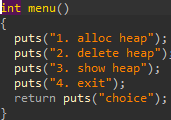
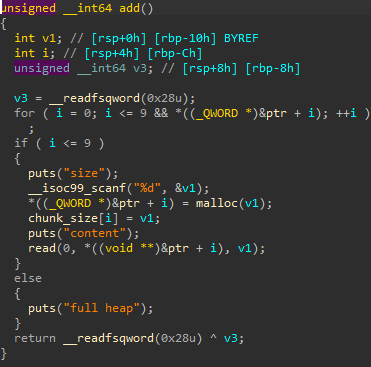
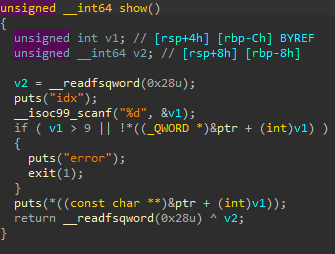
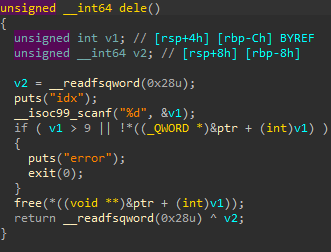
**省赛堆题free**



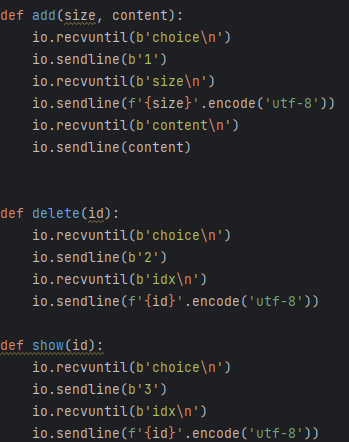
标准堆菜单



能分配九个堆

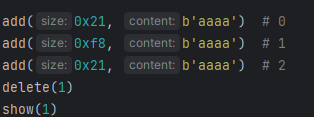


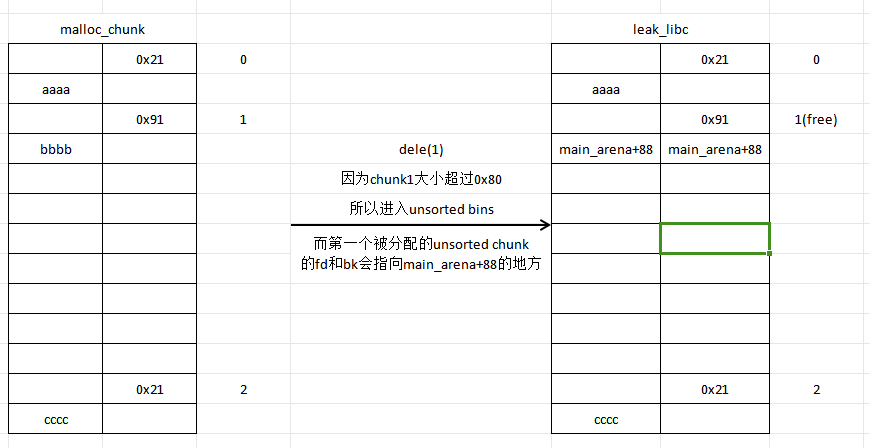
标准的利用函数



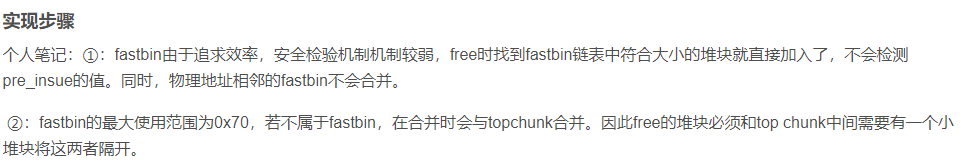
第一步：通过unsorted bins泄露libc地址

payload



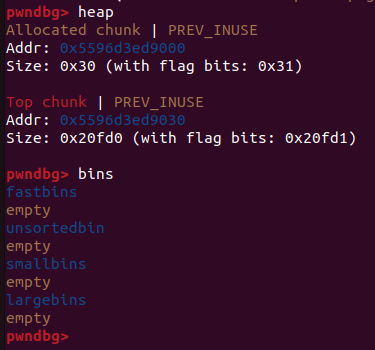
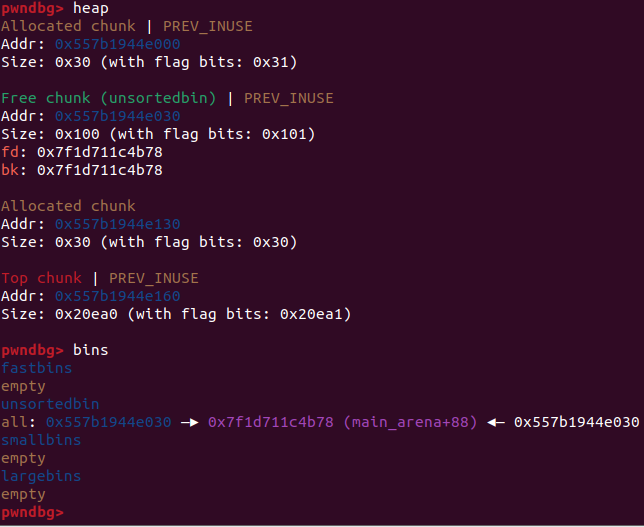


这样main\_arena的地址我们就知道了，并且我们还知道main\_arena-0x10的地方是malloc\_hook函数的地址，这样只要根据泄露出来的main\_arena减去(88+0x10)就能得到malloc\_hook函数的实际地址，再减去libc中的地址就能得到libc的基地址。

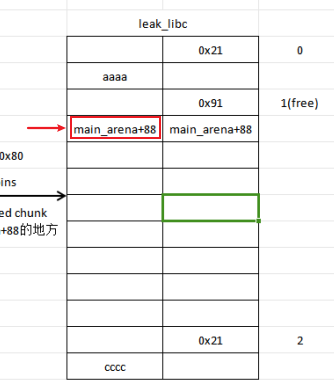
0号堆分不分配都无所谓，1号堆的要求是大于0x80，因为要让free的进unsorted bins，2号堆的大小没有要求，但是必须有，因为unsorted bins和top chunk之间没有堆的话，也就是物理地址相邻的话就会合并，见Loτυs爷的笔记：

<https://blog.csdn.net/Invin_cible/article/details/121322899>

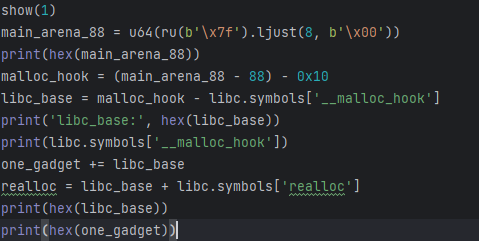
gdb看（左：分配012三个堆；右：不分配2，只分配01两个堆）



可以看到不加chunk2就直接跟top chunk合并了

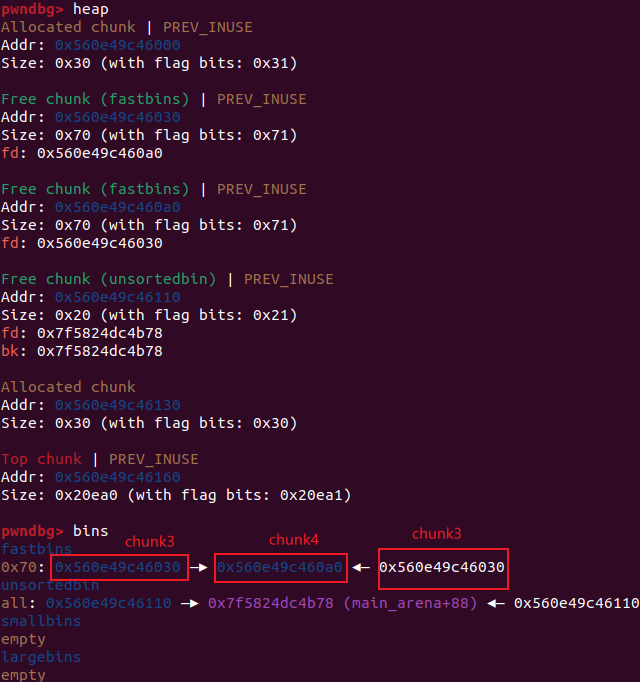


chunk1的指针就指向这里，所以show(1)的时候前八个字节就是我们要的，但是由于64位libc地址以0x7f开头，32位以0xf7开头（反正我见过的都是），所以更通用的写法是：ru(b'\x7f').ljust(8, b'\x00')，所以payload：



现在我们有libc的基地址，结合题目名可以选择打double free

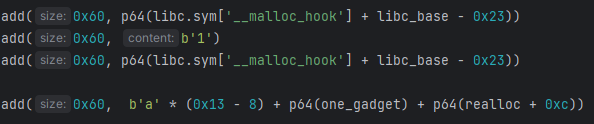
简单来说就是创建两个chunk：3和4，然后以343的顺序free，这样在bins里就会有这样的双向链表：



如果大小跟这些堆相同，堆申请会从左到右申请这些堆，所以我们申请第一次的时候，在chunk3对应fd的位置写入malloc\_hook附近的地址，这样我们申请完这三个堆，再申请第四个堆的时候就会往malloc\_hook附近的地址里写东西

（double free为什么要用fastbin也参考Loτυs爷的笔记）

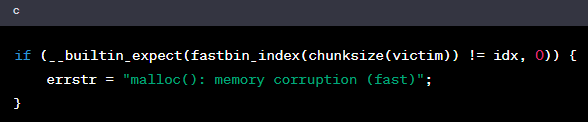
payload：



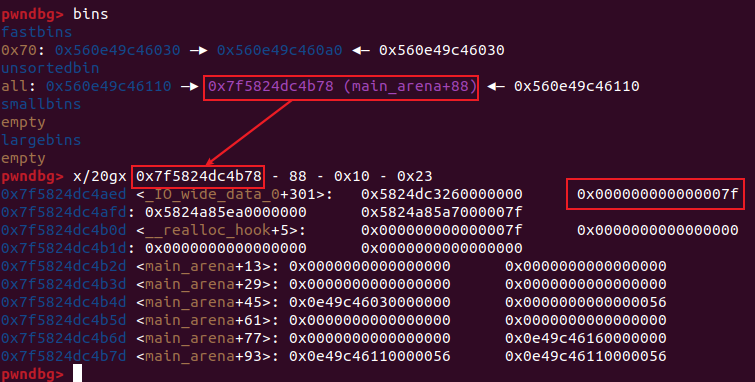
这里前三个add就对应chunk3 4 3，最后一个就是往malloc\_hook里写东西，这里要注意两点：

1. 为什么要将chunk3的fd指向malloc\_hook - 0x23

这里需要知道fastbin在分配的时候会检查size部分是否小于或等于最大的"fastbin"大小



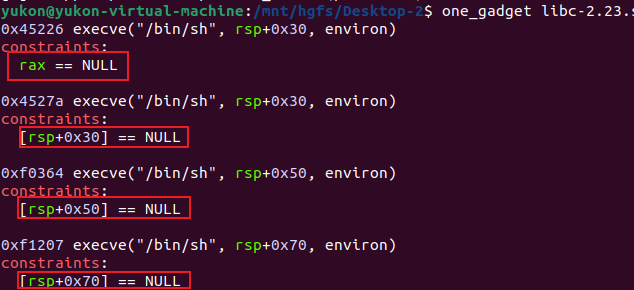
所以我们要找到一个满足要求的地址



这里的0x7f刚好就能绕过这个检测

堆的指针指向堆+0x10处，所以再填充0x13个字节就能填到malloc\_hook里面

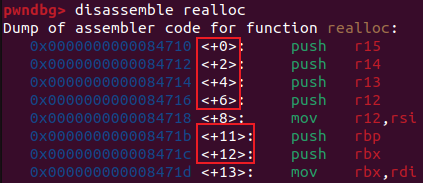
一般的思路就是填一个one\_gadget进去，但是这题所有one\_gadget的条件都不满足，所以需要一个realloc对栈进行调整，onegg的条件如下：



我们看realloc的汇编



这个函数中有6个push和6个pop，我们选择realloc+一定偏移就能实现减少几个push，由于栈是高地址向低地址生长，所以push压栈相当于降低栈顶（rsp），减少了push就相当于抬高了栈顶（rsp），具体抬几个能成功得计算，听说Loτυs爷建议爆破，那就爆破（见下图），最后得到offset为0xc的时候可以满足条件



1. malloc\_hook和realloc\_hook

执行malloc前会执行malloc\_hook，执行realloc之前会执行realloc\_hook，所以payload是这样写的： 先填充任意字符直到realloc\_hook前，填充realloc\_hook为onegg，

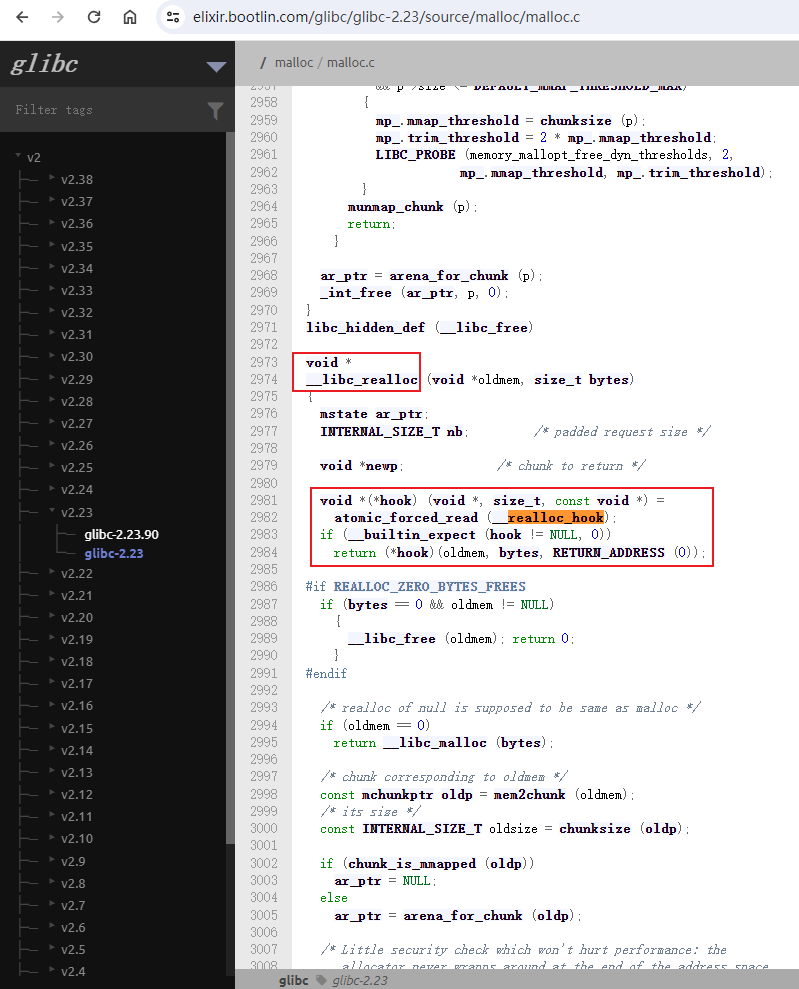
再填充任意字符直到malloc\_hook前，填充malloc\_hook为realloc，

这样执行顺序就是：

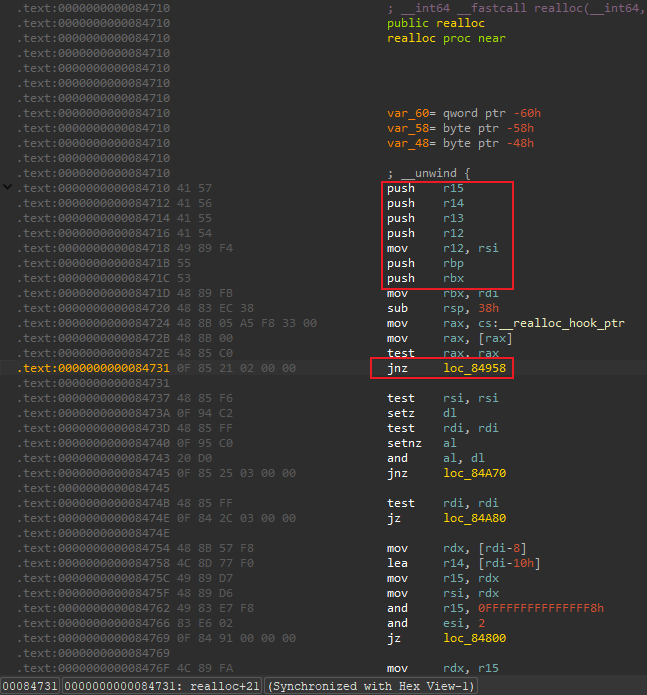
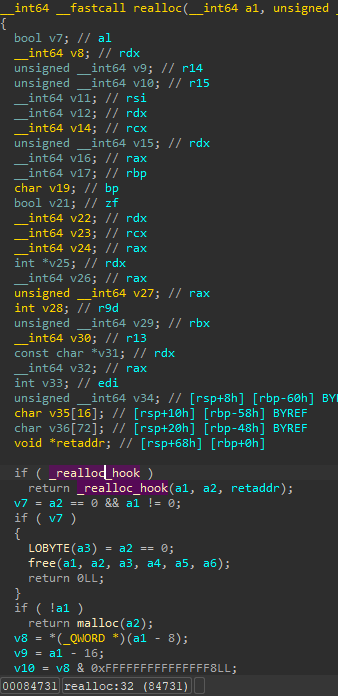
malloc -> malloc\_hook -> realloc -> realloc\_hook -> onegadget

这个顺序还有一点问题：realloc执行“前”会调用realloc\_hook，这样怎么realloc调整栈呢？

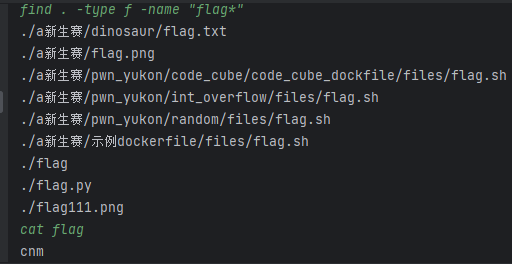
看到源码中是这样实现的：



用ida看可以发现在刚刚的push之后（见下图），所以是先调整了栈再进入的realloc\_hook，



也就是说已经使onegg条件满足了再调用的onegg，遂getshell



部分完整代码：

