Exercises

Day 3

Exercise #3a

- Write a small main program to create and initialize two large arrays and call a subroutine that swaps the two arrays.
- Choose an array size that exceeds the stack size (in linux check using: ulimit -a).
- Pass the arrays using assume shape (:-type notation) and an explicit interface.
- Declare the swap array as:
  - Automatic array.
  - POINTER and ALLOCATE/DEALLOCATE.
- Compile with different compilers and check if the code crashes.
  - For gfortran use:

    gfortran -ffree-form -fstack-arrays -frecursive program.f

    to ensure the arrays are forced to be on the stack (old versions of gfortran may not have this option; so make sure you use a recent version).

Exercise #3b

- Extend the program written in exercises #1+2:
  - Split the entire code into separate files – one file for each subprogram (functions, subroutines, and modules). Create a Makefile (e.g., copy the example on ~jhwa/Makefile and adjust it) for your source; recompile (make new) and rerun the simulation.
  - Create a subroutine with dummy arguments for allocating the field array. Overload the routine for single and double precision (example: *CALL alloc(field,Nx,Ny,info)*). Allow the routine to preserve any existing data (in case the field already has been allocated; this is not important for this specific application – but in general it is).
  - If you have a copy routine (from exercise #2b) then replace it with an ELEMENTAL subroutine. If you do not have a copy routine, then create a small separate program for swapping two arrays to test ELEMENTAL subroutines.