CSE 374: Programming Concepts and Tools

Eric Mullen

Spring 2017

Lecture 15: More Preprocessor, More Structs

Administrivia

- Midterm is over, you made it through
  - Grades published in grade book after class
  - Physical copies returned at end of class
  - Answer Key posted online
- Homework 5 out this Wed, due next Thursday
- Homework 6 is with a partner. Start thinking about who you want to work with

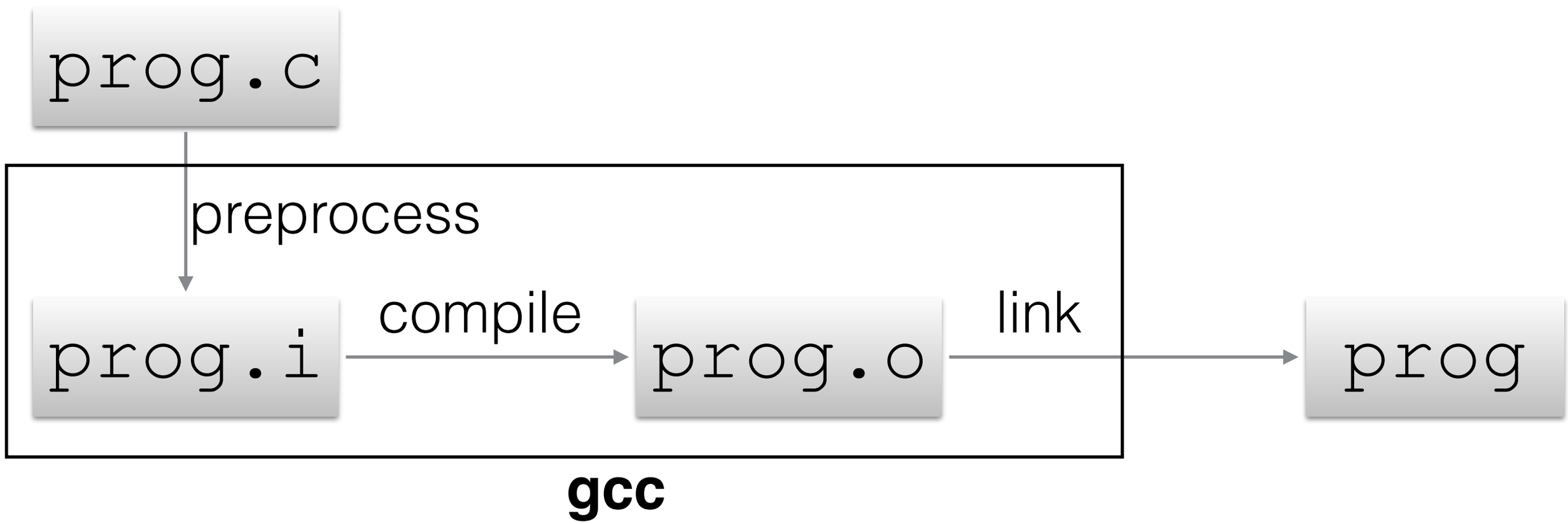
Today

- More about compilers
  - basic C compiler anatomy
  - more preprocessor features
- How to make large(r) programs
  - compiling multiple files

Compilation

- What happens when you type:

```
gcc prog.c -o prog -Wall -g -std=c11
```



Separate Compilati

probably never need

```
gcc -E prog.c -o prog.i -Wall -g -std=c11
```

- preprocess prog.c, generate result prog.i

quite common

```
gcc -c prog.c -o prog.o -Wall -g -std=c11
```

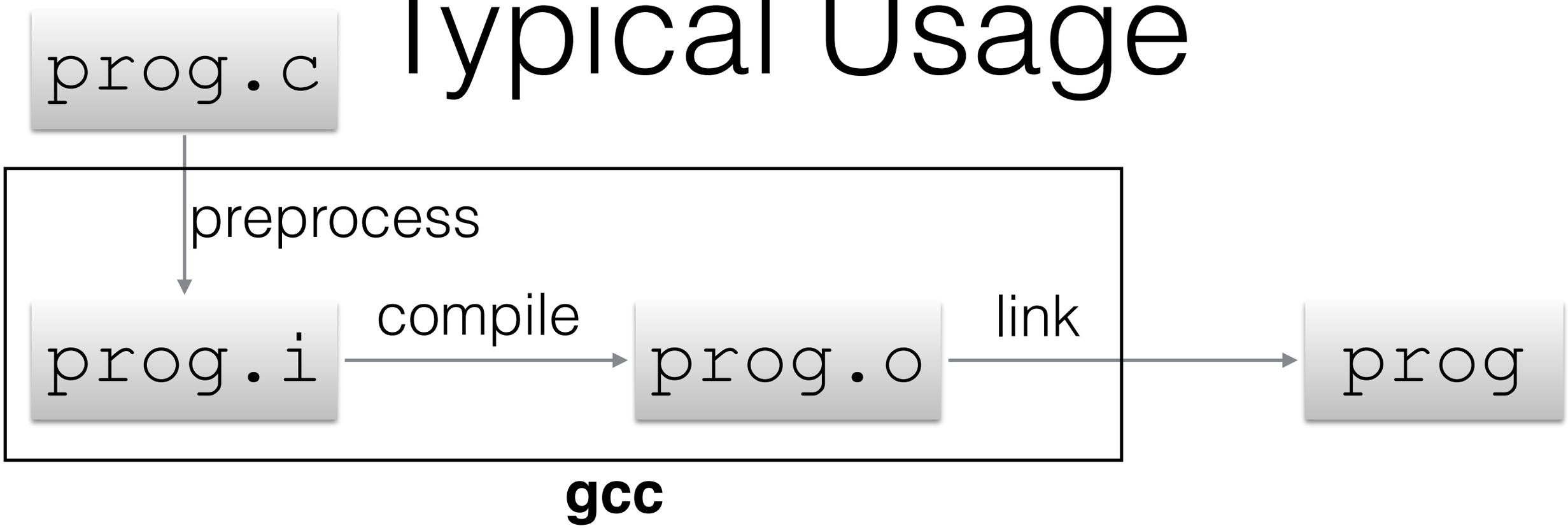
- preprocess and compile prog.c, generate result prog.o

```
gcc prog.c -o prog -Wall -g -std=c11
```

- preprocess, compile, and link prog.c, generate result prog

current use, less common

Typical Usage



- Preprocessor used to `#include` declarations describing code
- Linker combines all .o files and other code
  - C standard library
  - any .o files you give it (usually multiple)

The Preprocessor

- Rewrites your file before the compiler
  - Any lines starting with # are for the preprocessor
- Normal things to do:
  - `#include` header files
  - `#define` constants and parameterized macros
  - conditional compilation with `#if`
    - mostly for including header files exactly once

File Inclusion

```
#include <hdr.h>
```

- Search for hdr.h in the standard include directories, and paste the preprocessed contents of that file in this place

```
#include "hdr.h"
```

- Same as above, but look in current directory first

- use `gcc -I /path/to/dir` to specify additional directories where headers might be found (not used in this class)

Header File Conventions

1. Give included files a name ending in .h; only include these header files. NEVER include a .c source file.
2. Do not define functions in a header file; only struct definitions, function prototypes, and other includes
3. Put all your includes at the top of your files, before anything else
4. ALWAYS use include guards in every header file (next slide)

The Problem

```
int x = 5;

...
```

a.h

```
#include “a.h”

...
```

b.h

```
int x = 5;
int x = 5;

...
```

foo.c

Include Guards

- Make sure your header file is always included exactly once
- Use the preprocessor to make it happen

```
#ifndef FOO_H
#define FOO_H
<header file contents>
#endif //FOO_H
```

Use different variable for each header file (usually filename, foo.h -> FOO_H)

Simple Macros (review)

- Symbolic Constants

```
#define ABOUT_PI (22/7)

#define FEET_PER_MILE 5280

#define TIMEOUT_SEC 80
```

- Replaces all matching *tokens* in rest of file
- Has no notion of scope
- All caps not required, but is good style (pretty much universal agreement)

Macros with Parameters

```
#define TWICE_TERRIBLE(x) x+x
#define TWICE_AWFUL(x) (x)+(x)
#define TWICE_BAD(x) x*2
#define TWICE_OK(x) ((x)*2)
int twice_best(int x) { return x+x; }
```

- Replace all matching calls with body, use string substitution for arguments
- Many ways this can go wrong
- Common misconception: Macros avoid performance overhead (true in 1975, but not today)
- Macros can be more flexible (no types)

Macros: the dark side

```
#define TWICE_TERRIBLE(x) x+x

TWICE_TERRIBLE(3)*2 ==> 3+3*2 ==> 9

#define TWICE_AWFUL(x)  (x)+(x)

int x=4; TWICE_AWFUL(x++) ==>

   (x++)+(x++) ==> 9; x=6
```

Conditional Compilation

```
#ifdef FOO  (matching #endif later)
#ifndef FOO (matching #endif later)
#if FOO > 2 (matching #endif later)
```

Simple:
```
#ifdef DEBUG
    printf(…)
#endif
```

Fancy:
```
#ifdef DEBUG
    #define DBG_PRINT(x) printf("%s",x)
    #else
    #define DBG_PRINT(x)
    #endif
```

Header Files

```
#ifndef FOO_H
#define FOO_H
<header file contents>
#endif //FOO_H
```

- We want the freedom to include whatever we want, without worrying
- Thus every header must (and does) use include guards
- Be careful: use separate variable for each

Preprocessor Summary

- Runs before compilation
- `#include` for files
- `#define` for macros
- `#if` for conditional compilation

Midterms

- Class Average: 91.66/122 (75%)
- Std Deviation 18.3/122 (15%)
- How to talk to TAs/me about it
  - READ ANSWER KEY FIRST
  - We are happy to chat about answer key/grading/ anything once you've read the answer key