C++ Compilation and Linking Process

Humayun Kabir
Professor, CS, Vancouver Island University, BC, Canada
C++ Development Environment

Fig. 1.7 | Typical C development environment. (Part 1 of 3.)
C++ Compilation Process

1. Source
2. Preprocessor
3. Preprocessed Source
4. Compiler
5. ASM (.s)
6. Assembler
7. Linker
8. Executable (a.out)
9. Loader
10. Executable
11. In Memory
12. Program

Flowchart:
- Source -> Preprocessor
- Preprocessed Source -> Compiler
- Compiler -> ASM (.s) -> Assembler
- Assembler -> Linker
- Linker -> Executable (a.out) -> Loader
- Loader -> Executable
- Executable In Memory
- Program

Output:
- .o
- OBJECT

Error: No error detected.
C++ Compilation Process: Preprocessing

1. **Preprocessor**
   - Source
   - Preprocessed

2. **Compiler**
   - ASM .s
   - Executable
   - Linked
   - a.out

3. **Assembler**
   - .s
   - Executable
   - Linked
   - a.out

4. **Linker**
   - .o
   - Program
   - In Memory

5. **Loader**
   - Executable
   - Program
   - In Memory

6. **Executable**
   - Program
   - In Memory

Preprocessed Source

Executable Program

In Memory

C++ Compilation Process:

Preprocessing
C++ Compilation Process: Preprocessing

• Pass over source
  • Insert included files
  • Perform macro substitutions

• Define macros
  • #define NUM 100
  • #define TASK(op, d1, d2) \ (d1 op d2)

• gcc –E example.c sends preprocessor output to stdout
C++ Compilation Process: Compiling

source -> Preprocessor -> Preprocessed Source

Source -> Compiler -> ASM .s

ASM .s -> Assembler -> Executable .o

Executable .o -> Linker -> Executable a.out

a.out -> Loader -> Program In Memory
C++ Compilation Process: Compiling

• gcc actually name of a script
• Compiler translates one language to another
• gcc compiler translates C to assembler
• **gcc -S example.c** “saves” assembler output to **example.s**
• Compiler consists of
  • Parser
  • Code generation
  • Mysticism
C++ Compilation Process: Assembling

- Another translator??? (as example.s)
- Assembler to (binary) object
- Why not compile straight to binary?
- **gcc –c example.c** to “save” (binary) object **example.o**
- Use **nm** to look at object (**nm example.o**)
C++ Compilation Process: Linking

source → Preprocessor → Preprocessed

Preprocessed → Compiler → ASM

ASM → Assembler

. S

Executable

Program

Loader

Executable a.out

Linker

. O

In Memory

OBJ

ECT
C++ Compilation Process: Linking

• Combines objects, both user .o files and libraries; makes an executable file

• `gcc *.o` yields executable `a.out` linking user and standard library objects.

• `gcc -o myExec *.o` yields executable `myExec` linking user and standard library objects.

• `gcc -o myExec *.o -lm` yields executable `myExec` linking user, standard library, and `math library` objects.

• Use `nm` to look at executable
C++ Loading

source → Preprocessor → Preprocessed Source → Compiler → ASM .s → Assembler → Linker → Executable .o

Executable Program → Loader → a.out → In Memory
Loader

- Runs when you type ./a.out
- Gets an address to place program (from the operating system)
- Changes necessary addresses (if any)
- Places code into memory
Loader

Phase 5:
Loader puts program in memory.

Phase 6:
CPU takes each instruction and executes it, possibly storing new data values as the program executes.

Fig. 1.1 | Typical C development environment. (Part 2 of 2.)