CA review: CALL: Compiling + Assembler + Linker + Loading

Main Topic: C program from storage (disk or flash) into a program running on a computer

C program → Compiler → Assembly → Assembler

↓ machine code → Linker ← library routine (machine code)

Memory ← Loader ← 可执行机器码程序 ← Linker

Compiler: C → Assembly

Assembler: 汇编中可能有伪指令, Assembler 基本任务是: Assembly → machine code

具体而言: assembly → object file, a combination of machine 指令、数据 以及指令放在内存中位置的信息

指令中的 label 地址将会十分重要, 故 assemble 会将追踪 labels 并记录它们与 data transfer instructions* 在 symbol table 中

对于 UNIX, object file 有以下六个信息:

① header: the size and the position of the other pieces of obj file

② text segment: contains machine code

③ static data segment: data allocated for the life of program

④ relocation information: 当 program load 进内存时, identifies instructions and data word that 依赖于绝对路径.

⑤ symbol table ⑥ debugging info

Linker: 有时,要调库 (如头文件, 其它.o), 要把所有独立的 machine language programs 都"拼"在一起

它一共有三步:

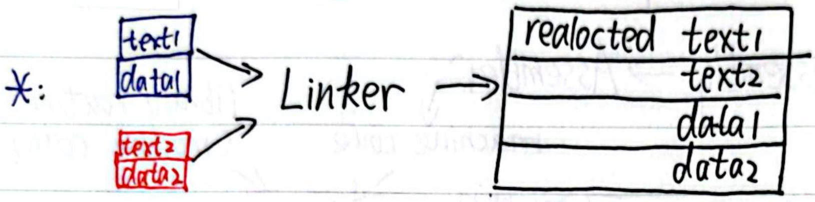
① * Place code and data modules symbolically in memory

② Determine the address of data & labels

③ Patch both the internal and externel references

↓ fill in absolute addresses ; must relocate to reflect location

*:



Loader: 可执行文件 : disk → memory , and start it.

Read header → create an address for text & data

→ copy data & 指令 到内存 → 初始化 reg, set stack pointer

→ Jump to start-up routine, calls main routine of program

Ex: When are the maehine codes generated?

1) add x5, x6, x7 ⇒ After assembly

2) jal x1, fprintf ⇒ After assembly

Determine? 1) After assembly 2) After linking