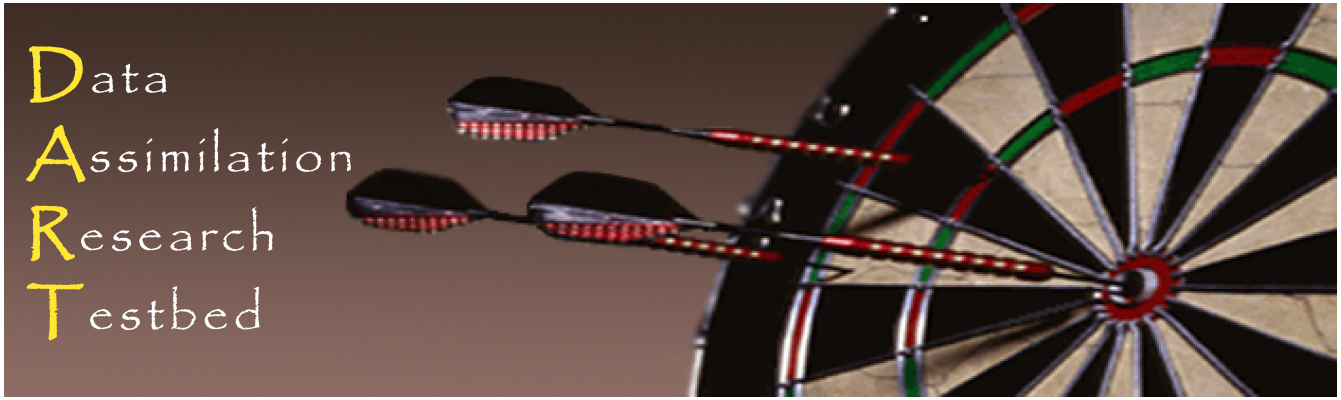
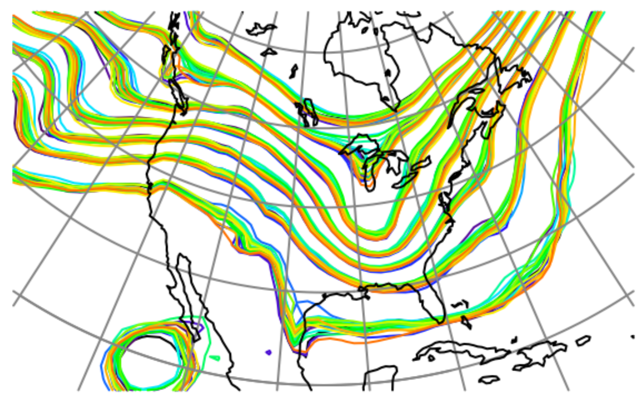


DART Tutorial Section 11: Creating DART Executables



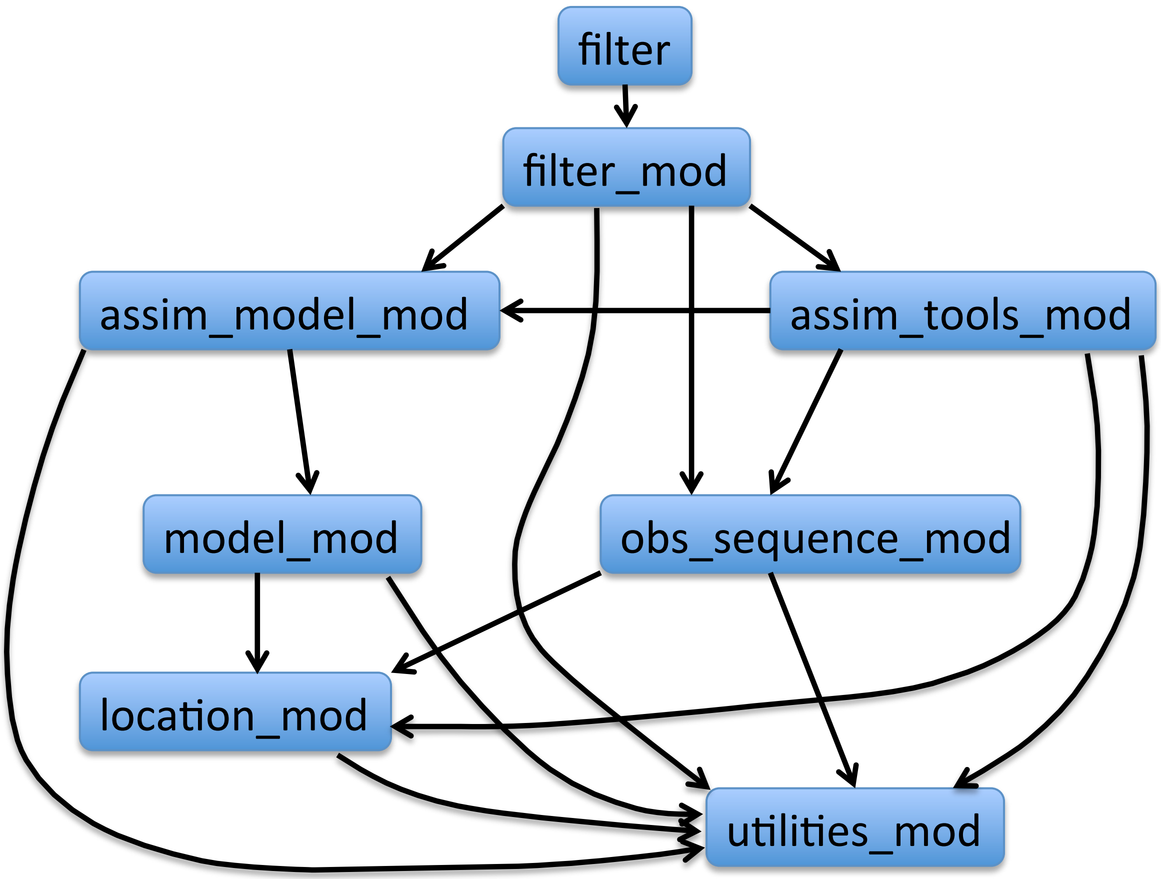
The National Center for Atmospheric Research is sponsored by the National Science Foundation. Any opinions, findings and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

NCAR | UCAR | National Center for Atmospheric Research

Fortran 90 'Use' Trees

DART requires use of F90 *use module_mod, only:*

No other mechanism for use of 'external' routine.



Program viewed as Directed Acyclic Graph (tree with shared leaves)

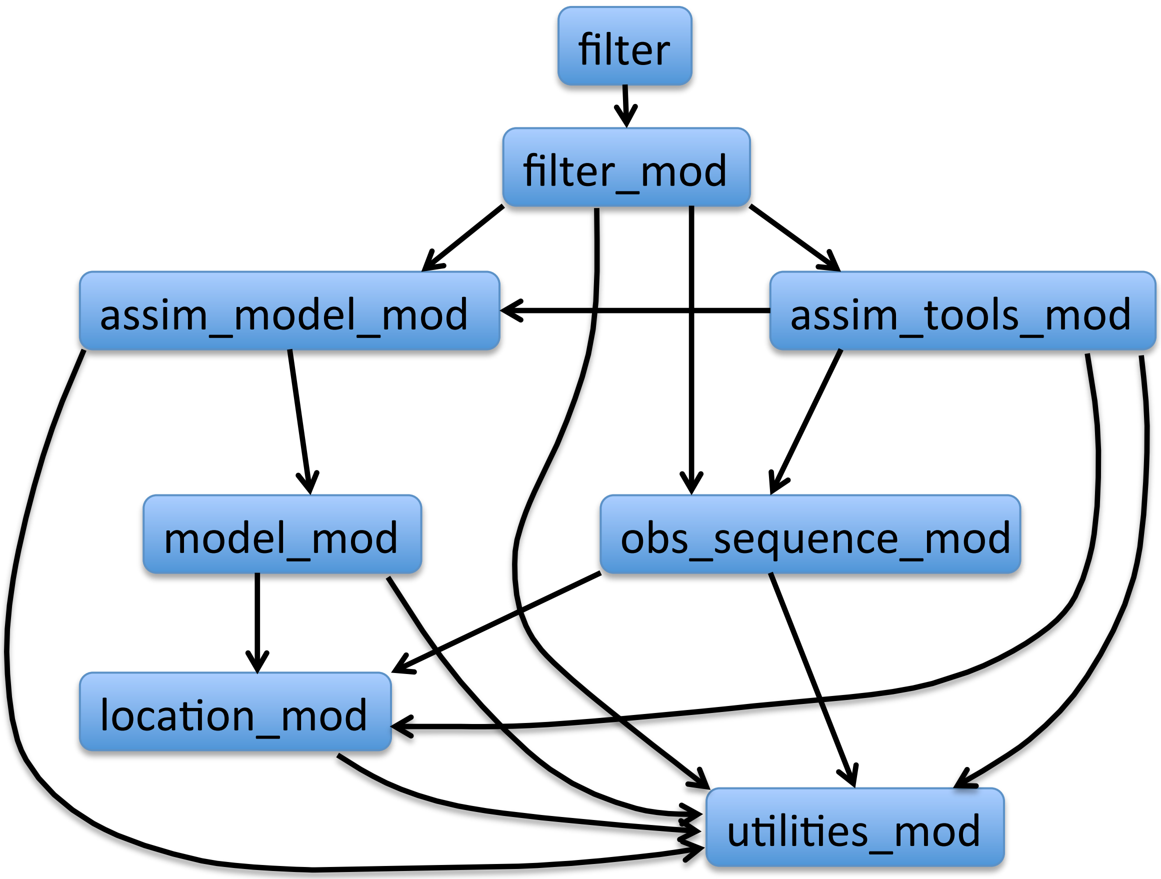
No cycles!

Partial tree for program filter is shown.

Vector represents use of target module.

Generating a Makefile with mkmf

DART requires use of F90 *use module_mod, only:*



Use ***mkmf*** Perl script to make a *Makefile*

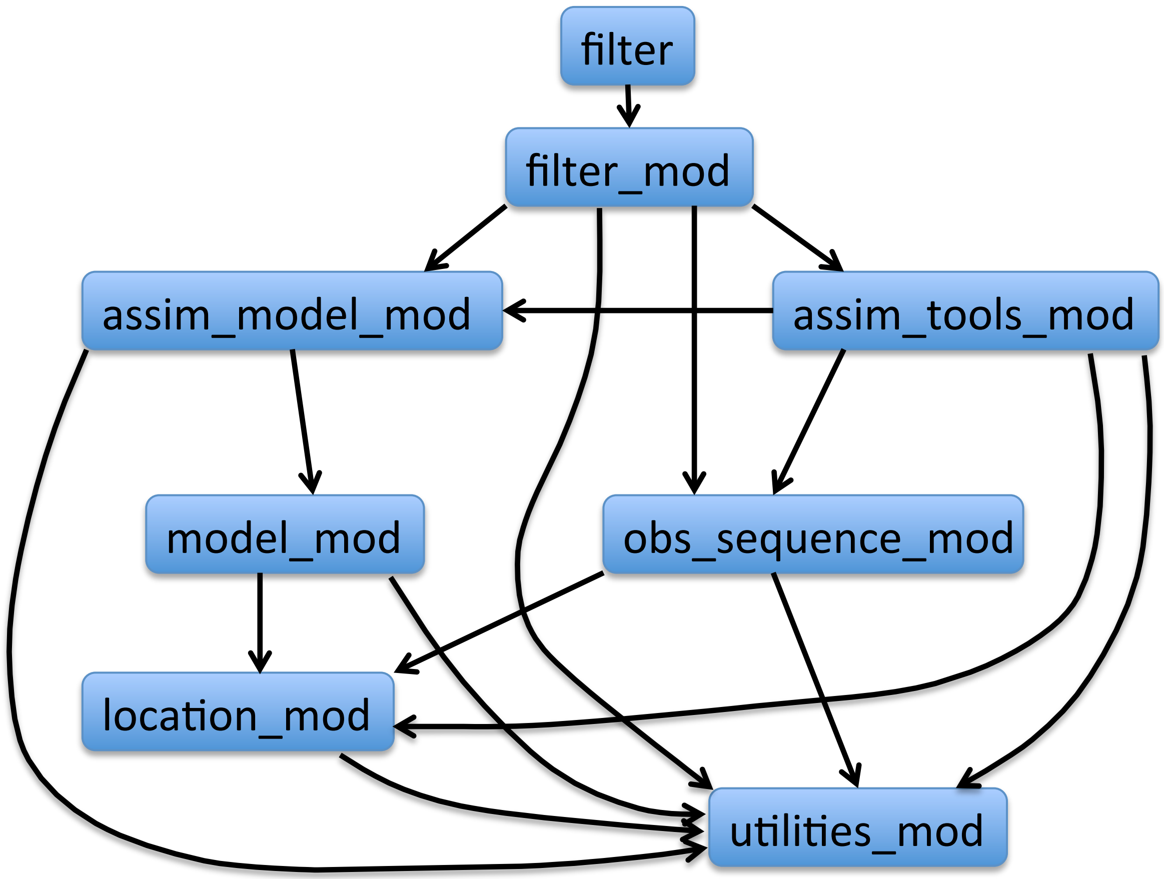
mkmf requires a list of files to search for the main program and modules that are used.

This is called a *path_names* file.

See *path_names_filter* in *models/lorenz_63/work.*

Generating a Makefile with mkmf

DART requires use of F90 *use module_mod, only:*



mkmf searches files in path_names for one that contains *program filter.*

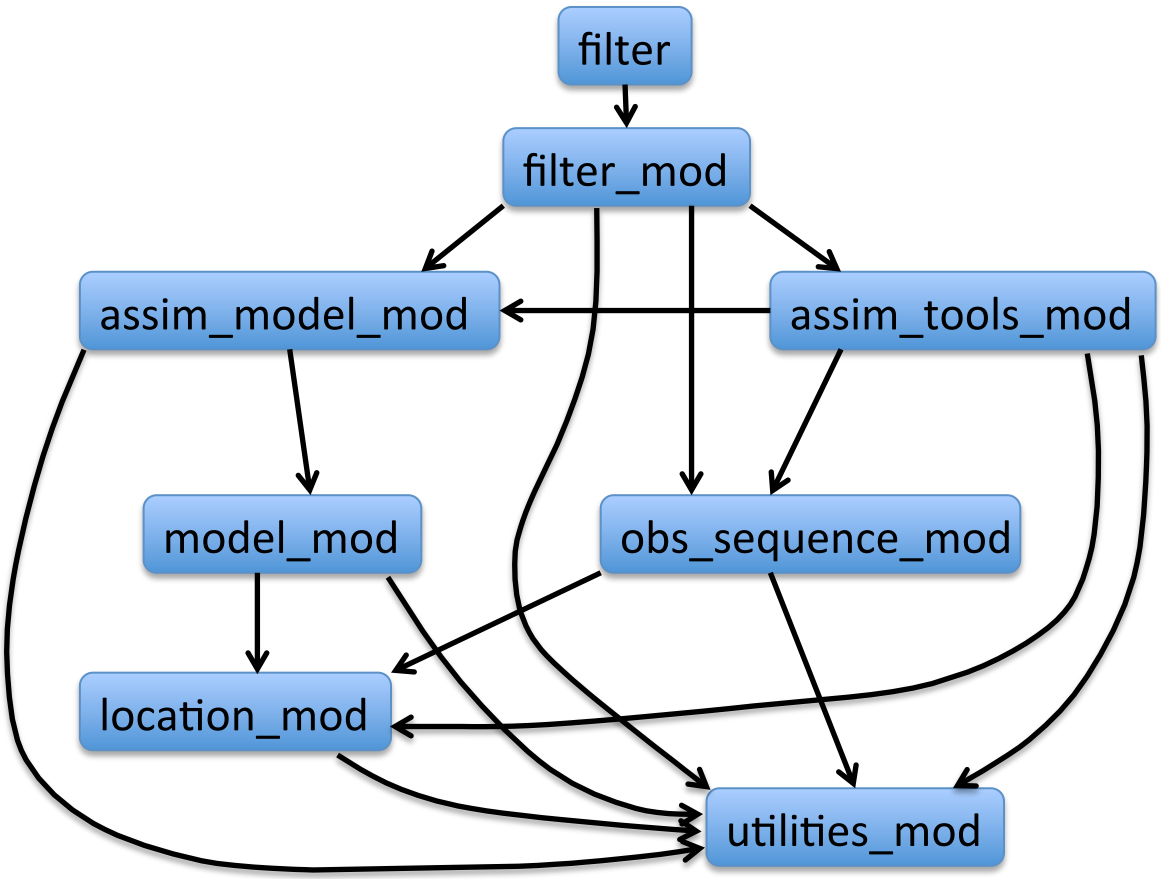
Finds first *use only* in filter.

Searches path_names files for this module recursively.

Builds a dependency graph like one at left.

Generating a Makefile with mkmf

DART requires use of F90 *use module_mod, only:*



From dependency graph, ***mkmf*** generates a stan-dard *Makefile*.

Also creates a default namelist file, *input.nml.filter_default.*

Examine to see namelists required, default values.

Enter ***make*** to create filter executable.

mkmf Details

Each DART program has *mkmf_* and *path_names_* files.
Can see a selection of these in *models/lorenz_63/work*.

Let's look at *mkmf_perfect_model_obs* first.

File *mkmf_perfect_model_obs* starts with scripting that controls whether code is built with mpi for parallel runs.

Key line starts with "../../../build" and contains (default values in parentheses):

1. Relative location of mkmf program (../../../build_templates/mkmf)
2. Name of executable program to create (-p perfect_model_obs)
3. Compiler options file (-t ../../../build_templates/mkmf.template)
4. Relative base location for file search (-a "../../.."),
5. Name of the path_names file (path_names_perfect_model_obs).

mkmf Details

A variety of mkmf templates for different machines and compilers are available in the directory *build_templates*
(You can also see the Perl script ***mkmf*** there).

mkmf templates specify:
What compiler to use,
Where to find the netCDF libraries,
Compile and link command line options.

Templates usually include comments about useful compiler and linker options for both debugging and production.

To change your template:
Copy the appropriate *mkmf.template.xxx* file to *mkmf.template* and make any needed changes.

mkmf Details

Two versions of the MPI module are available; one which really calls MPI and one with dummy routines.

The small models default to building a serial program.
The large models default to building an MPI parallel version.

To run ***mkmf*** for filter, execute shell script *mkmf_filter:*

./mkmf_filter OR ***csh mkmf_filter***

DART Modular Philosophy

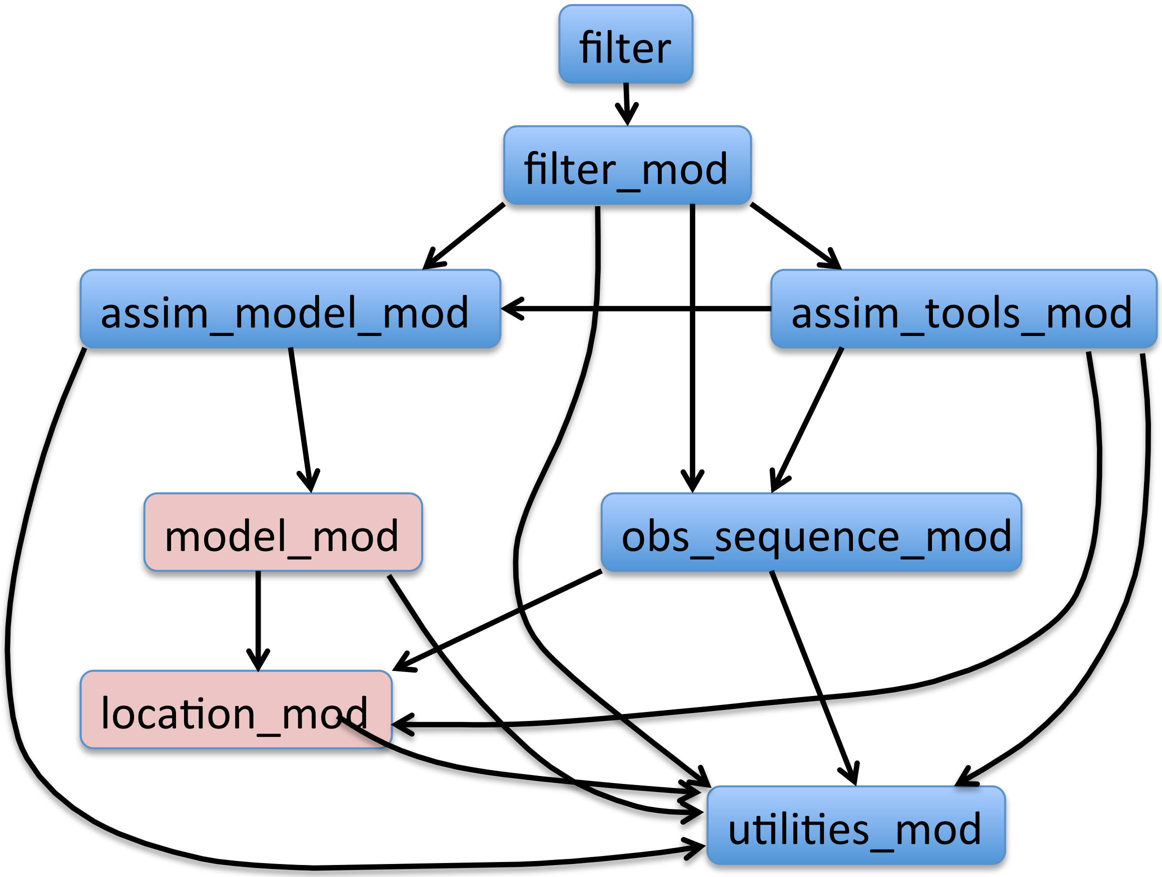
Can swap modules with same names and public interfaces.

Changing the following paths in *path_names_filter*:

models/lorenz_63/model_mod.f90 ⟹ models/cam-fv/model_mod.f90

assimilation_code/location/oned/location_mod.f90 ⟹

assimilation_code/location/threed_sphere/location_mod.f90



switches from Lorenz 63 to CAM-FV GCM!

Modules with multiple implementations have second directory level (see paths above).

Compare path_names files in *models/lorenz_63* with *models/cam-fv*

Exercise: Compiling lorenz_63 filter program

models/lorenz_63/work/

1. Go to *models/lorenz_63/work*

2. Remove all files with .o and .mod extensions.

3. Generate a *Makefile* and *input.nml.filter_default* by:
   csh mkmf_filter

4. Generate program filter:
   Enter ***make***

DART Tutorial Index to Sections

1. Filtering For a One Variable System
2. The DART Directory Tree
3. DART Runtime Control and Documentation
4. How should observations of a state variable impact an unobserved state variable? Multivariate assimilation.
5. Comprehensive Filtering Theory: Non-Identity Observations and the Joint Phase Space
6. Other Updates for An Observed Variable
7. Some Additional Low-Order Models
8. Dealing with Sampling Error
9. More on Dealing with Error; Inflation
10. Regression and Nonlinear Effects
11. Creating DART Executables
12. **Adaptive Inflation**
13. **Hierarchical Group Filters and Localization**
14. **Quality Control**
15. **DART Experiments: Control and Design**
16. **Diagnostic Output**
17. **Creating Observation Sequences**
18. **Lost in Phase Space: The Challenge of Not Knowing the Truth**
19. **DART-Compliant Models and Making Models Compliant**
20. **Model Parameter Estimation**
21. **Observation Types and Observing System Design**
22. **Parallel Algorithm Implementation**
23. Location module design (not available)
24. Fixed lag smoother (not available)
25. **A simple 1D advection model: Tracer Data Assimilation**