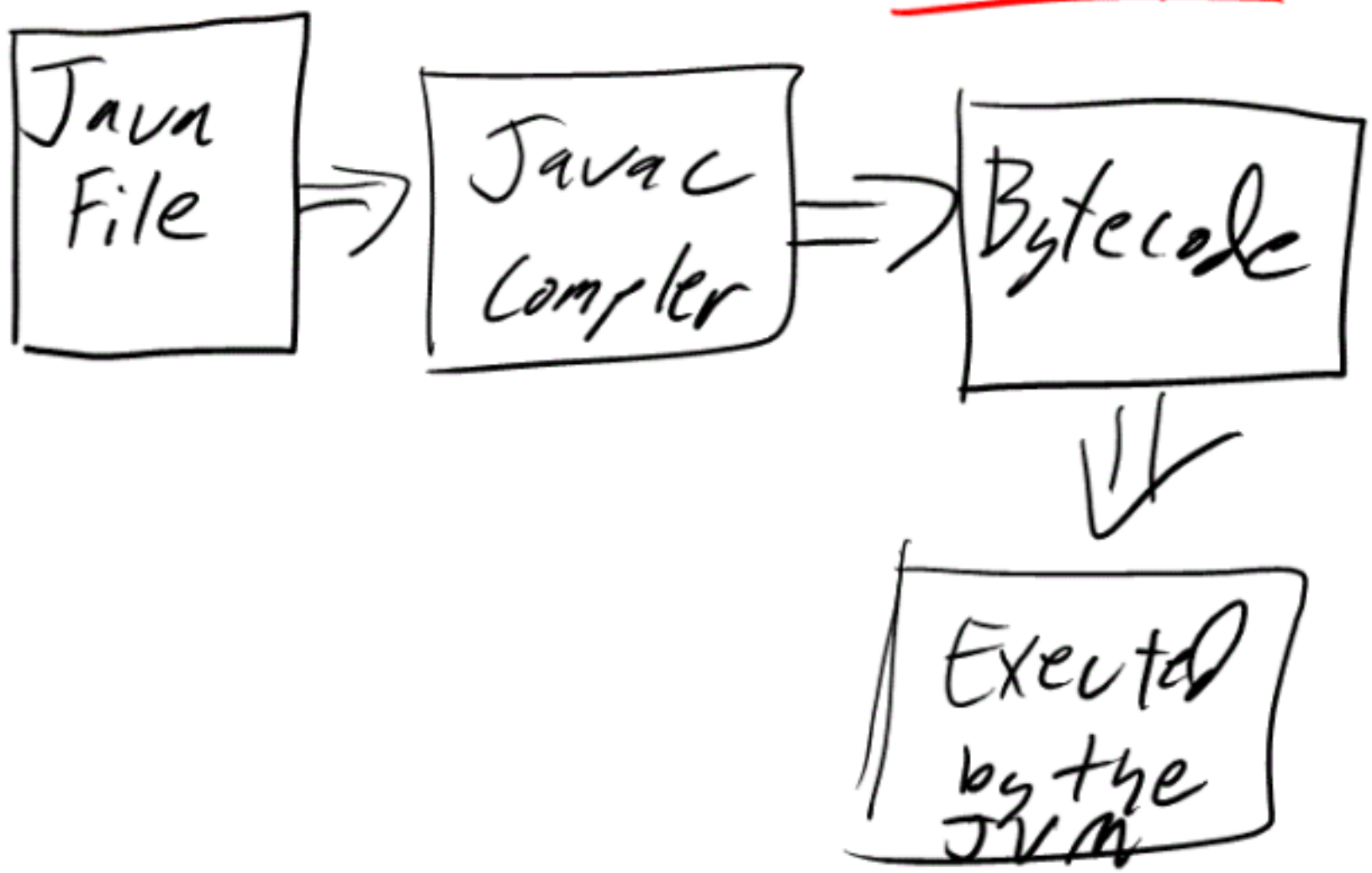
CS-3841: Operating Systems

- Objectives
  - Draw the C flow of C compilation from source code to object code.
  - Explain the purpose for the preprocessor, compiler, and linker within the C compilation model
  - Using the gcc compiler, generate the output for the preprocessor stage of compilation
  - Explain the concept of a dependency.
  - Create a GNU Make file which automatically generates dependencies, creates preprocessed source code, and links a given C application.

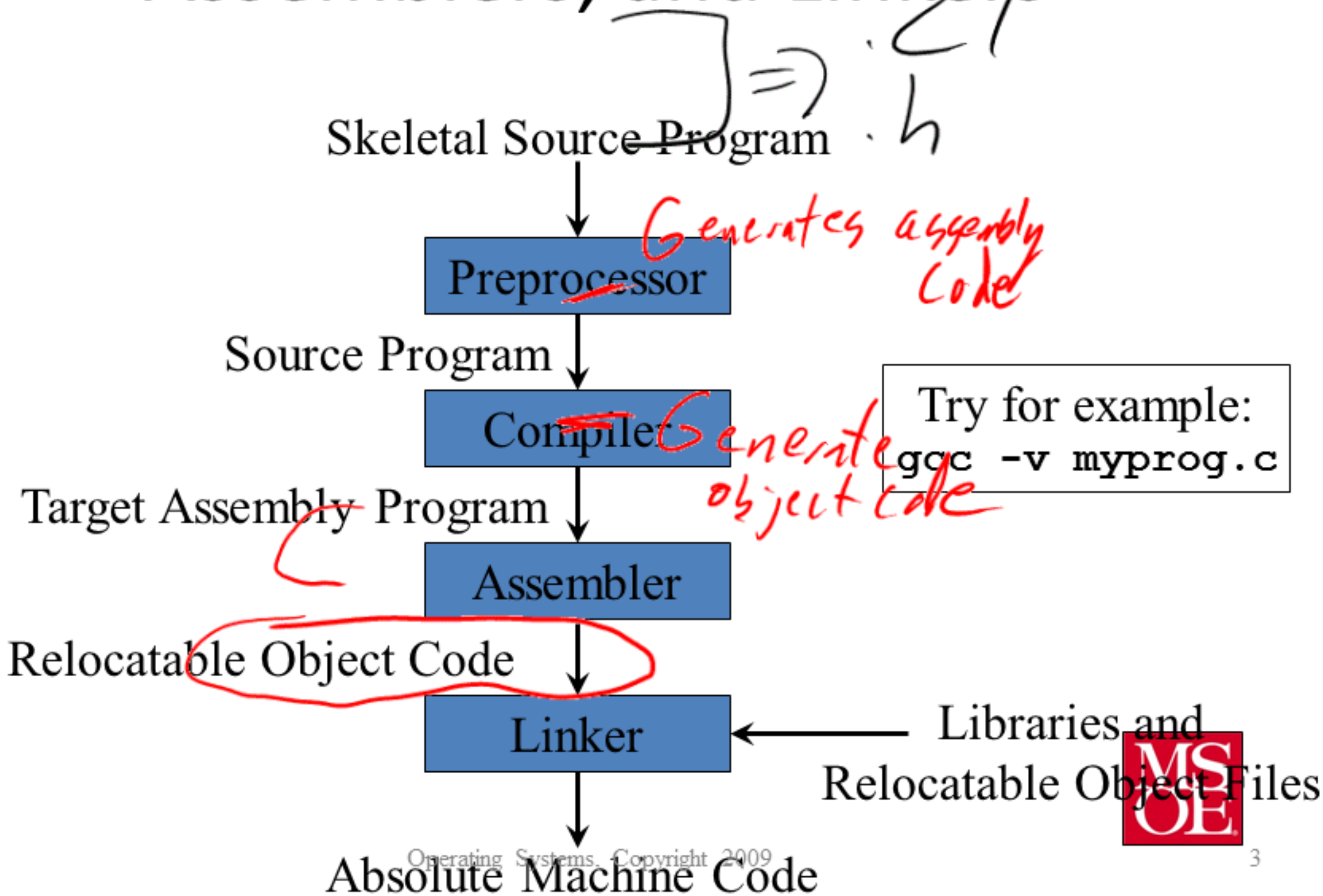
Discussion

- How does source code go from source code to an executable program?

Java



Preprocessors, Compilers, Assemblers, and Linkers



Try for example:
`gcc -v myprog.c`

Preprocessor

- Preprocessor : initial translator
  - Removes comments & white space
  - Groups characters into tokens (keywords, identifiers, numbers)
  - Expends macros and abbreviations
  - "Paste's" in #include files
  - Completes conditional compilation steps
- Gcc –E stops compilation after the preprocessing phase

Compilation

- Translates the Preprocessed code into assembly language instructions
  - Defines all variables appropriately
  - Defines all subroutines
  - Assigns tables for all constants
  - Generates assembly language code for the high level instructions provided previously
  - May optimize code as compilation occurs

Assembler

- Converts the assembly language code generated by the compiler into object code
  - Object code is specific to the given machine
  - Object code can be stored in libraries for usage by other programs
  - May optimize code depending on compiler setup

Linker

- combines the optimized object modules into an executable program
  - Resolves references to functions across files such as object modules or library files.
  - Program can be loaded into memory and run by the operating system.

Dependency

- An external file upon which impacts the compilation of a given source file.

Lets look at an example in Ubuntu Unix