12.010 Computational Methods of Scientific Programming

Lecturers
Thomas A Herring
Chris Hill

Review of Lecture 5

- Looked at Fortran commands in more detail
  - Control through `if` and `do` statements.
  - Logical expressions in Fortran `.eq. .ne. .gt. .ge. .lt. .le.`
  - Logical expressions `.not. .and. .or.`
  - Looping with `do j = start, end, inc` and `do while` constructions.
  - `Include, common`, and `parameter` statements as methods of communicating between modules
  - `Data` and `Save` statements as methods of initialization and ensuring that modules remember values.

Final topics

- Most of today's class will on trying Fortran program.
- Errors in Fortran programs can occur 3 places:
- Compile errors:
  - Nature of error is reported and the line number in the source code with the problem. It is best to work from the first to last error. In some cases, an early error can generate many later errors (so try to fix the major problems—often other compile errors disappear once these are fixed).
  - Forgetting to end `do` loops and `if` statements will often generate many errors.

Errors 02

- Linking errors:
  - Most common here is `undefined external` which means you seems to call a `subroutine` or `function` and the linker can not find the routine you are referring to.
  - In some machines, you might get errors about `real*8` variables not being on even byte boundaries. Usually only a problem when you are doing "tricky" Fortran code.

Errors 03

- Runtime errors:
  - IOSTAT errors associated with file reading and writing.
  - NaN - Not a number, generated when illegal operations are performed (e.g., sqrt(negative number) not stored to a complex variable
  - Inf - Infinite result (usually divide by zero or tan(π/2)
  - Segmentation violation/Bus error. These are worst types of errors because where the program stops may not be related at all to where the error is. Common causes are:
    - Incorrect calling arguments and subroutine/function definition
    - Exceeding the bounds of an array

Errors 04

- With optimized code (`-On` where n=0—5) can generate very strange errors which are not your fault (although the cleaner the code the less likely this is to occur)
- Try compiling with out optimization (this does not say that you don't have a bug since things move around in memory with different optimization.

Final comments

- Other useful utilities (on Unix systems)
- `size` — program to tell you the size of a program e.g.
  ```
  % size poly_area
  ```
- `make` — Used to organize large program by specification of dependency of modules (e.g., object modules depend of source code and include files, programs depend of libraries and object modules)
  - `make` checks the dependencies and re-compiles and links only those things that depend of things that have changed.
- `ar` and `ranlib` — create libraries and indexes them for faster loading.

Remainder of class

- Remainder of class we develop and debug fortran programs. Programs will be written in class to
  - Compute root-mean-square scatter (RMS) of random numbers generated with the intrinsic rand function
- Students with laptops might want to bring them along so that can work on their own system or on athena. There is wireless internet in the room.

RMS calculation

- Root mean square (RMS) scatter is computed by:

$$RMS = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{(n-1)}} \qquad \bar{x} = \sum_{i=1}^{n} x_i / n$$

$$RMS = \sqrt{\frac{\sum_{i=1}^{n} (x_i)^2 - n\bar{x}^2}{(n-1)}}$$

- Notice that in the second form, the values do not need to be saved.

12.010 Computational Methods of Scientific Programming
Fall 2011