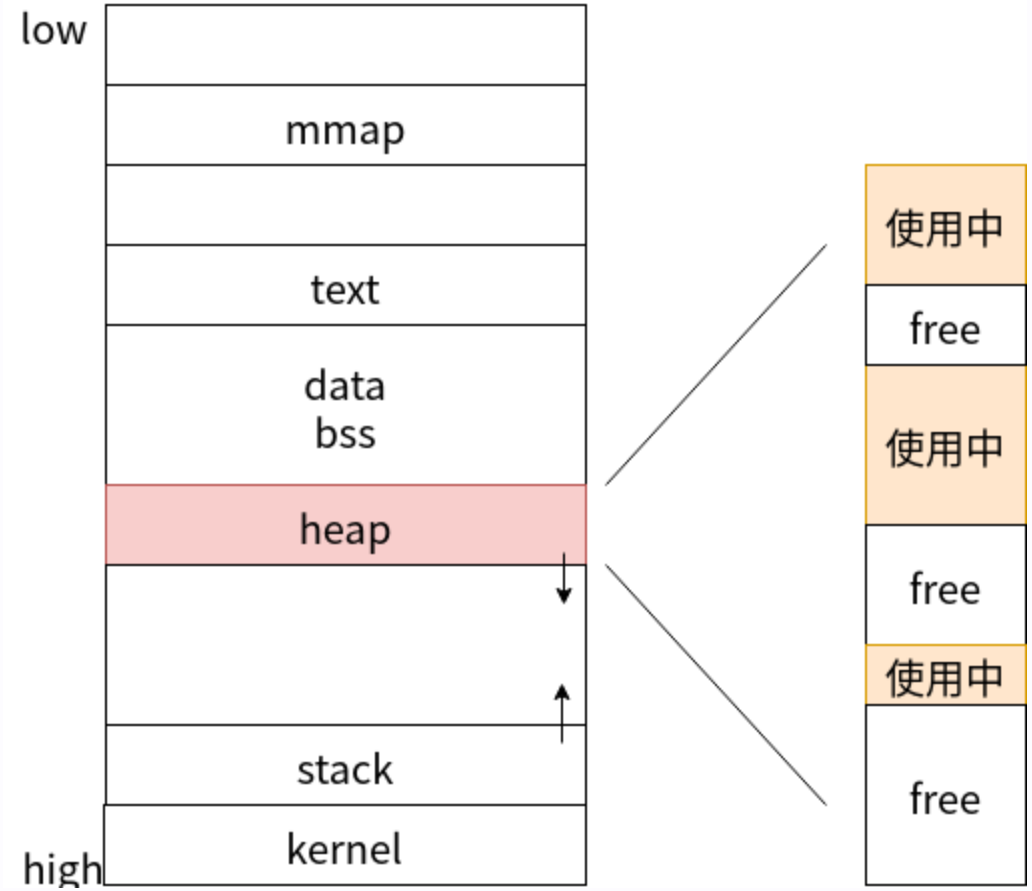
heap勉強会

- mallocやfreeに関して学ぶ
- `INTERNAL_SIZE_T prev_size`
- `INTERNAL_SIZE_T size`
- INTERNAL_SIZE_Tは64bit環境の場合は64bit
- 32bit環境の場合は32bit

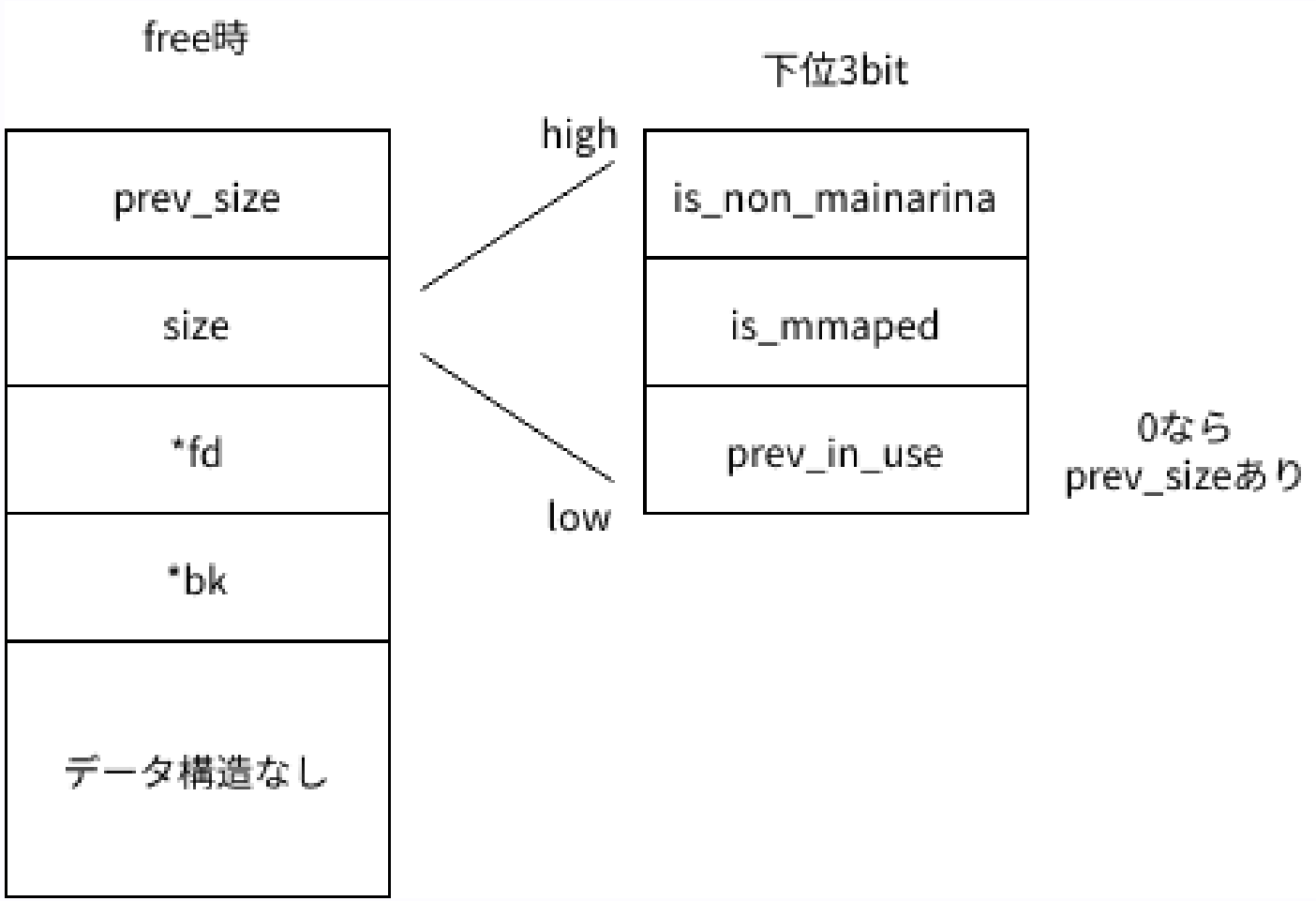
参考資料

- https://faraz.faith/2019-10-12-picoctf-2019-heap-challs/
- https://ctf-wiki.github.io/ctf-wiki/pwn/linux/glibc-heap/heap_structure/
- ptr-yudaiさんのCTFするぞやwrite-up
- https://furutsuki.hatenablog.com/entry/2019/02/26/112

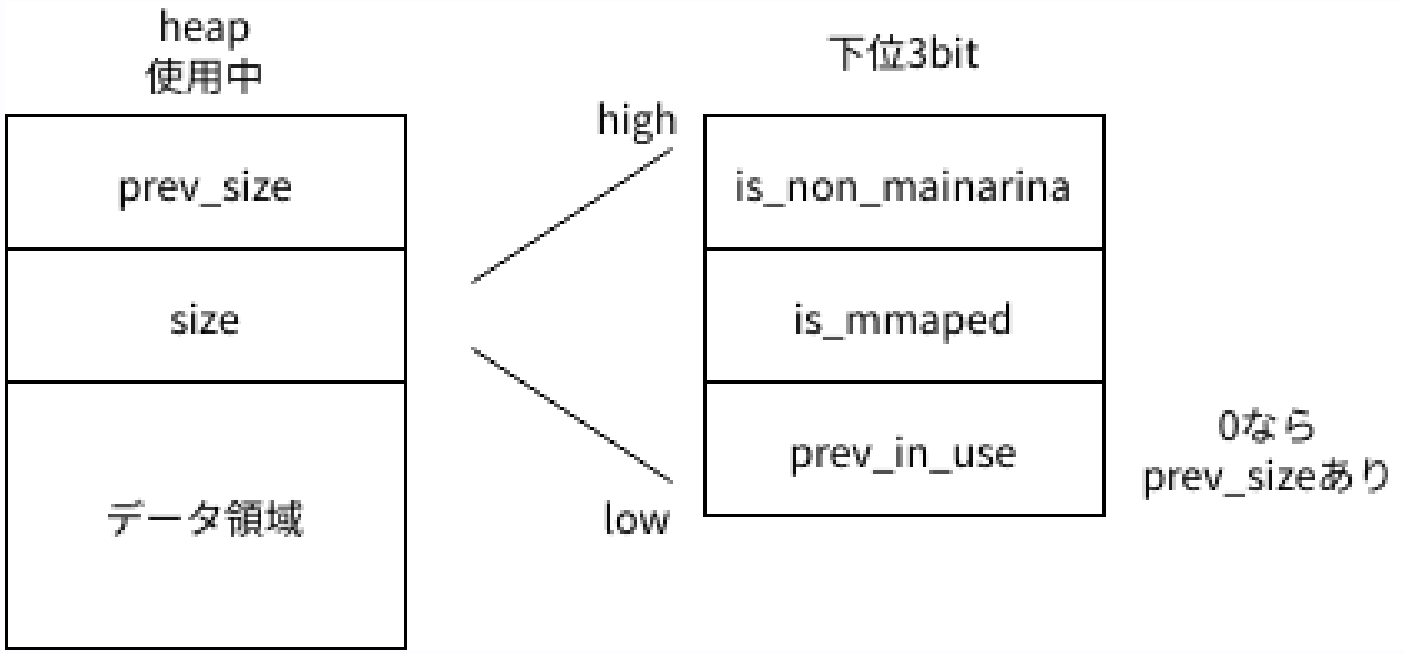
heap領域はどこ？

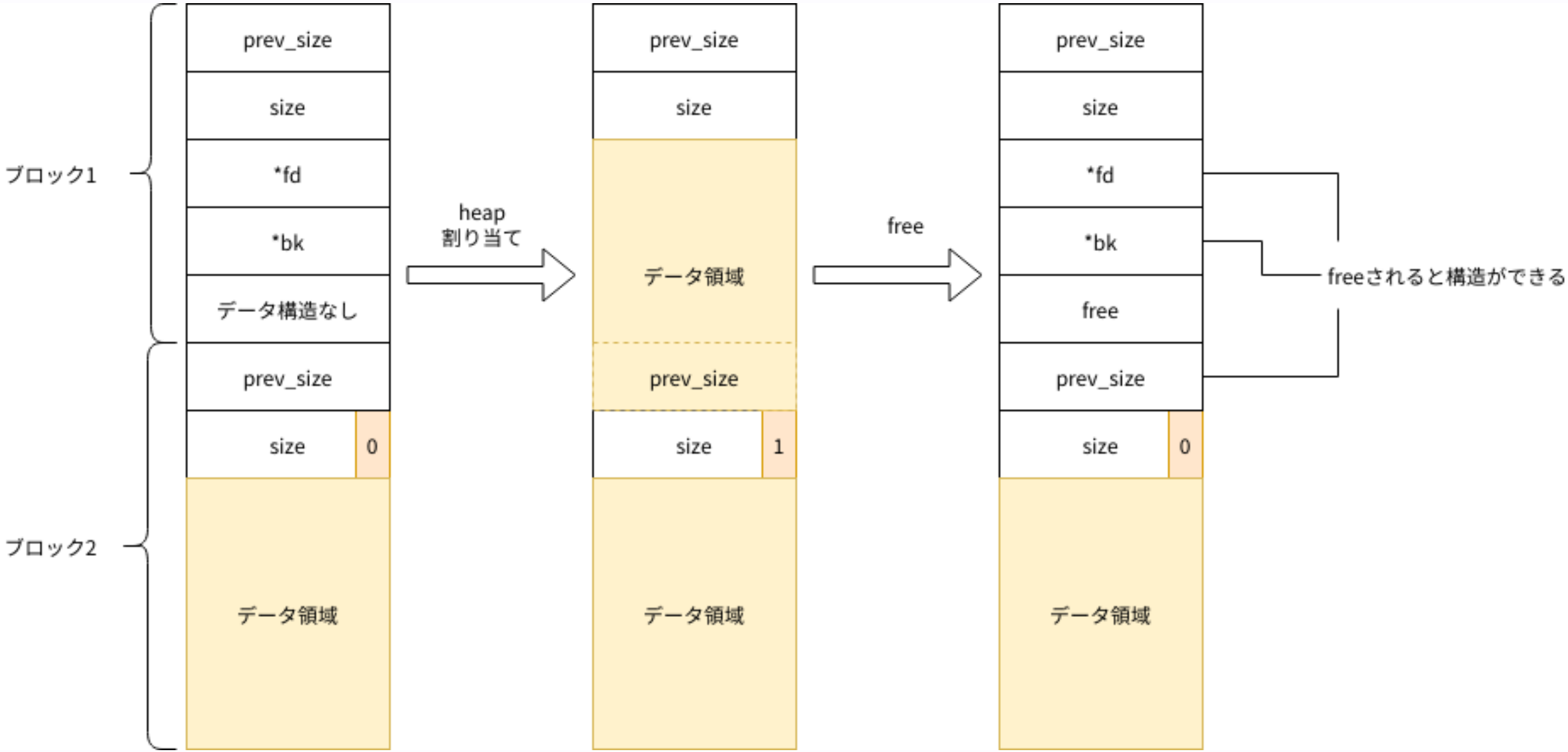

low
mmap
text
data
bss
heap
stack
kernel
high
使用中
free
使用中
free
使用中
free

free時のデータ構造



malloc時のデータ構造




ブロック1
ブロック2
prev_size
size
*fd
*bk
データ構造なし
prev_size
size
0
データ領域
heap
割り当て
prev_size
size
データ領域
prev_size
size
1
データ領域
free
prev_size
size
*fd
*bk
free
prev_size
size
0
データ領域
freeされると構造ができる

free listはこんなに単純？

- 実は数種類ある
  - t-cache
  - fast bin
  - small bin
  - large bin
  - unsorted bin

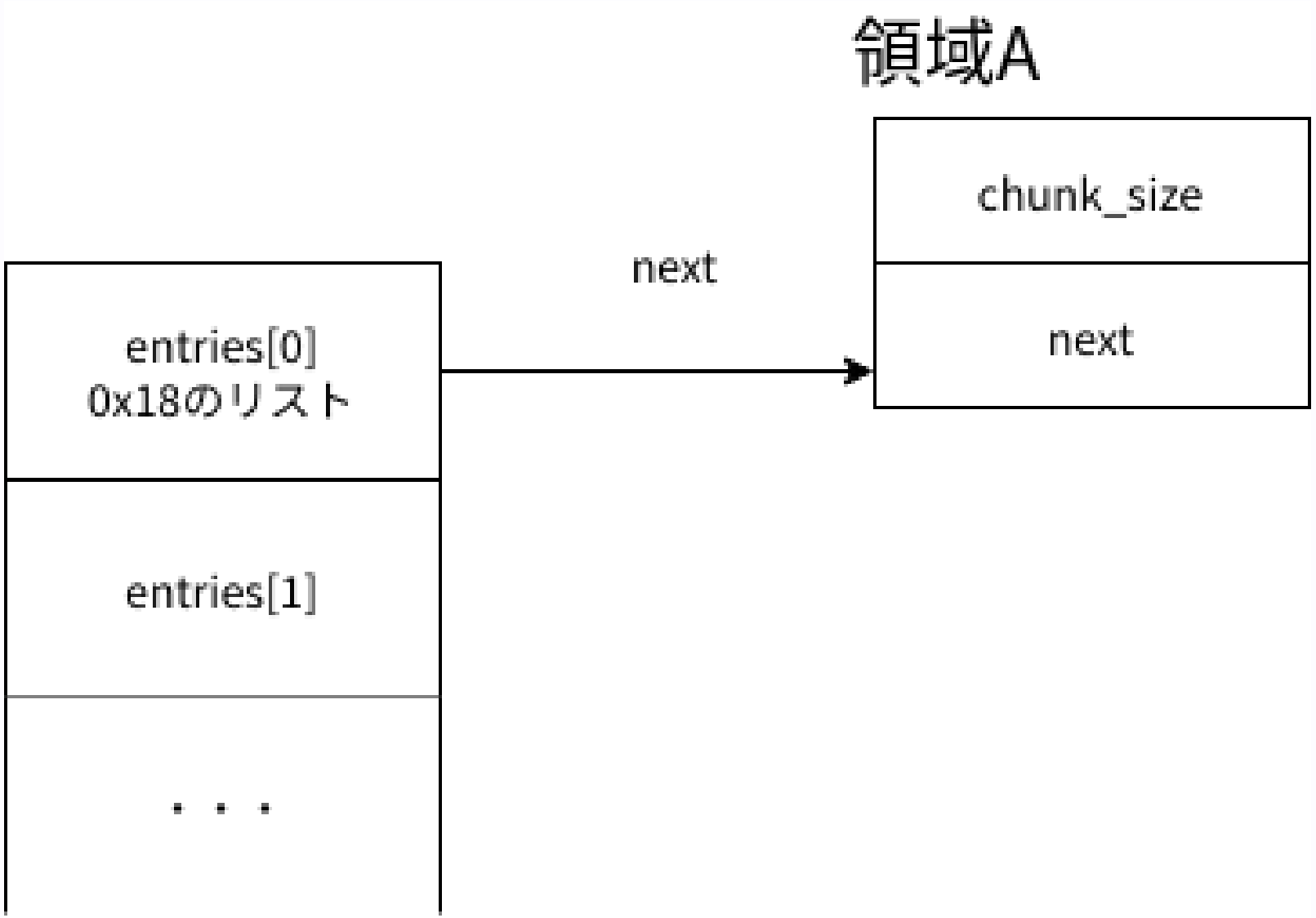
t-cache

- glibc 2.26から追加されたもの
  - スレッド毎のキャッシュ
  - 排他制御の必要がないので高速
- 64bitだと `TCACHE_MAX_BINS` は64になっている
  - キャッシュサイズは0x18, 0x28, 0x38, ... 0x408バイト以下というように区切られている
  - リンクリストの長さは `TCACHE_FILL_COUNT` によって制限されていて7になっている

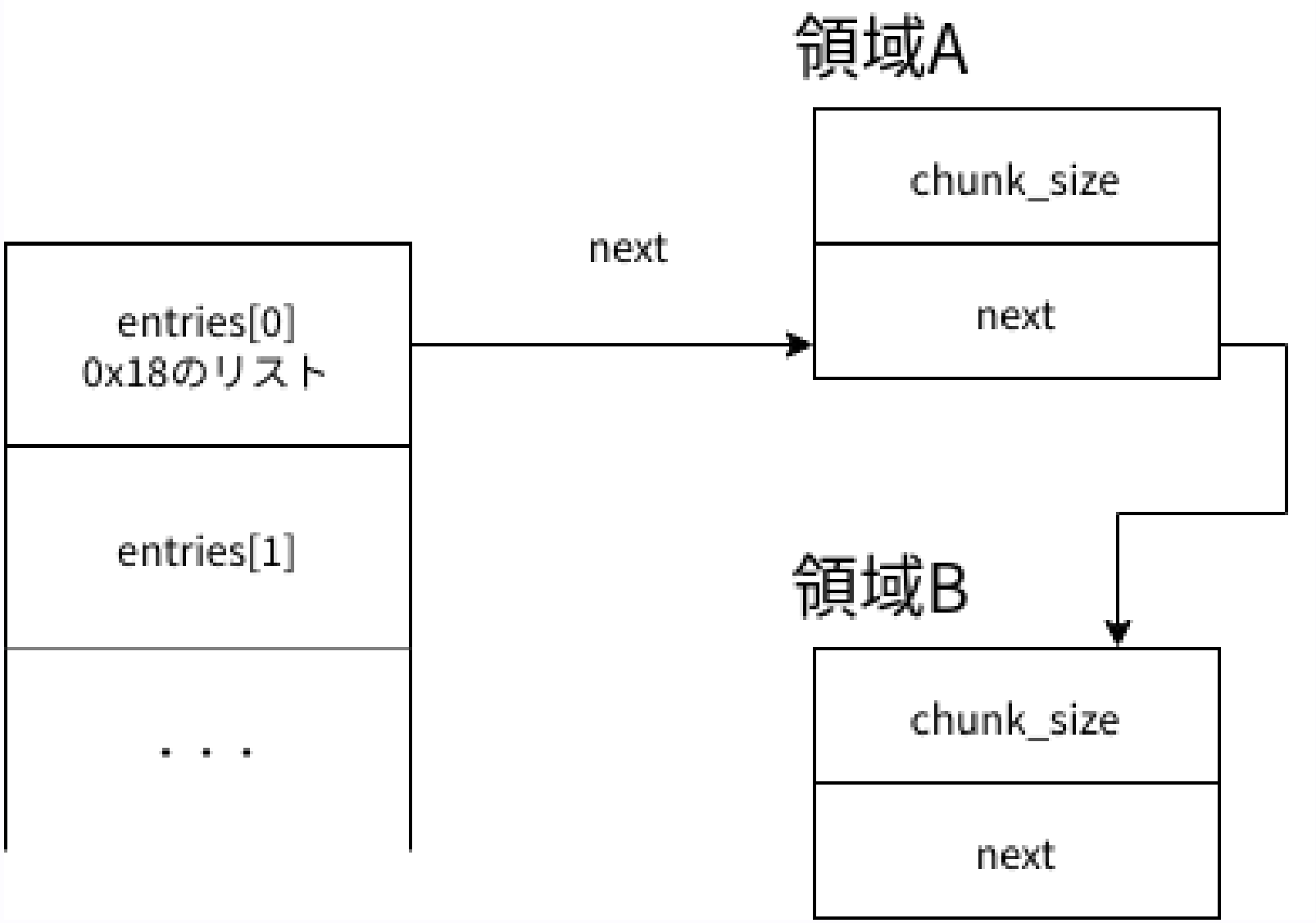
t-cache

- Aを0x10でmallocする
- Bを0x10でmallocする
- Aをfreeする
- Bをfreeする
- Cを0x10でmallocする
- この場合どんな感じになるか

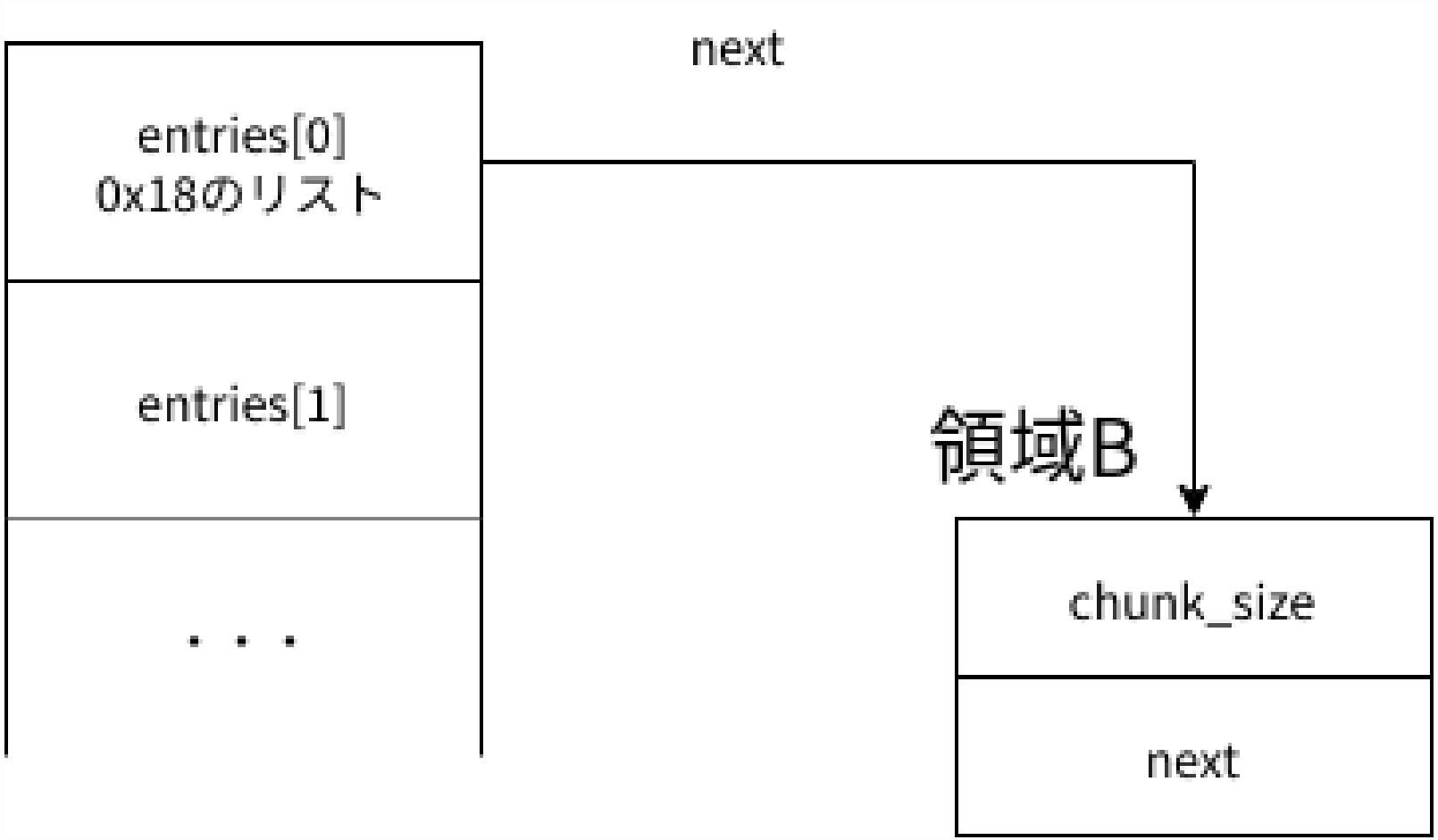
t-cache



t-cache


領域A
chunk_size
next
next
entries[0]
0x18のリスト
entries[1]
. . .
領域B
chunk_size
next

t-cache



fast bin

- 小さなメモリブロックをmallocしてはfreeするという処理がよくある
  - マウスの移動など
- freeリストで隣接している部分はくっつける

unsorted bin

- free後すぐに同じ大きさのmallocが呼ばれやすいことから作られた