4.

符号	是否在**SWAP.O**的符号表中	定义模块	符号类型	节
buf	在	main.o	extern	.data
bufp0	在	swap.o	global	.data
bufp1	在	swap.o	local	.bss
incr	在	swap.o	local	.text
count	在	swap.o	local	.data
swap	在	swap.o	global	.text
temp	不在	-	-	-

5.

(1) main.c中强符号有 x, y, main, 弱符号有 y 和 proc1; proc1.c中的强符号有 proc1, 弱符号有 x ;

(2)

	0	**1**	**2**	**3**
&z	02	00	...	...
&x	01	01	00	00

proc1()函数执行前

	0	**1**	**2**	**3**
&z	00	00	F8	BF
&x	00	00	00	00

proc1()函数执行后

故打印结果为 x = 0, z = 0;

改变后结果为 x = 0, z = -16392

(3) 将proc1.c中第一行改为"static double x"

7.

全局符号 main 在 m1 中是强符号, 在 m2 中弱符号, 因此在 m1 中全局符号 main 被定义在 .text 节中, 因为 main 函数对应的机器码开始两个字节为 55H 和 89H; 在 m2 中的 printf 语句中引用数组元素 main[0] 和 main[1] 时, main[0] = 55H, main[1] = 89H;

8.

.data 节中全局变量的初始值总的数据长度为 0xE8, 因此, 虚拟地址空间中长度为0x104 字节的可读写数据段中, 开始的 0xE8 个字节取自 .data 节, 后面的28 个字节是未初始化全局变量所在区域

9.

(1) -gcc -static -o p p.o libx.a liby.a p.o

(2) -gcc -static -o p p.0 libx.a liby.a libx.a

(3) -gcc -static -o p p.0 libx.a liby.a libx.a libz.a

10.

需要重定位的是全局符号 swap , 位置相对于 .text 节起始位置位移量 r_offset 为 7 , 指令行号是第 6 行 call 指令中的偏移量字段, 按照 R_386_PC32 进行重定位;

重定位前, 在位移量 7、8、9、a 处内容分别为 FC、FF、FF、FF, 因此初始值 init 的机器数为 0xFFFFFFFC , 值为 -4 ; 重定位后, 应该使 call 指令的目标转移地址指向 swap 函数的起始地址;

main 函数占 12H = 18 字节的储存空间, 起始地址ADDR(.text)为 0x8048386, 因此最后一条指令地址为 0x8048386 + 0x12 = 0x8048398; 因为 swap 函数代码紧跟在 main 后且首地址按 4 字节边界对齐, 故 swap 的起始地址 ADDR(swap) 就是 0x8048398

ADDR(swap) - ((ADDR(.text) + r_offset) - init)) = 0x8048398 - ((0x8048386 + 7) - (-4)) = 7

重定位后在位移量 7、8、9、a 处的 call 指令的偏移量字段为 07 00 00 00