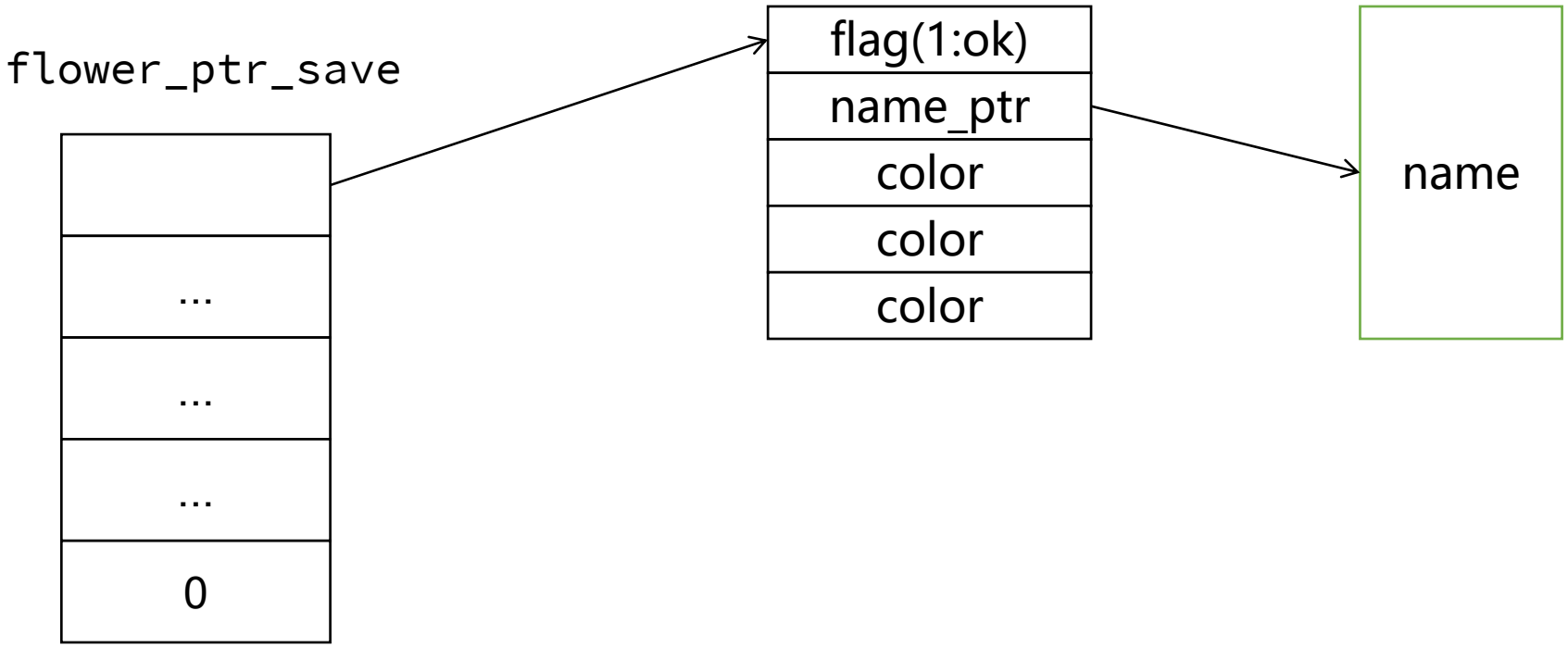
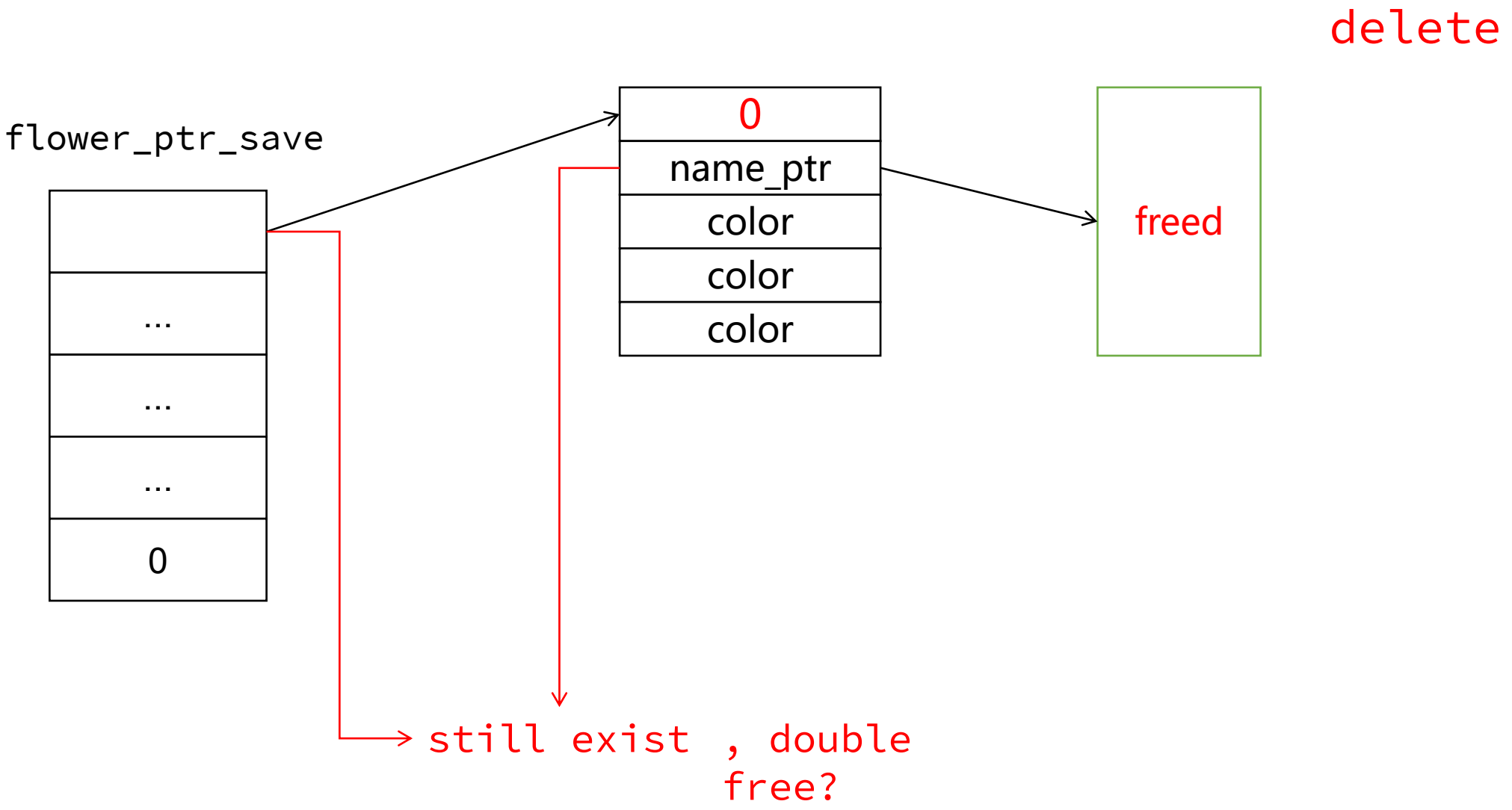

flower_ptr_save
...
...
...
0
flag(1:ok)
name_ptr
color
color
color
name


delete
flower_ptr_save
0
name_ptr
color
color
color
freed
...
...
...
0
still exist , double free?

能做哪些事情?

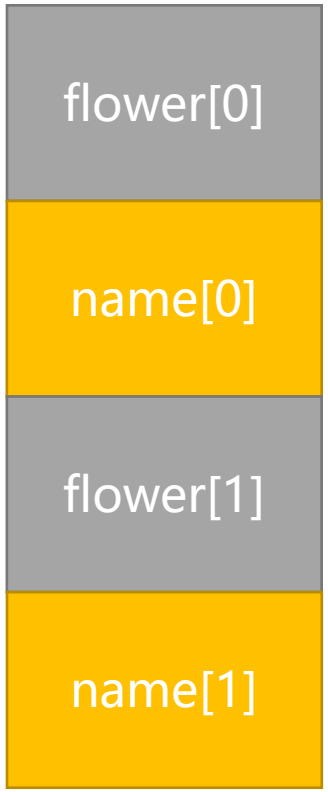
泄露Libc地址：是否可以通过输出来泄露？想方法获取name结构体。

泄露Stack地址：貌似不太行

泄露Heap地址：和泄露Libc地址异曲同工！！

执行System：很明显看到有Double Free

我开始的方法，类似hacknote，最终失败



two
malloc

two
free

one
malloc

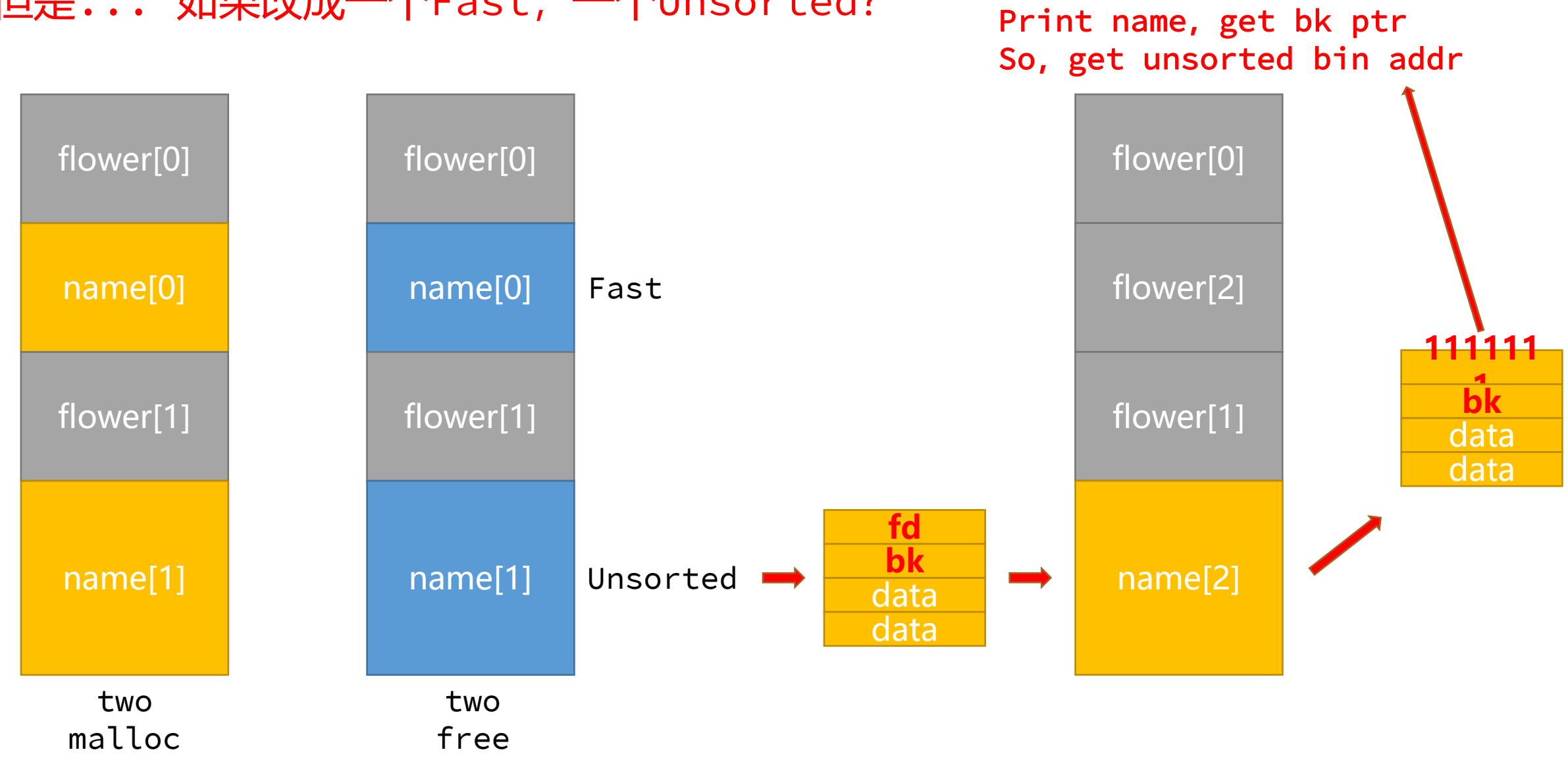
对于(指针)->(头部)->(内容)

hacknote之所以能够成功，是因为他可以控制(头部)

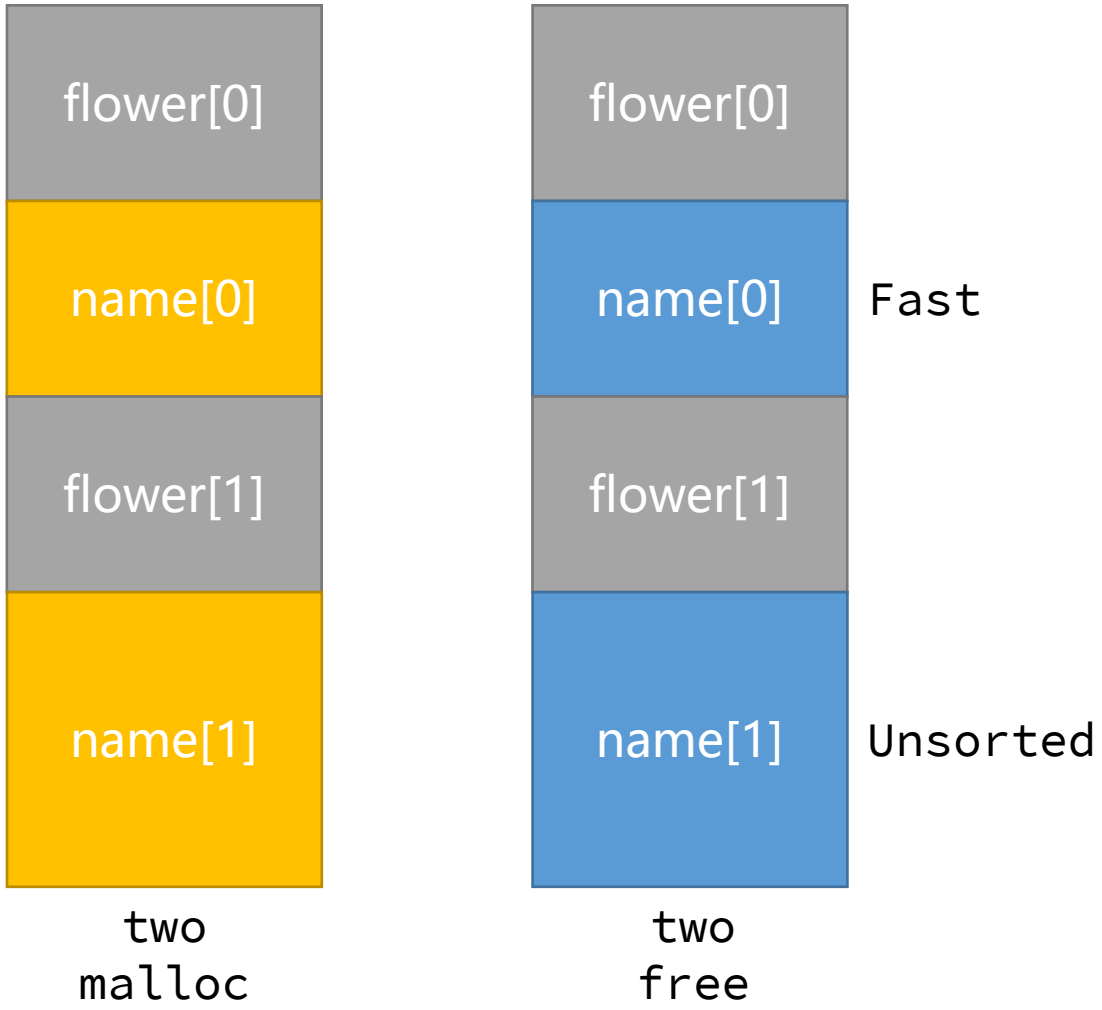
secretgarden不可以控制(头部)
所以这种方法不行

两题区别
这题没有free(flower)
只是单纯的让(flower->flag = 0)

但是... 如果改成一个Fast，一个Unsorted?



但是... 如果改成一个Fast，一个Unsorted?



有一个问题

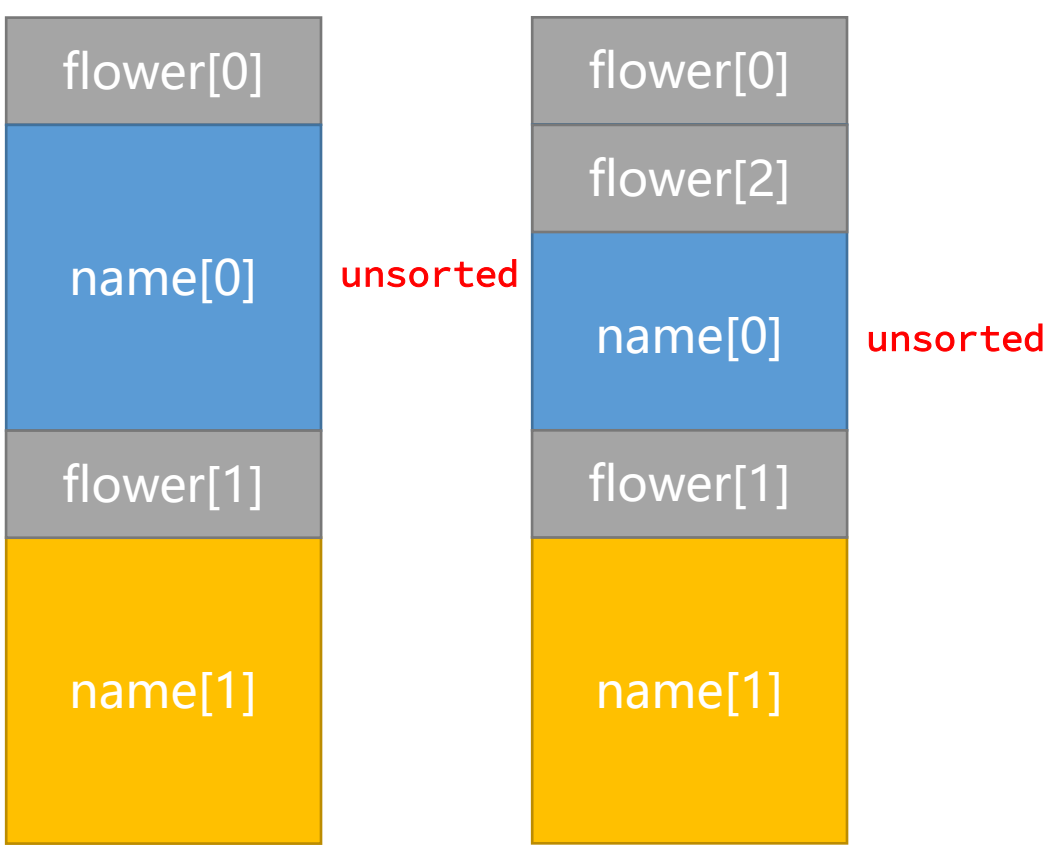
name[1] 属于最近一次申请，所以如果释放，就会合并到Top chunk

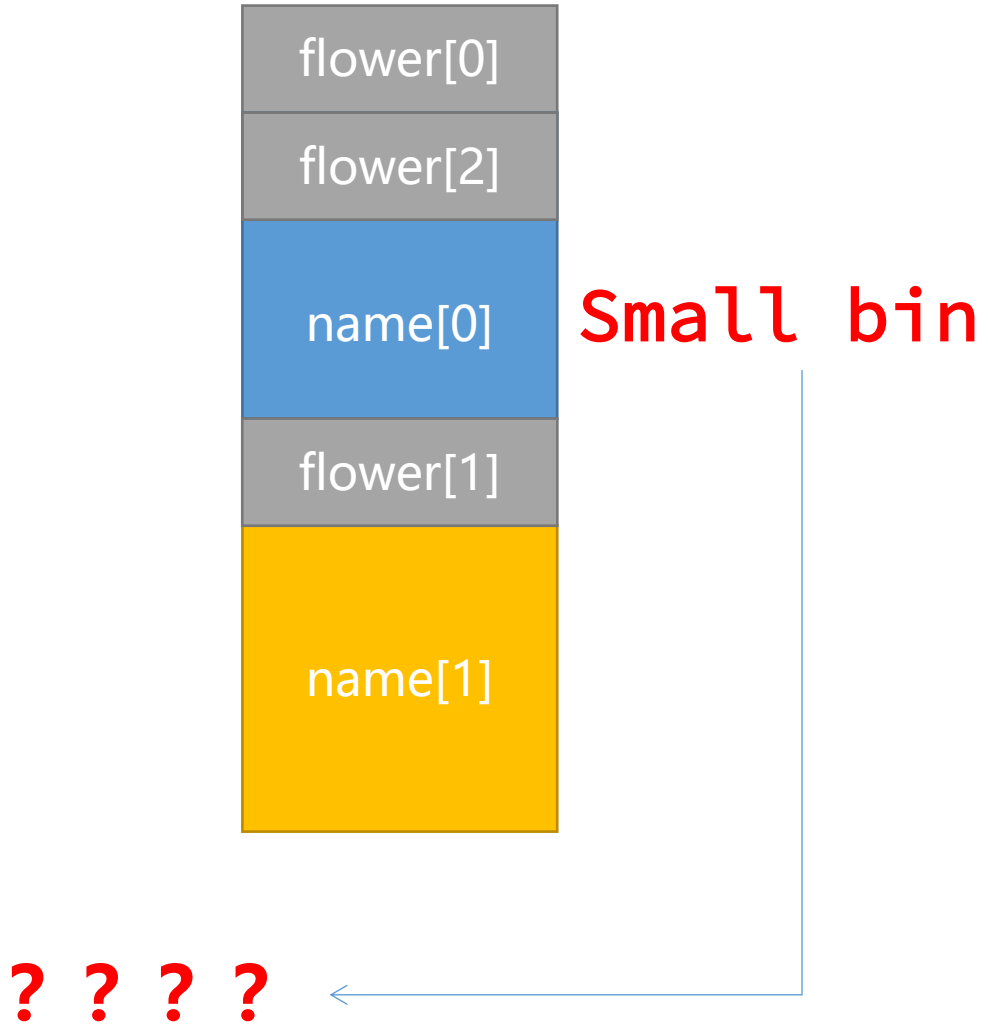
有两种解决方式

- 申请flower[2]，防止合并发生
- name[0] = Unsorted，name[1] = Fast

第二种利用了Fast 不会合并的特性，很赞!

还有一个方法，我看了一天才看懂！！





还有一个方法，我看了一天才看懂！！

这里要熟悉Malloc的流程，假设蓝块大小为0xe0，我们申请0xd0

- 查找对应Fastbin，Smallbin：失败

- 因为Unsorted bin中只有一块：尝试分裂。

但是！如果分裂后剩下的块小于0x20，是不会分裂的！！

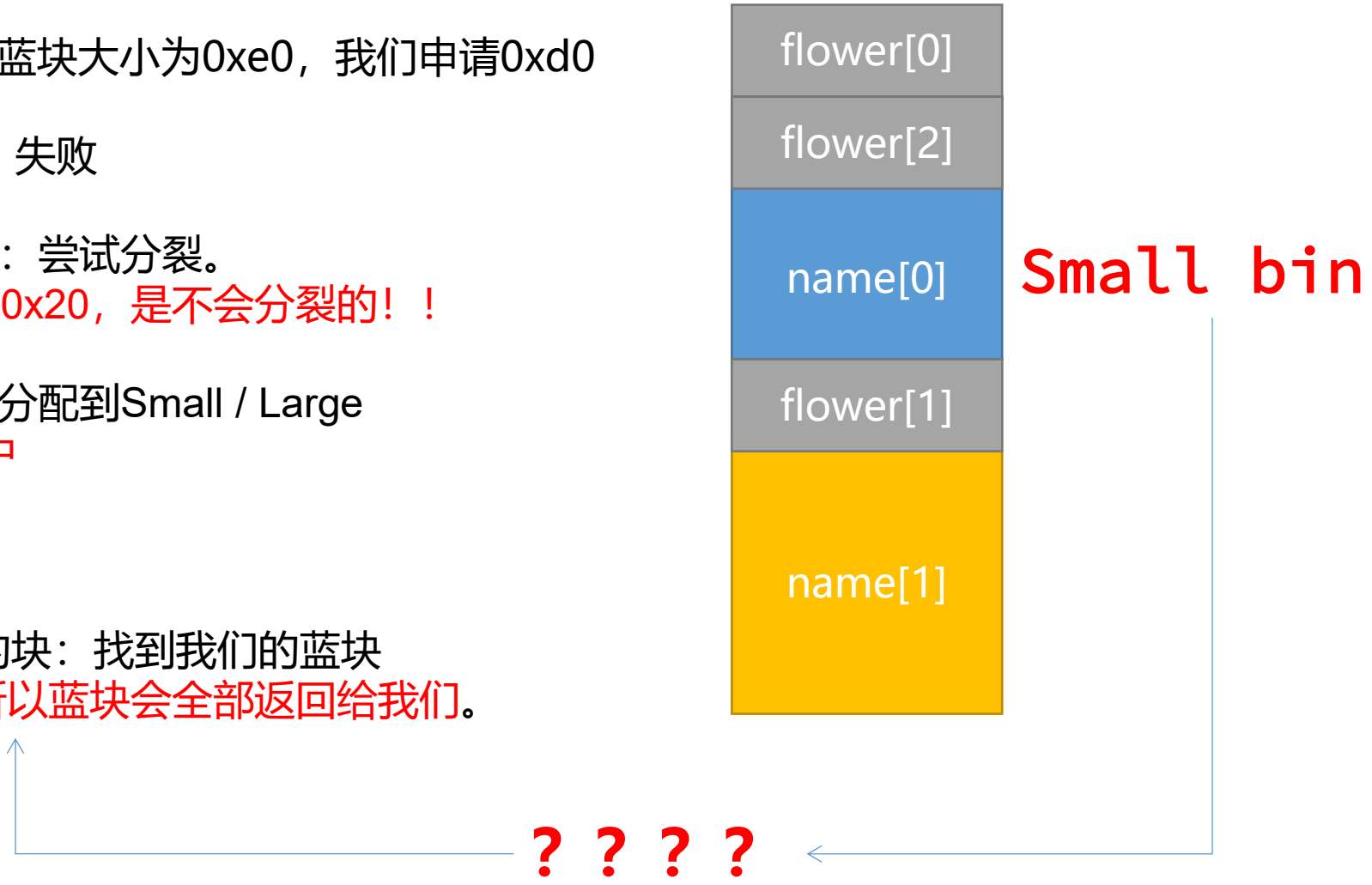
- 遍历Unsorted bin，根据大小分配到Small / Large

所以我们的蓝块会到Small bin 中

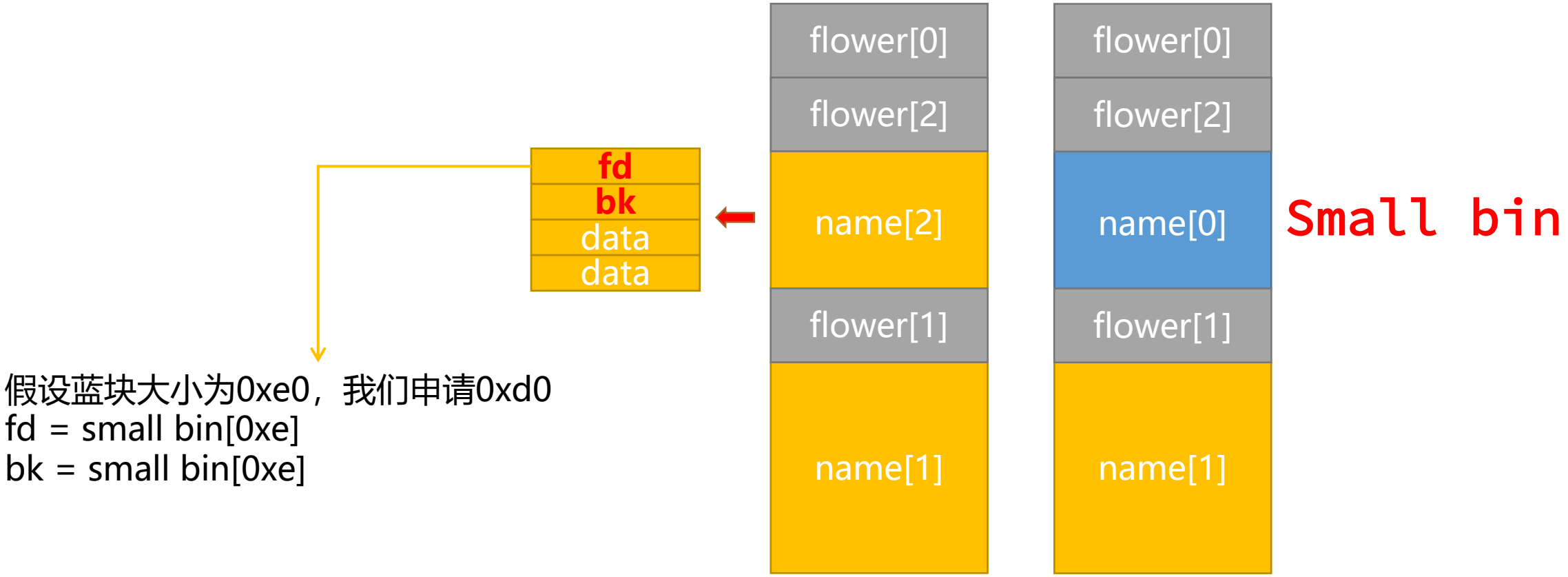
- 查找Large bin：失败

- 寻找是否有比我们申请块大的块：找到我们的蓝块

并且！由于分裂后小于0x20，所以蓝块会全部返回给我们。



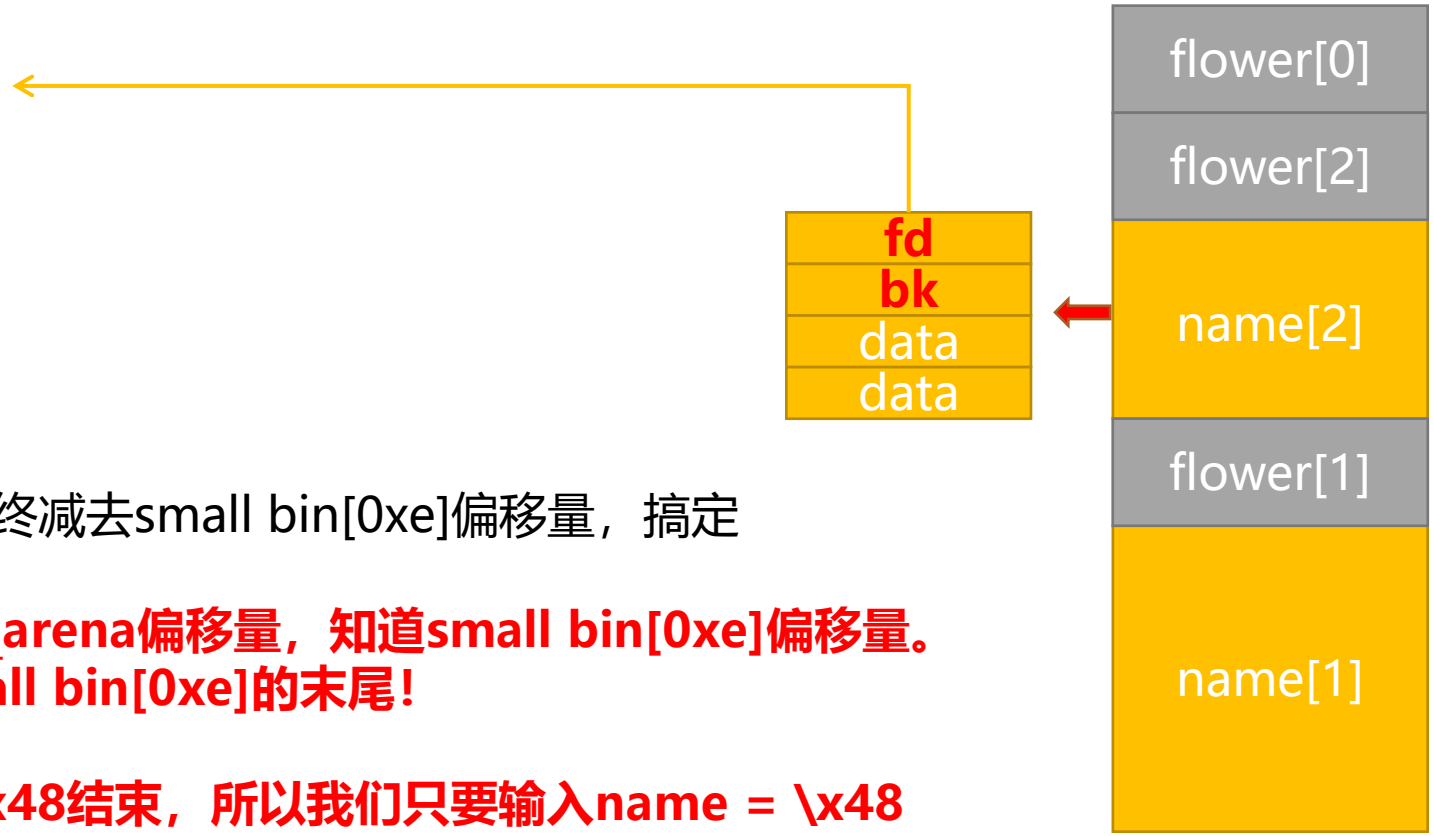
还有一个方法，我看了一天才看懂！！



假设蓝块大小为0xe0，我们申请0xd0
fd = small bin[0xe]
bk = small bin[0xe]

还有一个方法，我看了一天才看懂！！

假设蓝块大小为0xe0，我们申请0xd0
fd = small bin[0xe]
bk = small bin[0xe]



两种方法

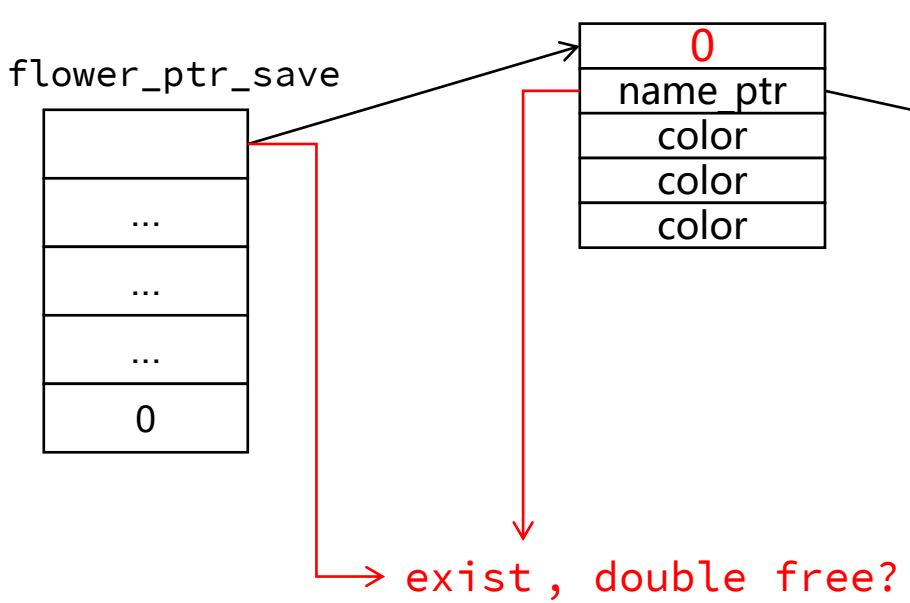
- 和前面一样，fd = 11111，可以得到bk，最终减去small bin[0xe]偏移量，搞定

- **对于libc，只要确定版本，就可以知道main_arena偏移量，知道small bin[0xe]偏移量。**

由于libc基址一定是\x00，所以就可以知道small bin[0xe]的末尾!

例子：在libc2.23中，small bin[0xe]一定是\x48结束，所以我们只要输入name = \x48
此时fd 仍然为 small bin[0xe]! 所以可以得到fd，得到small bin[0xe]，最终减去偏移，搞定

有了Libc基址，很显然开始执行System，很明显有个Double Free

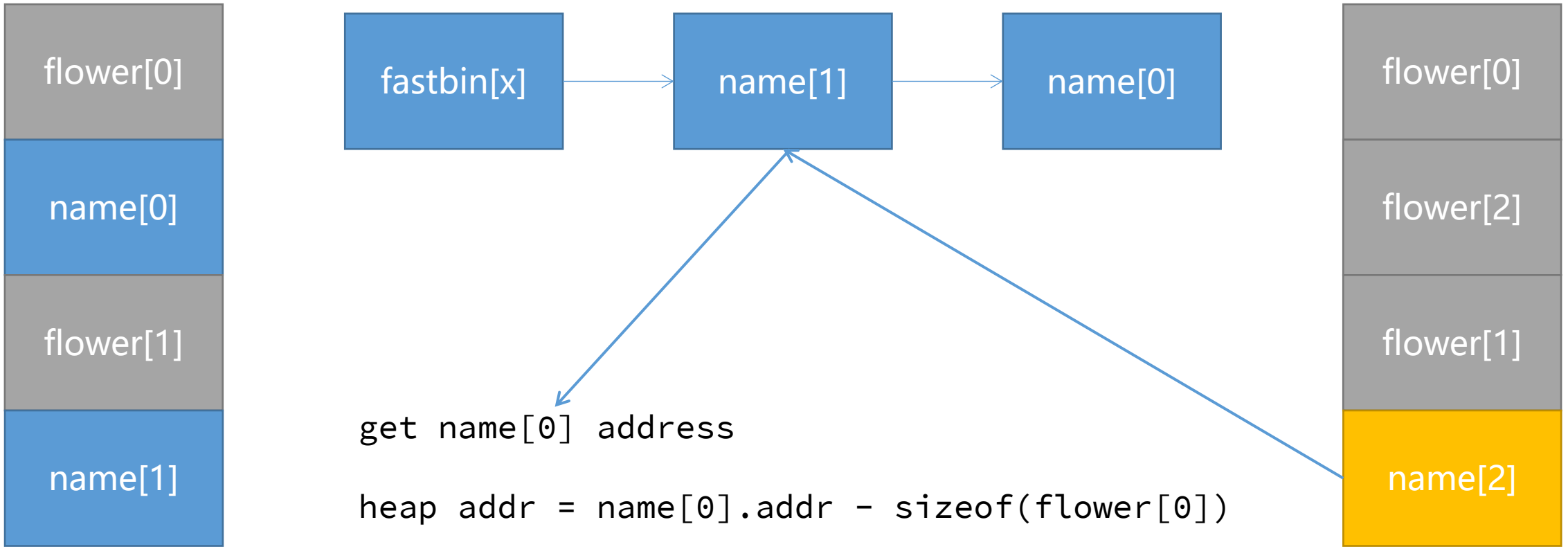


Suppose begin with index 4
- Malloc twice
- Free(4),Free(5),Free(4)
- Malloc(4, our_addr)
- Malloc(5, something)
- Malloc(4, something)
- Malloc(our_addr)

更改哪里的内容?

- GOT表：开了RELOAD，失败
- Libc_fini_main：ALSR，失败
- hook：尝试!
- IO_struct：尝试!

插句题外话，本题同样可以泄露Heap基址，方法同上面泄露Libc基址一样



Double Free，一开始搞错了，以为这就是House of spirit

Double Free ： 在链表上伪造一个fake_chunk，然后malloc(fake_chunk)
House of spirit ： 伪造一个fake_chunk，然后free(fake_chunk)

Double free check

- fake chunk 的 size 满足对应的 fastbin 需求

House of spirit check

- fake chunk 的 ISMMAP 位不能为 1
- fake chunk 地址需要对齐， MALLOC_ALIGN_MASK
- fake chunk 的 size 大小需要满足对应的 fastbin 的需求，同时也得对齐。
- fake chunk 的 next chunk 的大小 [2 * SIZE_SZ, av->system_mem]

开始修改hook，很快就找到了，malloc_hook - 0x23，可以当做头部，改成one_gadget即可。

但是...本题，one_gadget 无一可用。

解决方法
直接调用malloc，one_gadget 条件不满足
因此要先改变流程，**比如执行普通的Double Free，让系统去触发**malloc_printerr，此时会调用malloc，而这个时候one_gadget 条件得到了满足

成功解决!

开始修改IO_struct，很快就找到了，IO_stdout + 0x9d，可以当做头部

stdout 调用 printf 时，会调用vtable 表中的 xsputn.

处理步骤

- 先Double Free，可以操纵IO_stdout + 0x9d + 0x10 后面的数据内容
- vtable 在可修改范围内，所以修改 vtable 指针地址．该地址要能够让我们可以构建假的 xsputn.
- 假设 vtable 修改的地址为A，xsputn 偏移为B．那就把(A+B)那里，改成one_gadget 即可

最终很完美，成功解决!