5、

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

弱符号有x。根据多重定义符号处理规则，符号x的定义以main.c中的强符号x为准，即在main.o的.data节中分配x,占4个字节，随后是另一个强符号z占两个字节x和z都属于.data

节，随后是.bss节，其中只有一个变量y,按4字节对齐，因此，在z后面有两个字节空闲，

再后面是变量y的空间。

（2）程序执行时,在调用proc（）函数之前，&x中存放的是x的机器数：00000101H,

随后两个字节（地址为&z）存放z,即0002H,再后面两个字节空闲。

在调用procl()函数以后，因为procl()中的符号x是弱符号，所以x的定义以main

中的强符号x为准,执行x=-1.5后.便将BFF80000 00000000H存放到了&x开始的8个字节中。即&x中为其低32位的00000000H, &z中为高32位的BFF80000H中的低16位0000H, z后面的两个空闲字节中为高16位BFF8H。

最终打印结果，x=0，z=0.

(3）只要将文件procl.c中的第1行修改为“static double x”，就可以使得procl中的

x设定为本地变量，从而在procl.o的.data节中专门分配存放x的8个字节空间，而不会和

main中的x共用同一个存储地址，因此也就不会破坏main中x和z的值。

7、

全局符号main在m1中是强符号，在m2中是弱符号，因此以ml中main的定义为准。

在m1中全局符号main被定义在.text节中。出现本题所说结果的原因是，main函数对应的

机器码开始两个字节为55H和89H。在有些系统中，main函数最初两条指令如下：

1 Disassembly of section .text：

2 00000000 <main>:

3 0: 55 push %ebp

4 1: 89 e5 mov %esp4 %ebp

其中，55H是指令“push %ebp”的机器码，89E5H是指令“mov %e8p, %ebp”的机器码,

因此，可以看出在m2中的printf语引用数元素main[0]和main[1]时,main[0]=55H,

main[1]=89H。

8、

在可执行目标文件中描述的“可读写数据段”由所有可重定位目标文件中的.data节合并生成的.data节、所有可重定位目标文件中的.bss节合并生成的.bss节这两部分组成。.data节由初始化的全局变量组成，因此其初始值必须记录在可执行文件中；而.bss节由未初始化的全局变量组成，因此在可执行目标文件中无需记录其值，只要描述总的长度和每个变量的

起始位置即可。

根据图4.25中的内容可知，.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,o libx.a liby.a libx.a

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

10、

Swap，相对于起始位置偏移为7，所在指令行号第6行，PC相对寻址

重定位前，在位移量7、8、9、a处的内容分别为fc、ff、ff. ff,因此初始值init的机器

数为Oxfffffffc,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函数澈共占12H-18字节的存储空间，其起始地址ADDR(.text)为0x8048386.因

此main函数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。

重定位值的计算公式为

ADDR(swap)-((ADDR(.text)+r\_offset)-init)=()x8048398-((0x8048386+7)-(-4))=7o

因此，重定位后，在位移量7、8、9、a处的call指令的偏移暈字段为07 00 00 00。