Week 06 1

CSC209 Fall 2023

Dr. Demetres (dee) Kostas

October 17, 2023

Announcements

- A2 is due tomorrow
  - final office hours
  - 6pm tonight over zoom
- midterm
  - 1 week from now
  - 50 mins long
    - starts 2:10 sharp

Midterm (continued)

- material includes
  - PCRS, lectures, labs, A1+2
  - Everything before + **this lecture**
- **Primarily** materials
  - that have *come up* in class
  - or in the assignments
- Available on quercus **now**
  - some past tests
  - useful code from lectures

`#include`

- what does this really mean
- why don't we just include '.c' files

headers

- these are the '.h' files
- they define things like
  - types
  - functions available
    - to other source files
- largely for **declarations**
  - not implementations

what about implementations?

- we still need to provide these
- before/during compilation of
  - the final executable
- the names of functions
  - or variables
  - are recognized *with declaration*
  - and then linked to code
    - when the executable is made

ifndef

- notice that '.h' files
- are surrounded with this macro
- what is it?
  - in essence, it's so that we don't

    - redefine things, over and over

- we only want to include the header's
  - declarations once!

Makefiles

- at some point
  - compilation can be exhausting
- compiling multi-file programs
  - gets increasingly complex
  - some source files **depend**
    - on other ones
- you also type the same commands
  - into the terminal all the time

make

- 47 years old
- before this
  - mostly custom scripts
  - for compiling code
- standardized compilations
  - so you just have to type
    - `make`

make [target ...]

- as a command line program
  - make *specific* targets
- this looks for the file

    - `Makefile` or `makefile`

- without a target
    - it executes the first one
    - commonly `all`
        - which is `.PHONY`

Rules

- each `Makefile`
- primarily made up of **rules**

```
target: depedency # can be more than one
[shell command to execute 1]
...
[shell command N]
```

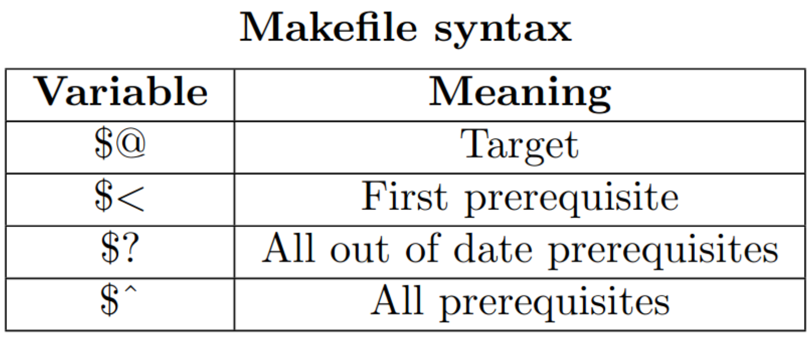
note actual TAB chars before commands

Macros (kinda variables)

```
MACRO = definition-${EXPAND_OTHER_MACRO}
```

- similar to the C macros
    - expanded (replacement) on demand

Some special ones



Makefile syntax

Variable	Meaning
$@	Target
$<	First prerequisite
$?	All out of date prerequisites
$^	All prerequisites

What is `.PHONY`

- a special target
- in case a target (e.g. `all`)
  - conflicts with a filename
- the dependencies for this target
  - don't (themselves)
    - create files

WORKSHEET

`makefile.pdf`