2020 春 第 7 章 家庭作业

这次作业只有两题（ \(^o^)/ ）：

第 2 版教材 P474~476 7.6 & 7.12

要求：一定要有写分析过程，不要光填表啊！

PS. 对于想要检验自己是否搞清楚 PC 相对地址引用重定位的同学，

还有一道自选题 P477 7.13 （建议不要错过）。

*7.6 考虑下面的 swap.c 函数版本，它计算自己被调用的次数：

```
1   extern int buf[];
2
3   int *bufp0 = &buf[0];
4   static int *bufp1;
5
6   static void incr()
7   {
8       static int count=0;
9
10      count++;
11  }
12
13  void swap()
14  {
15      int temp;
16
17      incr();
18      bufp1 = &buf[1];
19      temp = *bufp0;
20      *bufp0 = *bufp1;
21      *bufp1 = temp;
22  }
```

对于每个 swap.o 中定义和引用的符号，请指出它是否在模块 swap.o 的 .symtab 节中有符号表条目。如果是这样，请指出定义该符号的模块（swap.o 或 main.o）、符号类型（本地、全局或外部）以及它在模块中所处的节（.text、.data 或 .bss）。

符号	swap.o.symtab 条目？	符号类型	定义符号的模块	节
buf				
bufp0				
bufp1				
swap				
temp				
incr				
count				

**7.12 图 7-10 中的 swap 程序包含 5 个重定位的引用。对于每个重定位的引用，给出它在图 7-10 中的行号、运行时存储器地址和值。swap.o 模块中的原始代码和重定位条目如图 7-19 所示。

```
1  00000000 <swap>:
2     0:   55                          push   %ebp
3     1:   8b 15 00 00 00 00           mov    0x0,%edx          Get *bufp0=&buf[0]
4                                      3: R_386_32      bufp0   Relocation entry
5     7:   a1 04 00 00 00              mov    0x4,%eax          Get buf[1]
6                                      8: R_386_32      buf     Relocation entry
7     c:   89 e5                       mov    %esp,%ebp
8     e:   c7 05 00 00 00 00 04        movl   $0x4,0x0          bufp1 = &buf[1];
9    15:   00 00 00
10                                     10: R_386_32     bufp1   Relocation entry
11                                     14: R_386_32     buf     Relocation entry
12    18:   89 ec                      mov    %ebp,%esp
13    1a:   8b 0a                      mov    (%edx),%ecx       temp = buf[0];
14    1c:   89 02                      mov    %eax,(%edx)       buf[0]=buf[1];
15    1e:   a1 00 00 00 00             mov    0x0,%eax          Get *bufp1=&buf[1]
16                                     1f: R_386_32     bufp1   Relocation entry
17    23:   89 08                      mov    %ecx,(%eax)       buf[1]=temp;
18    25:   5d                         pop    %ebp
19    26:   c3                         ret
```

图 7-19　练习题 7.12 的代码和重定位条目

图 7-10 中的行号	地址	值

****7.13**　考虑图 7-20 中的 C 代码和相应的可重定位目标模块。

A. 确定当模块被重定位时，链接器将修改 .text 中的哪些指令。对于每条这样的指令，列出它的重定位条目中的信息：节偏移、重定位类型和符号名字。

B. 确定当模块被重定位时，链接器将修改 .data 中的哪些数据目标。对于每条这样的指令，列出它的重定位条目中的信息：节偏移、重定位类型和符号名字。

可以随意使用诸如 **OBJDUMP** 之类的工具来帮助你解答这个题目。

```
1   extern int p3(void);
2   int x = 1;
3   int *xp = &x;
4
5   void p2(int y) {
6   }
7
8   void p1() {
9       p2(*xp + p3());
10  }
```

a) C 代码

```
1   00000000 <p2>:
2      0:   55                      push   %ebp
3      1:   89 e5                   mov    %esp,%ebp
4      3:   89 ec                   mov    %ebp,%esp
5      5:   5d                      pop    %ebp
6      6:   c3                      ret

7   00000008 <p1>:
8      8:   55                      push   %ebp
9      9:   89 e5                   mov    %esp,%ebp
10     b:   83 ec 08                sub    $0x8,%esp
11     e:   83 c4 f4                add    $0xfffffff4,%esp
12    11:   e8 fc ff ff ff          call   12 <p1+0xa>
13    16:   89 c2                   mov    %eax,%edx
14    18:   a1 00 00 00 00          mov    0x0,%eax
15    1d:   03 10                   add    (%eax),%edx
16    1f:   52                      push   %edx
17    20:   e8 fc ff ff ff          call   21 <p1+0x19>
18    25:   89 ec                   mov    %ebp,%esp
19    27:   5d                      pop    %ebp
20    28:   c3                      ret
```

b）可重定位目标文件的 .text 节

```
1   00000000 <x>:
2      0:   01 00 00 00
3   00000004 <xp>:
4      4:   00 00 00 00
```

c）可重定位目标文件的 .data 节

图 7-20　练习题 7.13 的示例代码