Make

Can’t Live with It, Can’t Live Without It

- make is the standard Unix way to manage compilations with many files
- Even most IDEs use make under the hood (even on Windows)
- make has an antiquated and obscure syntax. Efforts have been made to replace it, but none has really been successful.
- Managing complex builds without it is tedious and error-prone.

makemake

- On the clusters we provide a script makemake
- Now works for C++ too!
- Creates a skeleton Makefile that you must edit.
- cd to the directory (must contain all the source) and invoke

```
makemake
```

Example

```
PROG =

SRCS =	csv_file.f90 csv_file_1d.f90 csv_file_2d.f90 Xmas_bird_count.f90

OBJS =	csv_file.o csv_file_1d.o csv_file_2d.o Xmas_bird_count.o

LIBS =

CC = cc
CXX = c++
CFLAGS = -O
CXXFLAGS = -O
FC = f77
FFLAGS = -O
F90 = f90
F90FLAGS = -O
LDFLAGS =
all: $(PROG)

$(PROG): $(OBJS)
	$(F90) $(LDFLAGS) -o $@ $(OBJS) $(LIBS)

.PHONY: clean
clean:
	rm -f $(PROG) $(OBJS) *.mod

.SUFFIXES: $(SUFFIXES) .f90 .F90 .f95
.SUFFIXES: $(SUFFIXES) .c .cpp .cxx

.f90.o .f95.o .F90.o:
	$(F90) $(F90FLAGS) -c $<

.c.o:
	$(CC) $(CFLAGS) -c $<
```

```
.cpp.o .cxx.o:
	$(CXX) $(CXXFLAGS) -c $<

csv_file.o: csv_file_1d.f90 csv_file_2d.f90
Xmas_bird_count.o: csv_file.o
```

Edit the Makefile

- Provide a program name
- Delete lines not relevant to your language
  - (F77/C/C++ in this case)
- Add any flags you need
  - Default flag is -O
  - This means optimize at the default level (high for the Intel compiler)

Debugging Flags

- I usually change the flags as follows:

```
DBG=-g -CB
#OPT=-O
F90FLAGS=$(DBG) $(OPT)
```

Comment out either `DBG` or `OPT` for development (use DBG) or production (use OPT)

Note: this is for the Intel compiler, other compilers use -C alone

Why CB?

- CB stands for check bounds
- Without CB, the compiler/runtime will not stop you from accessing memory outside of the block allocated for an array.
- If you attempt to read from that memory you will get random garbage
- If you attempt to write to that memory you will generate a **memory fault** (segmentation violation, segfault, `SIGSEGV`)

Debugging (Continued)

- You should compile with CB (or C, or whatever your compiler uses to check bounds--check the manpage for your compiler) *but* bounds checking results in much slower code. When you are ready for production,
- make clean
- Comment out DBG, uncomment OPT
- make