系统基础作业

4.

符号	是否在swap.o的符号表中	定义模块	符号类型	节
buf	是	main.o	extern	.data
bufp0	是	swap.o	global	.data
bufp1	是	swap.o	local	.bss
incr	是	swap.o	local	.text
count	是	swap.o	local	.data
swap	是	swap.o	global	.text
temp	否	\	\	\

5.

(1)

> main.c 中的强符号有 x, z, main，弱符号有 y, proc1；proc1.c 中强符号有 proc1，弱符号有 x。

(2)

> proc1()函数调用前
>
> | | 0 | 1 | 2 | 3 |
> | --- | --- | --- | --- | --- |
> | &z | 02 | 00 | ... | ... |
> | &x | 01 | 01 | 00 | 00 |
>
> proc1()函数调用后
>
> | | 0 | 1 | 2 | 3 |
> | --- | --- | --- | --- | --- |
> | &z | 00 | 00 | F8 | BF |
> | &x | 00 | 00 | 00 | 00 |
>
> 打印结果为 **x=0,z=-16392。**在调用 proc1()函数之前，&x 中存放的是 x 的机器数 0000 0101H；随后两个字节（&y）存放 y，未初始化，通常情况下为 0；再后面两个字节存放 z 的机器数 0002H。调用 proc1()函数后，因为 proc1.c 中的 x 是弱符号，proc1() 中 x 的定义以 main.c 中的强符号 x 为准，执行 `x = -1.5` 后，将 -1.5 的机器数 BFF8 0000 0000 0000H 存放到 &x 开始的 8 个字节中。这时 &x 中为低 32 位的 0000 0000H，&y 中为高 32 位的 BFF8 0000H 中的低 16 位 0000H，&z 中为高 32 位的 BFF8 0000H 中的高 16 位 BFF8H，均以小端方式存储。修改第 3 行不影响最终的结果，打印结果为 **x=0,z=-16392。**

(3)

> 将 proc1.c 中的第一行改为 `static double x`。这时 proc1 中的 x 为 local 变量，会在 proc1.o 的 .data 节中专门分配空间，不会和 main 中的 x 共用同一个地址，也就不会破坏 main 中的 x 和 z。

7.

> 全局符号 main 在 m1.c 中是强符号，在 m2.c 中是弱符号，以 m1 中 main 的定义为准。在 m1 中 main 定义在 .text 节中，通常 main 函数开始的两条指令为：
>
> - push %ebp
> - mov %esp, %ebp
>
> 第一条指令的机器码为 **55**，第二条指令的机器码为 **89 E5**。在 m2 中的 printf 语句中引用数组元素时，main[0]=55H，main[1]=89H。

8.

> 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 节中全局变量的初始值总的数据长度为 0xe8，故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 0x104 - 0xe8 = 0x1c = 28 字节是未初始化全局变量所在的区域。

9.

(1)

> gcc -static -o p p.o libx.a liby.a

(2)

> gcc -static -o p p.o libx.a liby.a libx.a

(3)

> gcc -static -o p p.o libx.a liby.a libz.a libx.a

10.

> main.o 的 .text 节中需要重定位的符号是在 main.c 中被引用的全局符号 swap，需要重定位的是第 6 行 call 指令中的偏移量字段，相对于 .text 节的起始位置的位移为 7，按照 PC 相对地址方式重定位（R_386_PC32）。重定位前，在位移量 7、8、9、a 处的初始值 init 的内容分别为 fc ff ff ff，其机器数为 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 函数共占 12h = 18 字节的存储空间，起始地址为 0x8048386，最后一条指令的地址为 0x8048386 + 0x12 = 0x8048398。又 swap 函数代码紧跟 main 后且首地址按 4 字节对齐，故 swap 的起始地址就是 0x8048398。重定位值的计算过程如下：
>
> ADDR(r_sym) - ((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。

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	0x 0000 0000	0x8049620
2	buf(.data)	0xe	6~7	R_386_32	0x 0000 0004	0x80495cc
3	bufp0(.data)	0x11	10	R_386_32	0x 0000 0000	0x80495d0
4	bufp0(.data)	0x1b	14	R_386_32	0x 0000 0000	0x80495d0
5	bufp1(.dss)	0x21	17	R_386_32	0x 0000 0000	0x8049620
6	bufp1(.dss)	0x2a	21	R_386_32	0x 0000 0000	0x8049620