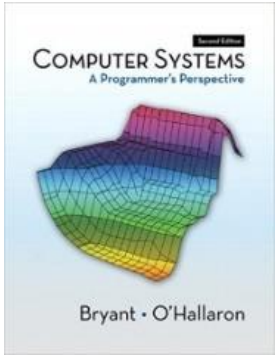
系统级编程

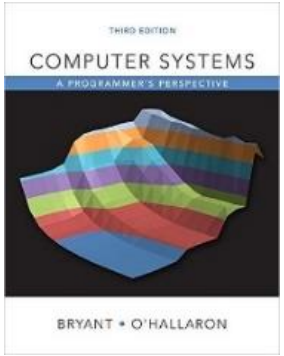
System Level Programming

李 靓 (Li Jing)

School of Computer Science and Software Engineering

lijing712@scu.edu.cn





Unit13 Linking and Linker

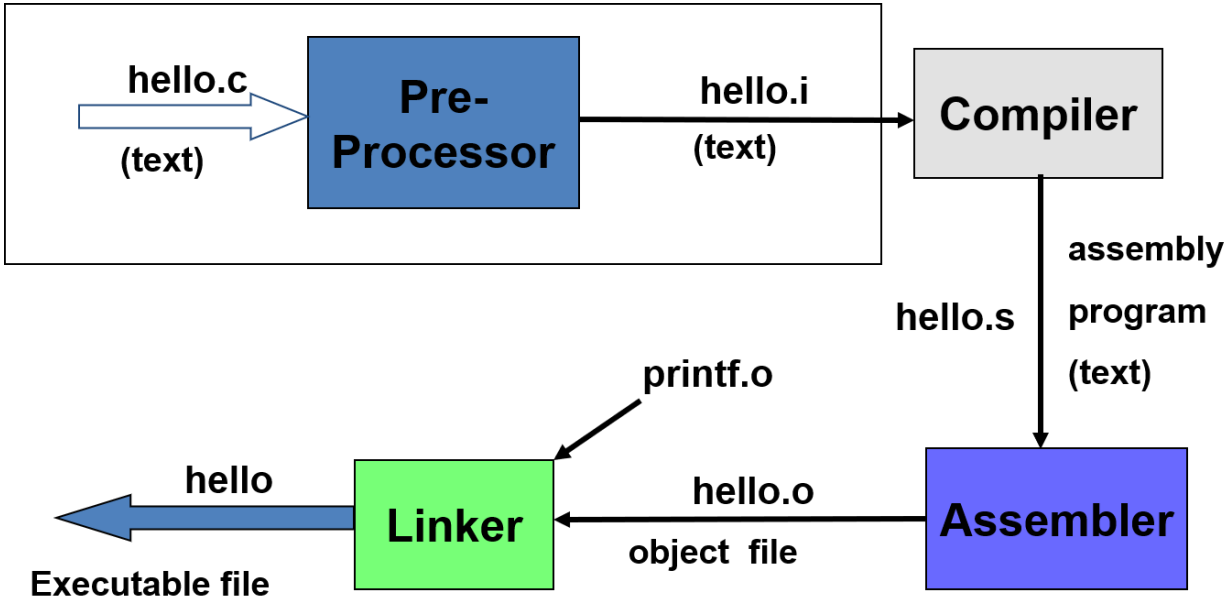
- 13.1 Linking and Linker Introduction
- 13.2 Object File Format
- 13.3 Symbol Resolution and Relocation
- 13.4 Loading and Loader Introduction

13.1 Linking and Linker Introduction

	说明	文件	行	列	项目
73	error LNK1120: 40 个无法解析的外部命令	3Dtext.exe			3Dtext
58	error LNK2019: 无法解析的外部符号 _cvCalibrateCamera2，该符号在函数 "void __cdecl StereoCalib(char const *,int,int,int)" (?StereoCalib@@YAXPBDHHH@Z) 中被引用	ch12_ex12_3.obj			3Dtext
55	error LNK2019: 无法解析的外部符号 _cvComputeCorrespondEpilines，该符号在函数 "void __cdecl StereoCalib(char const *,int,int,int)" (?StereoCalib@@YAXPBDHHH@Z) 中被引用	ch12_ex12_3.obj			3Dtext
36	error LNK2019: 无法解析的外部符号 _cvConvertScale，该符号在函数 "void __cdecl StereoCalib(char const *,int,int,int)" (?StereoCalib@@YAXPBDHHH@Z) 中被引用	ch12_ex12_3.obj			3Dtext
69	error LNK2019: 无法解析的外部符号 _cvCreateImage，该符号在函数 "void __cdecl StereoCalib(char const *,int,int,int)" (?StereoCalib@@YAXPBDHHH@Z) 中被引用	ch12_ex12_3.obj			3Dtext
38	error LNK2019: 无法解析的外部符号 _cvCreateImageHeader，该符号在函数 "void __cdecl StereoCalib(char const *,int,int,int)" (?StereoCalib@@YAXPBDHHH@Z) 中被引用	ch12_ex12_3.obj			3Dtext
54	error LNK2019: 无法解析的外部符号 _cvCreateMat，该符号在函数 "void __cdecl StereoCalib(char const *,int,int,int)" (?StereoCalib@@YAXPBDHHH@Z) 中被引用	ch12_ex12_3.obj			3Dtext
47	error LNK2019: 无法解析的外部符号 _cvCreateStereoBMState，该符号在函数 "void __cdecl StereoCalib(char const *,int,int,int)" (?StereoCalib@@YAXPBDHHH@Z) 中被引用	ch12_ex12_3.obj			3Dtext
65	error LNK2019: 无法解析的外部符号 _cvCvtColor，该符号在函数 "void __cdecl StereoCalib(char const *,int,int,int)" (?StereoCalib@@YAXPBDHHH@Z) 中被引用	ch12_ex12_3.obj			3Dtext
64	error LNK2019: 无法解析的外部符号 _cvDrawChessboardCorners，该符号在函数 "void __cdecl StereoCalib(char const *,int,int,int)" (?StereoCalib@@YAXPBDHHH@Z) 中被引用	ch12_ex12_3.obj			3Dtext
67	error LNK2019: 无法解析的外部符号 _cvFindChessboardCorners，该符号在函数 "void __cdecl StereoCalib(char const *,int,int,int)" (?StereoCalib@@YAXPBDHHH@Z) 中被引用	ch12_ex12_3.obj			3Dtext
61	error LNK2019: 无法解析的外部符号 _cvFindCornerSubPix，该符号在函数 "void __cdecl StereoCalib(char const *,int,int,int)" (?StereoCalib@@YAXPBDHHH@Z) 中被引用	ch12_ex12_3.obj			3Dtext
51	error LNK2019: 无法解析的外部符号 _cvFindFundamentalMat，该符号在函数 "void __cdecl StereoCalib(char const *,int,int,int)" (?StereoCalib@@YAXPBDHHH@Z) 中被引用	ch12_ex12_3.obj			3Dtext
45	error LNK2019: 无法解析的外部符号 _cvFindStereoCorrespondenceBM，该符号在函数 "void __cdecl StereoCalib(char	ch12_ex12_3.obj			3Dtext

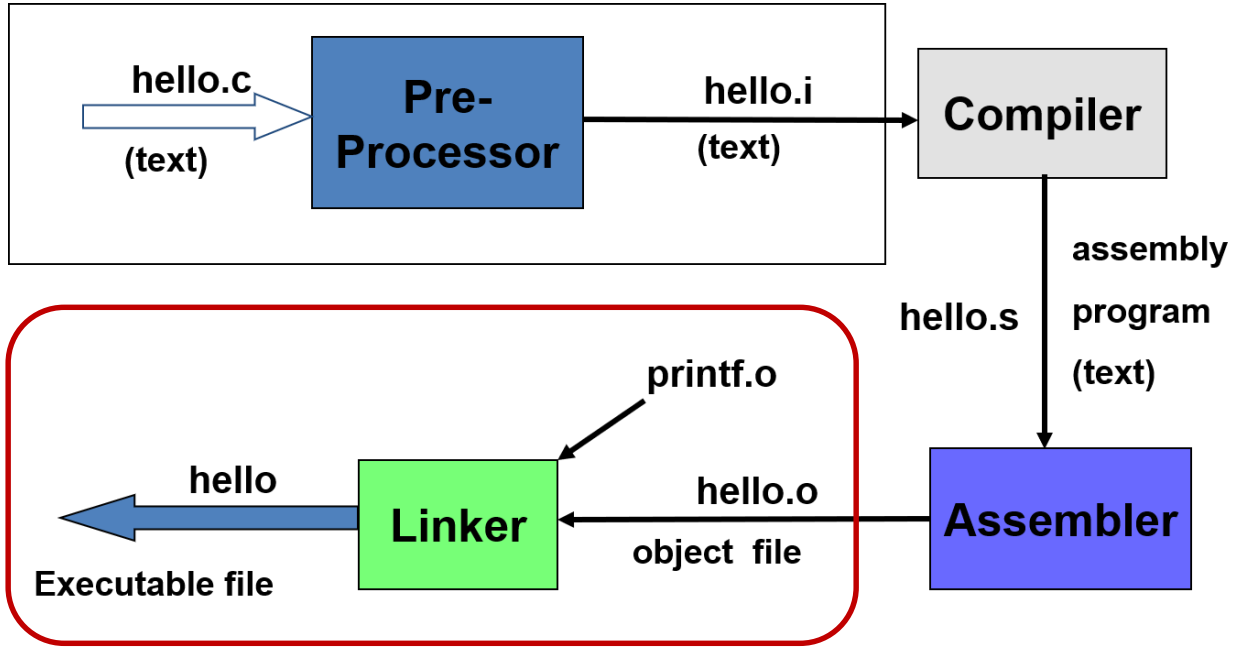
13.1 Linking and Linker Introduction

- Time consumption = compile + link + load + run



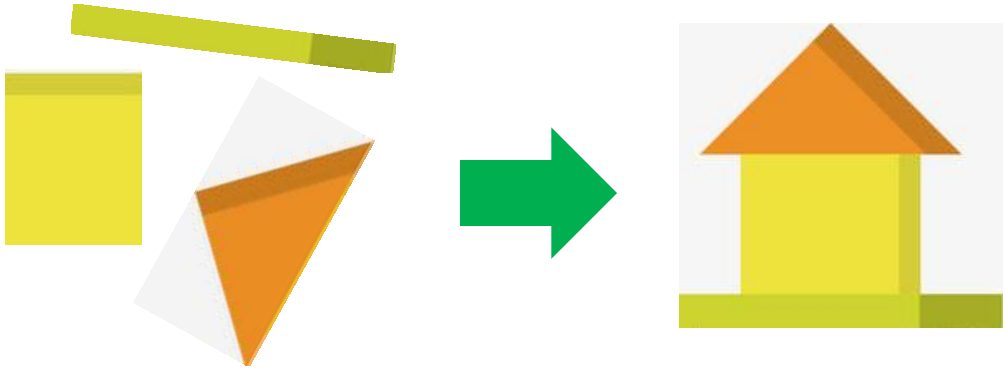
13.1 Linking and Linker Introduction

- Functionality: Object file(s) TO Executable file



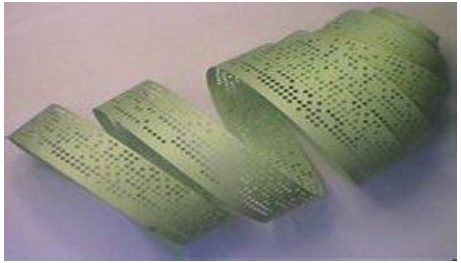
13.1 Linking and Linker Introduction

- Why linker?



13.1 Linking and Linker Introduction

- More on linker



```
MOV     AX, BX
AND     [BP+SI+98BC], DI
MOV     AH, 41
INT     21
MOV     BYTE PTR [96D1], 00
JMP     FB69
JMP     FFC3
AND     AL, 0C
JPE     0129
PUSH    AX
AND     AL, 08
XOR     AL, 00
AND     CX, [SI]
```

- Old times, there is no linker
- Linker is older than assembler and compiler

13.1 Linking and Linker Introduction

- What can linker do?

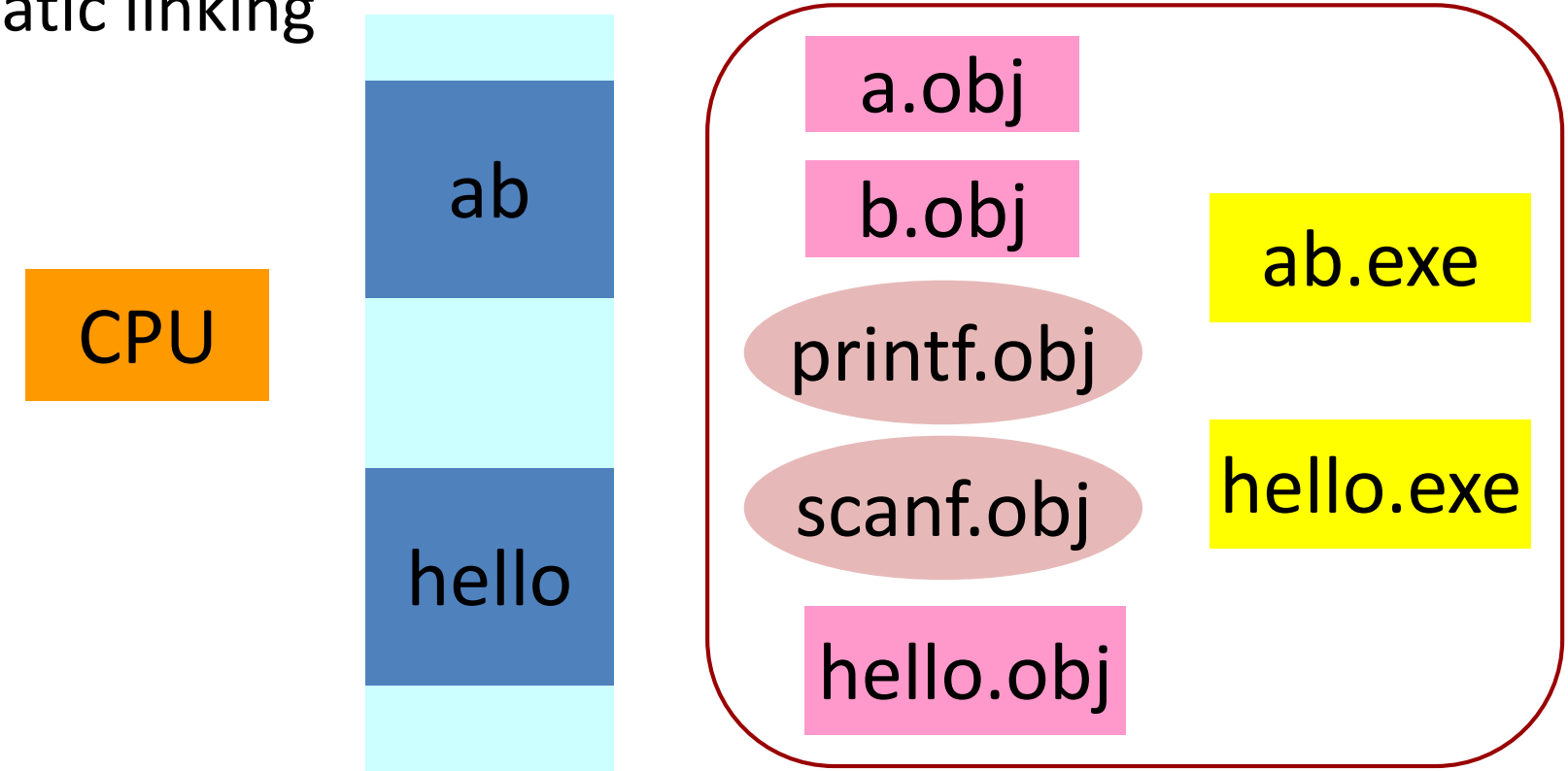
Relocatable **object** files (including library) → Linker → Executable **object** file

- When do linker work?

Static linking	Dynamic linking
☐ After compiling	☐ At loading time
	☐ At running time

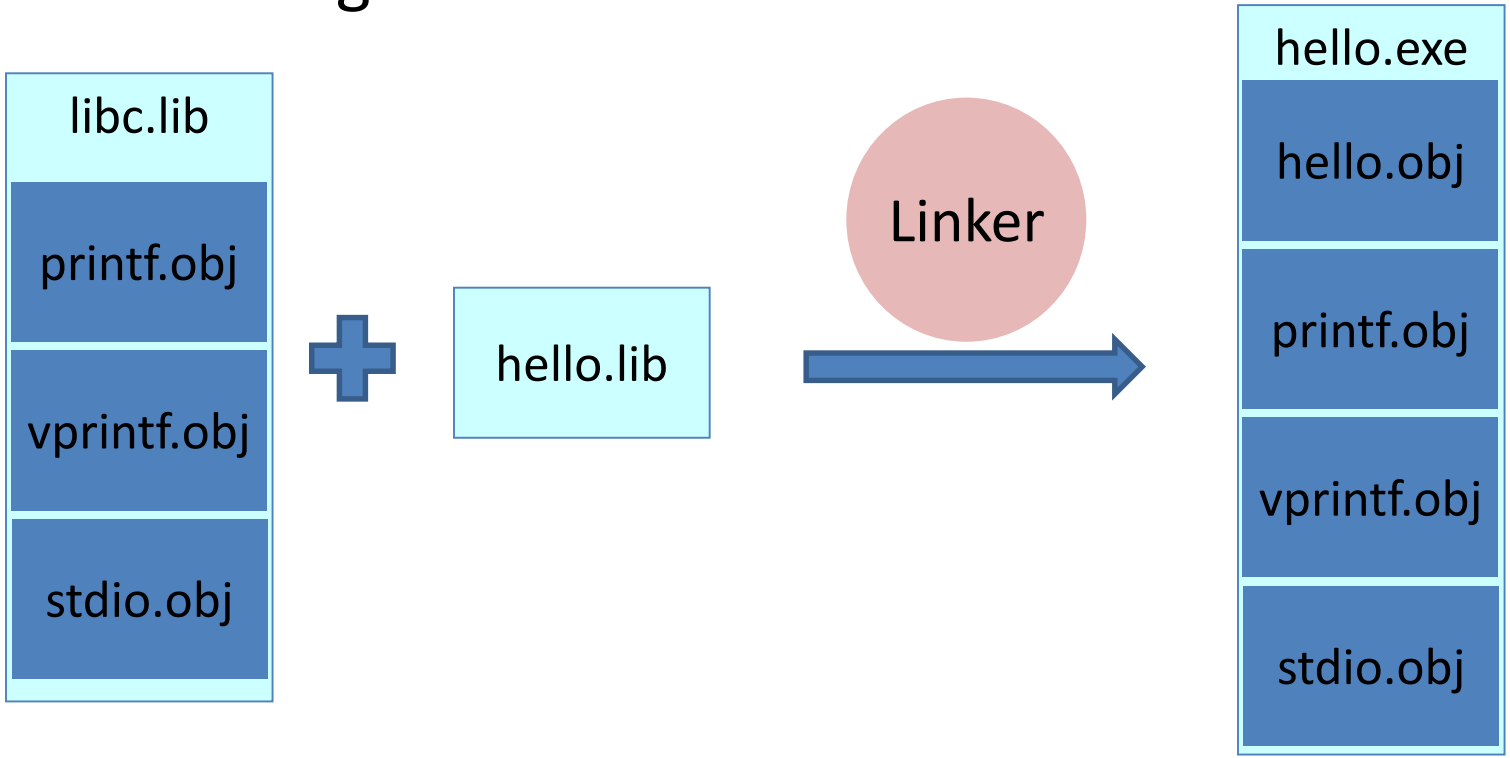
13.1 Linking and Linker Introduction

- Static linking



13.1 Linking and Linker Introduction

- Static linking



13.1 Linking and Linker Introduction

- Why study linker and loader?

Build large programs

Avoid dangerous program errors

How language scoping rules are implemented

Make the whole picture of computer system

Exploit shared libraries

13.1 Linking and Linker Introduction

- Platform

IA32 + ELF + GCC + objdump

IA32 + PE/COFF + VC + dumpbin

13.1 Linking and Linker Introduction

code/link/main.c

```c
/* main.c */
void swap();

int buf[2] = {1, 2};

int main()
{
    swap();
    return 0;
}
```

code/link/main.c

(a) main.c

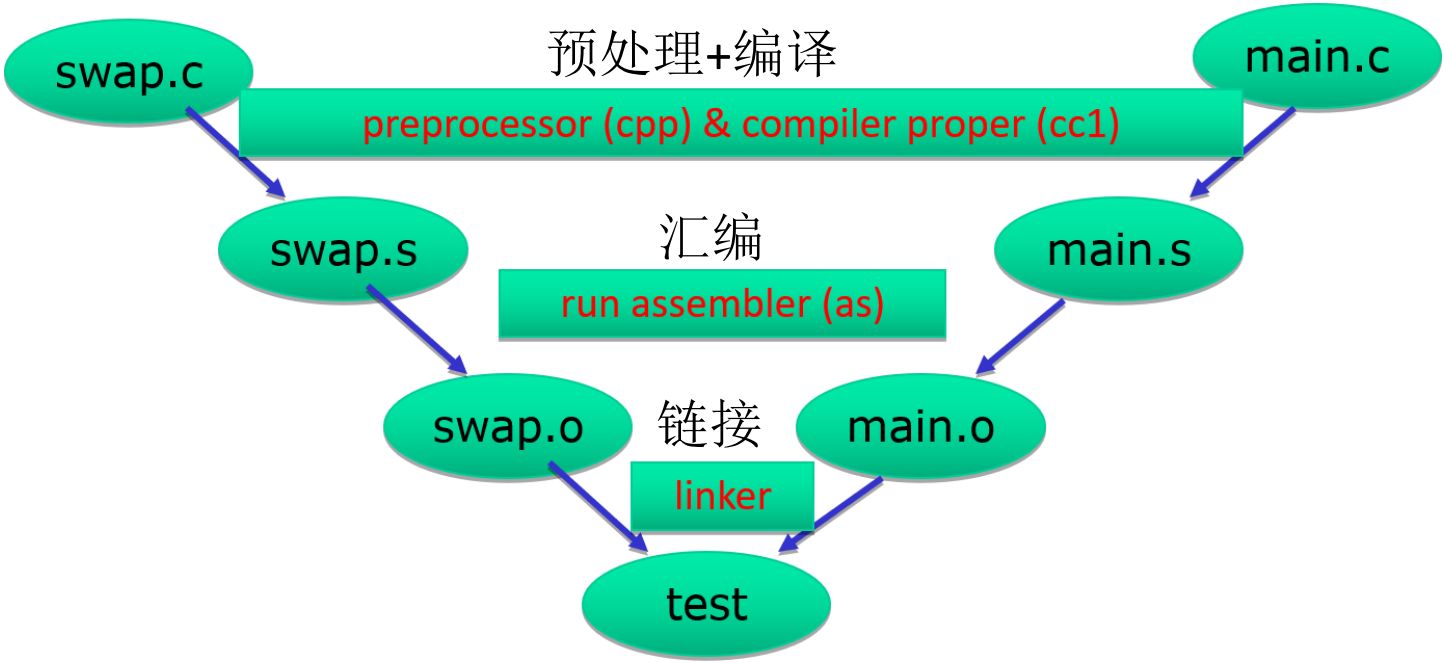
code/link/swap.c

```c
/* swap.c */
extern int buf[];

int *bufp0 = &buf[0];
int *bufp1;

void swap()
{
    int temp;

    bufp1 = &buf[1];
    temp = *bufp0;
    *bufp0 = *bufp1;
    *bufp1 = temp;
}
```

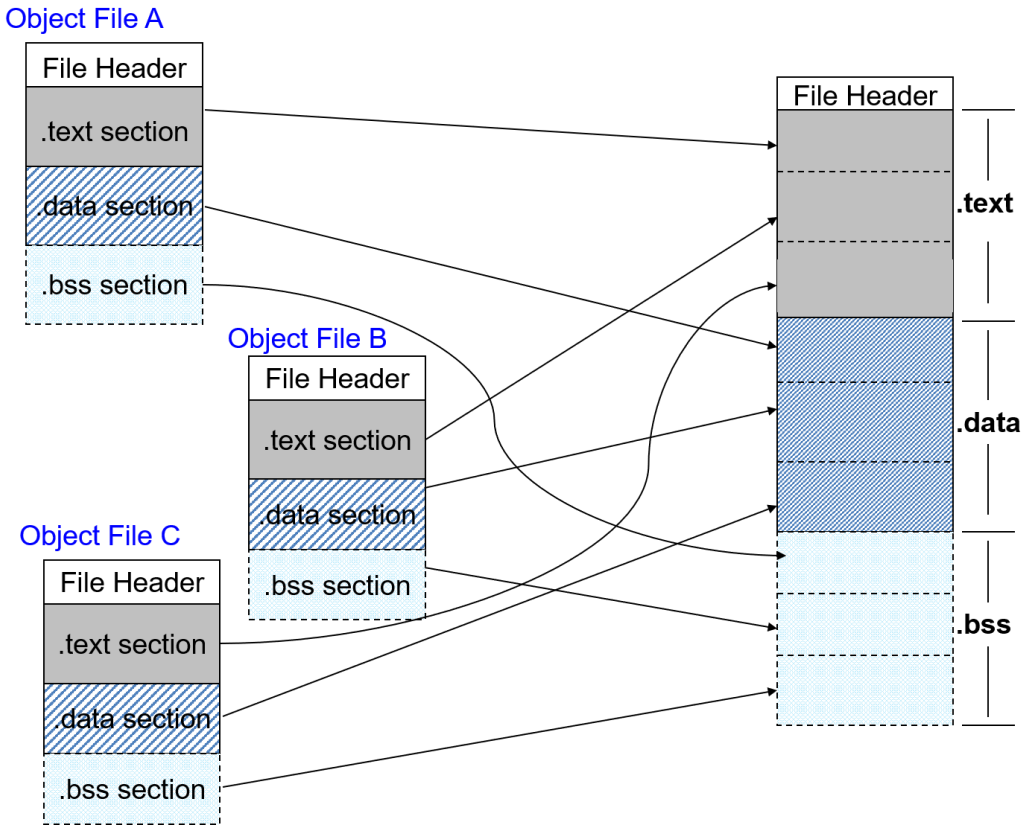
code/link/swap.c

(b) swap.c

13.1 Linking and Linker Introduction



13.1 Linking and Linker Introduction



13.1 Linking and Linker Introduction

- Linking process

Unit13 Linking and Linker

- 13.1 Linking and Linker Introduction
- 13.2 Object File Format
- 13.3 Symbol Resolution and Relocation
- 13.4 Loading and Loader Introduction

13.2 Object File Format

- What is object file?

Executable/ Object File
File Header
.text section
.data section
.bss section

```c
int global_int_var = 84;

int global_uninit_var;

void func1(int i)
{
    printf("%d\n", i);
}

int main(void)
{
    static int static_var = 85;

    static int static_var2;

    int a = 1;
    int b;

    func1(static_var + static_var2 + a + b);

    return 0;
}
```

```
4c01 0600 4cea b75d a305 0000 2100 0000
0000 0000 2e64 7265 6374 7665 0000 0000
0000 0000 3800 0000 0401 0000 0000 0000
0000 0000 0000 0000 000a 1000 2e64 6562
7567 2453 0000 0000 0000 0000 4903 0000
3c01 0000 8504 0000 0000 0000 0200 0000
4800 1042 2e64 6174 6100 0000 0000 0000
0000 0000 0800 0000 9904 0000 0000 0000
0000 0000 0000 0000 4000 30c0 2e74 6578
7400 0000 0000 0000 0000 0000 3000 0000
```

13.2 Object File Format

- Why we put code and data into different sections?
  - Protect the code
  - Improve catch hit rate
  - Share memory

13.2 Object File Format

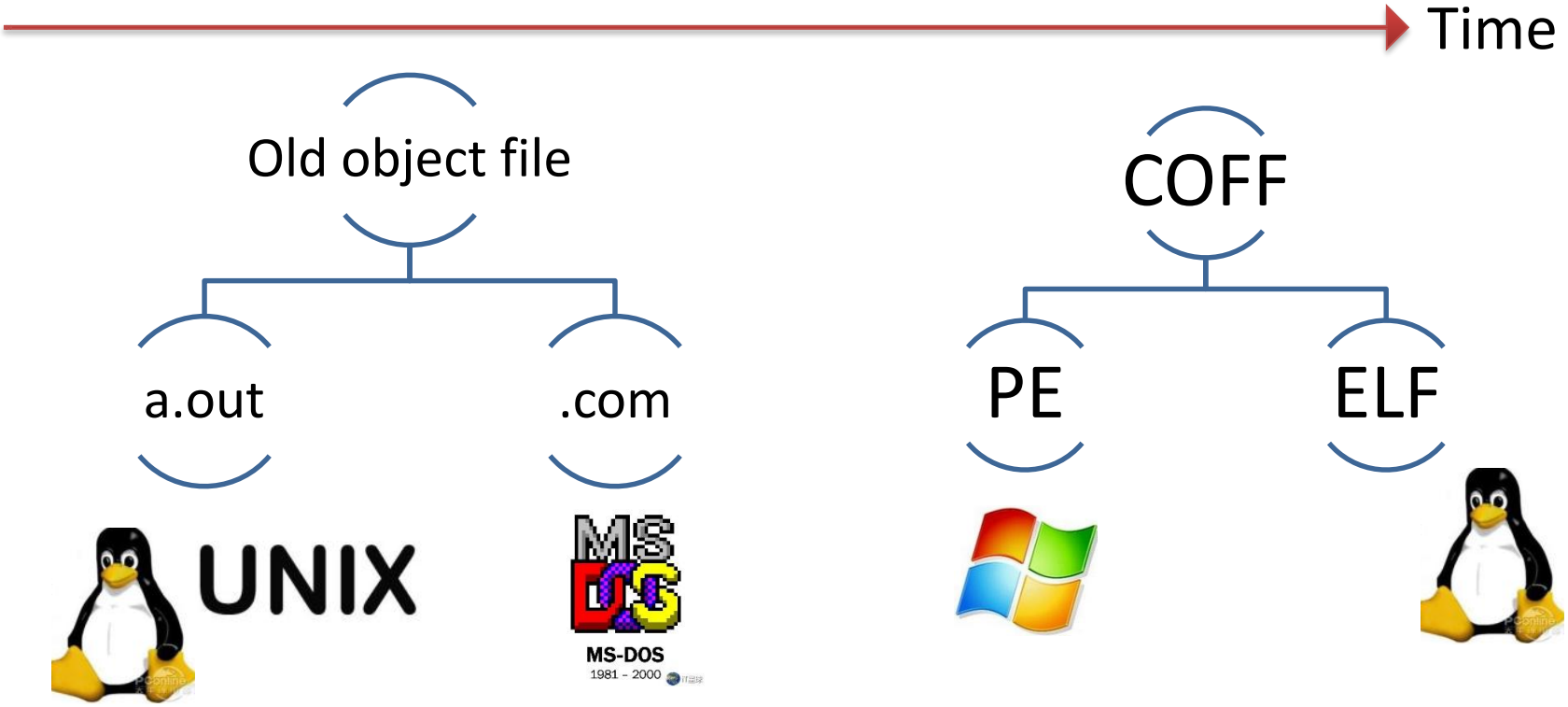
- Object file types

File type	Example
Relocatable object file	Linux .o | Win .obj Linux .a | Win .lib
Executable object file	Linux /bin/bash | Win .exe
Shared object file	Linux .so | Win .dll

Relocatable object files (including library) → Linker → Executable object file

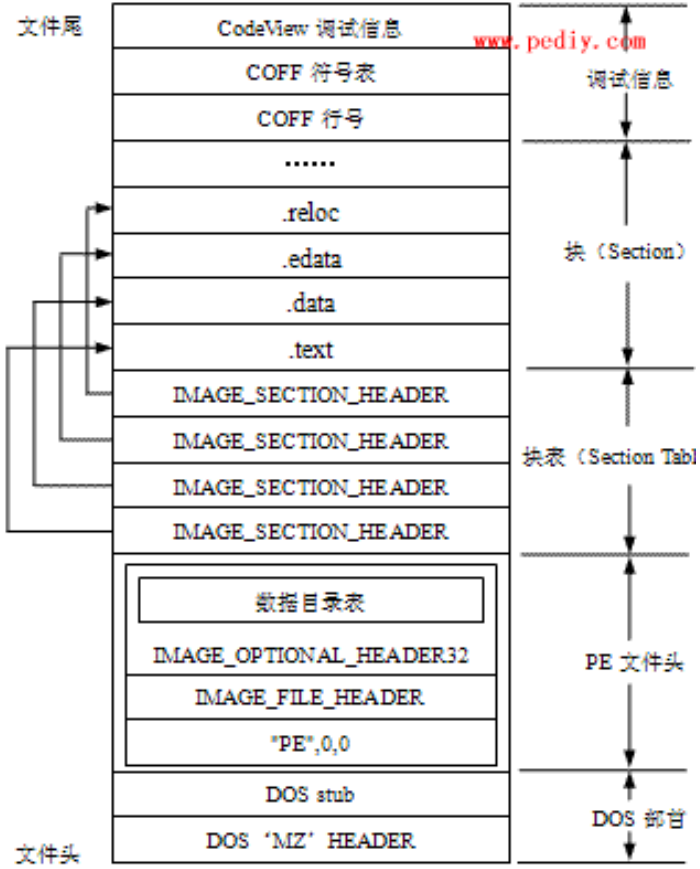
13.2 Object File Format

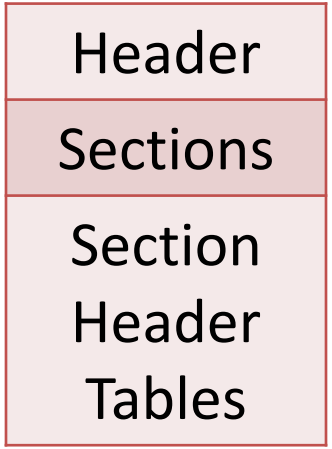
- Object file format standard



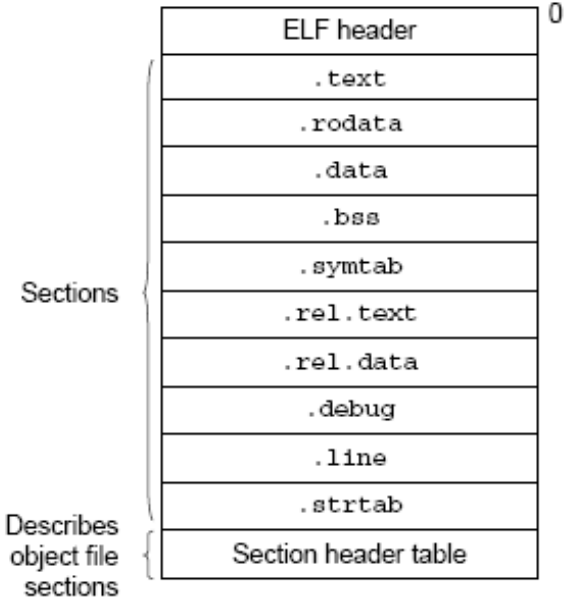
13.2 Object File Format

- PE





- ELF



13.2 Object File Format

- File property
  - Is it executable?
  - Static or dynamic linking?
  - Starting address?
  - Operating system

13.2 Object File Format

- ✓ .text | .data | .bss
- ✓ .strtab | .shstrtab
- ✓ Line, Debug
- ✓ .symtab | Relocation (Rel.text & Rel.data)

13.2 Object File Format

- Describe each of the sections
  - Section name
  - Section length
  - File offset
  - Read only / read & write

13.2 Object File Format

- dumpbin.exe - Microsoft binary dumper

  [C:\Program Files (x86)\Microsoft Visual Studio\VC98\BIN\DUMPBIN.EXE]

- ✓ dumpbin.exe /?
- ✓ dumpbin.exe /all
- ✓ dumpbin.exe /headers
- ✓ dumpbin.exe /disasm
- ✓ dumpbin.exe /relocations

```
/ALL
/ARCH
/ARCHIVEMEMBERS
/DEPENDENTS
/DIRECTIVES
/DISASM
/EXPORTS
/FPO
/HEADERS
/IMPORTS
/LINENUMBERS
/LINKERMEMBER[:{1|2}]
/LOADCONFIG
/OUT:filename
/PDATA
/RAWDATA[:{NONE|BYTES|SHORTS|LONGS}[,#]]
/RELOCATIONS
/SECTION:name
/SUMMARY
/SYMBOLS
```

13.2 Object File Format

code/link/main.c

```c
/* main.c */
void swap();

int buf[2] = {1, 2};

int main()
{
    swap();
    return 0;
}
```

code/link/main.c

(a) main.c

code/link/swap.c

```c
/* swap.c */
extern int buf[];

int *bufp0 = &buf[0];
int *bufp1;

void swap()
{
    int temp;

    bufp1 = &buf[1];
    temp = *bufp0;
    *bufp0 = *bufp1;
    *bufp1 = temp;
}
```

code/link/swap.c

(b) swap.c

13.2 Object File Format

➢ Header

```
typedef struct _IMAGE_FILE_HEADER {
    WORD    Machine;
    WORD    NumberOfSections;
    DWORD   TimeDateStamp;
    DWORD   PointerToSymbolTable;
    DWORD   NumberOfSymbols;
    WORD    SizeOfOptionalHeader;
    WORD    Characteristics;
} IMAGE_FILE_HEADER, *PIMAGE_FILE_HEADER;
```

```
Dump of file D:\work\SLP\2019\week10\linker_exp3\
Debug\main.obj

File Type: COFF OBJECT

FILE HEADER VALUES
             14C machine (i386)
               6 number of sections
        5DBAC710 time date stamp Thu Oct 31
        19:35:44 2019
             584 file pointer to symbol table
              25 number of symbols
               0 size of optional header
               0 characteristics
```

13.2 Object File Format

- Header

```
Dump of file
D:\work\SLP\2019\week10\linker_exp3\Debug\linker_exp3.exe

PE signature found

File Type: EXECUTABLE IMAGE

FILE HEADER VALUES
             14C machine (i386)
               5 number of sections
        5DBAC711 time date stamp Thu Oct 31 19:35:45 2019
               0 file pointer to symbol table
               0 number of symbols
              E0 size of optional header
             10E characteristics
                   Executable
                   Line numbers stripped
                   Symbols stripped
                   32 bit word machine
```

13.2 Object File Format

➢ Optional header

```
typedef struct _IMAGE_OPTIONAL_HEADER {
    //
    // Standard fields.
    //

    WORD    Magic;
    BYTE    MajorLinkerVersion;
    BYTE    MinorLinkerVersion;
    DWORD   SizeOfCode;
    DWORD   SizeOfInitializedData;
    DWORD   SizeOfUninitializedData;
    DWORD   AddressOfEntryPoint;
    DWORD   BaseOfCode;
    DWORD   BaseOfData;
```

```
OPTIONAL HEADER VALUES
             10B magic #
            6.00 linker version
           21000 size of code
            A000 size of initialized data
               0 size of uninitialized data
            1100 RVA of entry point
            1000 base of code
            1000 base of data
```

File format: 32b exe
Linker version
Code size
Data size
Program entry
Code section entry
Data section entry

13.2 Object File Format

- Optional header

```
  DWORD   ImageBase;
  DWORD   SectionAlignment;
  DWORD   FileAlignment;
  WORD    MajorOperatingSystemVersion;
  WORD    MinorOperatingSystemVersion;
  WORD    MajorImageVersion;
  WORD    MinorImageVersion;
  WORD    MajorSubsystemVersion;
  WORD    MinorSubsystemVersion;
  DWORD   Win32VersionValue;
  DWORD   SizeOfImage;
  DWORD   SizeOfHeaders;
  DWORD   CheckSum;
  WORD    Subsystem;
  WORD    DllCharacteristics;
  DWORD   SizeOfStackReserve;
  DWORD   SizeOfStackCommit;
  DWORD   SizeOfHeapReserve;
  DWORD   SizeOfHeapCommit;
  DWORD   LoaderFlags;
  DWORD   NumberOfRvaAndSizes;
  IMAGE_DATA_DIRECTORY DataDirectory[IMAGE_NUMBEROF_DIRECTORY_ENTRIES];
} IMAGE_OPTIONAL_HEADER32, *PIMAGE_OPTIONAL_HEADER32;
```

Alignment

Versions

Sizes

```
400000 image base
  1000 section alignment
  1000 file alignment
  4.00 operating system version
  0.00 image version
  4.00 subsystem version
     0 Win32 version
 2C000 size of image
  1000 size of headers
     0 checksum
     3 subsystem (Windows CUI)
     0 DLL characteristics
100000 size of stack reserve
  1000 size of stack commit
100000 size of heap reserve
  1000 size of heap commit
     0 loader flags
    10 number of directories
```

13.2 Object File Format

- Section header tables and .data

```
typedef struct _IMAGE_SECTION_HEADER {
    BYTE    Name[IMAGE_SIZEOF_SHORT_NAME];
    union {
            DWORD   PhysicalAddress;
            DWORD   VirtualSize;
    } Misc;
    DWORD   VirtualAddress;
    DWORD   SizeOfRawData;
    DWORD   PointerToRawData;
    DWORD   PointerToRelocations;
    DWORD   PointerToLinenumbers;
    WORD    NumberOfRelocations;
    WORD    NumberOfLinenumbers;
    DWORD   Characteristics;
} IMAGE_SECTION_HEADER, *PIMAGE_SECTION_
```

Address

Size

```
SECTION HEADER #3
    .data name
         0 physical address
         0 virtual address
         8 size of raw data
       47A file pointer to raw data
         0 file pointer to relocation table
         0 file pointer to line numbers
         0 number of relocations
         0 number of line numbers
C0300040 flags
         Initialized Data
         4 byte align
         Read Write

RAW DATA #3
  00000000: 01 00 00 00 02 00 00 00
```

13.2 Object File Format

- .text

code/link/main.c

```c
/* main.c */
void swap();

int buf[2] = {1, 2};

int main()
{
    swap();
    return 0;
}
```

code/link/main.c

```
_main:
00000000: 55                push        ebp
00000001: 8B EC             mov         ebp,esp
00000003: 83 EC 40          sub         esp,40h
00000006: 53                push        ebx
00000007: 56                push        esi
00000008: 57                push        edi
00000009: 8D 7D C0          lea         edi,[ebp-40h]
0000000C: B9 10 00 00 00    mov         ecx,10h
00000011: B8 CC CC CC CC    mov         eax,0CCCCCCCCh
00000016: F3 AB             rep stos    dword ptr [edi]
00000018: E8 00 00 00 00    call        0000001D
0000001D: 33 C0             xor         eax,eax
0000001F: 5F                pop         edi
00000020: 5E                pop         esi
00000021: 5B                pop         ebx
00000022: 83 C4 40          add         esp,40h
00000025: 3B EC             cmp         ebp,esp
00000027: E8 00 00 00 00    call        0000002C
0000002C: 8B E5             mov         esp,ebp
0000002E: 5D                pop         ebp
0000002F: C3                ret
```

13.2 Object File Format

Id offset location type range name

- Symbol tables

code/link/main.c

```c
/* main.c */
void swap();

int buf[2] = {1, 2};

int main()
{
    swap();
    return 0;
}
```

code/link/main.c

```
COFF SYMBOL TABLE
000 00000012 DEBUG  notype       Filename     | .file
    D:\work\SLP\2019\week10\linker_exp3\main.c
004 000A1FE8 ABS    notype       Static       | @comp.id
005 00000000 SECT1  notype       Static       | .drectve
    Section length   38, #relocs    0, #linenums    0, checksum        0
    Relocation CRC 00000000
008 00000000 SECT2  notype       Static       | .debug$S
    Section length  32A, #relocs    2, #linenums    0, checksum        0
    Relocation CRC 4B6CD82D
00B 00000000 SECT3  notype       Static       | .data
    Section length    8, #relocs    0, #linenums    0, checksum 66A3C815
    Relocation CRC 00000000
00E 00000000 SECT3  notype       External     | _buf
00F 00000000 SECT4  notype       Static       | .text
    Section length   30, #relocs    2, #linenums    4, checksum D3EA8E67,
    Relocation CRC 6BC48EBE
012 00000000 DEBUG  notype       Filename     | .file
    d:\work\slp\2019\week10\linker_exp3\main.c
016 00000000 SECT4  notype ()    External     | _main
    tag index 0000001A size 00000030 lines 000004C6 next function 00000000
018 00000000 UNDEF  notype ()    External     | _swap
019 00000000 UNDEF  notype ()    External     | __chkesp
01A 00000000 SECT4  notype       BeginFunction | .bf
    line# 0006 end 00000000
01C 00000004 SECT4  notype       .bf or.ef    | .lf
01D 00000030 SECT4  notype       EndFunction  | .ef
    line# 0009
01F 00000000 SECT5  notype       Static       | .debug$S
    Section length   4E, #relocs    2, #linenums    0, checksum        0,
    Relocation CRC 13D2BF73
022 00000000 SECT6  notype       Static       | .debug$T
    Section length   44, #relocs    0, #linenums    0, checksum        0
    Relocation CRC 00000000
```

13.2 Object File Format

➢ Relocation

```
SECTION HEADER #4
   .text name
       0 physical address
       0 virtual address
      30 size of raw data
     482 file pointer to raw data
     4B2 file pointer to relocation table
     4C6 file pointer to line numbers
       2 number of relocations
       4 number of line numbers
60501020 flags
         Code
         Communal; sym= _main
         16 byte align
         Execute Read

RAW DATA #4
  00000000: 55 8B EC 83 EC 40 53 56 57 8D 7D C0 B9 10 00 00
  00000010: 00 B8 CC CC CC CC F3 AB E8 00 00 00 00 33 C0 5F
  00000020: 5E 5B 83 C4 40 3B EC E8 00 00 00 00 8B E5 5D C3

RELOCATIONS #4
                                                  Symbol    Symbol
 Offset    Type              Applied To           Index     Name
 --------  ----------------  -----------------    --------  ------
 00000019  REL32                     00000000         18  _swap
 00000028  REL32                     00000000         19  __chkesp
```

```
_main:
00000000: 55                  push        ebp
00000001: 8B EC               mov         ebp,esp
00000003: 83 EC 40            sub         esp,40h
00000006: 53                  push        ebx
00000007: 56                  push        esi
00000008: 57                  push        edi
00000009: 8D 7D C0            lea         edi,[ebp-40h]
0000000C: B9 10 00 00 00      mov         ecx,10h
00000011: B8 CC CC CC CC      mov         eax,0CCCCCCCCh
00000016: F3 AB               rep stos    dword ptr [edi]
00000018: E8 00 00 00 00      call        0000001D
0000001D: 33 C0               xor         eax,eax
0000001F: 5F                  pop         edi
00000020: 5E                  pop         esi
00000021: 5B                  pop         ebx
00000022: 83 C4 40            add         esp,40h
00000025: 3B EC               cmp         ebp,esp
00000027: E8 00 00 00 00      call        0000002C
0000002C: 8B E5               mov         esp,ebp
0000002E: 5D                  pop         ebp
0000002F: C3                  ret
```

Unit13 Linking and Linker

- 13.1 Linking and Linker Introduction
- 13.2 Object File Format
- 13.3 Symbol Resolution and Relocation
- 13.4 Loading and Loader Introduction

13.3 Symbol Resolution and Relocation

- 13.3.0 Review
- 13.3.1 Symbol Resolution
- 13.3.2 Relocation

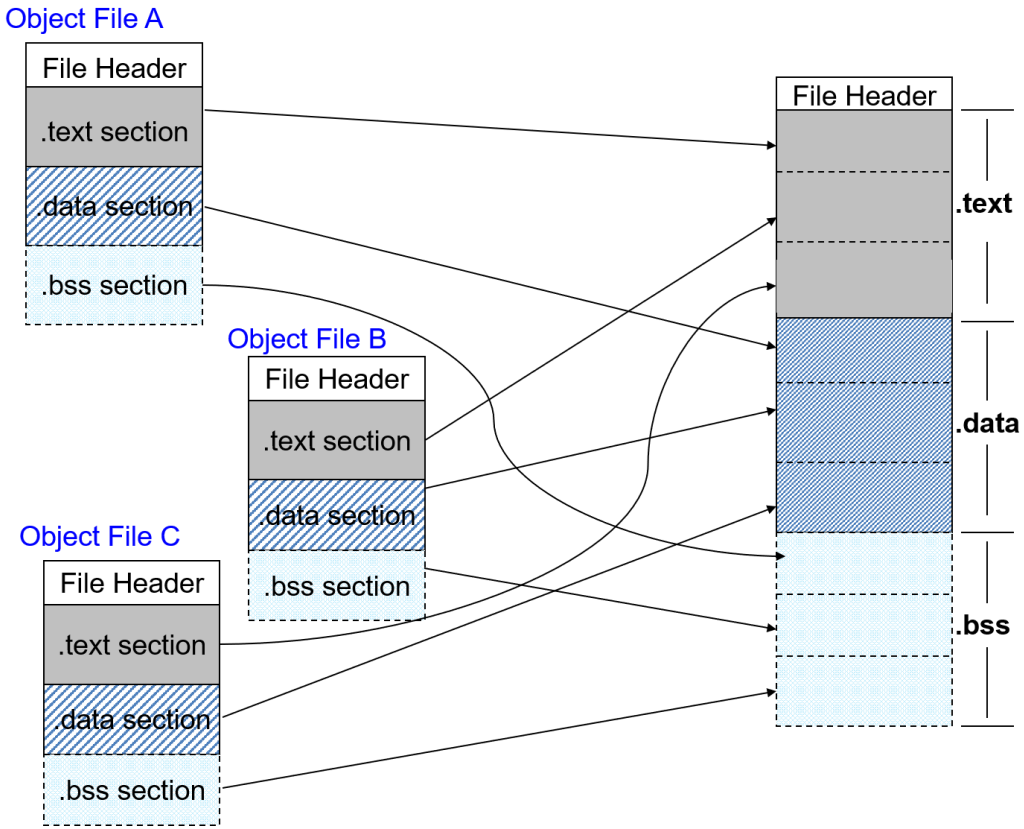
13.3.0 Review

- Linking process

Symbol Resolution

Combination

Relocation



13.3.0 Review

- Definition V.S. Declare V.S. Reference
  - Data

code/link/main.c

```c
/* main.c */
void swap();

int buf[2] = {1, 2};

int main()
{
    swap();
    return 0;
}
```

definition

code/link/main.c

(a) main.c

code/link/swap.c

```c
/* swap.c */
extern int buf[];

int *bufp0 = &buf[0];
int *bufp1;

void swap()
{
    int temp;

    bufp1 = &buf[1];
    temp = *bufp0;
    *bufp0 = *bufp1;
    *bufp1 = temp;
}
```

declare

reference

code/link/swap.c

(b) swap.c

13.3.0 Review

- Definition V.S. Declare V.S. Reference
  - Code

code/link/main.c

```c
/* main.c */
void swap();

int buf[2] = {1, 2};

int main()
{
    swap();
    return 0;
}
```

declare

reference

code/link/main.c

(a) main.c

code/link/swap.c

```c
/* swap.c */
extern int buf[];

int *bufp0 = &buf[0];
int *bufp1;

void swap()
{
    int temp;

    bufp1 = &buf[1];
    temp = *bufp0;
    *bufp0 = *bufp1;
    *bufp1 = temp;
}
```

definition

code/link/swap.c

(b) swap.c

13.3.0 Review

- static keyword: storage and linkage (存储域和作用域)

Where declared	Static modifies	Static applied?	Storage class	Linkage class
Inside a block	Storage class	Yes	Static	None
Inside a block	Storage class	No	Auto	None
Outside any block	Linkage class	Yes	Static	Internal
Outside any block	Linkage class	No	Static	External

13.3.0 Review

- main

```
[root@localhost eg2]# more Makefile
main:main.o swap.o
        ld -o main -static /usr/lib/crt1.o /usr/lib/crti.o /usr/lib/gcc-lib/i386
-redhat-linux/3.2.2/crtbeginT.o -L/usr/lib/gcc-lib/i386-redhat-linux/3.2.2 -L/us
r/lib -L /lib ./main.o ./swap.o --start-group -lgcc -lgcc_eh -lc --end-group /us
r/lib/gcc-lib/i386-redhat-linux/3.2.2/crtend.o /usr/lib/crtn.o
```

13.3 Symbol Resolution and Relocation

- 13.3.0 Review
- 13.3.1 Symbol Resolution
- 13.3.2 Relocation

13.3.1 Symbol Resolution

- Purpose is to associate each symbol reference with exactly one symbol definition according to symtab

```
typedef struct _IMAGE_SYMBOL {
    union {
        BYTE    ShortName[8];
        struct {
            DWORD   Short;     // if 0, use LongName
            DWORD   Long;      // offset into string table
        } Name;
        DWORD   LongName[2];    // PBYTE [2]
    } N;
    DWORD   Value;
    SHORT   SectionNumber;
    WORD    Type;
    BYTE    StorageClass;
    BYTE    NumberOfAuxSymbols;
} IMAGE_SYMBOL;
typedef IMAGE_SYMBOL UNALIGNED *PIMAGE_SYMBOL;
```

Id offset location type range name

```
00E 00000000 SECT3  notype       External     | _buf
00F 00000000 SECT4  notype       Static       | .text
    Section length   30, #relocs    2, #linenums    4, checksum D3EA8E67,
    Relocation CRC 6BC48EBE
012 00000000 DEBUG  notype       Filename     | .file
    d:\work\slp\2019\week10\linker_exp3\main.c
016 00000000 SECT4  notype ()    External     | _main
    tag index 0000001A size 00000030 lines 000004C6 next function 00000000
018 00000000 UNDEF  notype ()    External     | _swap
```

13.3.1 Symbol Resolution

- Global symbol, External symbol and Local symbol

code/link/main.c

```c
/* main.c */
void swap();

int buf[2] = {1, 2};

int main()
{
    swap();
    return 0;
}
```

code/link/main.c

(a) main.c

code/link/swap.c

```c
/* swap.c */
extern int buf[];

int *bufp0 = &buf[0];
int *bufp1;

void swap()
{
    int temp;

    bufp1 = &buf[1];
    temp = *bufp0;
    *bufp0 = *bufp1;
    *bufp1 = temp;
}
```

code/link/swap.c

Static int *bufp1;

(b) swap.c

13.3.1 Symbol Resolution

- Symbol tables

code/link/main.c

```c
/* main.c */
void swap();

int buf[2] = {1, 2};

int main()
{
    swap();
    return 0;
}
```

code/link/main.c

```
COFF SYMBOL TABLE
000 00000012 DEBUG  notype       Filename     | .file
    D:\work\SLP\2019\week10\linker_exp3\main.c
004 000A1FE8 ABS    notype       Static       | @comp.id
005 00000000 SECT1  notype       Static       | .drectve
    Section length   38, #relocs    0, #linenums    0, checksum        0
    Relocation CRC 00000000
008 00000000 SECT2  notype       Static       | .debug$S
    Section length  32A, #relocs    2, #linenums    0, checksum        0
    Relocation CRC 4B6CD82D
00B 00000000 SECT3  notype       Static       | .data
    Section length    8, #relocs    0, #linenums    0, checksum 66A3C815
    Relocation CRC 00000000
00E 00000000 SECT3  notype       External     | _buf
00F 00000000 SECT4  notype       Static       | .text
    Section length   30, #relocs    2, #linenums    4, checksum D3EA8E67, sel
    Relocation CRC 6BC48EBE
012 00000000 DEBUG  notype       Filename     | .file
    d:\work\slp\2019\week10\linker_exp3\main.c
016 00000000 SECT4  notype ()    External     | _main
    tag index 0000001A size 00000030 lines 000004C6 next function 00000000
018 00000000 UNDEF  notype ()    External     | _swap
019 00000000 UNDEF  notype ()    External     | __chkesp
01A 00000000 SECT4  notype       BeginFunction | .bf
    line# 0006 end 00000000
01C 00000004 SECT4  notype       .bf or.ef    | .lf
01D 00000030 SECT4  notype       EndFunction  | .ef
    line# 0009
01F 00000000 SECT5  notype       Static       | .debug$S
    Section length   4E, #relocs    2, #linenums    0, checksum        0, sel
    Relocation CRC 13D2BF73
022 00000000 SECT6  notype       Static       | .debug$T
    Section length   44, #relocs    0, #linenums    0, checksum        0
    Relocation CRC 00000000
```

13.3.1 Symbol Resolution

code/link/swap.c

```c
/* swap.c */
extern int buf[];

int *bufp0 = &buf[0];
int *bufp1;

void swap()
{
    int temp;

    bufp1 = &buf[1];
    temp = *bufp0;
    *bufp0 = *bufp1;
    *bufp1 = temp;
}
```

code/link/swap.c

```
COFF SYMBOL TABLE
000 00000000 DEBUG  notype       Filename     | .file
    D:\work\SLP\2019\week10\linker_exp3\swap.c
004 000A1FE8 ABS    notype       Static       | @comp.id
005 00000000 SECT1  notype       Static       | .drectve
    Section length   38, #relocs    0, #linenums    0, checksum        0
    Relocation CRC 00000000
008 00000000 SECT2  notype       Static       | .debug$S
    Section length  352, #relocs    6, #linenums    0, checksum        0
    Relocation CRC C6429972
00B 00000000 SECT3  notype       Static       | .data
    Section length    4, #relocs    1, #linenums    0, checksum        0
    Relocation CRC 4DE21D67
00E 00000000 SECT3  notype       External     | _bufp0
00F 00000000 UNDEF  notype       External     | _buf
010 00000004 UNDEF  notype       External     | _bufp1
011 00000000 SECT4  notype       Static       | .text
    Section length   4D, #relocs    6, #linenums    6, checksum 6A22A5A4, s
    Relocation CRC 1CD93523
014 00000000 SECT4  notype ()    External     | _swap
    tag index 00000016 size 0000004D lines 00000561 next function 00000000
016 00000000 SECT4  notype       BeginFunction | .bf
    line# 0007 end 00000000
018 00000006 SECT4  notype       .bf or.ef    | .lf
019 0000004D SECT4  notype       EndFunction  | .ef
    line# 000e
01B 00000000 SECT5  notype       Static       | .debug$S
    Section length   5F, #relocs    2, #linenums    0, checksum        0, s
    Relocation CRC B9C129AE
01E 00000000 SECT6  notype       Static       | .debug$T
    Section length   44, #relocs    0, #linenums    0, checksum        0
    Relocation CRC 00000000
```

13.3.1 Symbol Resolution

➢ Global symbol types

Strong symbol

Weak symbol

Local symbol

main.c

```c
int buf[2] = {1, 2};
void swap();

int main()
{
  swap();
  return 0;
}
```

swap.c

```c
extern int buf[];

int *bufp0 = &buf[0];
static int *bufp1;

void swap()
{
  int temp;

  bufp1 = &buf[1];
  temp = *bufp0;
  *bufp0 = *bufp1;
  *bufp1 = temp;
}
```

Reference

Local variable

Strong symbol

- Function name
- Initialized global variable

Weak symbol

- Uninitialized global variable

13.3.1 Symbol Resolution

- Rule

Rule1: Multiple strong symbols are not allowed

```
int buf[2] = {1, 2};
int buf[2] = {1, 2};
```

```
--------------------Configuration: linker_exp1 - Win32 Debug--------------------
Compiling...
main.cpp
D:\work\SLP\2019\week10\linker_exp1\main.cpp(6) : error C2374: 'buf' : redefinition; multiple initialization
        D:\work\SLP\2019\week10\linker_exp1\main.cpp(5) : see declaration of 'buf'
swap.cpp
Error executing cl.exe.

linker_exp1.exe - 1 error(s), 0 warning(s)
```

13.3.1 Symbol Resolution

- Rule

Rule2: Given a strong symbol and multiple weak symbols, choose the strong symbol.

```
int buf[2] = {1, 2};
int buf[];
int buf[];
```

```
--------------------Configuration: linker_exp1 - Win32 Debug--------------------
Compiling...
main.cpp
Linking...
LINK : LNK6004: Debug/linker_exp1.exe not found or not built by the last incremental link; performing full link

linker_exp1.exe - 0 error(s), 0 warning(s)
```

Rule3: Given multiple weak symbols, choose any of the weak symbols.

13.3.1 Symbol Resolution

```
Compiling...
bar.cpp
Linking...
bar.obj : error LNK2005: _main already defined in foo.obj
Debug/link_exp3.exe : fatal error LNK1169: one or more multiply defined symbols found
Error executing link.exe.

link_exp3.exe - 2 error(s), 0 warning(s)
```

```c
/* foo1.c */
int main()
{
    return 0;
}
```

```c
/* bar1.c */
int main()
{
    return 0;
}
```

13.3.1 Symbol Resolution

```
Compiling...
foo.cpp
bar.cpp
Linking...
bar.obj : error LNK2005: "int x" (?x@@3HA) already defined in foo.obj
Debug/link_exp3.exe : fatal error LNK1169: one or more multiply defined symbols found
Error executing link.exe.

link_exp3.exe - 2 error(s), 0 warning(s)
```

```c
/* foo2.c */
int x = 15213;

int main()
{
    return 0;
}
```

```c
/* bar2.c */
int x = 15213;

void f()
{
}
```

13.3.1 Symbol Resolution

"D:\work\SLP\2019\week1

```
x=15212
Press any key to continue
```

```c
/* foo3.c */
#include <stdio.h>
void f(void);

int x = 15213;

int main()
{
    f();
    printf("x = %d\n", x);
    return 0;
```

```c
/* bar3.c */
int x;

void f()
{
    x = 15212;
}
```

13.3.1 Symbol Resolution

"D:\work\SLP\2019\week1

```
x=15212
Press any key to continue
```

```c
/* foo3.c */
#include <stdio.h>
void f(void);

int x ;

int main()
{    x = 15213;
     f();
     printf("x = %d\n", x);
     return 0;
```

```c
/* bar3.c */
int x;

void f()
{
     x = 15212;
}
```

13.3.1 Symbol Resolution

/*foo.c*/

```c
#include <stdio.h>
int x=15213;
int y= 15212;
void f(void);

int main()
{
    f();
    printf("x=%d,y=%d\n", x,y);
    return 0;
}
```

"D:\work\SLP\2019\wee

```
x=15213, y=15212
Press any key to continue
```

/*bar.c*/

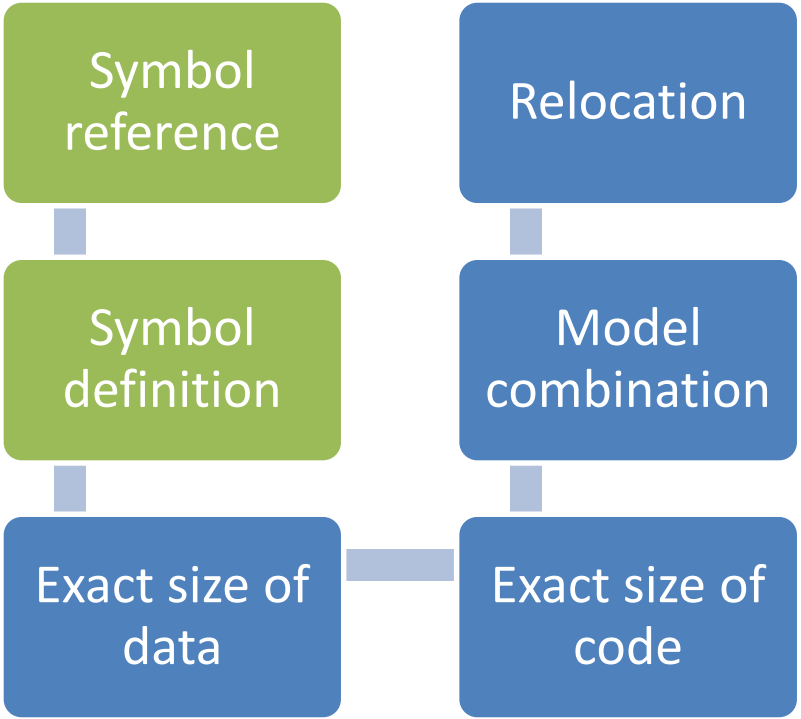
```c
double x;
void f()
{
    x = -0.0;
}
```

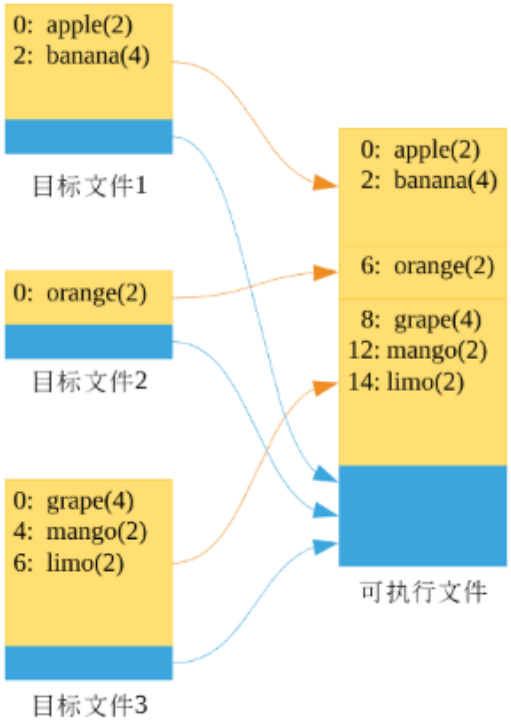
13.3 Symbol Resolution and Relocation

- 13.3.0 Review
- 13.3.1 Symbol Resolution
- 13.3.2 Relocation

13.3.2 Relocation

- Relocation: modify memory locations to each reference





13.3.2 Relocation

- The relocation table

```
typedef struct _IMAGE_RELOCATION {
    union {
        DWORD   VirtualAddress;
        DWORD   RelocCount;          // Set to the real count when IMAGE_S
    } DUMMYUNIONNAME;
    DWORD   SymbolTableIndex;
    WORD    Type;
} IMAGE_RELOCATION;
typedef IMAGE_RELOCATION UNALIGNED *PIMAGE_RELOCATION;
```

```
RELOCATIONS #4
                                                  Symbol    Symbol
 Offset    Type              Applied To           Index     Name
 --------  ----------------  -----------------    --------  ------
 00000019  REL32                      00000000         18  _swap
 00000028  REL32                      00000000         19  __chkesp
```

13.3.2 Relocation

- Type

```
//
// I386 relocation types.
//
#define IMAGE_REL_I386_ABSOLUTE      0x0000  // Reference is absolute, no relocation is necessary
#define IMAGE_REL_I386_DIR16         0x0001  // Direct 16-bit reference to the symbols virtual address
#define IMAGE_REL_I386_REL16         0x0002  // PC-relative 16-bit reference to the symbols virtual address
#define IMAGE_REL_I386_DIR32         0x0006  // Direct 32-bit reference to the symbols virtual address
#define IMAGE_REL_I386_DIR32NB       0x0007  // Direct 32-bit reference to the symbols virtual address, base not included
#define IMAGE_REL_I386_SEG12         0x0009  // Direct 16-bit reference to the segment-selector bits of a 32-bit virtual addre
#define IMAGE_REL_I386_SECTION       0x000A
#define IMAGE_REL_I386_SECREL        0x000B
#define IMAGE_REL_I386_TOKEN         0x000C  // clr token
#define IMAGE_REL_I386_SECREL7       0x000D  // 7 bit offset from base of section containing target
#define IMAGE_REL_I386_REL32         0x0014  // PC-relative 32-bit reference to the symbols virtual address
```

13.3.2 Relocation

code/link/main.c

```c
/* main.c */
void swap();

int buf[2] = {1, 2};

int main()
{
    swap();
    return 0;
}
```

code/link/main.c

```
RELOCATIONS #4
                                                  Symbol    Symbol
 Offset    Type              Applied To           Index     Name
 --------  ----------------  -----------------    --------  ------
 00000019  REL32                      00000000        18   _swap
 00000028  REL32                      00000000        19   __chkesp
```

13.3.2 Relocation

```
_main:
 00000000: 55                 push        ebp
 00000001: 8B EC              mov         ebp,esp
 00000003: 83 EC 40           sub         esp,40h
 00000006: 53                 push        ebx
 00000007: 56                 push        esi
 00000008: 57                 push        edi
 00000009: 8D 7D C0           lea         edi,[ebp-40h]
 0000000C: B9 10 00 00 00     mov         ecx,10h
 00000011: B8 CC CC CC CC     mov         eax,0CCCCCCCCh
 00000016: F3 AB              rep stos    dword ptr [edi]
 00000018: E8 00 00 00 00     call        0000001D
 0000001D: 33 C0              xor         eax,eax
 0000001F: 5F                 pop         edi
 00000020: 5E                 pop         esi
 00000021: 5B                 pop         ebx
 00000022: 83 C4 40           add         esp,40h
 00000025: 3B EC              cmp         ebp,esp
 00000027: E8 00 00 00 00     call        0000002C
 0000002C: 8B E5              mov         esp,ebp
 0000002E: 5D                 pop         ebp
 0000002F: C3                 ret
```

```
RELOCATIONS #4

                                                  Symbol    Symbol
 Offset    Type              Applied To           Index     Name
 --------  ----------------  -----------------    --------  ------
 00000019  REL32                     00000000         18   _swap
 00000028  REL32                     00000000         19   __chkesp
```

13.3.2 Relocation

code/link/swap.c

```c
/* swap.c */
extern int buf[];

int *bufp0 = &buf[0];
int *bufp1;

void swap()
{
    int temp;

    bufp1 = &buf[1];
    temp = *bufp0;
    *bufp0 = *bufp1;
    *bufp1 = temp;
}
```

code/link/swap.c

```
RELOCATIONS #3
                                                  Symbol    Symbol
 Offset    Type              Applied To           Index     Name
 --------  ----------------  -----------------   --------  ------
 00000000  DIR32                      00000000          F  _buf
RELOCATIONS #4
                                                 Symbol    Symbol
 Offset    Type              Applied To          Index     Name
 --------  ----------------  -----------------  --------  ------
 0000001A  DIR32                      00000000        10  _bufp1
 0000001E  DIR32                      00000004         F  _buf
 00000023  DIR32                      00000000         E  _bufp0
 0000002E  DIR32                      00000000         E  _bufp0
 00000033  DIR32                      00000000        10  _bufp1
 0000003D  DIR32                      00000000        10  _bufp1
```

13.3.2 Relocation

```
_swap:
  00000000: 55                  push        ebp
  00000001: 8B EC               mov         ebp,esp
  00000003: 83 EC 44            sub         esp,44h
  00000006: 53                  push        ebx
  00000007: 56                  push        esi
  00000008: 57                  push        edi
  00000009: 8D 7D BC            lea         edi,[ebp-44h]
  0000000C: B9 11 00 00 00      mov         ecx,11h
  00000011: B8 CC CC CC CC      mov         eax,0CCCCCCCCh
  00000016: F3 AB               rep stos    dword ptr [edi]
  00000018: C7 05 00 00 00 00   mov         dword ptr [_swap],4
            04 00 00 00
  00000022: A1 00 00 00 00      mov         eax,[_swap]
  00000027: 8B 08               mov         ecx,dword ptr [eax]
  00000029: 89 4D FC            mov         dword ptr [ebp-4],ecx
  0000002C: 8B 15 00 00 00 00   mov         edx,dword ptr [_swap]
  00000032: A1 00 00 00 00      mov         eax,[_swap]
  00000037: 8B 08               mov         ecx,dword ptr [eax]
  00000039: 89 0A               mov         dword ptr [edx],ecx
  0000003B: 8B 15 00 00 00 00   mov         edx,dword ptr [_swap]
  00000041: 8B 45 FC            mov         eax,dword ptr [ebp-4]
  00000044: 89 02               mov         dword ptr [edx],eax
  00000046: 5F                  pop         edi
  00000047: 5E                  pop         esi
  00000048: 5B                  pop         ebx
  00000049: 8B E5               mov         esp,ebp
  0000004B: 5D                  pop         ebp
  0000004C: C3                  ret
```

```
RELOCATIONS #3

                                                  Symbol    Symbol
 Offset    Type              Applied To           Index     Name
 --------  ----------------  -----------------    --------  ------
 00000000  DIR32                      00000000          F  _buf
RELOCATIONS #4

                                                  Symbol    Symbol
 Offset    Type              Applied To           Index     Name
 --------  ----------------  -----------------    --------  ------
 0000001A  DIR32                      00000000         10  _bufp1
 0000001E  DIR32                      00000004          F  _buf
 00000023  DIR32                      00000000          E  _bufp0
 0000002E  DIR32                      00000000          E  _bufp0
 00000033  DIR32                      00000000         10  _bufp1
 0000003D  DIR32                      00000000         10  _bufp1
```

13.3.2 Relocation

- Relocation algorithm

```
foreach section s {
    foreach relocation entry r {
        refptr = s + r.offset;  /* ptr to reference to be relocated */

        /* relocate a PC-relative reference */
        if (r.type == R_386_PC32) {
            refaddr = ADDR(s) + r.offset; /* ref's runtime address */
            *refptr = (unsigned) (ADDR(r.symbol) + *refptr - refaddr);
        }

        /* relocate an absolute reference */
        if (r.type == R_386_32)
            *refptr = (unsigned) (ADDR(r.symbol) + *refptr);
    }
}
```

13.3.2 Relocation

```
_swap:
  00000000: 55                  push        ebp
  00000001: 8B EC               mov         ebp,esp
  00000003: 83 EC 44            sub         esp,44h
  00000006: 53                  push        ebx
  00000007: 56                  push        esi
  00000008: 57                  push        edi
  00000009: 8D 7D BC            lea         edi,[ebp-44h]
  0000000C: B9 11 00 00 00      mov         ecx,11h
  00000011: B8 CC CC CC CC      mov         eax,0CCCCCCCCh
  00000016: F3 AB               rep stos    dword ptr [edi]
  00000018: C7 05 00 00 00 00   mov         dword ptr [_swap],4
            04 00 00 00
  00000022: A1 00 00 00 00      mov         eax,[_swap]
  00000027: 8B 08               mov         ecx,dword ptr [eax]
  00000029: 89 4D FC            mov         dword ptr [ebp-4],ecx
  0000002C: 8B 15 00 00 00 00   mov         edx,dword ptr [_swap]
  00000032: A1 00 00 00 00      mov         eax,[_swap]
  00000037: 8B 08               mov         ecx,dword ptr [eax]
  00000039: 89 0A               mov         dword ptr [edx],ecx
  0000003B: 8B 15 00 00 00 00   mov         edx,dword ptr [_swap]
  00000041: 8B 45 FC            mov         eax,dword ptr [ebp-4]
  00000044: 89 02               mov         dword ptr [edx],eax
  00000046: 5F                  pop         edi
  00000047: 5E                  pop         esi
  00000048: 5B                  pop         ebx
  00000049: 8B E5               mov         esp,ebp
  0000004B: 5D                  pop         ebp
  0000004C: C3                  ret
```

RELOCATIONS #4

Offset	Type	Applied To	Symbol Index	Symbol Name
0000001A	DIR32	00000000	10	_bufp1
0000001E	DIR32	00000004	F	_buf
00000023	DIR32	00000000	E	_bufp0
0000002E	DIR32	00000000	E	_bufp0
00000033	DIR32	00000000	10	_bufp1
0000003D	DIR32	00000000	10	_bufp1

```
_swap:
  00401060: 55                  push        ebp
  00401061: 8B EC               mov         ebp,esp
  00401063: 83 EC 44            sub         esp,44h
  00401066: 53                  push        ebx
  00401067: 56                  push        esi
  00401068: 57                  push        edi
  00401069: 8D 7D BC            lea         edi,[ebp-44h]
  0040106C: B9 11 00 00 00      mov         ecx,11h
  00401071: B8 CC CC CC CC      mov         eax,0CCCCCCCCh
  00401076: F3 AB               rep stos    dword ptr [edi]
  00401078: C7 05 44 7E 42 00   mov         dword ptr [_bufp1],424A34h
            34 4A 42 00
  00401082: A1 3C 4A 42 00      mov         eax,[_bufp0]
```

```
00424A10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00424A20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00424A30: 01 00 00 00 02 00 00 00 00 00 00 00 30 4A 42 00
00424A40: A0 1A 40 00 01 00 00 00 FF FF FF FF 02 00 00 00
00424A50: 04 00 00 00 04 00 00 00 FF FF FF FF FF FF FF FF
00424A60: FF FF FF FF 38 21 42 00 30 21 42 00 1C 21 42 00
```

13.3.2 Relocation

```
_main:
 00000000: 55                  push        ebp
 00000001: 8B EC               mov         ebp,esp
 00000003: 83 EC 40            sub         esp,40h
 00000006: 53                  push        ebx
 00000007: 56                  push        esi
 00000008: 57                  push        edi
 00000009: 8D 7D C0            lea         edi,[ebp-40h]
 0000000C: B9 10 00 00 00      mov         ecx,10h
 00000011: B8 CC CC CC CC      mov         eax,0CCCCCCCCh
 00000016: F3 AB               rep stos    dword ptr [edi]
 00000018: E8 00 00 00 00      call        0000001D
 0000001D: 33 C0               xor         eax,eax
 0000001F: 5F                  pop         edi
 00000020: 5E                  pop         esi
 00000021: 5B                  pop         ebx
 00000022: 83 C4 40            add         esp,40h
 00000025: 3B EC               cmp         ebp,esp
 00000027: E8 00 00 00 00      call        0000002C
 0000002C: 8B E5               mov         esp,ebp
 0000002E: 5D                  pop         ebp
 0000002F: C3                  ret
```

```
RELOCATIONS #4

                                                  Symbol    Symbol
 Offset   Type              Applied To            Index     Name
 -------- ---------------   -----------------     --------  ------
 00000019 REL32                     00000000          18   _swap
 00000028 REL32                     00000000          19   __chkesp
```

```
@ILT+5(_swap):
  0040100A: E9 51 00 00 00      jmp         _swap
  0040100F: CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC
```

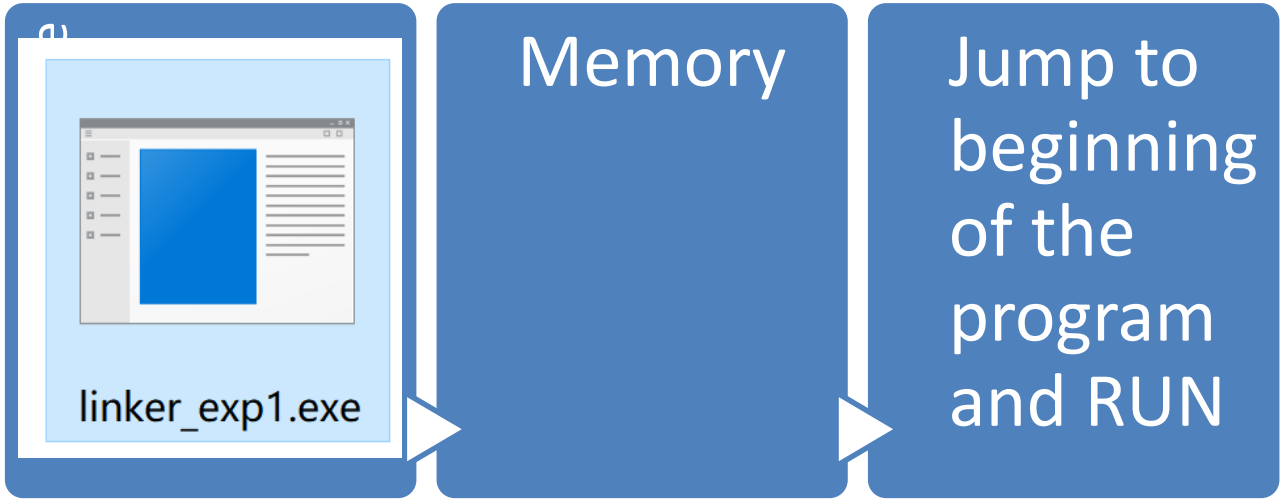
```
_main:
  00401020: 55                  push        ebp
  00401021: 8B EC               mov         ebp,esp
  00401023: 83 EC 40            sub         esp,40h
  00401026: 53                  push        ebx
  00401027: 56                  push        esi
  00401028: 57                  push        edi
  00401029: 8D 7D C0            lea         edi,[ebp-40h]
  0040102C: B9 10 00 00 00      mov         ecx,10h
  00401031: B8 CC CC CC CC      mov         eax,0CCCCCCCCh
  00401036: F3 AB               rep stos    dword ptr [edi]
  00401038: E8 CD FF FF FF      call        @ILT+5(_swap)
  0040103D: 33 C0               xor         eax,eax
  0040103F: 5F                  pop         edi
  00401040: 5E                  pop         esi
  00401041: 5B                  pop         ebx
```

Unit13 Linking and Linker

- 13.1 Linking and Linker Introduction
- 13.2 Object File Format
- 13.3 Symbol Resolution and Relocation
- 13.4 Loading and Loader Introduction

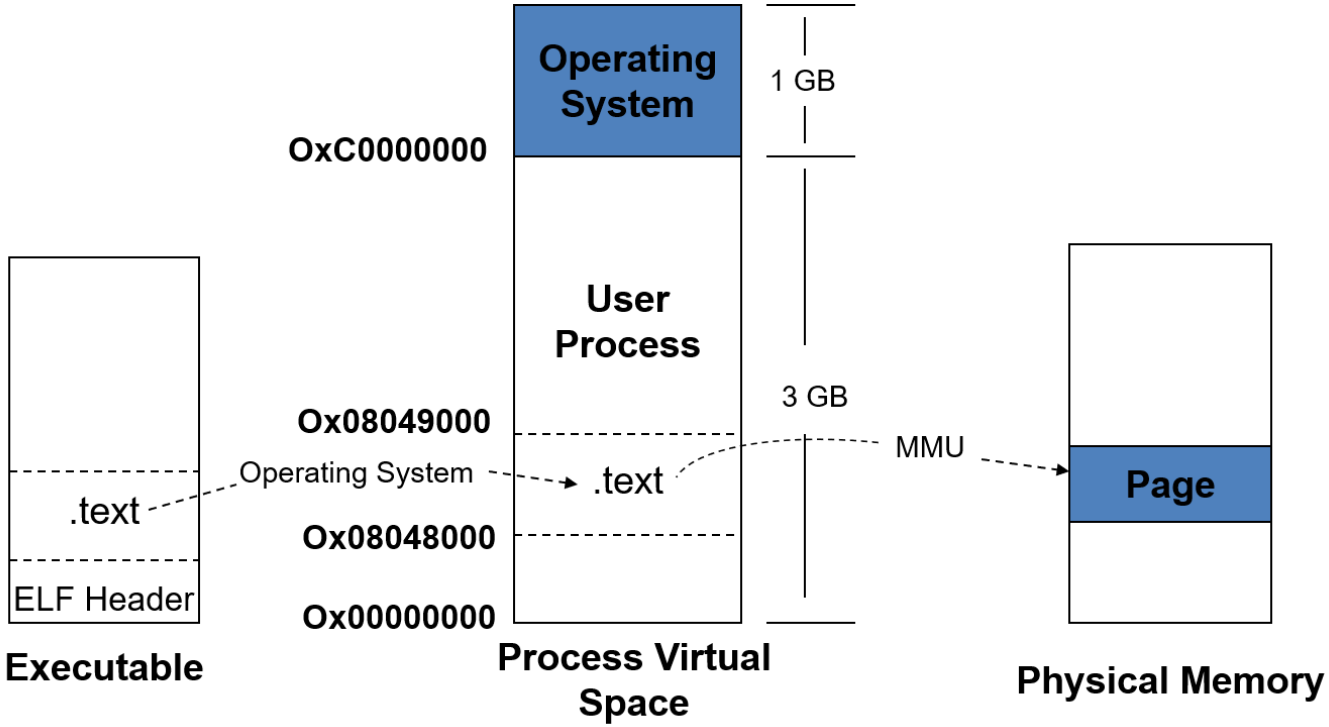
13.4 Loading and Loader Introduction

- Loading and loader



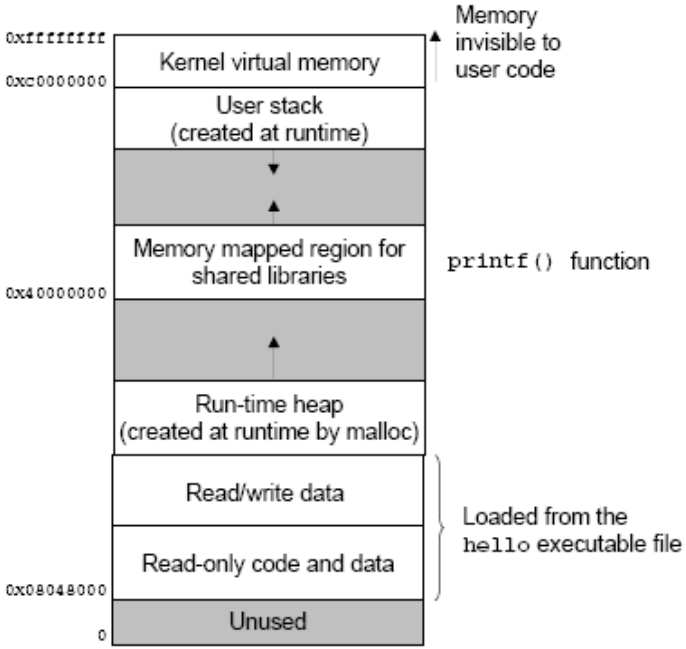
13.4 Loading and Loader Introduction

- Loading and loader



13.4 Loading and Loader Introduction

- Virtual memory



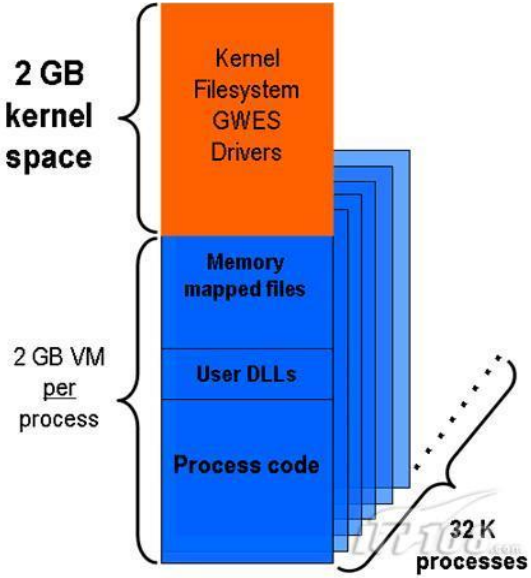


Figure 1.13: **Process virtual address space.**

13.4 Loading and Loader Introduction

➢ Format standard

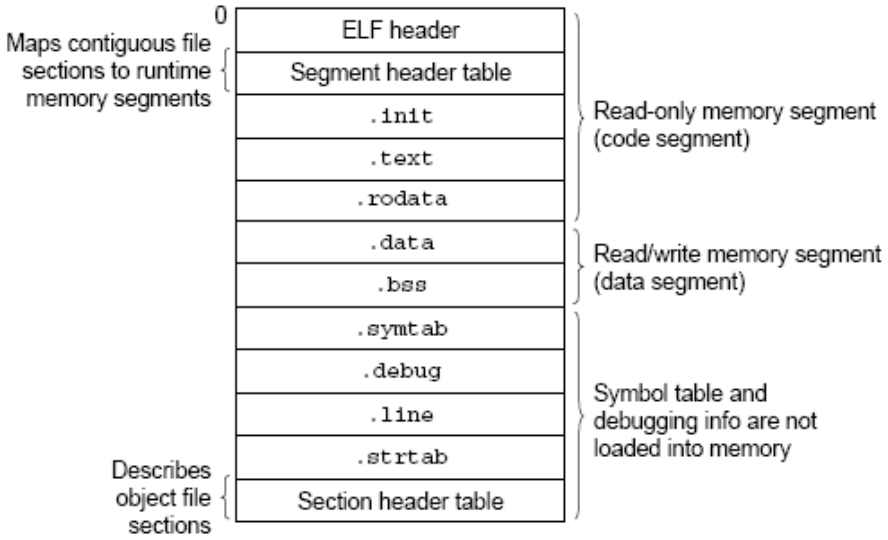
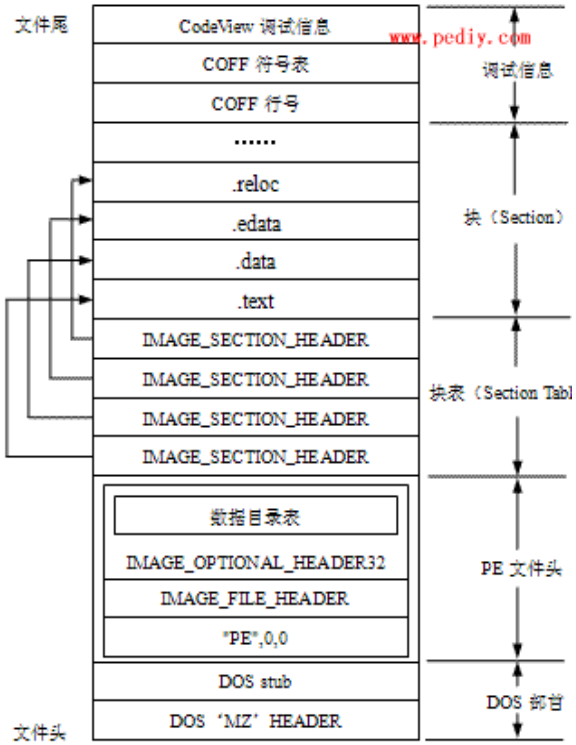


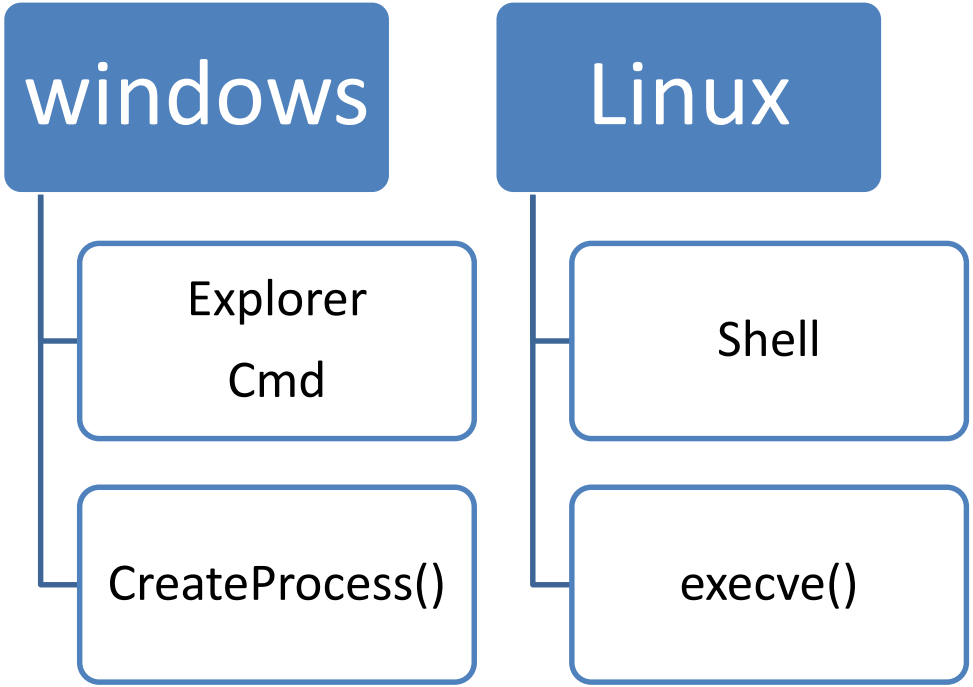
Figure 7.11: **Typical ELF executable object file**



PE 文件框架结构

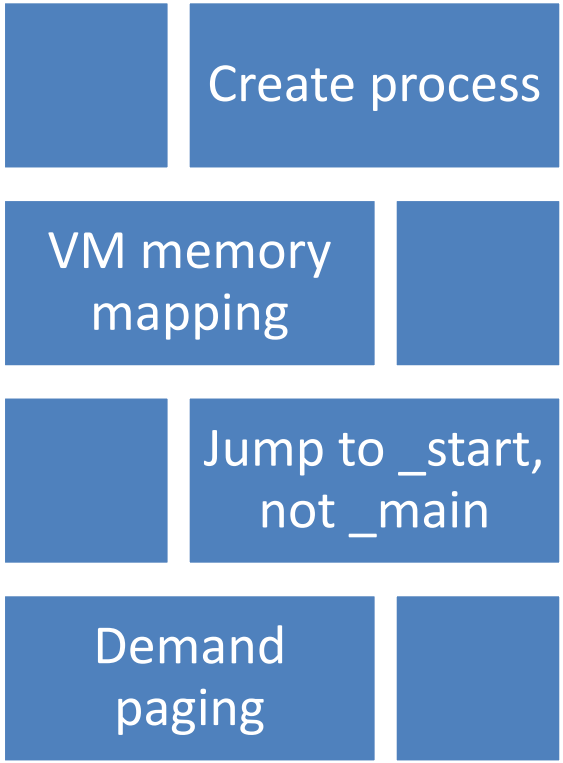
13.4 Loading and Loader Introduction

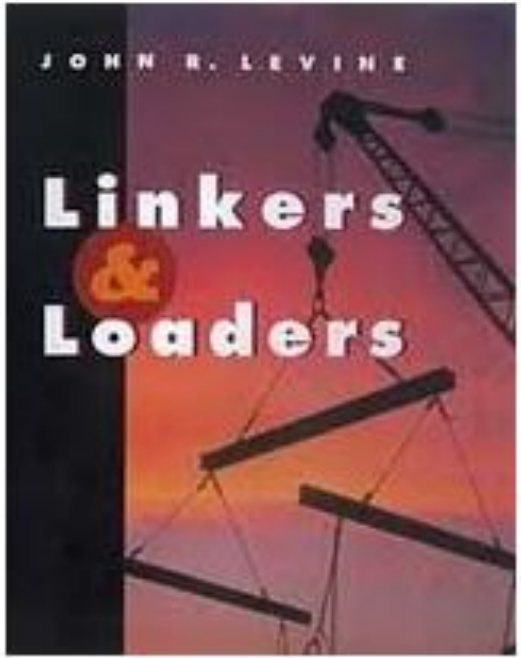
- How to use it



13.4 Loading and Loader Introduction

- Loading



13.4 Loading and Loader Introduction

- Related books