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 | 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)

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

(2)

按小端模式存放. x和z为定义在main.c中的强符号, x以强符号为主, 则x和z存放在main.o的.data节中, 且x和z的内存地址连续, x的地址小于z. 执行proc1时, 将-1.5的机器数BFF8 0000 0000 0000H存到x的地址开始的8个字节中, 于是执行完后, x的机器数变为 0000 0000H, z的机器数变为0000H. 打印的结果为x=0,z=0.

|  |  |  |
| --- | --- | --- |
|  | 执行前 | 执行后 |
| &x中内容 | 0000 0101H | 0000 0000H |
| &z中内容 | ---- 0002H | BFF8 0000H |

(3)

将x = -1.5改成 x = 0x0000 0002 0000 0101

或者将double x改为static double x.

7

m1.c中的main为强符号, m2.c中的main为弱符号, 则编译链接后, main的定义以m1.c中的main为主. m1.o中的main是全局符号, 类型是FUNC, 储存在.text节中, 出现该结果的原因应该是main符号对应的机器数的前两个字节为55H和89H.

10

由图4.7第7行可以知道需要重定位的符号名为swap, 由第6行可知起始位置的位移为7, 所在指令的行号为6, 重定位类型为R\_386\_PC32, 重定位前的内容是ff ff ff fc(十进制-4).

对于重定位后的内容: 可知main函数起始地址是0x8048386, 大小为18字节, 又swap函数紧跟的main函数且对齐方式为4, 则swap函数的起始地址为0x8048398, 于是重定位值为: ADDR(swap) – ( ( ADDR(.text) + r\_offset ) – init ) = 0x8048398 – ( 0x8048386 + 7 – (-4)) = 7. 于是重定位后的内容为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 |