Pg.216

4.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 符号 | 是否在swap.o符号表中 | 定义模块 | 符号类型 | 节 |
| buf | 在 | main.o | 外部 | .data |
| bufp0 | 在 | swap.o | 全局 | .data |
| bufp1 | 在 | swap.o | 本地 | .bss |
| incr | 在 | swap.o | 本地 | .text |
| count | 在 | swap.o | 本地 | .data |
| swap | 在 | swap.o | 全局 | .text |
| temp | 不在 | - | - | - |

5.

1)

|  |  |  |
| --- | --- | --- |
|  | main.c | proc1.c |
| 强符号 | x, z, main | proc1 |
| 弱符号 | proc1, y | x |

x .data节4字节

y .data节2字节

z .bss节2字节

2)

(double)-1.5对应的机器码为0xbff8000000000000，在程序执行完proc1()中语句后,由于x只有4字节的空间，故只能存放其低32位0x00000000, 剩余的32位溢出至z和x,z之间的空间中，使得z = 0x0000, z和x之间的2字节空位为0xbff8.

故最终的打印结果为x = 0, z = 0.

3)

类似2)中的表述，double(-1.5)的高32位会分别溢出至y, z的空间中，使得y = 0x0000, z = 0xbff8.

故最终的打印结果为x = 0, z = -16392.

4)

将proc1.c对x的声明改为

static double x;

即可让proc1中的x变为local型，proc1.o会另外给x分配8字节空间而不与main中的x共用地址.

7.

main是m1.c中的强符号，在m2.c中是弱符号，所以m2中的main符号的地址为m1.c的main()函数在 m1.o的.text节的地址. 故打印的结果是main函数对应机器码的首两个字节，是0x55和0x89, 分别对应push %ebp和mov指令.

10.

需要重定位的符号名为swap, 相对于.text节起始位置的位移为0x7, 所在指令行号为6, 重定位类型为R\_386\_PC32.

重定位前在位移量7~a处有0xfcffffff,其值为-4作为初始值, 重定位后，应使call的目标地址为swap的首地址. main函数的首地址为0x8048386, 占0x12字节空间，故main函数最后一条指令结束的地址为0x8048386+0x12=0x8048398, 即是swap的起始地址（以4字节对齐）

故重定位值为

ADDR(swap) - ((ADDR(.text) + r\_offset) - init) = 0x8048398 - ((0x8048386 + 7) - 0xfcffffff = 0x7

故重定位后call指令处的偏移量修改为 07 00 00 00.

11.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 符号 | 位移 | 指令所在行号 | 重定位类型 | 重定位前内容 | 重定位后内容 |
| 1 | bufp1(.bss) | 0x8 | 6~7 | R\_386\_32 | 0x00000000 | 0x8049620 |
| 2 | buf(.data) | 0xc | 6~7 | R\_386\_32 | 0x00000004 | 0x80495cc |
| 3 | bufp0(.data) | 0x11 | 10 | R\_386\_32 | 0x00000000 | 0x80495d0 |
| 4 | bufp0(.data) | 0x1b | 14 | R\_386\_32 | 0x00000000 | 0x80495d0 |
| 5 | bufp1(.bss) | 0x21 | 16 | R\_386\_32 | 0x00000000 | 0x8049620 |
| 6 | bufp1(.bss) | 0x2a | 20 | R\_386\_32 | 0x00000000 | 0x8049620 |