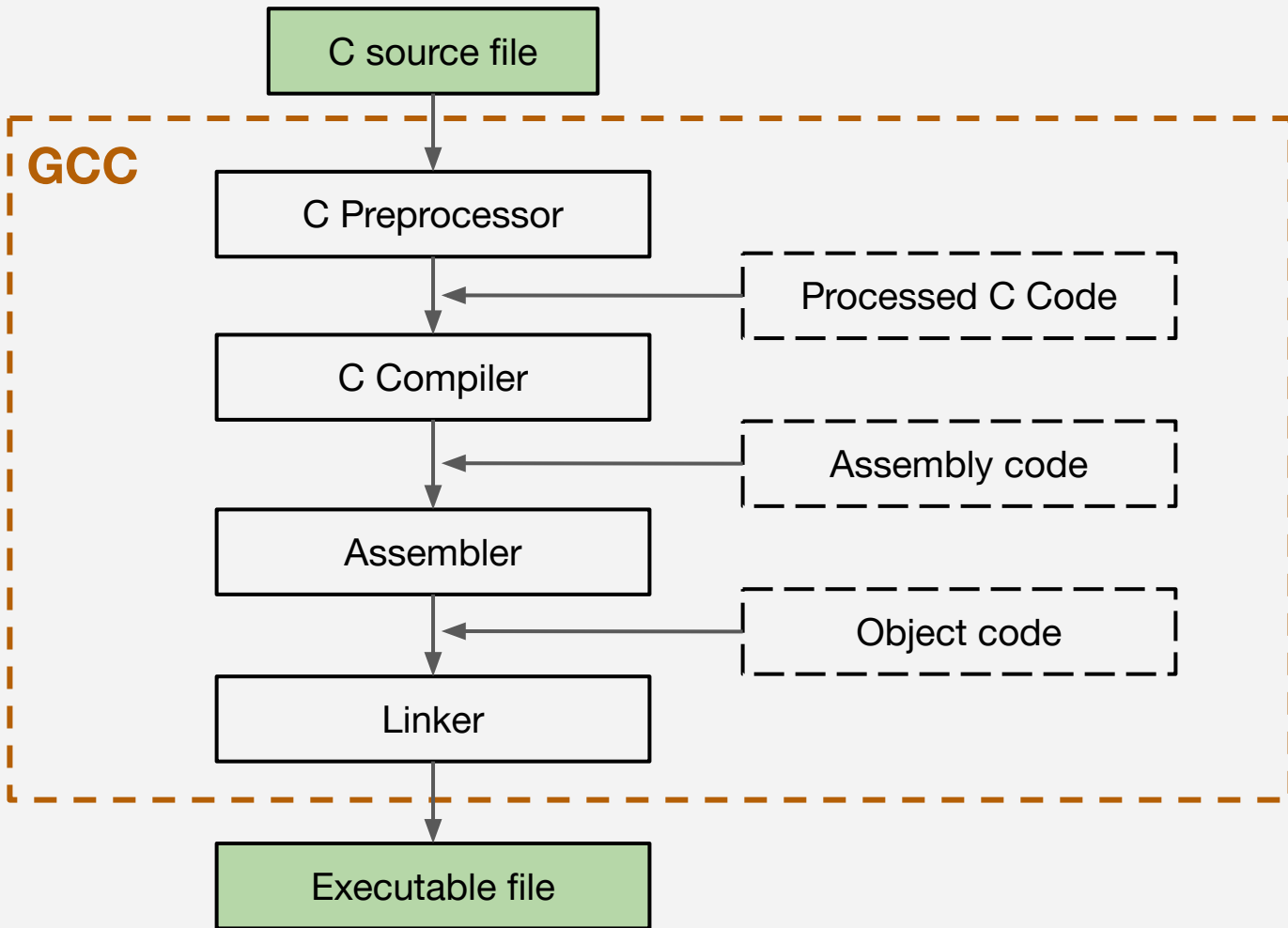


CSc 352

The C Preprocessor

Benjamin Dicken



The C Preprocessor

- A “mini” language that gives the programmer the ability to include other source files, conditionally include code, define literals, etc
- Many creative ways it can be used, but there are some common patterns

Some C Preprocessor Directives

`#include`

`#define`

`#undef`

`#ifdef`

`#ifndef`

`#endif`

`#error`

`#pragma`

Includes

- `#include` allows you to include (copy) code from one file into another
 - Use `< >` for standard library files
 - Use `" "` for files within the source code for your project
 - Operates recursively

Defines

- **#define** allows you to define keywords that can then be found and replaced throughout the source
 - Useful for constants, debug prints, repetitive sequences of code
 - Use `\` for multi-line constants
 - Can have parameters too!

Checking / Modifying Definitions

- **#ifdef** and **#ifndef** check if a keyword is or is not defined currently
 - Can conditionally include code depending on answer
- **#undef** to un-define a previously defined keyword
 - Useful for constants, debug prints, repetitive sequences of code
 - Use `\` for multi-line constants
 - Can have parameters too!

Fix the program

- COPY the files in `/tmp/352cptest` to your home directory
`$ cp -r /tmp/352cptest ~/`
- Compile the code with **make**
- What do you see?
- How can you fix it using the preprocessor?

Implement Debug Define

- Implement a **DEBUG_PRINT** directive that prints out a debug line, only if **DEBUG_MODE** is enabled
- Write a makefile to have the option to build in **DEBUG_MODE** or not