Compiling C++

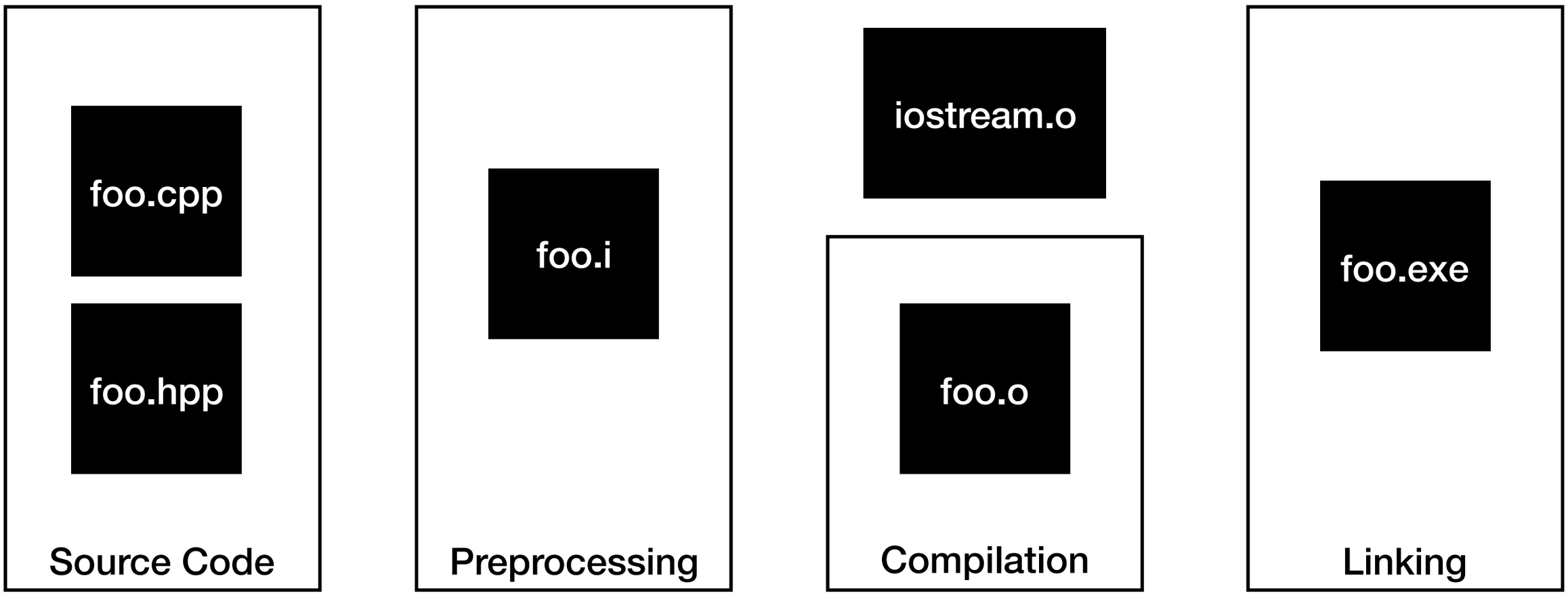
Source Code to Executable

Jonathan I. Maletic
Kent State University

Three Basic Steps

- **Preprocessing**
  - Includes, preprocessor commands
  - Template preprocessing
- **Compilation**
  - Lexical analysis
  - Parsing
  - Assembly code generation
- **Linking**
  - Combines compilation units and libraries
  - Generates executable

High Level View



Programs are Multi-file

Need to link multiple files plus all the included libraries (iostream, vector, etc.)

Build Management

- Building an entire program requires that each .cpp/.hpp pair be compiled separately.
  - `clang++ foo.cpp -c -Wall -o foo.o`
- These are then linked together after everything is compiled.
  - `clang++ foo.o foo1.o main.o -Wall -o prog`
- make (Makefile) is used to manage this process

Preprocessor

foo.cpp

foo.hpp

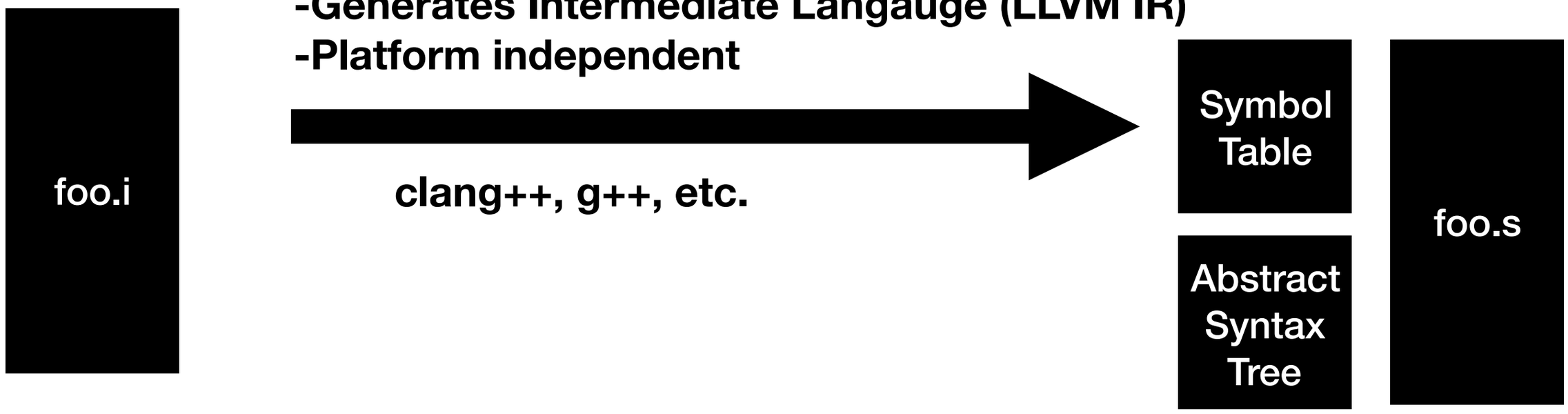
-Include headers
-Evaluates #ifndef, #define
-Instantiates templates
-Platform independent

cpp

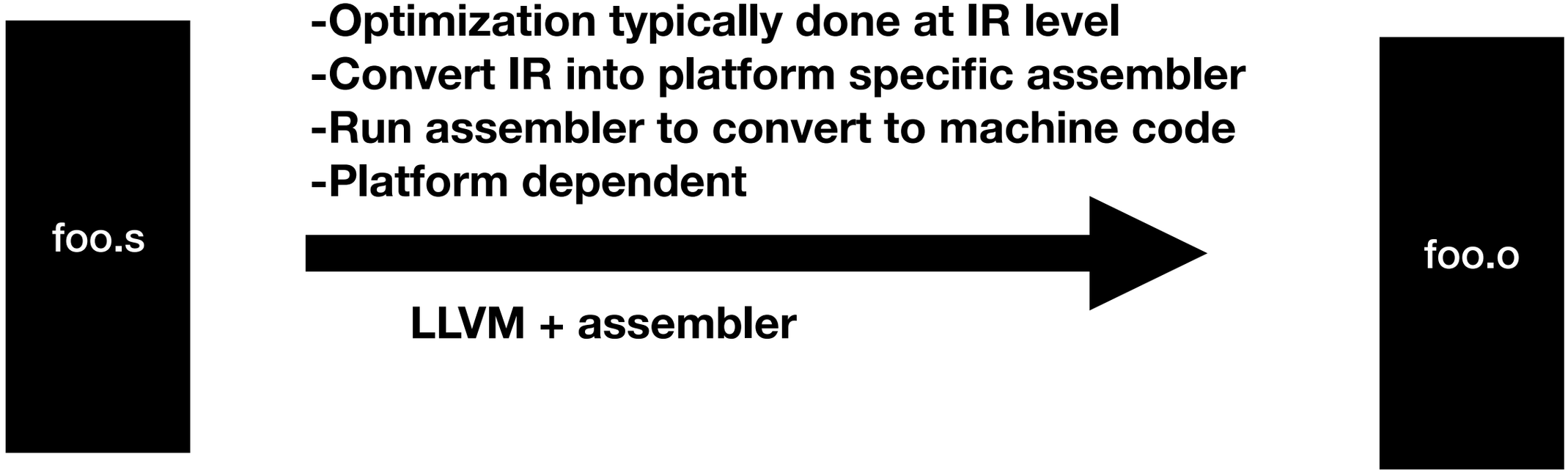
foo.i

Compilation

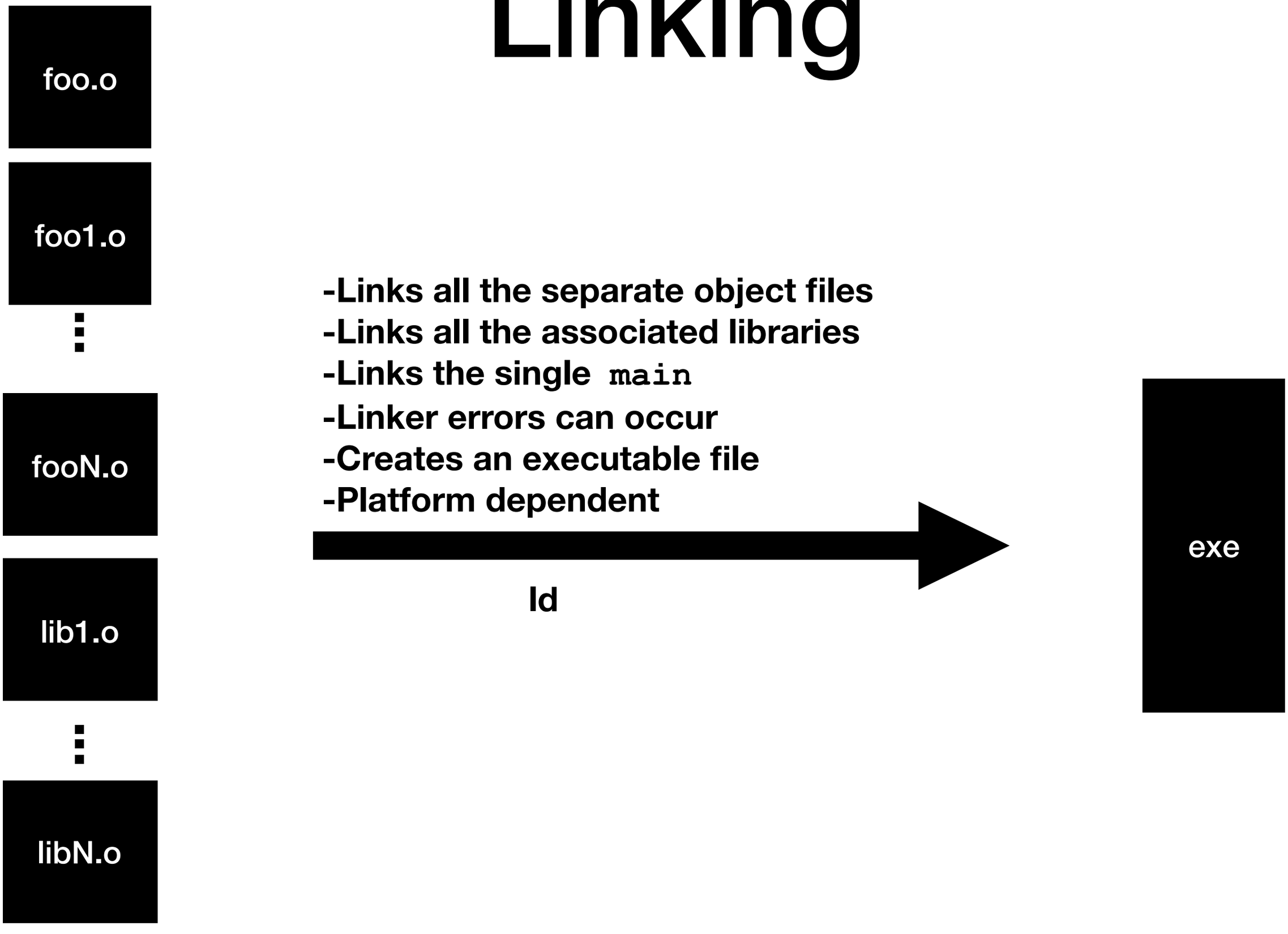
-Lexical Analysis - finds tokens (names, symbols), builds Symbol Table
-Syntactic Analysis - parses code (using grammar), builds Abstract Syntax Tree (AST)
-Semantic Analysis, type resolution, semantics
-Syntax errors can occur
-Generates Intermediate Langauge (LLVM IR)
-Platform independent



Code Generation



Linking


foo.o
foo1.o
fooN.o
lib1.o
libN.o
-Links all the separate object files
-Links all the associated libraries
-Links the single main
-Linker errors can occur
-Creates an executable file
-Platform dependent
ld
exe

Execution

-Loader reads exe from disk and loads into memory
-CPU executes each machine instruction
-Runtime errors can occur