Make and Makefiles

Makefile Disclaimer

This course will give a brief overview of how to use make with Fortran

Will cover the basics only!

Then look at how modules complicate the use of make

What is Make?

Make is a tool which controls the generation of executables from a program's source files

It gets its knowledge of how to build your program from a file called the makefile

The compilation procedure is much faster!

- The compilation is done with a single command
- Only files that have been modified are recompiled
- Allows managing large programs with lots of dependencies

Makefile Basics (1)

A rule in the makefile tells Make how to execute a series of commands in order to build a target file from source files

It also specifies a list of dependencies of the target file

Here is what a simple rule looks like:

```
target : dependencies ... (also called prerequisites)
    <tab> commands
```

The <tab> is absolutely necessary!

Makefile Basics (2)

Make uses timestamps to locate the files that have been modified since the last time make was executed

By default when you type make it looks for the file makefile or Makefile. You can designate a specific name with make -f <thismakefile>

Can also use macros to give names to variables within the makefile. NOTE these are case-sensitive!

If no specific target is given in the make command then Make starts with the first target listed in the makefile

Let's start with a very simple example (**example1**)

Makefile Basics (3)

Comments are delimited by the # symbol

A backslash \ can be used as a continuation character

Common extra tidbit: Create a "phony target" called clean which can be run to do a fresh recompile of all source code

Makefile Automatic Variables

These can only be values in the recipe. They cannot be used in the target list of a rule

$<	The name of the first prerequisite
$^	The names of the all prerequisites
$@	The file name of the target of the rule

And there are even more available

Compiling Modules

When modules are compiled both a .o and .mod file are created

A .mod file is like a compiled header. This is what the compiler searches for when it sees a USE statement

The dependencies can start to get cumbersome and complicated when many modules are USED and inherited

Make has no method for determining these for you.

Take a look at **example2**

Compiling Modules (2)

If you edit a module but do not change the interface then there's no need to update the .mod file.

But this is compiler specific behavior:

- gfortran has been updated to handle this
- ifort always updates both the .o and .mod files

There are some software build tools that try to handle this complexities to try to reduce "cascading compilation".

Want it to compile fast, but really we want it correct!

Helpful Tools

mkDepends - generate a list of dependencies

mkSrcfiles - generate a list of all source files

Versions of these perl scripts are used in atmospheric models like SAM and CAM

mkdep - requires both GNU make and Python

fmfmk.pl - generate a makefile

foraytool - made especially for compiling large Fortran codes