Makefiles

Dr. Hyrum Carroll

September 22, 2016

Exercise

Write down the major steps to make a (Thanksgiving) dinner with two to three items

Makefiles

`make`

- A Unix system utility to aid in compiling complex codes
- If setup properly, only compiles the necessary parts
- Not limited to compiling code
- Searches for file named “Makefile” in your current directory

Makefile Rules

Rules

```
target ... : prerequisites ...
        recipe
```

Target:
What to generate if it doesn't exist or is older than `prerequisites` (*e.g.* object files, executables)
Prerequisites:
Items which need to be generated before executing the recipe
Recipe:
Instructions that `make` executes (must have a TAB character before)

Basic Makefile

```
1  # VARIABLES = definition
2  F90 = gfortran
3  TARGET = myProg
4
5  # definitions for compiling
6
7  # compile any .f90 file into a .o file
8  .f90.o:
9      $(F90) -c $<
10
11 # rules
12 # making $(TARGET) depends on obj{1,2,3}
13 # Execute recipe after generating those files
14 $(TARGET):  obj1.o obj2.o obj3.o
15     $(F90)  -o $(TARGET) obj1.o obj2.o obj3.o -llapack
```

Compiling

Object and Executable Files

Up until now, the compiler has really been doing two things:

1. Compiling code into an **object** file
2. Linking object files into an **executable** file

Example:
helloWorld.f90:

```
1 program helloWorld
2 print*,'Hello World!'
3 end program helloWorld
```

```
$ gfortran -c helloWorld.f90 -o helloWorld.o
$ gfortran -o helloWorld helloWorld.o
```

“Hello World” Makefile

helloWorld.f90:

```
1 program helloWorld
2 print*,'Hello World!'
3 end program helloWorld
```

Makefile:

```
1 helloWorld: helloWorld.o
2     gfortran -o helloWorld helloWorld.o
3
4 helloWorld.o: helloWorld.f90
5     gfortran -c helloWorld.f90 -o helloWorld.o
```

“Hello World” Makefile

Makefile:

```
1 helloWorld: helloWorld.o
2     gfortran -o helloWorld helloWorld.o
3
4 helloWorld.o: helloWorld.f90
5     gfortran -c helloWorld.f90 -o helloWorld.o
```

```
$ make
gfortran -c helloWorld.f90 -o helloWorld.o
gfortran -o helloWorld helloWorld.o
$ make
make: ‘helloWorld’ is up to date.
```

"External Call" Makefile

externalCall.f90:

```
1 program externalCall
2
3 print*,'Main program'
4 call aux('externalCall')
5
6 end program externalCall
```

aux.f90:

```
1 subroutine aux(who)
2   character(len=*) :: who
3
4   print*,'aux(',who,')'
5
6 end subroutine aux
```

Makefile:

```
1 externalCall: externalCall.o aux.o
2     gfortran -o externalCall externalCall.o aux.o
3
4 aux.o: aux.f90
5     gfortran -c aux.f90
6
7 externalCall.o: externalCall.f90
8     gfortran -c externalCall.f90
```

"External Call" Makefile

Makefile:

```
externalCall:  externalCall.o  aux.o
    gfortran -o externalCall externalCall.o aux.o

aux.o: aux.f90
    gfortran -c aux.f90

externalCall.o: externalCall.f90
    gfortran -c externalCall.f90
```

```
$ make
gfortran -c aux.f90
gfortran -c externalCall.f90
gfortran -o externalCall externalCall.o aux.o
$ ./externalCall
 Main program
 aux(externalCall)
```

"Module Call" Makefile

moduleCall.f90:

```
1 program moduleCall
2 use mod
3
4 print*,'moduleCall: i=',i
5
6 end program moduleCall
```

mod.f90:

```
1 module mod
2
3 integer :: i=4
4
5 end module mod
```

Makefile:

```
1 moduleCall: moduleCall.o mod.o
2     gfortran -o moduleCall moduleCall.o mod.o
3
4 mod.o: mod.f90
5     gfortran -c mod.f90
6
7 moduleCall.o: moduleCall.f90 mod.o
8     gfortran -c moduleCall.f90
```

(Notice `mod.o` on line 7)

"Module Call" Makefile

Makefile:

```
moduleCall: moduleCall.o mod.o
    gfortran -o moduleCall moduleCall.o mod.o

mod.o: mod.f90
    gfortran -c mod.f90

moduleCall.o: moduleCall.f90 mod.o
    gfortran -c moduleCall.f90
```

```
$ make
gfortran -c mod.f90
gfortran -c moduleCall.f90
gfortran -o moduleCall moduleCall.o mod.o
$ ./moduleCall
 Main program, i =                4
```

Makefiles

Makefile2:

```
1  TARGET = moduleCall
2  OBJECTS = $(TARGET).o mod.o
3
4  $(TARGET): $(OBJECTS)
5      gfortran -o $(TARGET) $(OBJECTS)
6
7  mod.o: mod.f90
8      gfortran -c mod.f90
9
10 $(TARGET).o: $(TARGET).f90 mod.o
11     gfortran -c $(TARGET).f90
12
13 .PHONY : clean
14 clean:
15     rm -f $(TARGET) $(OBJECTS)
```

```
$ make -f Makefile2 clean
rm -f moduleCall moduleCall.o mod.o
```

Makefiles: Variables

Makefile3:

```
1  TARGET = moduleCall
2  OBJECTS = $(TARGET).o mod.o
3
4  # $@ is the file name of the target of the rule
5  $(TARGET): $(OBJECTS)
6      gfortran -o $@ $(OBJECTS)
7
8  # $< is the first prerequisite
9  mod.o: mod.f90
10     gfortran -c $< -o $@
11
12 $(TARGET).o: $(TARGET).f90 mod.o
13     gfortran -c $< -o $@
14
15 .PHONY : clean
16 clean:
17     rm -f $(TARGET) $(OBJECTS)
```

Equivalent to Makefile2

Makefiles: Flags (*e.g.*, debugging)

Makefile4:

```
1  TARGET = moduleCall.debug
2  OBJECTS = moduleCall.o mod.o
3  FC = gfortran
4  FCFLAGS = -g -fbounds-check
5
6  $(TARGET): $(OBJECTS)
7      $(FC) $(FCFLAGS) -o $(TARGET) $(OBJECTS)
8
9  mod.o: mod.f90
10     $(FC) $(FCFLAGS) -c mod.f90 -o mod.o
11
12 moduleCall.o: moduleCall.f90 mod.o
13     $(FC) $(FCFLAGS) -c moduleCall.f90
14
15 .PHONY : clean
16 clean:
17     rm -f $(TARGET) $(OBJECTS)
```

Equivalent to Makefile2

Generic Fortran Makefile

`http://www.cs.mtsu.edu/~hcarroll/6100/lectures/makefiles/makeData/Makefile`

Makefiles Exercise

Convert your (Thanksgiving) meal planning steps into a `Makefile`. Execute it!