2021. 7. 13

Heap

SCP 이예준

Contents

- flag
- bins
- Fast bin
- Unsorted bin

flag

prev_size (if prev_chunk free)	size A M P
fd	bk
fd_nextsize	bk_nextsize
data	

NON_MAIN_ARENA(A) – 0x100
- 현재 청크가 thread_arena에 위치하는 경우 -> 1로 세팅

IS_MMAPPED(M) – 0x10
- 현재 청크가 mmap을 통해 할당된 경우 -> 1로 세팅
- 큰 메모리 요청할 경우 mmap함수를 사용하고 bin 내에 속하지 않음.
- munmap함수로 호출 해제

PREV_INUSE(P) – 0x1
- 현재 청크 바로 이전 청크가 할당되어 있는 경우 -> 1로 세팅

Bins

Bin이라는 구조를 이용하여 해제된 청크를 관리 -> 메모리를 효율적으로 사용

Fast bin

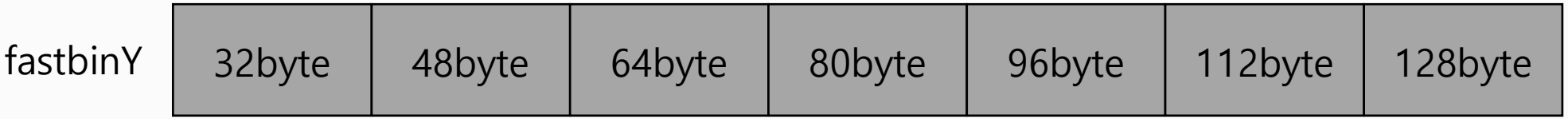
Unsorted bin

Small bin

Large bin

Bins - Fast bin

- LIFO 방식
- 청크의 크기가 16~80byte(32bit), 32~128byte(64bit)인 경우
- 메모리 할당과 해제가 가장 빠르다.
- bin의 개수는 10개(64bit default : 7)이고, 단일 연결리스트로 연결된다.
- 해제된 2개의 청크가 서로 인접해 있을 수 있고, 결합되지 않는다.



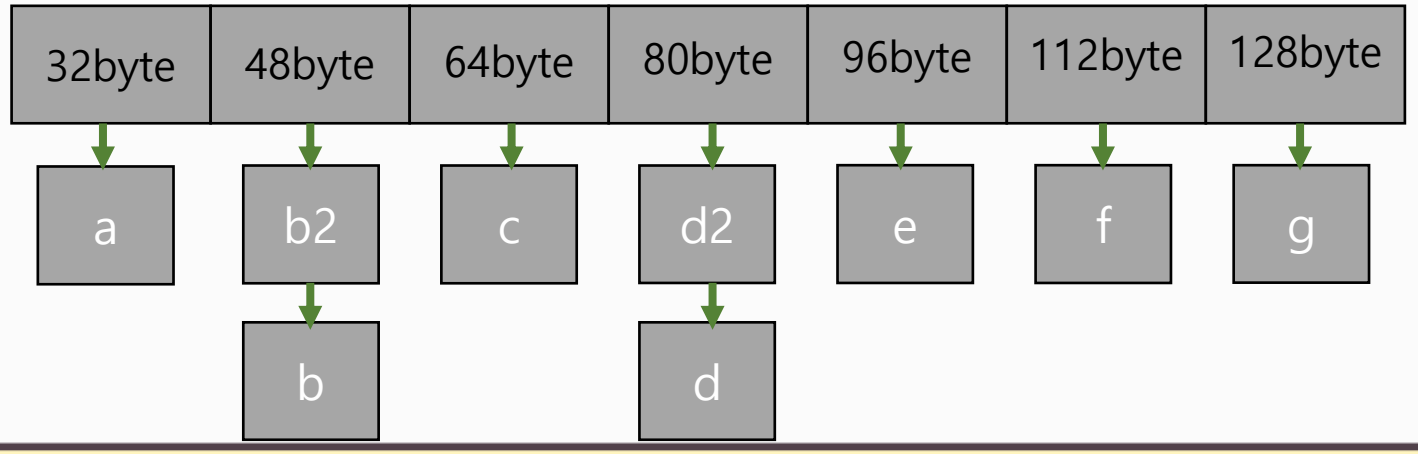
Bins - Fast bin

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  int main(){
5      char* a=(char*)malloc(0x10);
6      char* b=(char*)malloc(0x20);
7      char* b2=(char*)malloc(0x25);
8      char* c=(char*)malloc(0x30);
9      char* d=(char*)malloc(0x40);
10     char* d2=(char*)malloc(0x45);
11     char* e=(char*)malloc(0x50);
12     char* f=(char*)malloc(0x60);
13     char* g=(char*)malloc(0x70);
14     free(a);
15     free(b);
16     free(b2);
17     free(c);
18     free(d);
19     free(d2);
20     free(e);
21     free(f);
22     free(g);
23     return 0;
24 }
```

break point

0x25 + 0x10(청크 헤더)
= 0x35 -> 0x40 할당?

```
gdb-peda$ heapinfo
(0x20)     fastbin[0]: 0x602000 --> 0x0
(0x30)     fastbin[1]: 0x602050 --> 0x602020 --> 0x0
(0x40)     fastbin[2]: 0x602080 --> 0x0
(0x50)     fastbin[3]: 0x602110 --> 0x6020c0 --> 0x0
(0x60)     fastbin[4]: 0x602160 --> 0x0
(0x70)     fastbin[5]: 0x6021c0 --> 0x0
(0x80)     fastbin[6]: 0x602230 --> 0x0
(0x90)     fastbin[7]: 0x0
(0xa0)     fastbin[8]: 0x0
(0xb0)     fastbin[9]: 0x0
                  top: 0x6022b0 (size : 0x20d50)
       last_remainder: 0x0 (size : 0x0)
            unsortbin: 0x0
```



Bins - Fast bin

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main(){
    char* a=(char*)malloc(0x10);
    char* b=(char*)malloc(0x20);
    char* b2=(char*)malloc(0x28);
    char* c=(char*)malloc(0x30);
    char* d=(char*)malloc(0x40);
    char* d2=(char*)malloc(0x45);
    char* e=(char*)malloc(0x50);
    char* f=(char*)malloc(0x60);
    char* g=(char*)malloc(0x70);
```

```
gdb-peda$ heapinfo
(0x20)     fastbin[0]: 0x602000 --> 0x0
(0x30)     fastbin[1]: 0x602050 --> 0x602020 --> 0x0
(0x40)     fastbin[2]: 0x602080 --> 0x0
(0x50)     fastbin[3]: 0x602110 --> 0x6020c0 --> 0x0
(0x60)     fastbin[4]: 0x602160 --> 0x0
(0x70)     fastbin[5]: 0x6021c0 --> 0x0
(0x80)     fastbin[6]: 0x602230 --> 0x0
(0x90)     fastbin[7]: 0x0
(0xa0)     fastbin[8]: 0x0
(0xb0)     fastbin[9]: 0x0
                top: 0x6022b0 (size : 0x20d50)
      last_remainder: 0x0 (size : 0x0)
           unsortbin: 0x0
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main(){
    char* a=(char*)malloc(0x10);
    char* b=(char*)malloc(0x20);
    char* b2=(char*)malloc(0x29);
    char* c=(char*)malloc(0x30);
    char* d=(char*)malloc(0x40);
    char* d2=(char*)malloc(0x45);
    char* e=(char*)malloc(0x50);
    char* f=(char*)malloc(0x60);
    char* g=(char*)malloc(0x70);
```

```
gdb-peda$ heapinfo
(0x20)     fastbin[0]: 0x602000 --> 0x0
(0x30)     fastbin[1]: 0x602020 --> 0x0
(0x40)     fastbin[2]: 0x602090 --> 0x602050 --> 0x0
(0x50)     fastbin[3]: 0x602120 --> 0x6020d0 --> 0x0
(0x60)     fastbin[4]: 0x602170 --> 0x0
(0x70)     fastbin[5]: 0x6021d0 --> 0x0
(0x80)     fastbin[6]: 0x602240 --> 0x0
(0x90)     fastbin[7]: 0x0
(0xa0)     fastbin[8]: 0x0
(0xb0)     fastbin[9]: 0x0
                top: 0x6022c0 (size : 0x20d40)
      last_remainder: 0x0 (size : 0x0)
           unsortbin: 0x0
```

Bins - Fast bin

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main(){
    char* a=(char*)malloc(0x10);
    char* b=(char*)malloc(0x20);
    char* b2=(char*)malloc(0x28);
    char* c=(char*)malloc(0x30);
    char* d=(char*)malloc(0x40);
    char* d2=(char*)malloc(0x45);
    char* e=(char*)malloc(0x50);
    char* f=(char*)malloc(0x60);
    char* g=(char*)malloc(0x70);
    strcpy(b2,"123456789012345678901234567890123456781");

    free(a);
    free(b);
    free(b2);
    free(c);
    free(d);
    free(d2);
    free(e);
    free(f);
    free(g);
    return 0;
}
```

38byte

break pointer

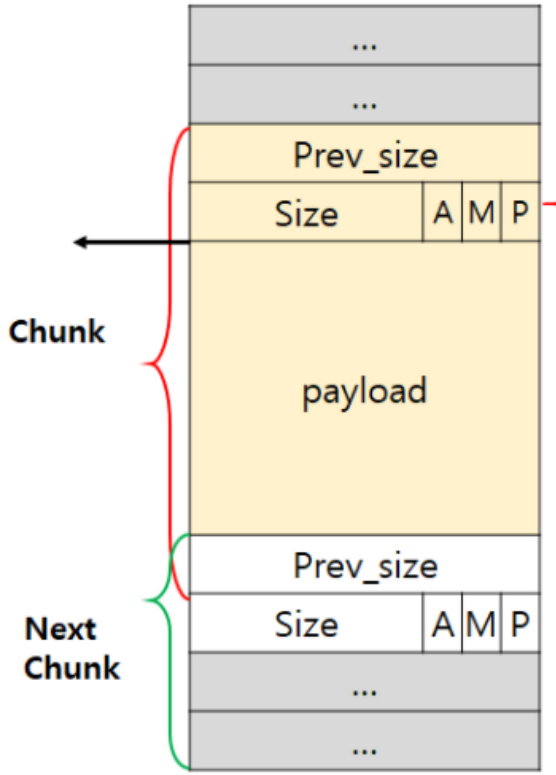
```
Breakpoint 1, 0x000000000040063f in main ()
gdb-peda$ x/32gx 0x602000
0x602000:       0x0000000000000000      0x0000000000000021
0x602010:       0x0000000000000000      0x0000000000000000
0x602020:       0x0000000000000000      0x0000000000000031
0x602030:       0x0000000000000000      0x0000000000000000
0x602040:       0x0000000000000000      0x0000000000000000
0x602050:       0x0000000000000000      0x0000000000000031
0x602060:       0x3837363534333231      0x3635343332313039
0x602070:       0x3433323130393837      0x3231303938373635
0x602080:       0x0000383736353433      0x0000000000000041
0x602090:       0x0000000000000000      0x0000000000000000
0x6020a0:       0x0000000000000000      0x0000000000000000
0x6020b0:       0x0000000000000000      0x0000000000000000
0x6020c0:       0x0000000000000000      0x0000000000000051
0x6020d0:       0x0000000000000000      0x0000000000000000
0x6020e0:       0x0000000000000000      0x0000000000000000
0x6020f0:       0x0000000000000000      0x0000000000000000
```

Bins - Fast bin

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main(){
    char* a=(char*)malloc(0x10);
    char* b=(char*)malloc(0x20);
    char* b2=(char*)malloc(0x28);
    char* c=(char*)malloc(0x30);
    char* d=(char*)malloc(0x40);
    char* d2=(char*)malloc(0x45);
    char* e=(char*)malloc(0x50);
    char* f=(char*)malloc(0x60);
    char* g=(char*)malloc(0x70);
    strcpy(b2,"123456789012345678901234567890123456789012345678");

    free(a);
    free(b);
    free(b2);
    free(c);
    free(d);
    free(d2);
    free(e);
    free(f);
    free(g);
    return 0;
}
```

38byte

break pointer



0x28+0x10
= 0x38

0x38
= 0x30+0x8(NextChunk->Prev_size)

Bins - Fast bin

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main(){
    char* a=(char*)malloc(0x10);
    char* b=(char*)malloc(0x20);
    char* b2=(char*)malloc(0x25);
    char* c=(char*)malloc(0x30);
    char* d=(char*)malloc(0x40);
    char* d2=(char*)malloc(0x45);
    char* e=(char*)malloc(0x50);
    char* f=(char*)malloc(0x60);
    char* g=(char*)malloc(0x70);
    strcpy(b2,"123456789012345678901234567890123456789012345678");
    free(a);
    free(b);
    free(b2);
    free(c);
    free(d);
    free(d2);
    free(e);
    free(f);
    free(g);

    char* re_b = (char*) malloc(0x20);
    char* re_g = (char*) malloc(0x70);
    strcpy(re_b,"AAAAAAAA");
    strcpy(re_g,"BBBBBBBB");
    free(re_b);
    free(re_g);

    return 0;
}
```

break pointer

```
gdb-peda$ heapinfo
(0x20)     fastbin[0]: 0x602000 --> 0x0
(0x30)     fastbin[1]: 0x602050 --> 0x602020 --> 0x0
(0x40)     fastbin[2]: 0x602080 --> 0x0
(0x50)     fastbin[3]: 0x602110 --> 0x6020c0 --> 0x0
(0x60)     fastbin[4]: 0x602160 --> 0x0
(0x70)     fastbin[5]: 0x6021c0 --> 0x0
(0x80)     fastbin[6]: 0x602230 --> 0x0
(0x90)     fastbin[7]: 0x0
(0xa0)     fastbin[8]: 0x0
(0xb0)     fastbin[9]: 0x0
            top: 0x6022b0 (size : 0x20d50)
      last_remainder: 0x0 (size : 0x0)
           unsortbin: 0x0
```

```
gdb-peda$ heapinfo
(0x20)     fastbin[0]: 0x602000 --> 0x0
(0x30)     fastbin[1]: 0x602020 --> 0x0
(0x40)     fastbin[2]: 0x602080 --> 0x0
(0x50)     fastbin[3]: 0x602110 --> 0x6020c0 --> 0x0
(0x60)     fastbin[4]: 0x602160 --> 0x0
(0x70)     fastbin[5]: 0x6021c0 --> 0x0
(0x80)     fastbin[6]: 0x0
(0x90)     fastbin[7]: 0x0
(0xa0)     fastbin[8]: 0x0
(0xb0)     fastbin[9]: 0x0
            top: 0x6022b0 (size : 0x20d50)
      last_remainder: 0x0 (size : 0x0)
           unsortbin: 0x0
```

```
gdb-peda$ x/80gx 0x602000
0x602000:       0x0000000000000000      0x0000000000000021
0x602010:       0x0000000000000000      0x0000000000000000
0x602020:       0x0000000000000000      0x0000000000000031
0x602030:       0x0000000000000000      0x0000000000000000
0x602040:       0x0000000000000000      0x0000000000000000
0x602050:       0x0000000000000000      0x0000000000000031
0x602060:       0x4141414141414141      0x3635343332313000
0x602070:       0x4342333133323037      0x3231303938373635
0x602080:       0x0000383736353433      0x0000000000000041
0x602090:       0x0000000000000000      0x0000000000000000
0x6020a0:       0x0000000000000000      0x0000000000000000
0x6020b0:       0x0000000000000000      0x0000000000000000
0x6020c0:       0x0000000000000000      0x0000000000000051
0x6020d0:       0x0000000000000000      0x0000000000000000
0x6020e0:       0x0000000000000000      0x0000000000000000
0x6020f0:       0x0000000000000000      0x0000000000000000
0x602100:       0x0000000000000000      0x0000000000000000
0x602110:       0x0000000000000000      0x0000000000000051
0x602120:       0x00000000006020c0      0x0000000000000000
0x602130:       0x0000000000000000      0x0000000000000000
0x602140:       0x0000000000000000      0x0000000000000000
0x602150:       0x0000000000000000      0x0000000000000000
0x602160:       0x0000000000000000      0x0000000000000061
0x602170:       0x0000000000000000      0x0000000000000000
0x602180:       0x0000000000000000      0x0000000000000000
0x602190:       0x0000000000000000      0x0000000000000000
0x6021a0:       0x0000000000000000      0x0000000000000000
0x6021b0:       0x0000000000000000      0x0000000000000000
0x6021c0:       0x0000000000000000      0x0000000000000071
0x6021d0:       0x0000000000000000      0x0000000000000000
0x6021e0:       0x0000000000000000      0x0000000000000000
0x6021f0:       0x0000000000000000      0x0000000000000000
0x602200:       0x0000000000000000      0x0000000000000000
0x602210:       0x0000000000000000      0x0000000000000000
0x602220:       0x0000000000000000      0x0000000000000000
0x602230:       0x0000000000000000      0x0000000000000081
0x602240:       0x4242424242424242      0x0000000000000000
0x602250:       0x0000000000000000      0x0000000000000000
0x602260:       0x0000000000000000      0x0000000000000000
0x602270:       0x0000000000000000      0x0000000000000000
```

Bins - Unsorted bin

- FIFO 방식
- 1개의 bin만 사용
- 이중 연결리스트로 구성
- small, large chunk를 보관
- 할당과 해제의 처리속도가 빠름
- 크기에 대한 제한이 없어서 다양한 크기의 청크가 저장됨
- 검색된 청크는 바로 재할당되거나 실패하면 원래의 bin으로 돌아감
- NON_MAIN_ARENA 플래그 세팅되지 않음

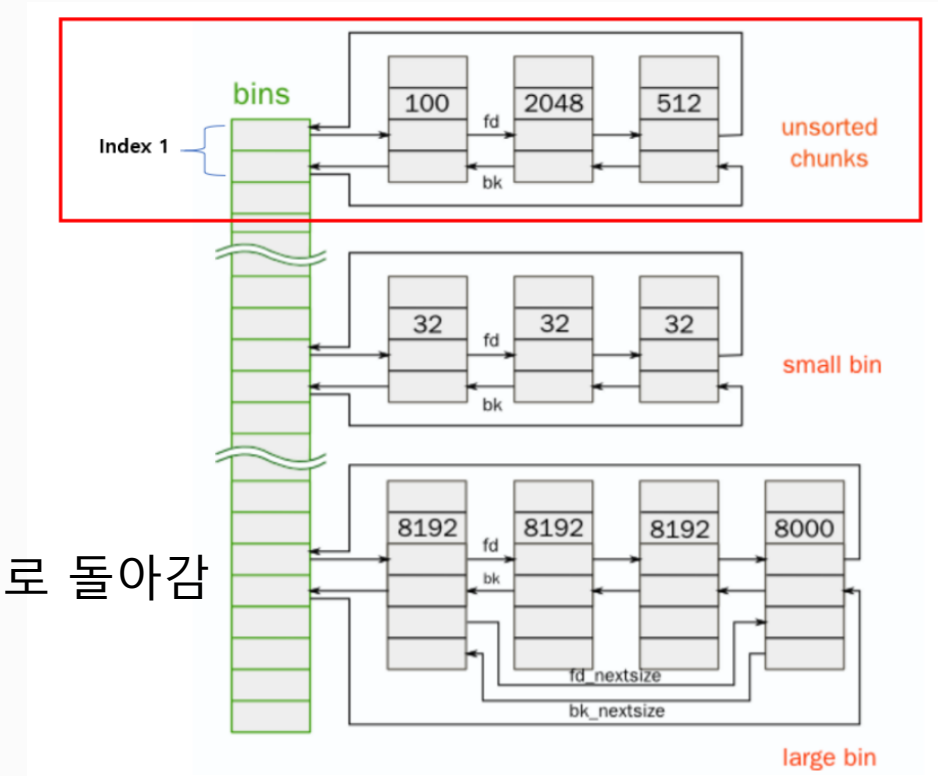


사진 출처 : http://studyfoss.egloos.com/5206220

Bins - Unsorted bin

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{
        char* a=(char*)malloc(0x100);
        char* b=(char*)malloc(0x122);
        char* c=(char*)malloc(0x144);

        free(a);
        free(b);
        free(c);

        char* f=(char*)malloc(0x100);
        char* g=(char*)malloc(0x122);

        free(g);

        return 0;
}
```

break pointer

```
gdb-peda$ heapinfo
(0x20)     fastbin[0]: 0x0
(0x30)     fastbin[1]: 0x0
(0x40)     fastbin[2]: 0x0
(0x50)     fastbin[3]: 0x0
(0x60)     fastbin[4]: 0x0
(0x70)     fastbin[5]: 0x0
(0x80)     fastbin[6]: 0x0
(0x90)     fastbin[7]: 0x0
(0xa0)     fastbin[8]: 0x0
(0xb0)     fastbin[9]: 0x0
                  top: 0x602390 (size : 0x20c70)
       last_remainder: 0x0 (size : 0x0)
            unsortbin: 0x602000 (size : 0x110)
```

Bins - Unsorted bin

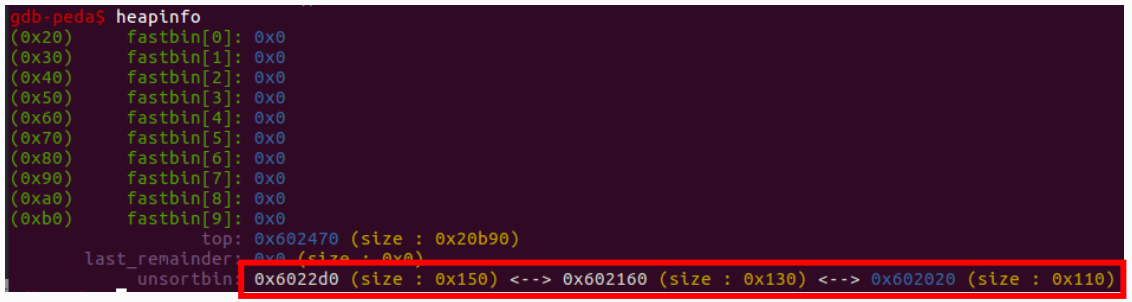
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{
        char* a=(char*)malloc(0x100);
        char* b=(char*)malloc(0x122);
        char* c=(char*)malloc(0x144);

        free(a);
        free(b);
        free(c);

        char* f=(char*)malloc(0x100);
        char* g=(char*)malloc(0x122);

        free(g);

        return 0;
}
```

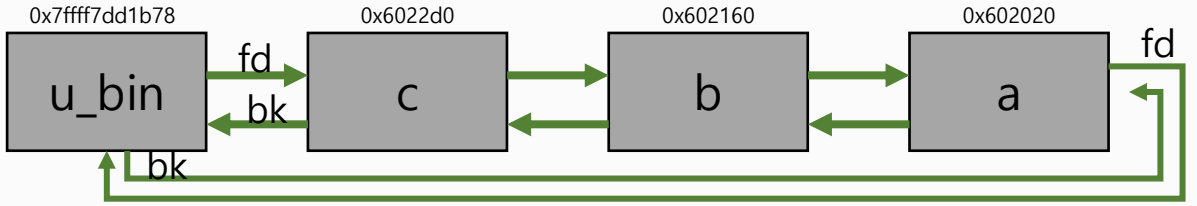
break pointer

```
gdb-peda$ heapinfo
(0x20)     fastbin[0]: 0x0
(0x30)     fastbin[1]: 0x0
(0x40)     fastbin[2]: 0x0
(0x50)     fastbin[3]: 0x0
(0x60)     fastbin[4]: 0x0
(0x70)     fastbin[5]: 0x0
(0x80)     fastbin[6]: 0x0
(0x90)     fastbin[7]: 0x0
(0xa0)     fastbin[8]: 0x0
(0xb0)     fastbin[9]: 0x0
                  top: 0x602390 (size : 0x20c70)
        last_remainder: 0x0 (size : 0x0)
             unsortbin: 0x602000 (size : 0x240)
```

Bins - Unsorted bin

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{
        char* fast1=(char*)malloc(0x10);
        char* a=(char*)malloc(0x100);
        char* fast2=(char*)malloc(0x20);
        char* b=(char*)malloc(0x122);
        char* fast3=(char*)malloc(0x30);
        char* c=(char*)malloc(0x144);
        char* fast4=(char*)malloc(0x40);

        free(a);
        free(b);
        free(c);

        char* f=(char*)malloc(0x100);
        char* g=(char*)malloc(0x122);

        free(g);

        return 0;
}
```





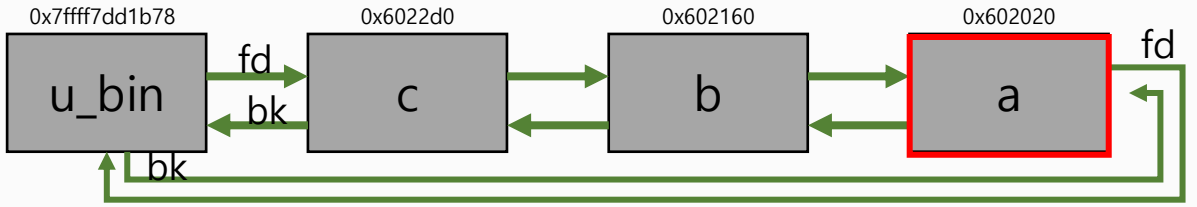

```
gdb-peda$ x/6gx 0x602020
0x602020:       0x0000000000000000      0x0000000000000111
0x602030:       0x00007ffff7dd1b78      0x0000000000602160
0x602040:       0x0000000000000000      0x0000000000000000
gdb-peda$ x/6gx 0x6022d0
0x6022d0:       0x0000000000000000      0x0000000000000151
0x6022e0:       0x0000000000602160      0x00007ffff7dd1b78
0x6022f0:       0x0000000000000000      0x0000000000000000
```

Bins - Unsorted bin

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{
        char* fast1=(char*)malloc(0x10);
        char* a=(char*)malloc(0x100);
        char* fast2=(char*)malloc(0x20);
        char* b=(char*)malloc(0x122);
        char* fast3=(char*)malloc(0x30);
        char* c=(char*)malloc(0x144);
        char* fast4=(char*)malloc(0x40);

        free(a);
        free(b);
        free(c);

        char* f=(char*)malloc(0x100);
        char* g=(char*)malloc(0x122);

        free(g);

        return 0;
}
```

break pointer

```
gdb-peda$ heapinfo
(0x20)     fastbin[0]: 0x0
(0x30)     fastbin[1]: 0x0
(0x40)     fastbin[2]: 0x0
(0x50)     fastbin[3]: 0x0
(0x60)     fastbin[4]: 0x0
(0x70)     fastbin[5]: 0x0
(0x80)     fastbin[6]: 0x0
(0x90)     fastbin[7]: 0x0
(0xa0)     fastbin[8]: 0x0
(0xb0)     fastbin[9]: 0x0
                top: 0x602470 (size : 0x20b90)
      last_remainder: 0x0 (size : 0x0)
           unsortbin: 0x6022d0 (size : 0x150) <--> 0x602160 (size : 0x130)
```

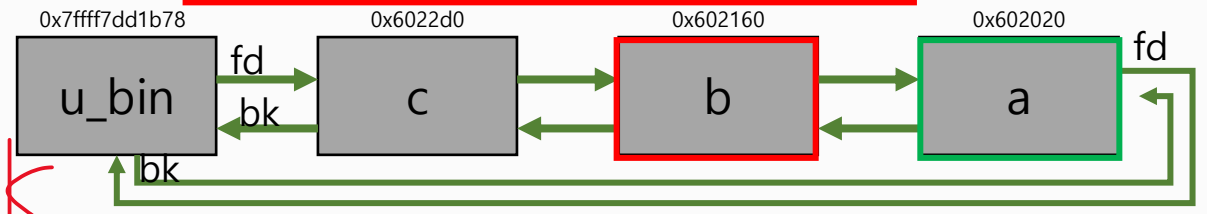


Bins - Unsorted bin

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{
        char* fast1=(char*)malloc(0x10);
        char* a=(char*)malloc(0x100);
        char* fast2=(char*)malloc(0x20);
        char* b=(char*)malloc(0x122);
        char* fast3=(char*)malloc(0x30);
        char* c=(char*)malloc(0x144);
        char* fast4=(char*)malloc(0x40);

        free(a);
        free(b);
        free(c);

        char* f=(char*)malloc(0x122);
        char* g=(char*)malloc(0x100);

        free(g);

        return 0;
}
```

break pointer

```
gdb-peda$ heapinfo
(0x20)     fastbin[0]: 0x0
(0x30)     fastbin[1]: 0x0
(0x40)     fastbin[2]: 0x0
(0x50)     fastbin[3]: 0x0
(0x60)     fastbin[4]: 0x0
(0x70)     fastbin[5]: 0x0
(0x80)     fastbin[6]: 0x0
(0x90)     fastbin[7]: 0x0
(0xa0)     fastbin[8]: 0x0
(0xb0)     fastbin[9]: 0x0
                  top: 0x602470 (size : 0x20b90)
        last remainder: 0x0 (size : 0x0)
               unsortbin: 0x6022d0 (size : 0x150)
(0x110)  smallbin[15]: 0x602020
```



청크 검색은 u_bin에서부터 bk를 타고 검색하는가 보다.

Bins

Bin이라는 구조를 이용하여 해제된 청크를 관리 -> 메모리를 효율적으로 사용

Fast bin

Unsorted bin

Small bin

Large bin