第三题（10 分）

本题基于下列 m.c 及 swap.c 文件所编译生成的 m.o 和 swap.o，编译和运行在 Linux/X86-64 下使用 GCC 完成，编译过程未加优化选项。

```
//m.c
void myswap();
void func();
char buf[2] = {1, 2};
void *bufp0;
void *bufp1;
char temp;
int main(){
    func();
    ......//略去题目无关代码
    myswap(temp);
    ......//略去题目无关代码
    return 0;
}
```

```
//swap.c
extern int buf[];
char *bufp0;
......//略去题目无关代码
char *bufp1 = &buf[1];
void myswap(char temp){
    static int count = 0;
    ......//略去题目无关代码
    bufp0 = &buf[0];
    temp = *bufp0;
    ......//略去题目无关代码
    *bufp0 = *bufp1;
    *bufp1 = temp;
    ......//略去题目无关代码
    count++;
}
void func(){
    ......//略去题目无关代码
}
```

1）对于每个 swap.o 中定义和引用的符号，请用“是”或“否”指出它是否在模块 swap.o 的.symtab 节中存在符号表条目。如果存在条目，则请指出定义该符号的模块（swap.o 或 m.o）、符号类型（局部、全局或外部）以及该符号在所属的模块（“即该符号在该模块中被定义”）中所处的节；如果不存在条目，则请将该行后继空白处标记为“/”。

符号	.symtab 有条目？	符号类型	定义符号的模块	节
buf				
bufp0				
count				
func				
temp				

2）下图左边给出了 m.o 和 swap.o 的反汇编文件，右边给出了采用某个配置链接成可执行程序后再反汇编出来的文件。根据答题需要，其中的信息略有删减。

```
0000000000......<main>:
55                      push   %rbp
48 89 e5                mov    %rsp,%rbp
b8 00 00 00 00          mov    $0x0,%eax
e8 00 00 00 00          callq  e <main+0xe>     ①
......
0f b6 05 00 00 00 00    movzbl 0x0(%rip),%eax  ②
0f be c0                movsbl %al,%eax
89 c7                   mov    %eax,%edi
b8 00 00 00 00          mov    $0x0,%eax
e8 00 00 00 00          callq  26 <main+0x26>  ③
......
b8 00 00 00 00          mov    $0x0,%eax
5d                      pop    %rbp
c3                      retq
```

```
0000000000......<myswap>:
55                push   %rbp
48 89 e5           mov    %rsp,%rbp
89 f8              mov    %edi,%eax
88 45 fc           mov    %al,-0x4(%rbp)
......
48 c7 05 00 00 00 00 00 00 00 00          ⑤④
                    movq   $0x0,0x0(%rip)
......
48 8b 05 00 00 00 00  mov    0x0(%rip),%rax    ⑥
0f b6 00              movzbl (%rax),%eax
88 45 fc              mov    %al,-0x4(%rbp)
......
48 8b 05 00 00 00 00 mov    0x0(%rip),%rax    ⑦
48 8b 15 00 00 00 00 mov    0x0(%rip),%rdx    ⑧
0f b6 12             movzbl (%rdx),%edx
88 10                mov    %dl,(%rax)
48 8b 05 00 00 00 00 mov    0x0(%rip),%rax    ⑨
0f b6 55 fc          movzbl -0x4(%rbp),%edx
88 10                mov    %dl,(%rax)
......
8b 05 00 00 00 00     mov    0x0(%rip),%eax    ⑩
83 c0 01              add    $0x1,%eax
89 05 00 00 00 00     mov    %eax,0x0(%rip)    ⑪
90                    nop
5d                    pop    %rbp
c3                    retq

0000000000......<func>:
......
```

```
0000000000001000 <main>:
1000: 55                  push  %rbp
1001: 48 89 e5            mov   %rsp,%rbp
1004: b8 00 00 00 00      mov   $0x0,%eax
1009: e8 ____①____        callq 302e <func>
......
1010:  0f  b6  05 _____________②_____________  movzbl
0x??????(%rip),%eax
1017: 0f be c0            movsbl %al,%eax
101a: 89 c7               mov    %eax,%edi
101c: b8 00 00 00 00      mov    $0x0,%eax
1021: e8 ____③____        callq  10e4 <myswap>
......
10db: b8 00 00 00 00       mov    $0x0,%eax
10e0: 5d                   pop    %rbp
10e1: c3                   retq

00000000000010e4<myswap>:
10e4: 55                   push   %rbp
10e5: 48 89 e5             mov    %rsp,%rbp
10e8: 89 f8                mov    %edi,%eax
10ea: 88 45 fc             mov    %al,-0x4(%rbp)
......
10f4: 48 c7 05 ____⑤____ ____④____
                    movq   ___④___,___⑤___(%rip)
......
1104: 48 8b 05 ____⑥____ mov   ___⑥___(%rip),%rax
110b: 0f b6 00             movzbl (%rax),%eax
110e: 88 45 fc             mov    %al,-0x4(%rbp)
......
1229: 48 8b 05 ____⑦____ mov   ___⑦___(%rip),%rax
1230: 48 8b 15 ____⑧____ mov   ___⑧___(%rip),%rdx
1237: 0f b6 12             movzbl (%rdx),%edx
123a: 88 10                mov    %dl,(%rax)
123c: 48 8b 05 ____⑨____ mov   ___⑨___(%rip),%rax
1243: 0f b6 55 fc          movzbl -0x4(%rbp),%edx
1247: 88 10                mov    %dl,(%rax)
......
124b: 8b 05 ____⑩____    mov    ___⑩___(%rip),%eax
1251: 83 c0 01             add    $0x1,%eax
1254: 89 05 ____⑪____    mov    %eax,___⑪___(%rip)
125a: 90                   nop
125b: 5d                   pop    %rbp
125c: c3                   retq

000000000000302e<func>:
......
   ......略去部分和答题无关的信息......
000000000000a000 <buf>:
   ......略去部分和答题无关的信息......
000000000000a250 <bufp1>:
   ......略去部分和答题无关的信息......
0000000000edd38 <count.1837>:
   ......略去部分和答题无关的信息......
0000000000edd40 <bufp0>:
   ......略去部分和答题无关的信息......
```

在上图中对所涉及到的重定位条目进行用数字①至⑪进行了标记，请根据下表中所提供的重定位条目信息，计算相应的重定位引用值并填写下表。

编号	重定位条目信息	应填入的重定位引用值 一律填写 32 位 16 进制数
①	r.offset = 0x0a　　r.symbol = 本题不提供 r.type = R_X86_64_PC32　　r.addend = -4	
③	r.offset = 0x22　　r.symbol = 本题不提供 r.type = R_X86_64_PC32　　r.addend = -4	
④	r.offset = 0x17　　r.symbol = 本题不提供 r.type = R_X86_64_32　　r.addend = 0	
⑨	r.offset = 0x15b　　r.symbol = 本题不提供 r.type = R_X86_64_PC32　　r.addend = -4	
⑪	r.offset = 0x172　　r.symbol = 本题不提供 r.type = R_X86_64_PC32　　r.addend = -4	

得分

第四题（10 分）

在 x86_64 环境下，考虑如下 4 个文件（main.c, value.c, f1.c, f2.c）：

```
/* main.c */
#include <stdio.h>

extern void f_void_void();
extern int f_int_void();
void *f;

int main()
{
    int a = 1, b, c;

    f = (void *)f_void_void;

    ((void (*)(int))f)(a);
    b = ((int (*)())f)();
    printf("b = %d\n", b);

    f = (void *)f_int_void;

    ((void (*)())f)(a);
    c = ((int (*)(int))f)(a);
printf("c = %d\n", c);

    return 0;
}

/* value.c */
int BIG;

/* f1.c */
#include <stdio.h>

extern int BIG;
int small = 1;

void f_void_void() {
    small += 1;
    BIG += 1;
    printf("small = %d, BIG = %d\n", small, BIG);
}
```

```c
/* f2.c*/
#include <stdio.h>

extern int BIG;
static int small;

int f_int_void() {
    small += 1;
    BIG += 1;
    printf("small = %d, BIG = %d\n", small, BIG);
    return small + 1;
}
```

使用命令

`gcc -o main main.c f1.c f2.c value.c`

编译这四个文件。

使用

`./main`

运行编译好的程序。

1. 对于程序中的相应符号，请给出它的属性（局部或全局，强符号或弱符号），不确定的请画 X。

源文件	符号名	局部或全局？	强符号或弱符号？
main.c	f		
value.c	BIG		
f1.c	small		
f2.c	small		

2. 请补全程序运行的输出，不确定的请画 X。

```
small = _____, BIG = _____
small = _____, BIG = _____
b = _____
small = _____, BIG = _____
small = _____, BIG = _____
c = _____
```