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