CSE 333: GCC and Make

Dylan Johnson

Exercise 0

Due tomorrow @ 11:15 AM in the dropbox

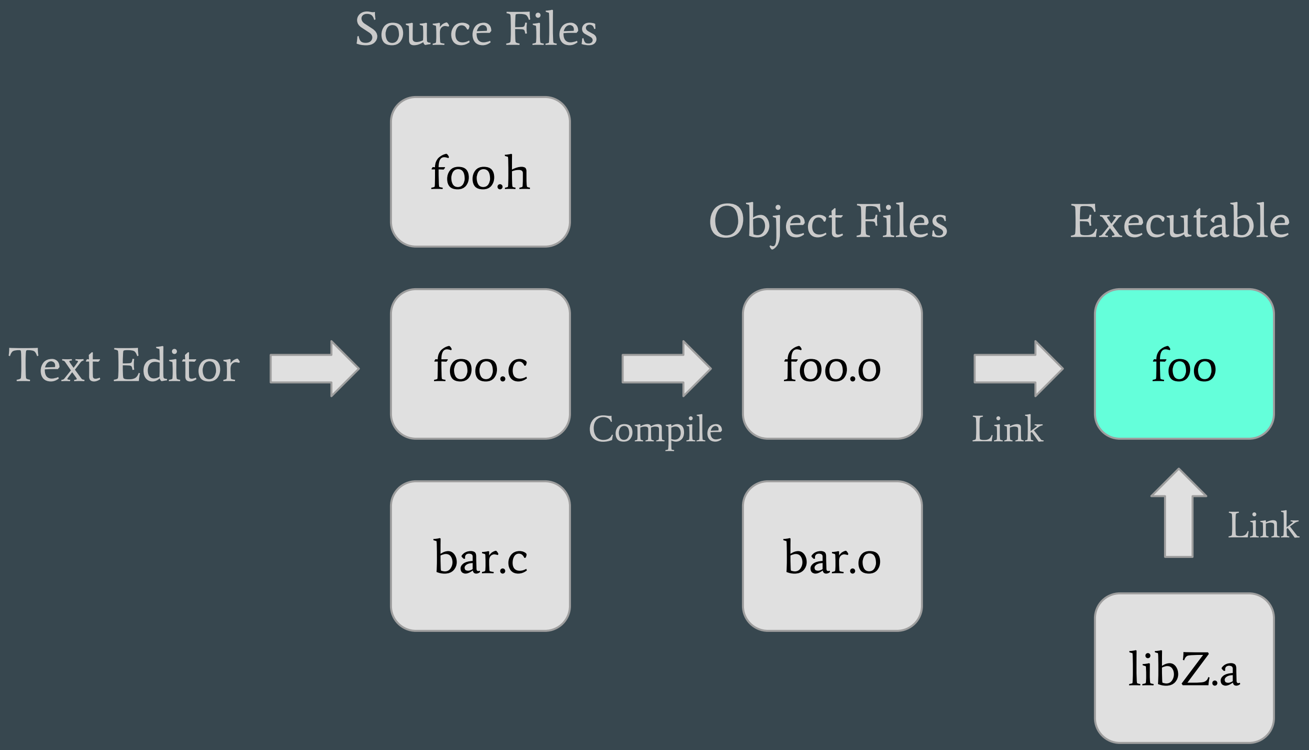
It's a simple comparison between 3 different programs

- C implementation
- Optimized C implementation
- Java implementation

Mini version of CSE 351 HW

Useful as a quick review of compiling using GCC

GCC Workflow



GCC: The C multitool

It can cut and paste (Preprocessor)

- gcc -E ex.c -o out

It can translate C into assembly code or machine code (Compiler)

- gcc -S ex.c -o out.s
- gcc -c ex.c -o out.o

It can combine compiled files into a single executable (Linker)

- gcc ex.c -o out

Common GCC Flags

Produce debugging information for use in GDB (-g)

Optimize the program (-Ox where x is the optimization factor)

Specify which C standard to use (-std=xxx)

Enable optional warnings desirable for normal code (-Wall)

Search for a library when linking (-l library)

- Use -L path flag to specify new library locations
- Searches for library named “liblibrary.a”
- Order matters! Always specify libraries last to avoid problems.

Make

make is a classic program for controlling what gets (re) compiled and how. Many other such programs exist (e.g., ant, maven, “projects” in IDEs, ...)

make has tons of fancy features, but only two basic ideas:

1. Scripts for executing commands
2. Dependencies for avoiding unnecessary work

Makefile

A makefile contains a bunch of triples

```
target: sources
	command
```

Example:

```
foo.o: foo.c foo.h bar.h
	gcc -Wall -o foo.o -c foo.c
```

Syntax gotchas:

The colon after the target is required

Command lines must start with a TAB NOT SPACES

You can actually have multiple commands (executed in order); if one command spans lines you must end the previous line with \

Make Variables

You can define variables in a Makefile. Example:

```
CC = gcc
CFLAGS = -Wall -std=c11
foo.o: foo.c foo.h bar.h
	$(CC) $(CFLAGS) -c foo.c -o foo.o
```

Why?

- Easy to change things
- Can change on make command line (CFLAGS=g)

Lots of Crazy Characters

- $@ for target
- $^ for all sources
- $< for left-most source
- % for pattern matching

Examples:

```
widget: foo.o bar.o
	$(CC) $(CFLAGS) -o $@ $^
foo.o: foo.c foo.h bar.h
	$(CC) $(CFLAGS) -c $<
```

Example Makefile