i_days .LT. 100 ) then
    write(filename,FMT="('winds',I4,'0',I2,'.dat')") year, i_days
  else
    write(filename,FMT="('winds',I4,I3,'.dat')") year, i_days
  endif
  file = '/u/a/met3220-02/'// filename

  !open a file for reading
  OPEN (7, STATUS='old', FILE=file)
  !More code here to work with data in the file
ENDDO
```


# Calling UNIX Commands From A Program


- In most programming languages (e.g., FORTRAN, C, IDL) there is a way of having program call UNIX commands.
  - In these languages, the command is SYSTEM.
- Example:

  CALL SYSTEM( "ls /Net/data/winds/*.dat > filenames" )
- It is possible to replace the hard coded character string with a variable.

  command = "ls /Net/data/winds/*.dat > filenames"

  CALL SYSTEM( command )
- Character string manipulation can be used to have the program set the UNIX command.