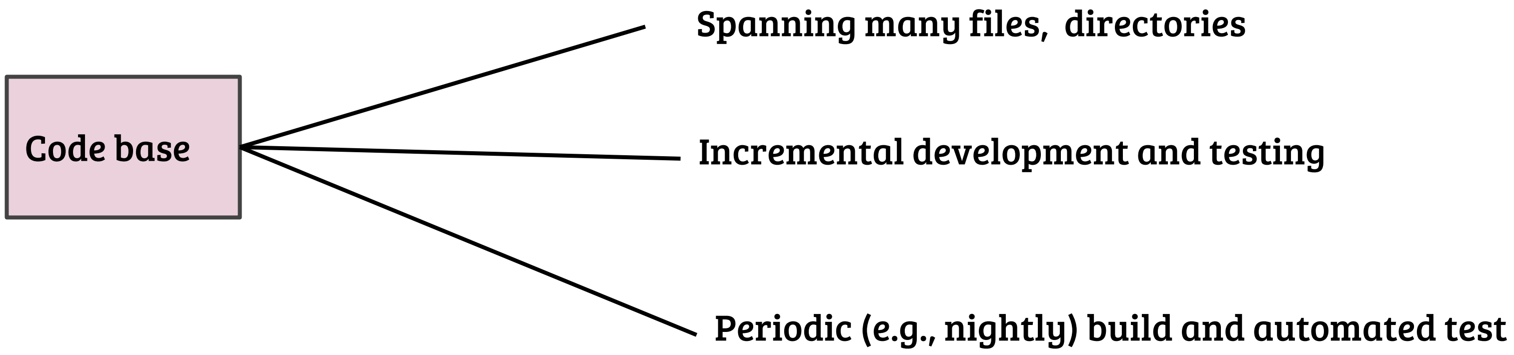
CS251 - Computing Laboratory

GNU Make and GNU Plot

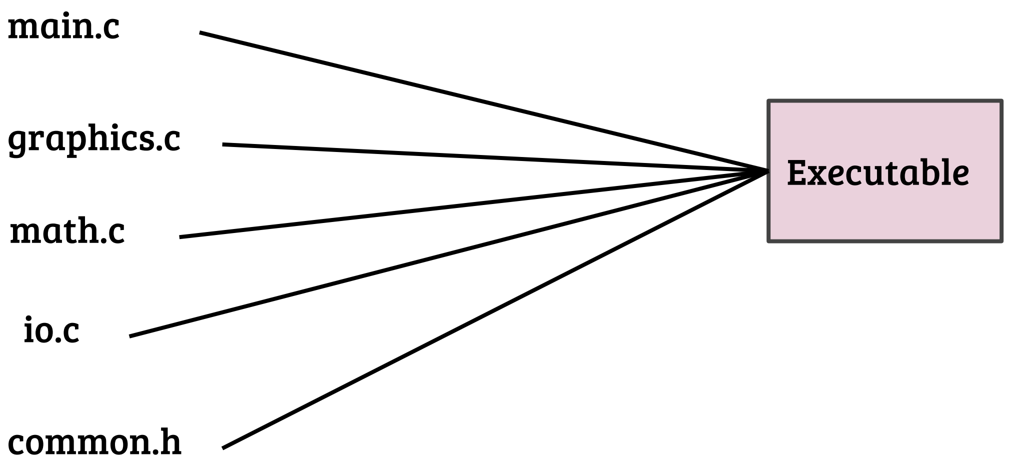
GNU Make

Build environments


Code base
Spanning many files, directories
Incremental development and testing
Periodic (e.g., nightly) build and automated test

- Challenges
  - Resource efficient build process → automated, incremental
  - Chance of build failure is high → fix and build

Motivating make



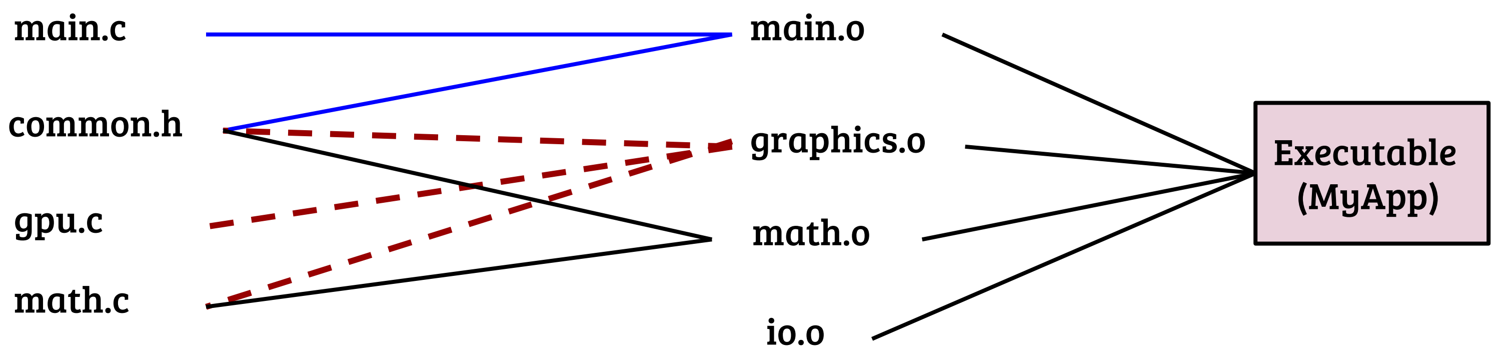
- ➔ Compilation using GCC, How?
  - ◆ What goes on in the background?
- ➔ What if you change only **main.c** file?

What is make?

- ➔ A set of rules to build a program
  - ◆ Expressed in a file, typically named Makefile
  - ◆ Contains dependencies
  - ◆ Can build more than one **target**
- ➔ Basic elements
  - ◆ **Target**
  - ◆ Dependencies
  - ◆ Commands

```
MyApp : core.c gui.c common.h
	gcc core.c gui.c -o MyApp
{TAB}
```

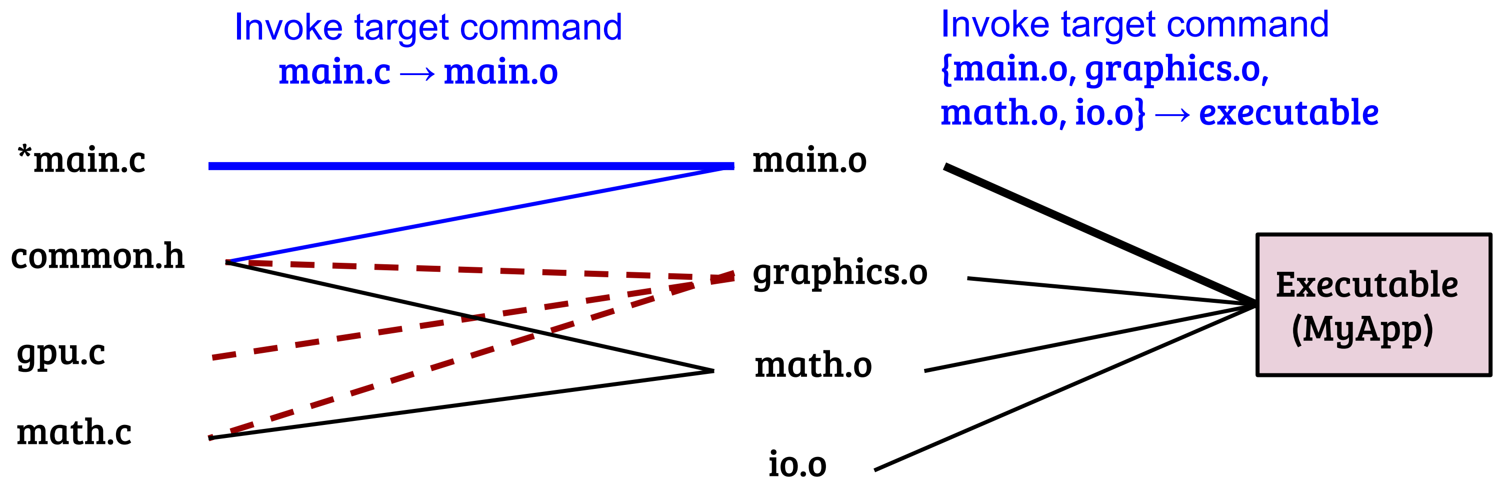
Build dependency tree



```
main.o: main.c common.h
          gcc -c main.c -o main.o
gpu.o: gpu.c math.c
         gcc -c gpu.c -o gpu.o
……………………………………..
MyApp: main.o graphics.o math.o io.o
        gcc main.o graphics.o math.o io.c -o MyApp
```

- ➔ What if there are circular dependencies?
- ➔ How helpful in (re)building only the necessary?

Targeted rebuilding

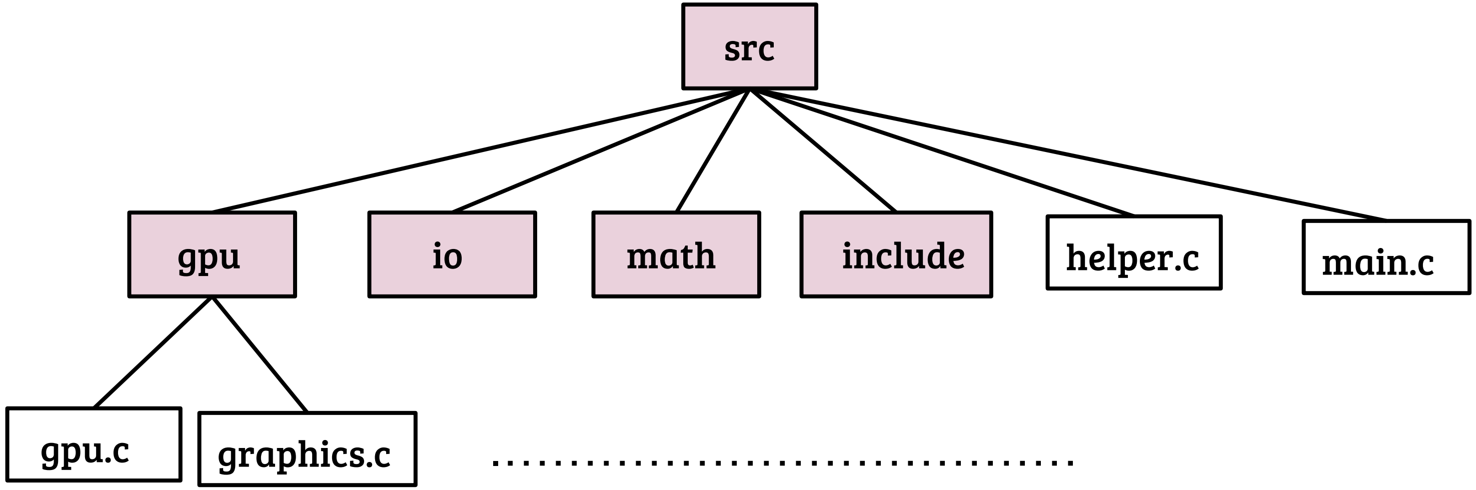


```
main.o: main.c common.h
         gcc -c main.c -o main.o
gpu.o: gpu.c math.c
        gcc -c gpu.c -o gpu.o
……………………………………..
MyApp: main.o graphics.o math.o io.o
       gcc main.o graphics.o math.o io.c -o MyApp
```

Makefile examples

- **Makefile (basic)**
  - All rules explicitly written
  - No variables, wildcards, functions etc.
- **Makefile.vars**
  - Variables
  - Rules using variables
- **Makefile.abbrs, Makefile.more**
  - **$@, $<, $^, %**
- **Makefile.multi**
  - Multiple targets
  - Phony targets
- **Makefile.wc**
  - Functions: **patsubst, wildcard** . . .

Multi-directory build



➔ Many strategies possible

◆ A single makefile at root-level with explicit rules

◆ Example: Makefile in each sub-folder invoked from root-level makefile (commonly used)

◆ **Makefile** for multi-directory

GNU Plotting utilities (gnuplot)

Gnuplot building blocks

- **Terminal**
  - Specify output format, size, color, font etc.
  - Examples: jpg, eps, png ...
- **Plotting styles**
  - Scatter, line, bar, box ...
- **Commands**
  - Set axes (labels, values, ranges ...)
  - key (legend) placement, spacing
  - Data source and plot
  - ....
- Examples
  - Scatter, CDF, lines, barchart, error bars