Makefiles

Make is a common tool used by programmers to automate tasks such as compiling code. Make is configured by writing down a set of rules that specify how target files are to be built from lists of dependencies. The rules are stored inside a `Makefile` that lives next to the program source code. This file is then processed by the `make` shell command. A basic Makefile provides two important benefits: a productivity boost by eliminating the need to re-type complicated commands and protection from typos that can cause critical files to be overwritten. As projects grow more complicated, Makefiles provide additional benefits by ensuring components are assembled in the correct order and that no file is re-compiled unless the file has been modified more recently than a target that depends on it.

A simple template for a Makefile has the following form:

Makefile Template

```
# All make commands are saved into a file named
# Makefile that is saved next to the project
# source code.
#
# Makefile comments start with a hash sign: '#'
#
# Variables may be assigned values for quick
# reference later. Values are retrieved from
# variables by wrapping the variable
# name inside '$()'.
LDFLAGS = <library search directories go here>

# The append operator '+=' is shorthand for
#     VAR = $(VAR) <stuff to append>
LDFLAGS += <library names go here>
FCFLAGS = <gfortran compiler flags go here>
FCFLAGS += <module search directories go here>

# A "make target" is a file, such as an executable
# program. The shell commands required to build
# the target follow in a _tab indented_ list.
<target name>: <list of files or target names>
        # '$@' is shorthand for the target name
        # and '$^' is shorthand for the list of
        # dependencies.
  Tab→ gfortran $(FCFLAGS) $^ -o $@ $(LDFLAGS)
```

Simple Makefile

```
LDFLAGS =
LDFLAGS +=
FCFLAGS = -Wall -fbounds-check
FCFLAGS +=

pipe_flow: pipe_flow.f90
  Tab→ gfortran $(FCFLAGS) $^ -o $@ $(LDFLAGS)
```

Makefile with Multiple Source Files

```
LDFLAGS =
LDFLAGS +=
FCFLAGS = -Wall -fbounds-check
FCFLAGS +=

pipe_flow: types.f90 integration_subs.f90 pipe_main.f90
  Tab→ gfortran $(FCFLAGS) $^ -o $@ $(LDFLAGS)
```

Makefile with Module Targets

```
LDFLAGS =
LDFLAGS +=
FCFLAGS = -Wall -fbounds-check
FCFLAGS +=

pipe_flow: pipe_main.f90 integration_subs.o types.o
  Tab→ gfortran $(FCFLAGS) $^ -o $@ $(LDFLAGS)

integration_subs.o: integration_subs.f90 types.o
        # '$<' is shorthand for the first dependency
        # in the list. This is used instead of '$^'
        # because only the '.f90' file needs to
        # be compiled.
  Tab→ gfortran $(FCFLAGS) -c $<

types.o: types.f90
  Tab→ gfortran $(FCFLAGS) -c $<
```

Makefile with External Libraries

```
LDFLAGS = -L$(HOME)/lib
LDFLAGS += -lfintegrate
FCFLAGS = -Wall -fbounds-check
FCFLAGS += -J$(HOME)/modules

pipe_flow: pipe_main.f90
  Tab→ gfortran $(FCFLAGS) $^ -o $@ $(LDFLAGS)
```