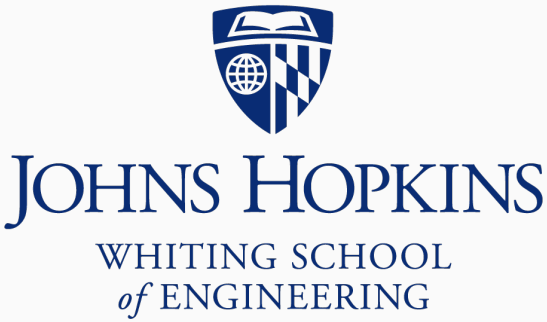
Makefiles

Ben Langmead

ben.langmead@gmail.com

www.langmead-lab.org



Source markdown available at github.com/BenLangmead/c-cpp-notes

Makefiles

For programs spread across many `.c` & `.h` files and executables, `make` and `Makefiles` tie things together

`Makefile` expresses *dependence relationships* between files

Defines *rules* for building one file from others

Makefiles

Example that performs 3 steps (preprocess, compile, link) at once:

- Some gcc arguments omitted for conciseness

```
interest-example: interest.c interest.h main.c
	gcc -o interest-example interest.c main.c -lm
```

First line says `interest-example` depends on `interest.c`, `interest.h` and `main.c`

Second line gives command for building `interest-example`

Makefiles

```
$ make interest-example
gcc -o interest-example interest.c main.c -lm
$ ./interest-example
110.46
```

Makefiles

This version performs steps 1 & 2 (preprocess and compile) separately from step 3 (link)

```
interest-example: interest.o main.o
	gcc -o interest-example interest.o main.o -lm

interest.o: interest.c interest.h
	gcc -c interest.c

main.o: main.c interest.h
	gcc -c main.c
```

Makefiles

```
$ make interest-example
gcc -c interest.c
gcc -c main.c
gcc -o interest-example interest.o main.o -lm
$ ./interest-example
110.46
```

Makefiles

Makefiles can get repetitive, e.g. if we re-write the gcc arguments for every rule. Use *variables* to simplify:

```
CC = gcc
CFLAGS = -std=c99 -Wall -Wextra -pedantic

interest-example: interest.c interest.h main.c
    $(CC) -o interest-example interest.c main.c $(CFLAGS) -lm
```

`$(CC)` is replaced by `gcc`

`$(CFLAGS)` is replaced by `-std=c99 -Wall -Wextra -pedantic`

Vocabulary

```
interest-example: interest.c interest.h main.c
    $(CC) -o interest-example interest.c main.c $(CFLAGS) -lm
```

`interest-example` is the *target*

`interest.c interest.h main.c` are the *prerequisites*

`$(CC) -o interest-example ...` is the *command*

All these together make up a *rule*

Makefiles

```
interest-example: interest.c interest.h main.c
    $(CC) $(CFLAGS) -o interest-example interest.c main.c
```

`make interest-example` asks make to build `interest-example` target, **if...**

- `interest-example` doesn't exist, *or*
- `interest.c` has changed more recently than `interest-example`, *or*
- `interest.h` has changed more recently than `interest-example`, *or*
- `main.c` has changed more recently than `interest-example`

If *any* are true, corresponding command is run

Makefiles

`make` looks at the "whole picture" of how targets are interrelated when deciding what to build

If you want to build target `A` **and**

- `A` depends on `B` *and*
- `B` depends on `C` *and*
- `B` is out of date (i.e. `C` is newer than `B`)

. . . then `make` builds `B` first, then builds `A`

Makefiles

For `make` to work, you must be in the same directory with the `Makefile`

- Advanced: use `make -C` if in a different directory

`Makefile` has to be called `Makefile`

- Advanced: use `make -f <name>` if it's not called `Makefile`

Typing `make` without specifying a target builds the *default target*; whichever appears first in the `Makefile`

Makefiles

A *very* common mistake is to use spaces instead of tabs

```
CC = gcc
CFLAGS = -std=c99 -Wall -Wextra -pedantic -lm

interest-example: interest.c interest.h main.c
    $(CC) $(CFLAGS) -o interest-example interest.c main.c
```

You can't tell by looking, but I put 4 spaces instead of a tab before the `$(CC)` command

Makefiles

```
$ make interest-example
Makefile:5: *** missing separator.  Stop.
```

"`missing separator`" usually means you used spaces instead of tab

emacs *should* notice you're editing a `Makefile` and use tabs where appropriate; you can force emacs to use tab with `Ctrl-q <tab>`

Makefile tutorials

mrbook.org/blog/tutorials/make/

- Uses g++/.cpp instead of gcc/.c, but ideas are the same

www.cs.bu.edu/teaching/cpp/writing-makefiles/

- Uses g++/.cpp instead of gcc/.c, but ideas are the same

cslibrary.stanford.edu/107/UnixProgrammingTools.pdf

- Section 2