

## 9. MAKE AND MM5

Dave Gill  
gill@ucar.edu

### 9.1 make and MM5

- Why use make with the MM5 system?
- MM5: 556 subroutine files, 135 C files, 91 makefiles, 74 directories, and zillions of include files: organizational and administrative necessity
- (P.S. 875 GOTO statements!)
- Nested tree structure of Makefiles mimics hierarchical directory structure (build everything from here down)

NCAR/MM5

### 9.1 make and MM5

- Hierarchical system of Makefiles allows recursive builds from top-level: make goes to other directories and issues other make commands
- Portability concerns: more than source code  
→ compiler/loader options, libraries, idiosyncracies, single/threaded/distributed

NCAR/MM5

### 9.1 make and MM5

- MM5 supported on a dozen different architectures, each with single-processor/OpenMP options, and MPI capabilities  
→ handled through an include file to all three levels of the MM5 Makefiles
- Makefiles only use minimal set of capabilities permitted to ensure that options do not exclude port to other architectures – particularly now with flavors of Linux running on several chip sets with multiple compiler choices

NCAR/MM5

### 9.1 make and MM5

- MM5 has several dozen physical parameterizations, many of which are mutually exclusive → no need to compile unnecessary files
- Combination of include files, make and CPP provides the conditional compilation capability

NCAR/MM5

### 9.1 make and MM5

- Conditional compilation: removes sections of Fortran from source, and skips entire directories

NCAR/MM5

## 9.2 configure.user File

- Included into each of the 91 Makefiles
- Single point: default rules, suffixes, compiler/loader options, CPP directives, maximum domain sizes, compilable physics options
- Since it does everything: confuser.user

NCAR/MM

## 9.2 configure.user File

- What's in configure.user?
- Compiler/loader options, library choices, parallelization, optimization (paralyzation), debugging, statically allocated space for grid sizes, domain numbers, and physics
- Macros that subsequent Makefiles inherit
- Suffixes and rules
- Specific complex scheme chosen for activating options and choices: uncommenting (RELAX)

NCAR/MM

## 9.2 configure.user File

- Mostly SYDGC
- Sample of rules:

F.F:

```
<tab> $(RM) $@
<tab> $(CPP) $(CPPFLAGS) $*.F > $@
```

NCAR/MM

## 9.3 Makefiles

- MM5 uses a three-tiered Makefile structure
- Top-level: for target (all, code, clean), go into main directories (memory, fdca, domain, physics, dynamics), responsible for mmlif and mm5.deck, and MPI installations

NCAR/MM

## 9.3 Makefiles

- Middle-level: branching into specific directories, such as for selected physics options chosen in configure.user (this is the structure that permits conditional compilation of entire directories)
- Modifications are required if adding new schemes to existing genres/suites

NCAR/MM

## 9.3 Makefiles

- Low-level: compilation of everything in the directory
- When adding files, low-level Makefiles are directly modified

NCAR/MM

### 9.3.1 Top-level Makefile

- Things to note: include and targets
- `include ./configure.user` similar to `#include`
- `all`: first target
- `mm5.deck`: `makedeck.csh` executed
- `mm1if`: you can make a text file, too

NCAR/MM

### 9.3.2 Mid-level Makefile

- Down one more level in directory structure, reflected in include and `DEVTOP`

```
DEVTOP = ../..
include .././configure.user
```

- `lib` is the first target, default

NCAR/MM

### 9.3.2 Mid-level Makefile

- `IBLTYP` macro set in `configure.user`, 8-40

```
IBLTYP = "5,5,2,0,0,0,0,0,0"
```

- Macro is expanded in the mid-level Makefile
- Return code 0 is "successfully found" in `grep`

NCAR/MM

### 9.3.2 Mid-level Makefile

```
echo $(IBLTYP) > .tmpfile ; \
$(GREP) "0" .tmpfile ; \
if [ $$? = 0 ] ; then \
echo "IBLTYP = 0" ; \
( cd dry ; $(MAKE) all ) ; \
else \
```

NCAR/MM

### 9.3.2 Mid-level Makefile

- All compilable physics options treated similarly: if an option is requested that directory's Fortran files are compiled
- Sequential if loops in shell, so all requested options are compiled (for `IBLTYP` example, option for 0, 2, and 5)

NCAR/MM

### 9.3.3 Low-level Makefile

- Example is for the MRF PBL scheme
  - Down an additional level:
- ```
DEVTOP = ../../..
include ../.././configure.user
```
- `OBJS`, `SRC`, `SRCF` macros
  - Compilation rules come from included `configure.user`

NCAR/MM

### 9.3.3 Low-level Makefile

- Target all depends on the `$(OBS)` macro
- UNIX archive `ar` used to build library of compiled Fortran routines
- Dependencies listed at bottom of file:  
`mrfpbl.o: ../../include/param.inc`
- If any of the .F files (or .inc files) are out of date wrt the .o files, new object code is compiled – the library is always updated

NCAR/MM

### 9.4 CPP

- The “C” pre-processor
- Used for textual modification to source code prior to compilation
- MM5 uses `cpp` to either include extra code or delete extraneous code

NCAR/MM

### 9.4 CPP

`#include <param.inc>`

- Expanded after `cpp`, more standard directory searching than Fortran INCLUDE statement, and .f file has included text

NCAR/MM

### 9.4 CPP

- Based on the `configure.user` physics options, `#define` commands are inserted into various routines, such as the main `solve.F`

`#define ICUPA3 1`

- Then later in the code appears

`#ifdef ICUPA3`

*Joe-Fortran calls to this cumulus scheme*

`#endif`

NCAR/MM

`exit ( 0 )`

NCAR/MM