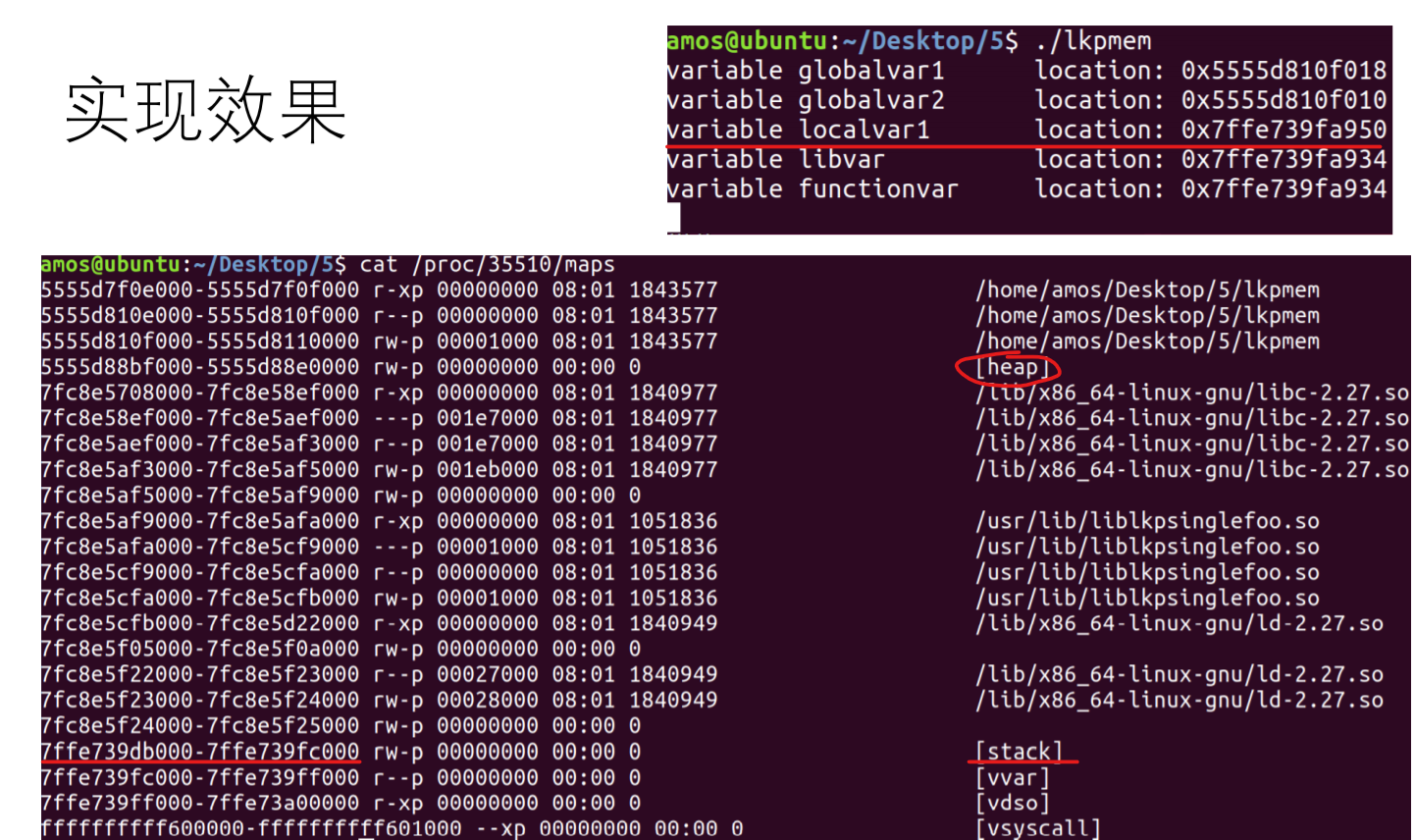
**内存映射：**

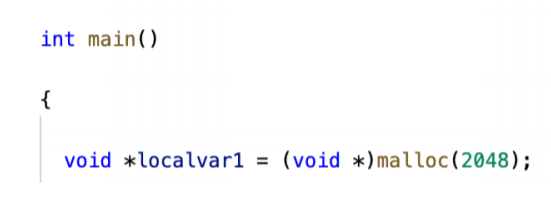
**1.实验结果显示由malloc动态分配的内存是在stack里，而不是在heap里**



原因：指针本身在stack里，指针指向的malloc分配的内存在heap里。

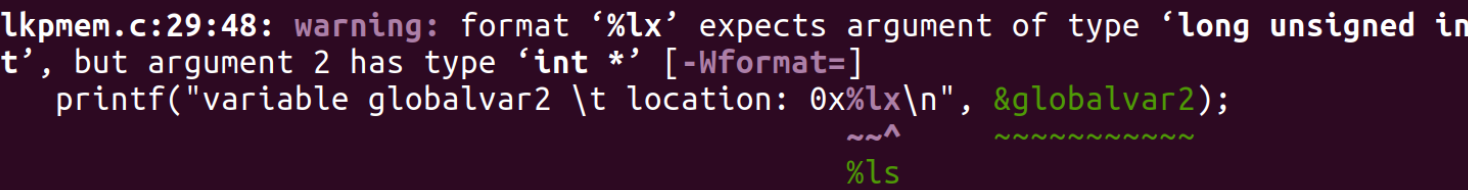
本实验中，localvar的值就是我们需要的地址；而对其再取地址符运算取到的是存放这个指针的地址。

pointers are (usually) stored on the stack. The memory they point to (usually allocate via malloc/calloc) is (usually) on the heap



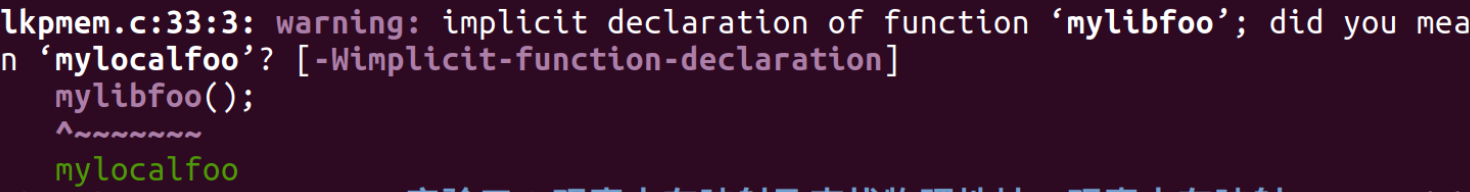
解决：直接取localvar值即可

**2.编译器类型转换warning，打印指针**



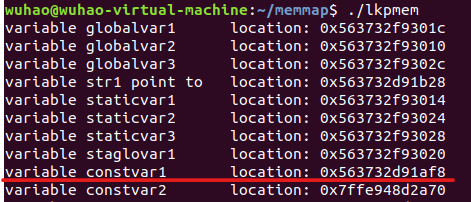
解决：对指针变量格式化打印改用%p

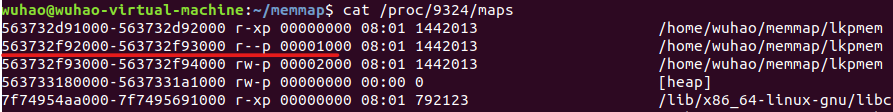
**3.编译器warning，函数未声明**



解决：添加声明void mylibfoo();

**4.const变量（已初始化的）对应哪个section？**

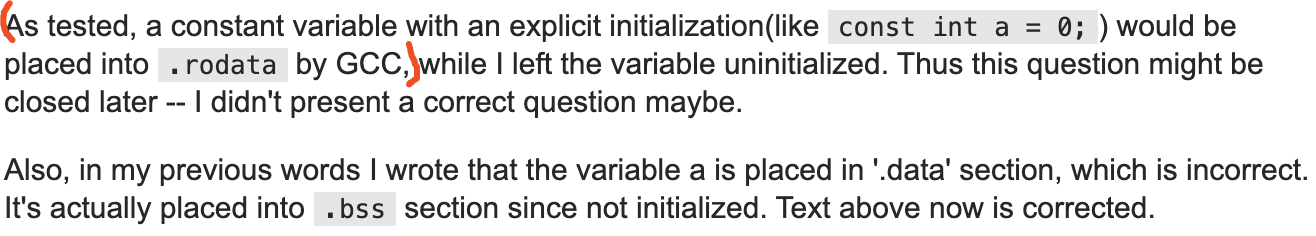




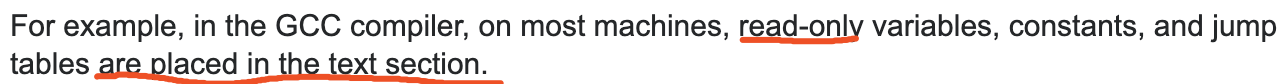
stackflow论坛关于这个有不同说法：

具体根据gcc版本或处理器而定，有的是在text域，有的在data域，我验证的是在text域

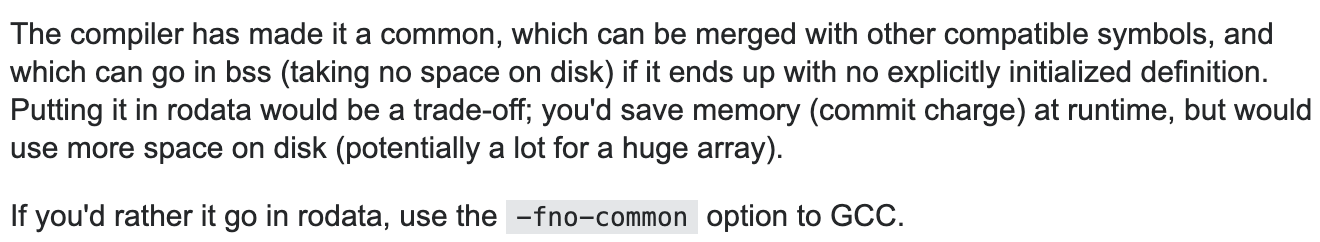
评论一：应该是在.rodata域



评论二：大部分机器是在text域



评论三：放到哪里由编译器所决定的，放到rodata可以节省内存，但可能会消耗更多存储空间，比如一个大数组。



**查找物理地址实验：**

**1.代码填充部分，对/proc文件进行读写用fread/fwrite可不可以**

回答：可以的，把/proc当成普通文件处理即可。fread/read一个是c库函数（高级IO），一个是系统调用（低级IO）。fread是包装了read，他们参数上是有区别的。

做法：

int fd 改掉=> FILE \* restrict fd;

fd = fopen("/proc/logadd2phyadd","rb+");

open read write close都要改成加f的，并注意参数。

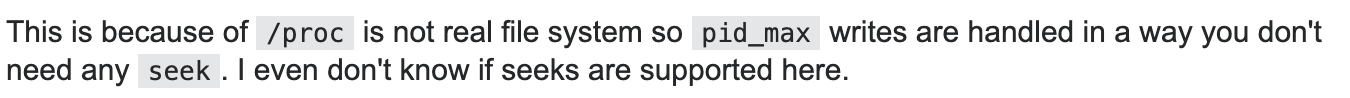
另外使用c库函数系列的读写在写之后还需要将指针归0，rewind(fd)（？问题2）

**2.（待解答）read/write时不需要lseek，在写之后指针自动回到起始位置，**

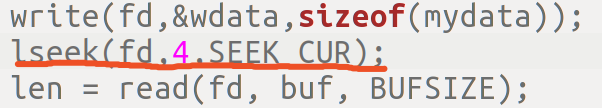
**但是fread/write时则需要rewind，否则输出就会少一部分。why?**

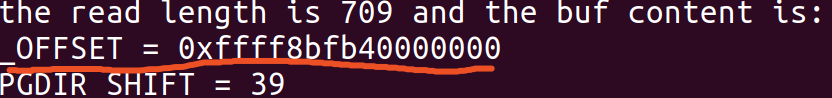
**结论：/proc有时可能不能简单的当作普通文件处理。**

stackflow一个评论说/proc不需要任何seek，并且他不都确定/proc是否支持seek



(我尝试的seek是支持的)





(可以看到PAGE被截掉了)