Heap_analysis_0

07-25 서동훈

glibc 2.25 source code

발표가 아파요.....

malloc

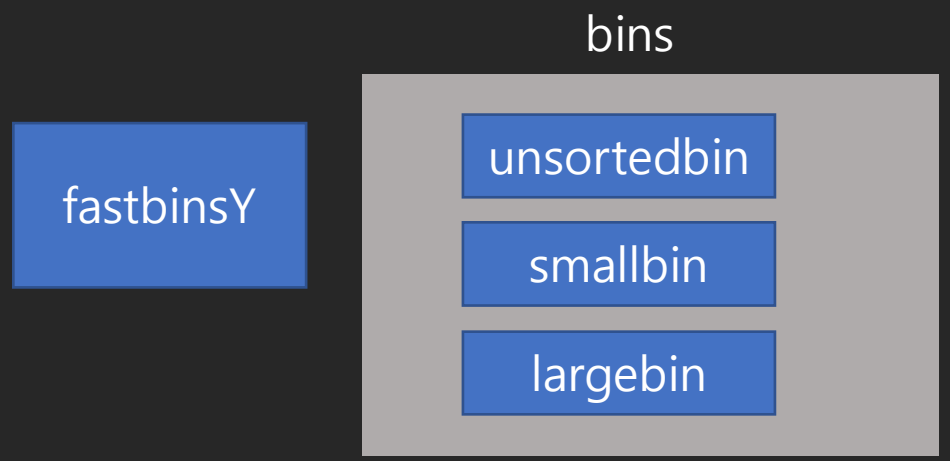
malloc(size)

__libc_malloc() -> _int_malloc() -> sysmalloc()

__libc_malloc

```c
struct malloc_state
{
  /* Serialize access.  */
  mutex_t mutex;

  /* Flags (formerly in max_fast).  */
  int flags;

  /* Fastbins */
  mfastbinptr fastbinsY[NFASTBINS];

  /* Base of the topmost chunk -- not otherwise kept in a bin */
  mchunkptr top;

  /* The remainder from the most recent split of a small request */
  mchunkptr last_remainder;

  /* Normal bins packed as described above */
  mchunkptr bins[NBINS * 2 - 2];

  /* Bitmap of bins */
  unsigned int binmap[BINMAPSIZE];

  /* Linked list */
  struct malloc_state *next;

  /* Linked list for free arenas.  Access to this field is serialized
     by free_list_lock in arena.c.  */
  struct malloc_state *next_free;

  /* Number of threads attached to this arena.  0 if the arena is on
     the free list.  Access to this field is serialized by
     free_list_lock in arena.c.  */
  INTERNAL_SIZE_T attached_threads;

  /* Memory allocated from the system in this arena.  */
  INTERNAL_SIZE_T system_mem;
  INTERNAL_SIZE_T max_system_mem;
};
```

```c
#ifndef _MALLOC_H
#include <malloc/malloc.h>


/* In the GNU libc we rename the global variable
   `__malloc_initialized' to `__libc_malloc_initialized'.  */
#define __malloc_initialized __libc_malloc_initialized
/* Nonzero if the malloc is already initialized.  */
extern int __malloc_initialized attribute_hidden;

struct malloc_state;
typedef struct malloc_state *mstate;

#endif
```



__libc_malloc

```
1   void *
2   __libc_malloc (size_t bytes)
3   {
4     mstate ar_ptr;
5     void *victim;
6     void *(*hook) (size_t, const void *)
7       = atomic_forced_read (__malloc_hook);
8     if (__builtin_expect (hook != NULL, 0))
9       return (*hook)(bytes, RETURN_ADDRESS (0));
10    arena_get (ar_ptr, bytes);
11    victim = _int_malloc (ar_ptr, bytes);
12    /* Retry with another arena only if we were able to find a usable arena
13       before.  */
14    if (!victim && ar_ptr != NULL)
15      {
16        LIBC_PROBE (memory_malloc_retry, 1, bytes);
17        ar_ptr = arena_get_retry (ar_ptr, bytes);
18        victim = _int_malloc (ar_ptr, bytes);
19      }
20    if (ar_ptr != NULL)
21      (void) mutex_unlock (&ar_ptr->mutex);
22    assert (!victim || chunk_is_mmapped (mem2chunk (victim)) ||
23            ar_ptr == arena_for_chunk (mem2chunk (victim)));
24    return victim;
25  }
26  libc_hidden_def (__libc_malloc)
```

_int_malloc

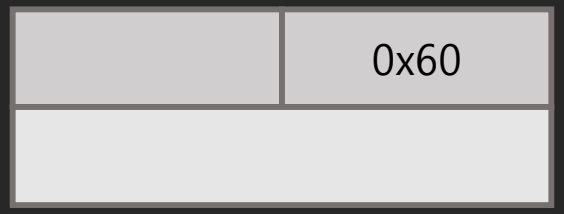
```
1  static void *
2  _int_malloc (mstate av, size_t bytes)
3  {
4    INTERNAL_SIZE_T nb;               /* normalized request size */
5    unsigned int idx;                 /* associated bin index */
6    mbinptr bin;                      /* associated bin */
7
8    mchunkptr victim;                 /* inspected/selected chunk */
9    INTERNAL_SIZE_T size;             /* its size */
10   int victim_index;                 /* its bin index */
11
12   mchunkptr remainder;              /* remainder from a split */
13   unsigned long remainder_size;     /* its size */
14
15   unsigned int block;               /* bit map traverser */
16   unsigned int bit;                 /* bit map traverser */
17   unsigned int map;                 /* current word of binmap */
18
19   mchunkptr fwd;                    /* misc temp for linking */
20   mchunkptr bck;                    /* misc temp for linking */
21
22   const char *errstr = NULL;
```

_int_malloc

```
1  /*
2       Convert request size to internal form by adding SIZE_SZ bytes
3       overhead plus possibly more to obtain necessary alignment and/or
4       to obtain a size of at least MINSIZE, the smallest allocatable
5       size. Also, checked_request2size traps (returning 0) request sizes
6       that are so large that they wrap around zero when padded and
7       aligned.
8     */
9
10    checked_request2size (bytes, nb);
```

malloc(80) ➡ (80+16)mod 16 == 0

bytes=80 ➡ nb = 0x60(96)



_int_malloc

```
/* There are no usable arenas.  Fall back to sysmalloc to get a chunk from
      mmap.  */
  if (__glibc_unlikely (av == NULL))
    {
      void *p = sysmalloc (nb, av);
      if (p != NULL)
    alloc_perturb (p, bytes);
      return p;
    }
```

_int_malloc

```
1   if ((unsigned long) (nb) <= (unsigned long) (get_max_fast ()))
2     {
3       idx = fastbin_index (nb);
4       mfastbinptr *fb = &fastbin (av, idx);
5       mchunkptr pp = *fb;
6       do
7         {
8           victim = pp;
9           if (victim == NULL)
10            break;
11        }
12      while ((pp = catomic_compare_and_exchange_val_acq (fb, victim->fd, victim))
13             != victim);
14      if (victim != 0)
15        {
16          if (__builtin_expect (fastbin_index (chunksize (victim)) != idx, 0))
17            {
18              errstr = "malloc(): memory corruption (fast)";
19            errout:
20              malloc_printerr (check_action, errstr, chunk2mem (victim), av);
21              return NULL;
22            }
23          check_remalloced_chunk (av, victim, nb);
24          void *p = chunk2mem (victim);
25          alloc_perturb (p, bytes);
26          return p;
27        }
28    }
```

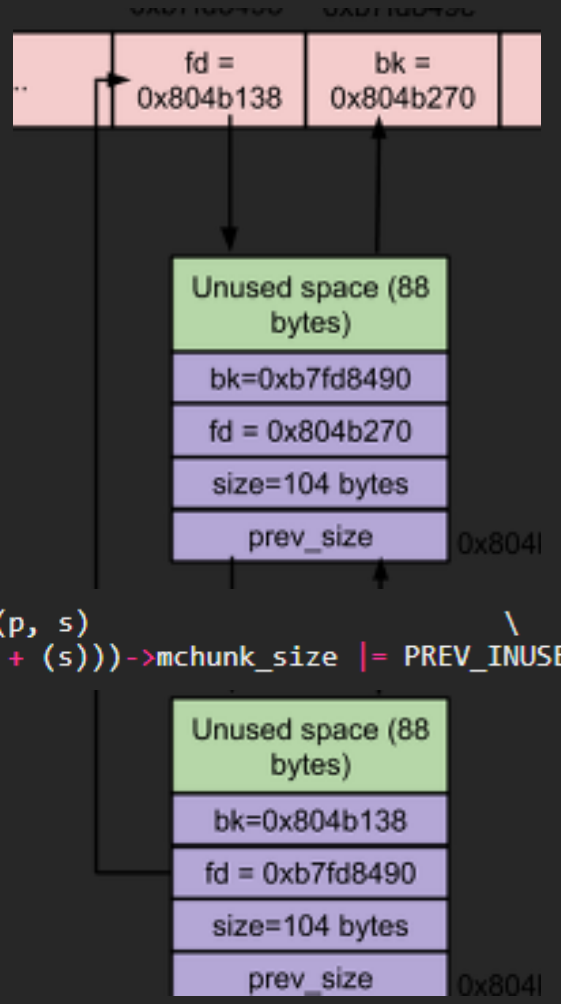
_int_malloc

```c
if (in_smallbin_range (nb))
    {
      idx = smallbin_index (nb);
      bin = bin_at (av, idx);

      if ((victim = last (bin)) != bin)
        {
          if (victim == 0) /* initialization check */
            malloc_consolidate (av);
          else
            {
              bck = victim->bk;
    if (__glibc_unlikely (bck->fd != victim))
                  {
                    errstr = "malloc(): smallbin double linked list corrupted";
                    goto errout;
                  }
                set_inuse_bit_at_offset (victim, nb);
                bin->bk = bck;
                bck->fd = bin;

                if (av != &main_arena)
          set_non_main_arena (victim);
                check_malloced_chunk (av, victim, nb);
                void *p = chunk2mem (victim);
                alloc_perturb (p, bytes);
                return p;
            }
        }
```

```c
#define in_smallbin_range(sz)  \
  ((unsigned long) (sz) < (unsigned long) MIN_LARGE_SIZE)
```

```c
#define last(b)      ((b)->bk)
```

```c
#define set_inuse_bit_at_offset(p, s)                                 \
  (((mchunkptr) (((char *) (p)) + (s)))->mchunk_size |= PREV_INUSE)
```



_int_malloc

```
1  else
2      {
3        idx = largebin_index (nb);
4        if (have_fastchunks (av))
5          malloc_consolidate (av);
6      }
```

끝 & 질문?