程式設計師的自我修養

Ch4 靜態連結

Samuel Chen

January, 8, 2013

Outline

source code

```c
/* a.c */
extern int shared;

int main()
{
    int a = 100;
    swap( &a, &shared );
}
```

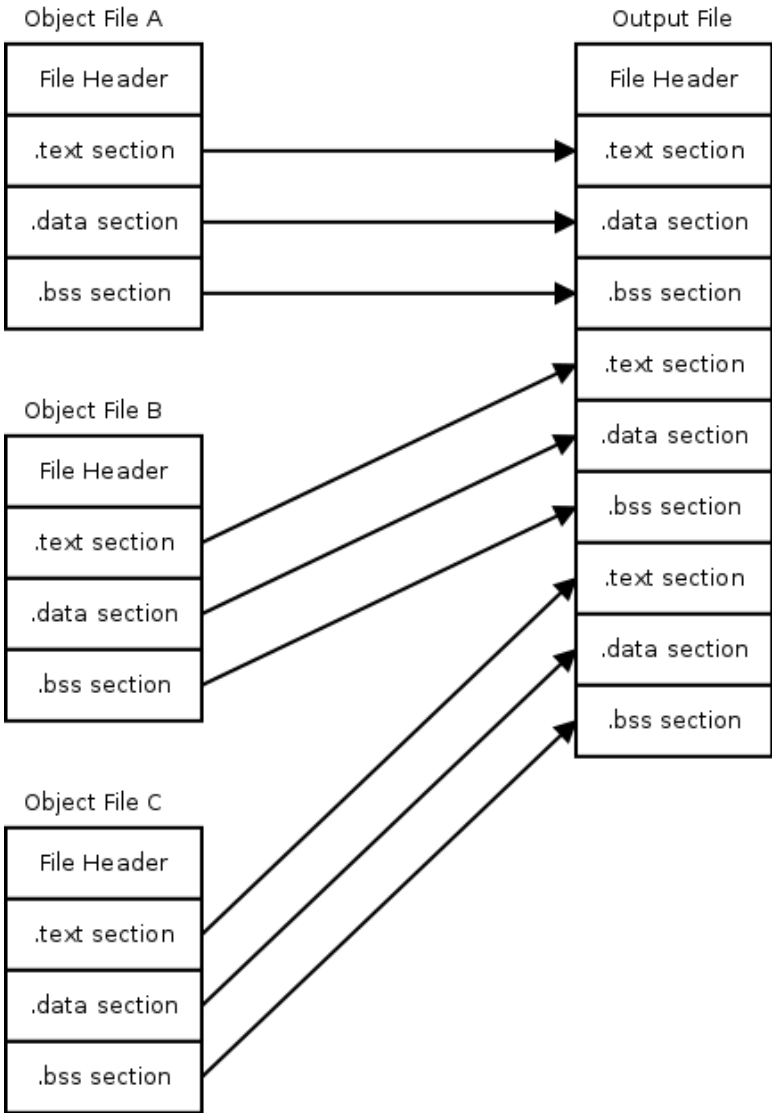
```c
/* b.c */
int shared = 1;

void swap( int* a, int* b
{
    *a ^= *b ^= *a ^= *b;
}
```

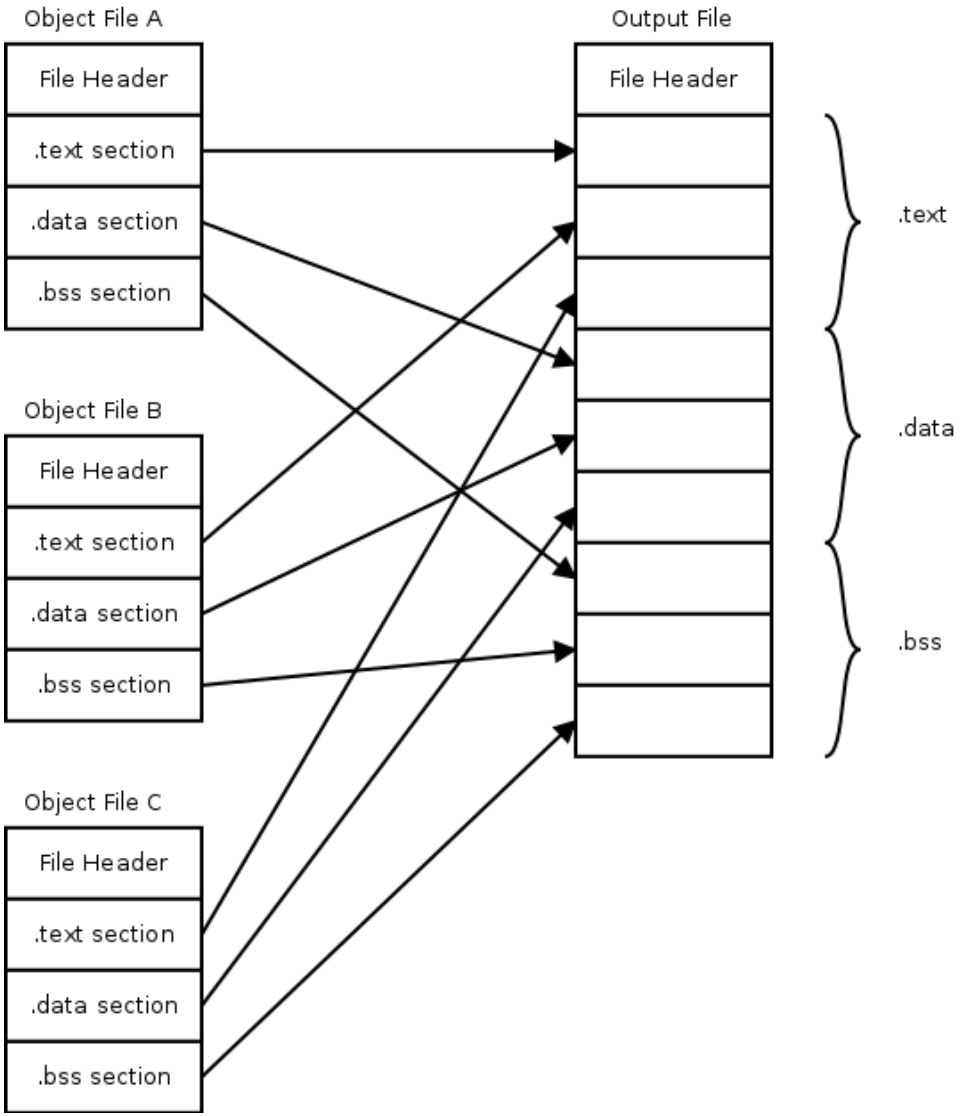
合併目的檔

- 按序累加(圖4-1)
- 相似區段合併(圖4-2)

按序累加


Object File A
File Header
.text section
.data section
.bss section
Object File B
File Header
.text section
.data section
.bss section
Object File C
File Header
.text section
.data section
.bss section
Output File
File Header
.text section
.data section
.bss section
.text section
.data section
.bss section
.text section
.data section
.bss section

相似區段合併


Object File A
File Header
.text section
.data section
.bss section
Object File B
File Header
.text section
.data section
.bss section
Object File C
File Header
.text section
.data section
.bss section
Output File
File Header
.text
.data
.bss

相似區段合併

- Step1 空間與位置分配[1]
- Step2 符號解析與重定

[1]在Linux 下, ELF 可執行檔預設從位置0x08048000開始分配

$ gcc -c a.c

$ objdump -d a.o

起始位置

```
a.o:     file format elf32-i386


Disassembly of section .text:

00000000 <main>:
   0:	55                      	push   %ebp
   1:	89 e5                   	mov    %esp,%ebp
   3:	83 e4 f0                	and    $0xfffffff0,%esp
   6:	83 ec 20                	sub    $0x20,%esp
   9:	c7 44 24 1c 64 00 00    	movl   $0x64,0x1c(%esp)
  10:	00 
  11:	c7 44 24 04 00 00 00    	movl   $0x0,0x4(%esp)
  18:	00 
  19:	8d 44 24 1c             	lea    0x1c(%esp),%eax
  1d:	89 04 24                	mov    %eax,(%esp)
  20:	e8 fc ff ff ff          	call   21 <main+0x21>
  25:	c9                      	leave  
  26:	c3                      	ret    
```

next instruction -4

25 + (-4) = 21

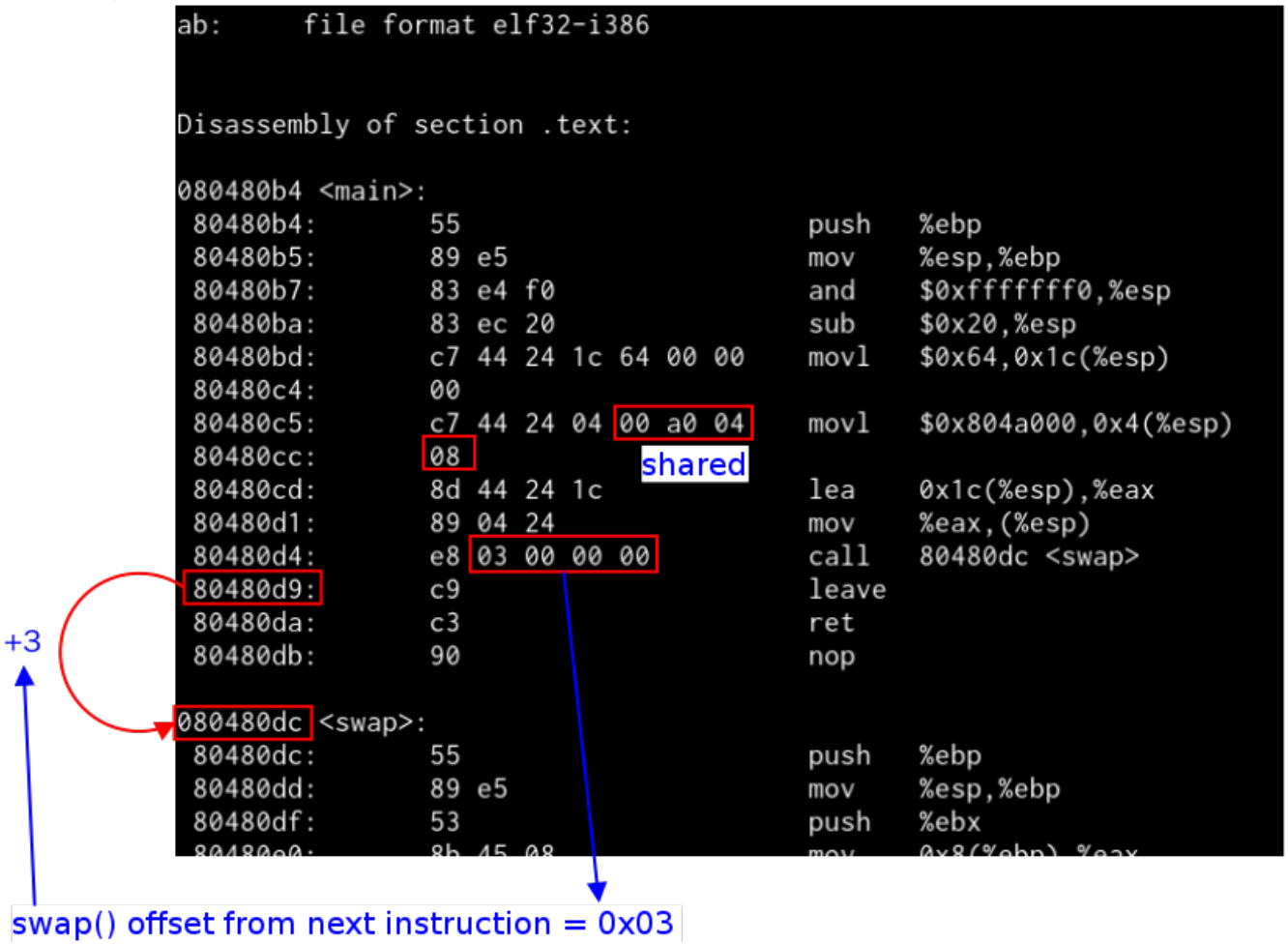
mechine code format

- mov指令
  - offset address: 0x11
  - opcode: C4 44 24 04 (mov)
  - 'shared': 00 00 00 00
- 相對偏移呼叫指令call
  - offset address: 0x20
  - opcode: E8 (call)
  - 'swap': FC FF FF FF[2]

[2] 目標位置相對於下一個指令的offset

連結後的輸出程式'ab'反組譯

$ ld a.o b.o -e main -o ab
$ objdump -d ab



重定表

$ objdump -d a.o
$ ld a.o #(error)
$ objdump -r a.o

a.o反組譯

```
a.o:     file format elf32-i386


Disassembly of section .text:

00000000 <main>:
   0:   55                      push   %ebp
   1:   89 e5                   mov    %esp,%ebp
   3:   83 e4 f0                and    $0xfffffff0,%esp
   6:   83 ec 20                sub    $0x20,%esp
   9:   c7 44 24 1c 64 00 00    movl   $0x64,0x1c(%esp)
  10:   00
  11:   c7 44 24 04 00 00 00    movl   $0x0,0x4(%esp)
  18:   00
  19:   8d 44 24 1c             lea    
  1d:   89 04 24                mov    
  20:   e8 fc ff ff ff          call   
  25:   c9                      leave  
  26:   c3                      ret
```

嘗試連結錯誤

```
i686-pc-linux-gnu-ld: warning: cannot find entry symbol _start; defaulting to 0000000008048094
a.o: In function `main':
a.c:(.text+0x15): undefined reference to `shared'
a.c:(.text+0x21): undefined reference to `swap'
--

a.o:     file format elf32-i386

RELOCATION RECORDS FOR [.text]:
OFFSET   TYPE              VALUE
00000015 R_386_32          shared
00000021 R_386_PC32        swap


RELOCATION RECORDS FOR [.eh_frame]:
OFFSET   TYPE              VALUE
00000020 R_386_PC32        .text
```

重定表

符號解析

$ readelf -s a.o

```
Symbol table '.symtab' contains 11 entries:
   Num:    Value  Size Type    Bind   Vis      Ndx Name
     0: 00000000     0 NOTYPE  LOCAL  DEFAULT  UND
     1: 00000000     0 FILE    LOCAL  DEFAULT  ABS a.c
     2: 00000000     0 SECTION LOCAL  DEFAULT    1
     3: 00000000     0 SECTION LOCAL  DEFAULT    3
     4: 00000000     0 SECTION LOCAL  DEFAULT    4
     5: 00000000     0 SECTION LOCAL  DEFAULT    6
     6: 00000000     0 SECTION LOCAL  DEFAULT    7
     7: 00000000     0 SECTION LOCAL  DEFAULT    5
     8: 00000000    39 FUNC    GLOBAL DEFAULT    1 main
     9: 00000000     0 NOTYPE  GLOBAL DEFAULT  UND shared
    10: 00000000     0 NOTYPE  GLOBAL DEFAULT  UND swap
```

目的檔案中有
關於它的重定項

指令修正方式

- 絕對近址32位定址
  R_386_32 = S + A
- 相對近址32位定址
  R_386_PC32 = S + A - P
  - S: 符號的實際位置
  - A: 保存在被修正位置的值
  - P: 相對於區段開始的偏移量

example

- main(): VA at 0x1000
- swap(): VA at 0x2000
- shared: VA at 0x3000
- relocation table:
  - shared: R_386_32
    S + A = 0x3000 + 0x00 = 0x3000
  - swap: R_386_PC32
    P = 0x1000 + 0x21 = 0x1021
    S + A - P = 0x2000 + (-4) - 0x1021 = 0x0fdb

Alignment

- 強符號個數大於2
  illegle
- 強符號*1, 弱符號*N
  以強符號為準
  (如果弱符號的大小，大於強符號，linker會有警告:
  ld: warning: alignment 4 of symbol "global" in a.o is smaller than 8 in b.o)
- 弱符號*N
  選大小最大的

COMMON to BSS

- 未初始化的全域變數放進: COMMON
- 涉及強弱符號，linking 的時候確認之後放進: BSS
- 總體來看，還是放在BSS
- GCC 參數設定不允許放COMMON:
  "-fno-common"

smallest program

inline assembly/linker script

Inline Assembly

```
1  char* str =  "Hello world\n";
2
3  void print()
4  {
5      asm( "movl $13,%%edx \n\t"
6           "movl %0,%%ecx \n\t"
7           "movl $0,%%ebx \n\t"
8           "movl $4,%%eax \n\t"
9           "int $0x80 \n\t"
10          ::"r"(str):"edx","ecx","ebx");
11 }
12
13 void exit()
14 {
15     asm ( "movl $42,%ebx \n\t"
16           "movl $1,%eax \n\t"
17           "int $0x80 \n\t");
18 }
19
20 void nomain()
21 {
22     print();
23     exit();
24 }
```

Figure : TinyHelloWorld.c

Linker Script

```
1 ENTRY(nomain)
2
3 SECTIONS
4 {
5     . = 0x0804800 + SIZEOF_HEADERS;
6     tinytext  : { *(.text) *(.data) *(.rodata)  }
7     /DISCARD/ : { *(.comment) }
8 }
9
```

Figure : TinyHelloWorld.lds

Thank you!