Module 9

Linking - Loading

Module Nine

- Linking – Loading - Part Two

- In this presentation, we are going to talk about :

- Program Linking
- Additional Object file record formats

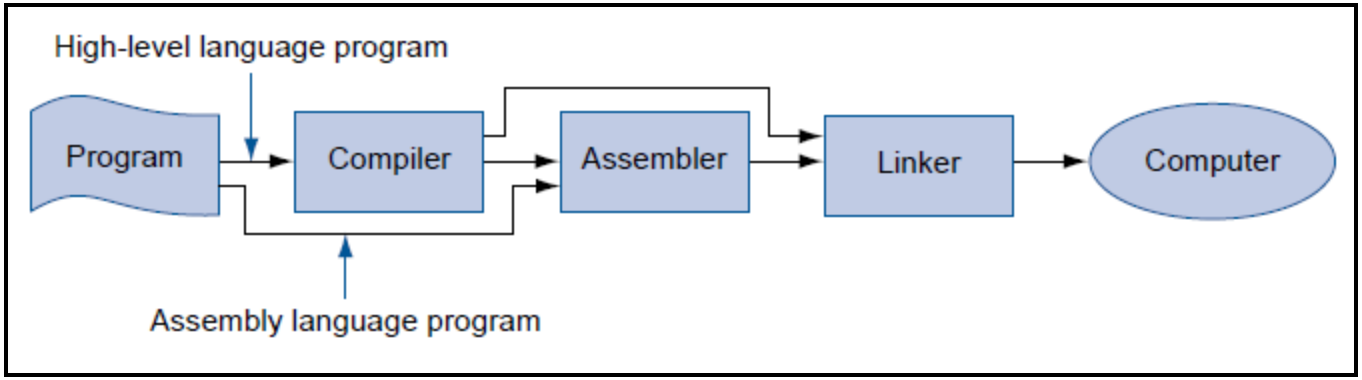
Overview

- Previously we talked about:
- Program Loading
  - Basic Function
  - Object file
  - Algorithm

Now: Program Linking

Program Linking

- Program Linking
  - Combine two or more separate object programs and supply the information needed to allow references between the programs.
  - LINK EDITOR



Program Linking

- Combine separately assembled or compiled programs into an executable program.
- Create the .exe

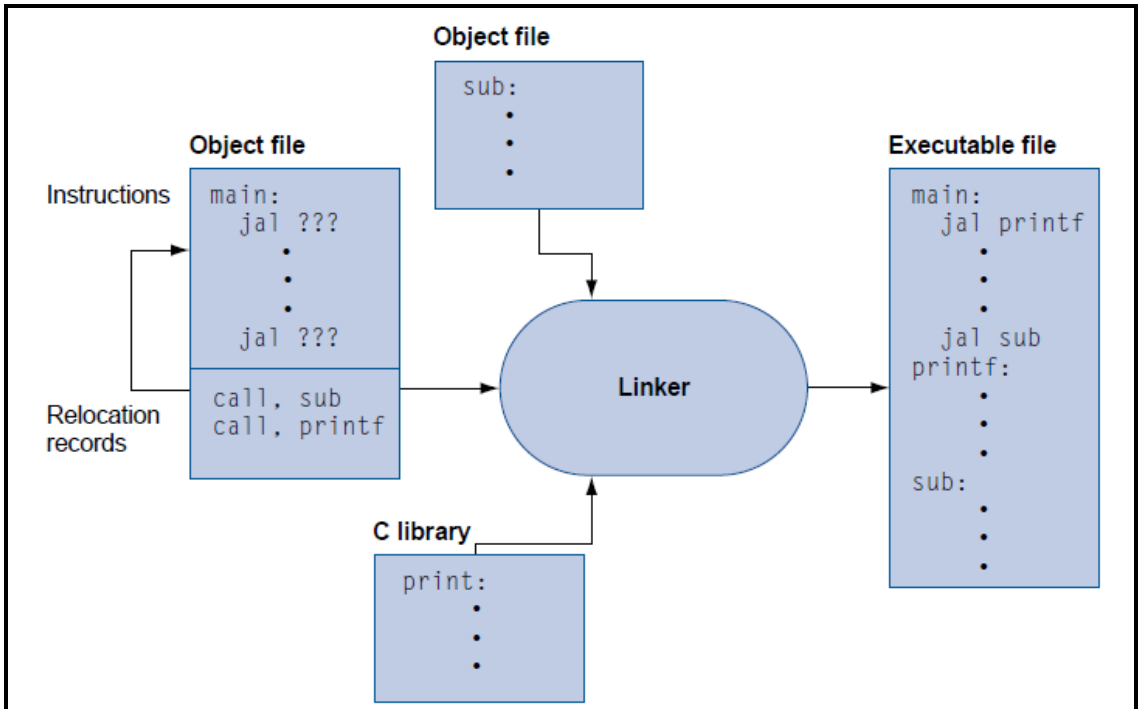


FIGURE A.3.1 The linker searches a collection of object files and program libraries to find nonlocal routines used in a program, combines them into a single executable file, and resolves references between routines in different files.

Program Linking

- Programs and subprograms assembled separately
- Symbol records
  - External Definitions
- External References
- The goal of the Linker is to resolve the External Addresses
- Search the Subroutine Libraries

Algorithm

- Logic for a Link Editor
  - Use a two pass method.
  - Read the object program files.
  - Assign addresses to external symbols.
  - Write .exe file with resolved external addresses.

Simple Object File Format

- Header Record H COPY 001000 00107A
- Text Record T 001000 1C 27BDFFE0 AFBF0014 AFA40020
- Define Record D BUFFER 000470 LENGTH 00107A
- Refer Record R BUFEND 000234 SUBONE 000580
- End Record E 001000

Another Object File Format (continue)

- Define Symbol Record

(1)	D
(2) - (7)	Name of symbol defined
(8) - (13)	Relative Address (hex)
(14) - (73)	Additional symbols and addresses

```
D BUFFER  000470  LENGTH  00107A
^         ^       ^       ^       ^
```

Another Object File Format (continue)

- Refer Record

(1)	R
(2) - (7)	Name of symbol referred to in section
(8) - (13)	Relative Address (hex)
(14) - (73)	Additional symbols and addresses

```
R  BUFEND  000234  SUBONE  000580
^          ^       ^       ^       ^
```

Example Object file

```
H  MAIN   000000  000378
D  BUFFER 000040 BUFRND   000140   LENGTH   000150
R  BLOCK  000150  TABLE    000180   START    0001C5
T  00014D   0C    000ACE00  12000000  03000110
E  000200

H  INIT   000000  000220
D  TABLE  0001A0
T  000070   10    17202DEF  69202D32  4B101036  3B2FEA45
T  000100   04    00001000
T  0001A0   04    FFFFEFFF
E  000000

H  READER 000000 0010AC
D  BLOCK  000370  START    000607
T  000600   0C    3B222FFF  B440B410  4B101000
E  000000
```

Algorithm

- Structure
  - **ExtSymTable** - External Symbol table.
    Name, address, subroutine name.
  - **Program Address** - Starting address to load the program,
    Supplied by the Operating System.
  - **SubPgmAddress** - SubProgram address
    Start address of SubProgram, value to be used to revise external reference addresses.

Algorithm

- Process PASS ONE - Build the External Symbol Table
  - Get **Program Address** value from Operating system
  - Set **SubPgmAddress** equal to **Program Address**
  - Read Header record

    Add Name to the **ExtSymTable** with SubPgmAddress value
  - Read Define Symbol record

    Add name to the **ExtSymTable** with SubPgmAddress value plus relative value from the record

    At end of subroutine update SubPgmAddress with length of subroutine; read next set of records.

Algorithm

- Process PASS TWO - Create the .exe file
  - Read Header record
  - Read Text records
    Copy code values to specified locations in the .exe file
  - Read Refer records
    Look-up symbols in **ExtSymTable**
    Add the symbol value to specified location in the .exe file

Program Linking

- Combine separately assembled or compiled programs into an executable program.
- Create the .exe

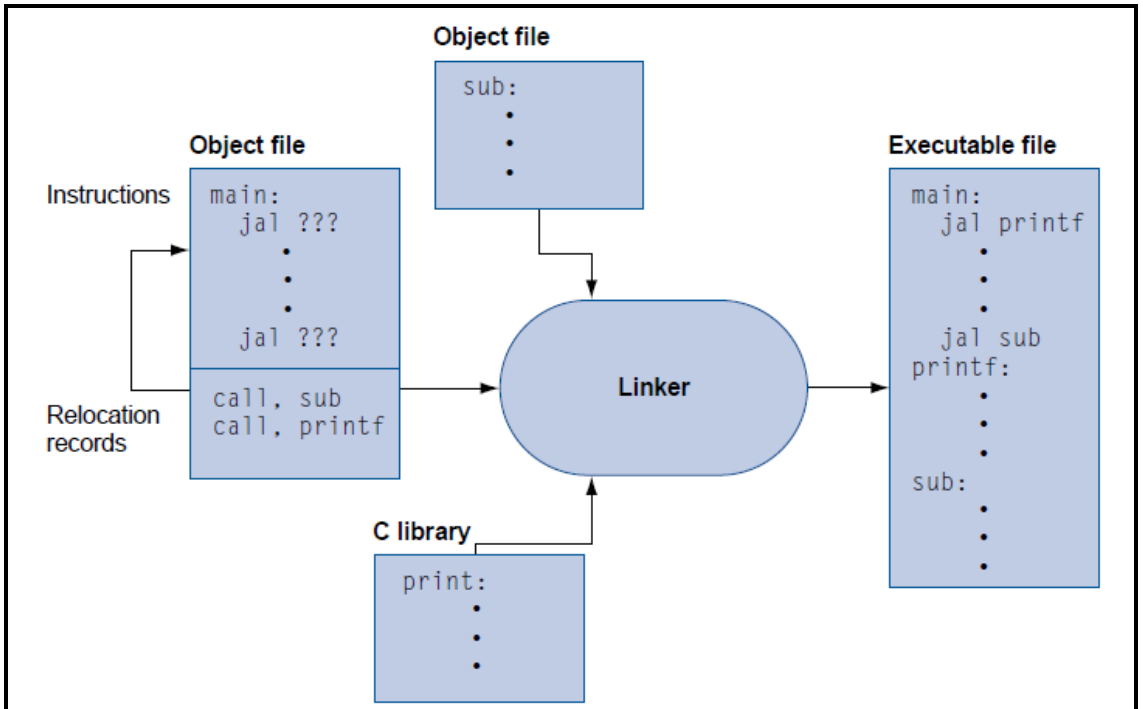


FIGURE A.3.1 The linker searches a collection of object files and program libraries to find nonlocal routines used in a program, combines them into a single executable file, and resolves references between routines in different files.

Summary

- Program Linking
- Link-Editor
  - Basic Functions
  - Algorithm

Next: Program Relocation