Make: How Does It Work?

- Automates program compilation
- Series of inter-dependent build rules to compile a program
- Builds only what it needs to
- Name it "`Makefile`" or use `make -f`

Rule Syntax

- A build rule has a target, prerequisites and a series of tab-indented shell commands:

```
target: prereq1 prereq2 ...
        cc -o target prereq1 prereq2
```

- Chain rules together by writing rules to build those prerequisites

Variables

- Case-sensitive
- Setting: `VAR = value`
- Reference: `$(VAR)`
- Environment variables automatically accessible in a makefile

Automatic Variables

- $@ - name of target
- $< - first prerequisite
- $^ - all prerequisites, space-separated
- $* - the ‘stem’ of a file name in a suffix rule

Common Variables

- `FC, CC` - Fortran and C compilers
- `FFLAGS, CFLAGS` - Fortran/C compile flags
- `LFLAGS` - link flags; language-agnostic
- `OBJS` - list of object files to link

Suffix Rules

- Files with similar extensions often compiled the same way; these let you stop repeating yourself
- Fortran example:

```
.f.o:
        $(FC) $(FFLAGS) -o $*.o -c $*.f
```

Suffix Rules (cont.)

- Extensive list of default suffixes and rules
- Override the defaults (no .mod problem):

```
.SUFFIXES:
.SUFFIXES: .f .F .o
```

Common Rules

- all - builds all relevant targets, e.g. model + utilities; should be first rule in file
- clean: cleans up programs, object files, etc. to prepare for a rebuild
- `.phony: all clean`
- Now I can "`make clean all`"

Other Syntax

- Comments start with #
- Continue long lines with “ \”

```
# Build these objects
OBJS = rsetgen.o BTree.o Stack.o \
    xmalloc.o
```

- `@` suppresses command echoing

Recursive Make

- Modularize your build process into multiple Makefiles
- Run make with `$(MAKE)`:

```
model_library:
        cd library
        $(MAKE) all
```

Extra Dependencies

- Specify "non-obvious" dependencies like modules that live in other Fortran files
- Ex: makesfc.f90 uses the module in types.f90, add the rule:

```
makesfc.f90: types.o
```

Includes

- Include another file with “`include filename`”

Conditions

```
HOST = $(shell hostname)

ifeq ($(HOST),bamboo)
    FC = gfortran
else ifeq ($(HOST),ilex)
    FC = pgf90
else
    FC = f77
endif
```

Bringing It All Together

- Example Makefile