第七章 链接 家庭作业

7.7

修改之前链接之后输出结果为：

```
linda:chapter7>gcc -o test foo5.c bar5.c
/usr/bin/ld: Warning: alignment 4 of symbol `x' in /tmp/ccsWRlWy.o is smaller th
an 8 in /tmp/cc8wVDAc.o
linda:chapter7>./test
x = 0x0 y=0x3b6c
```

分析：之所以会出现调用 bar5 中的 f()会修改 foo5 中声明的 x，是因为未初始化的全局变量是弱符号，当遇到在 bar5 中声明的已经被赋过初值的强符号 int x，会被覆盖。我们将 bar5 中声明的 int x，加上 static 修饰即可。

代码：

```
static double x;
void f() {
    x = -0.0;
}
```

修改之后代码测试：

```
linda:chapter7>gcc -o test foo5.c bar5.c
linda:chapter7>./test
x = 0x3b6d y=0x3b6c
```

7.9

分析：如果在 bar6 中定义的 main 不进行赋值，那么 bar6 中的 main 就是一个弱类型，而 foo6 中的 main 是一个强类型，所以在链接之后，main 中保存的就是 foo6 中 main 的含义，即 main 函数的地址。

7.11

分析：这个题目的答案在书上有原话：该段中剩下的 8 个字节对应于运行时将被初始化为 0 的.bss 数据。

7.13

A：

```
linda:x86_64-linux-gnu>ar -t /usr/lib/x86_64-linux-gnu/libc.a | wc -l
1690
```

```
linda:x86_64-linux-gnu>ar -t /usr/lib/x86_64-linux-gnu/libm-2.27.a | wc -l
794
```

B:

不同，原因如下:

```
linda:chapter7>gcc -Og -o a.out foo5.c bar5.c
linda:chapter7>gcc -Og -g -o b.out foo5.c bar5.c
```

```
linda:chapter7>sha256sum a.out b.out
f7df26beaef6af84e41d962a4c15946a0d6fee26c1ddfe85ee12dd51c3b45575  a.out
583ac84b6333f5a0ade54f27b779ab39baaae2dbfac96b9b11eed6f72b6a8822  b.out
```

C:

```
linda:chapter7>ldd /usr/bin/gcc
        linux-vdso.so.1 (0x00007ffe40bc2000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007fa0d9750000)
        /lib64/ld-linux-x86-64.so.2 (0x00007fa0d9b41000)
```