how2heap

2019-01-31 서동훈

- First_fit
- Fastbin_dup
- Fastbin_dup_into_stack
- Fastbin_dup_consolidate

GNU_LIBC_VERSION 2.11.2

```
(gdb) x/200wx 0x0804a000
0x804a000:    0x00000000    0x00000209    0x00000000    0x00000000
0x804a010:    0x00000000    0x00000000    0x00000000    0x00000000
0x804a020:    0x00000000    0x00000000    0x00000000    0x00000000
0x804a030:    0x00000000    0x00000000    0x00000000    0x00000000
0x804a040:    0x00000000    0x00000000    0x00000000    0x00000000
0x804a050:    0x00000000    0x00000000    0x00000000    0x00000000
0x804a060:    0x00000000    0x00000000    0x00000000    0x00000000
```


```
root@protostar:/opt/how2heap# ./firtst_fit_2
1st malloc(512): 0x804a008
2nd malloc(256): 0x804a210
first allocation 0x804a008 points to this is A!
3rd malloc(500): 0x804a008
3rd allocation 0x804a008 points to this is C!
first allocation 0x804a008 points to this is C!
```

first_fit.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main()
{
    char* a = malloc(512);
    char* b = malloc(256);
    char* c;

    fprintf(stderr, "1st malloc(512): %p\n", a);
    fprintf(stderr, "2nd malloc(256): %p\n", b);

    strcpy(a, "this is A!");
    fprintf(stderr, "first allocation %p points to %s\n", a, a);

    free(a);


    c = malloc(500);
    fprintf(stderr, "3rd malloc(500): %p\n", c);

    strcpy(c, "this is C!");
    fprintf(stderr, "3rd allocation %p points to %s\n", c, c);
    fprintf(stderr, "first allocation %p points to %s\n", a, a);

}
```

```
(gdb) x/200wx 0x0804a000
0x804a000:    0x00000000    0x00000209    0x73696874    0x20736920
0x804a010:    0x00002141    0x00000000    0x00000000    0x00000000
0x804a020:    0x00000000    0x00000000    0x00000000    0x00000000
0x804a030:    0x00000000    0x00000000    0x00000000    0x00000000
0x804a040:    0x00000000    0x00000000    0x00000000    0x00000000
0x804a050:    0x00000000    0x00000000    0x00000000    0x00000000
0x804a060:    0x00000000    0x00000000    0x00000000    0x00000000
```

first_fit.c

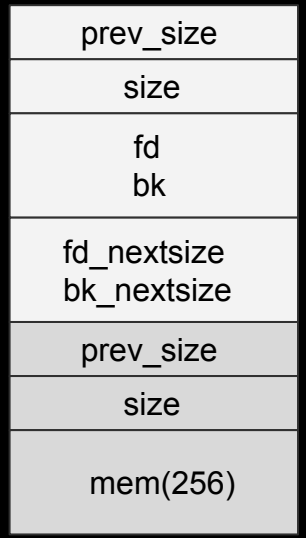
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main()
{
    char* a = malloc(512);
    char* b = malloc(256);
    char* c;

    fprintf(stderr, "1st malloc(512): %p\n", a);
    fprintf(stderr, "2nd malloc(256): %p\n", b);

    strcpy(a, "this is A!");
    fprintf(stderr, "first allocation %p points to %s\n", a, a);

    free(a);


    c = malloc(500);
    fprintf(stderr, "3rd malloc(500): %p\n", c);

    strcpy(c, "this is C!");
    fprintf(stderr, "3rd allocation %p points to %s\n", c, c);
    fprintf(stderr, "first allocation %p points to %s\n", a, a);

}
```

```
(gdb) x/200wx 0x0804a000
0x804a000:    0x00000000    0x00000209    0xb7fd93d0    0xb7fd93d0
0x804a010:    0x00000000    0x00000000    0x00000000    0x00000000
0x804a020:    0x00000000    0x00000000    0x00000000    0x00000000
0x804a030:    0x00000000    0x00000000    0x00000000    0x00000000
0x804a040:    0x00000000    0x00000000    0x00000000    0x00000000
0x804a050:    0x00000000    0x00000000    0x00000000    0x00000000
0x804a060:    0x00000000    0x00000000    0x00000000    0x00000000
```

first_fit.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main()
{
    char* a = malloc(512);
    char* b = malloc(256);
    char* c;

    fprintf(stderr, "1st malloc(512): %p\n", a);
    fprintf(stderr, "2nd malloc(256): %p\n", b);

    strcpy(a, "this is A!");
    fprintf(stderr, "first allocation %p points to %s\n", a, a);

    free(a);


    c = malloc(500);
    fprintf(stderr, "3rd malloc(500): %p\n", c);

    strcpy(c, "this is C!");
    fprintf(stderr, "3rd allocation %p points to %s\n", c, c);
    fprintf(stderr, "first allocation %p points to %s\n", a, a);

}
```

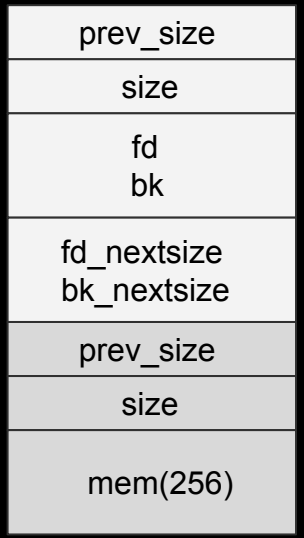
```
(gdb) x/200wx 0x0804a000
0x804a000:    0x00000000    0x000001f9    0xb7fd95c8    0xb7fd95c8
0x804a010:    0x0804a000    0x0804a000    0x00000000    0x00000000
0x804a020:    0x00000000    0x00000000    0x00000000    0x00000000
0x804a030:    0x00000000    0x00000000    0x00000000    0x00000000
0x804a040:    0x00000000    0x00000000    0x00000000    0x00000000
0x804a050:    0x00000000    0x00000000    0x00000000    0x00000000
0x804a060:    0x00000000    0x00000000    0x00000000    0x00000000
```



first_fit.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main()
{
    char* a = malloc(512);
    char* b = malloc(256);
    char* c;

    fprintf(stderr, "1st malloc(512): %p\n", a);
    fprintf(stderr, "2nd malloc(256): %p\n", b);

    strcpy(a, "this is A!");
    fprintf(stderr, "first allocation %p points to %s\n", a, a);

    free(a);


    c = malloc(500);
    fprintf(stderr, "3rd malloc(500): %p\n", c);

    strcpy(c, "this is C!");
    fprintf(stderr, "3rd allocation %p points to %s\n", c, c);
    fprintf(stderr, "first allocation %p points to %s\n", a, a);

}
```

```
(gdb) x/32wx 0x0804a000
0x804a000:      0x00000000      0x000001f9      0x73696874      0x20736920
0x804a010:      0x08002143      0x0804a000      0x00000000      0x00000000
0x804a020:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a030:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a040:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a050:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a060:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a070:      0x00000000      0x00000000      0x00000000      0x00000000
(gdb)
```



first_fit.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main()
{
    char* a = malloc(512);
    char* b = malloc(256);
    char* c;

    fprintf(stderr, "1st malloc(512): %p\n", a);
    fprintf(stderr, "2nd malloc(256): %p\n", b);

    strcpy(a, "this is A!");
    fprintf(stderr, "first allocation %p points to %s\n", a, a);

    free(a);


    c = malloc(500);
    fprintf(stderr, "3rd malloc(500): %p\n", c);

    strcpy(c, "this is C!");
    fprintf(stderr, "3rd allocation %p points to %s\n", c, c);
    fprintf(stderr, "first allocation %p points to %s\n", a, a);

}
```

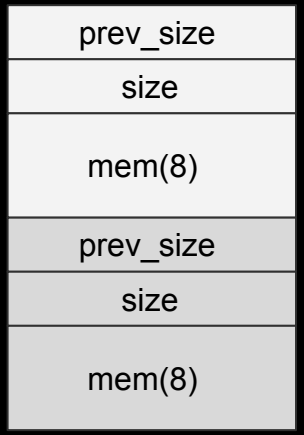
```
(gdb) x/32wx 0x0804a000
0x804a000:      0x00000000      0x000001f9      0x73696874      0x20736920
0x804a010:      0x08002143      0x0804a000      0x00000000      0x00000000
0x804a020:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a030:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a040:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a050:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a060:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a070:      0x00000000      0x00000000      0x00000000      0x00000000
(gdb)
```

```
root@protostar:/opt/how2heap# ./firtst_fit_2
1st malloc(512): 0x804a008
2nd malloc(256): 0x804a210
first allocation 0x804a008 points to this is A!
3rd malloc(500): 0x804a008
3rd allocation 0x804a008 points to this is C!
first allocation 0x804a008 points to this is C!
```

first_fit.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main()
{
    char* a = malloc(512);
    char* b = malloc(256);
    char* c;

    fprintf(stderr, "1st malloc(512): %p\n", a);
    fprintf(stderr, "2nd malloc(256): %p\n", b);

    strcpy(a, "this is A!");
    fprintf(stderr, "first allocation %p points to %s\n", a, a);

    free(a);


    c = malloc(500);
    fprintf(stderr, "3rd malloc(500): %p\n", c);

    strcpy(c, "this is C!");
    fprintf(stderr, "3rd allocation %p points to %s\n", c, c);
    fprintf(stderr, "first allocation %p points to %s\n", a, a);

}
```

```
(gdb) p main_arena.fastbinsY
$9 = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0}
(gdb) x/24wx 0x0804a000
0x804a000:      0x00000000      0x00000011      0x00000000      0x00000000
0x804a010:      0x00000000      0x00000011      0x00000000      0x00000000
0x804a020:      0x00000000      0x00000011      0x00000000      0x00000000
0x804a030:      0x00000000      0x00020fd1      0x00000000      0x00000000
0x804a040:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a050:      0x00000000      0x00000000      0x00000000      0x00000000
(gdb)
```

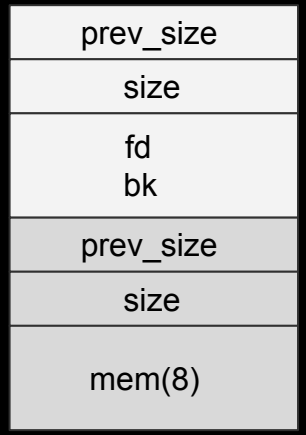


fastbin_dup_into_stack.c

```c
#include<stdio.h>
#include<stdlib.h>

int main()
{
    unsigned long long stack_var;

    int *a = malloc(8);
    int *b = malloc(8);
    int *c = malloc(8);

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

    unsigned long long *d = malloc(8);

    malloc(8);

    stack_var = 0x20;

    *d = (unsigned long long ) (((char *)&stack_var) - sizeof(d));

    malloc(8);

    printf("malloc(8): %p\n",malloc(8));

}
```

```
(gdb) p main_arena.fastbinsY
$10 = {0x804a000, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0}
(gdb) x/24wx 0x0804a000
0x804a000:      0x00000000      0x00000011      0x00000000      0x00000000
0x804a010:      0x00000000      0x00000011      0x00000000      0x00000000
0x804a020:      0x00000000      0x00000011      0x00000000      0x00000000
0x804a030:      0x00000000      0x00020fd1      0x00000000      0x00000000
0x804a040:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a050:      0x00000000      0x00000000      0x00000000      0x00000000
(gdb)
```

fastbin영역에는 최근에 해제된 heap영역이 등록됨.



fastbin_dup_into_stack.c

```c
#include<stdio.h>
#include<stdlib.h>

int main()
{
    unsigned long long stack_var;

    int *a = malloc(8);
    int *b = malloc(8);
    int *c = malloc(8);

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

    unsigned long long *d = malloc(8);

    malloc(8);

    stack_var = 0x20;

    *d = (unsigned long long ) (((char *)&stack_var) - sizeof(d));

    malloc(8);

    printf("malloc(8): %p\n",malloc(8));

}
```

```
(gdb) p main_arena.fastbinsY
$11 = {0x804a010, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0}
(gdb) x/24wx 0x0804a000
0x804a000:      0x00000000      0x00000011      0x00000000      0x00000000
0x804a010:      0x00000000      0x00000011      0x0804a000      0x00000000
0x804a020:      0x00000000      0x00000011      0x00000000      0x00000000
0x804a030:      0x00000000      0x00020fd1      0x00000000      0x00000000
0x804a040:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a050:      0x00000000      0x00000000      0x00000000      0x00000000
(gdb)
```

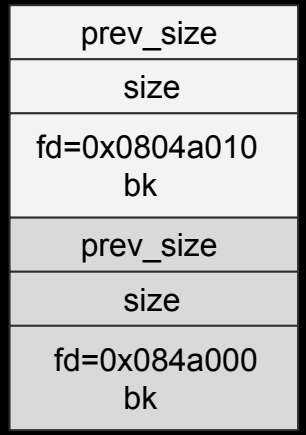
fastbin영역에는 최근에 해제된 heap영역이 등록됨.

fastbin_dup_into_stack.c

```c
#include<stdio.h>
#include<stdlib.h>

int main()
{
    unsigned long long stack_var;

    int *a = malloc(8);
    int *b = malloc(8);
    int *c = malloc(8);

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

    unsigned long long *d = malloc(8);

    malloc(8);

    stack_var = 0x20;

    *d = (unsigned long long ) (((char *)&stack_var) - sizeof(d));

    malloc(8);

    printf("malloc(8): %p\n",malloc(8));

}
```

```
(gdb) p main_arena.fastbinsY
$12 = {0x804a000, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0}
(gdb) x/24wx 0x0804a000
0x804a000:      0x00000000      0x00000011      0x0804a010      0x00000000
0x804a010:      0x00000000      0x00000011      0x0804a000      0x00000000
0x804a020:      0x00000000      0x00000011      0x00000000      0x00000000
0x804a030:      0x00000000      0x00020fd1      0x00000000      0x00000000
0x804a040:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a050:      0x00000000      0x00000000      0x00000000      0x00000000
(gdb)
```

fastbin영역에는 최근에 해제된 heap영역이 등록됨.

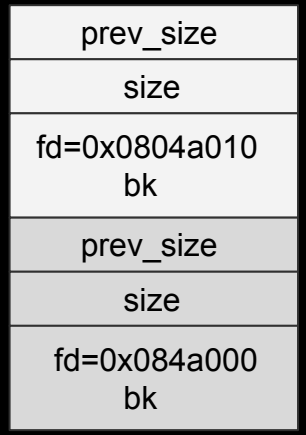


fastbin_dup_into_stack.c

```c
#include<stdio.h>
#include<stdlib.h>

int main()
{
    unsigned long long stack_var;

    int *a = malloc(8);
    int *b = malloc(8);
    int *c = malloc(8);

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

    unsigned long long *d = malloc(8);

    malloc(8);

    stack_var = 0x20;

    *d = (unsigned long long ) (((char *)&stack_var) - sizeof(d));

    malloc(8);

    printf("malloc(8): %p\n",malloc(8));

}
```

```
(gdb) p main_arena.fastbinsY
$13 = {0x804a010, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0}
(gdb) x/24wx 0x0804a000
0x804a000:      0x00000000      0x00000011      0x0804a010      0x00000000
0x804a010:      0x00000000      0x00000011      0x0804a000      0x00000000
0x804a020:      0x00000000      0x00000011      0x00000000      0x00000000
0x804a030:      0x00000000      0x00020fd1      0x00000000      0x00000000
0x804a040:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a050:      0x00000000      0x00000000      0x00000000      0x00000000
(gdb)
```

fastbin영역에는 최근에 해제된 heap영역이 등록됨.

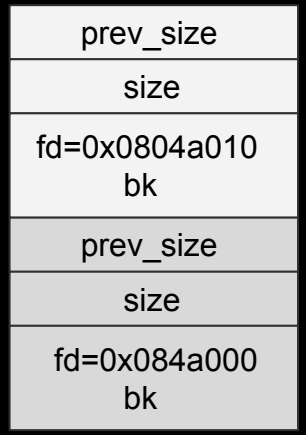


fastbin_dup_into_stack.c

```c
#include<stdio.h>
#include<stdlib.h>

int main()
{
    unsigned long long stack_var;

    int *a = malloc(8);
    int *b = malloc(8);
    int *c = malloc(8);

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

    unsigned long long *d = malloc(8);

    malloc(8);

    stack_var = 0x20;

    *d = (unsigned long long ) (((char *)&stack_var) - sizeof(d));

    malloc(8);

    printf("malloc(8): %p\n",malloc(8));

}
```

```
(gdb) p main_arena.fastbinsY
$12 = {0x804a000, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0}
(gdb) x/24wx 0x0804a000
0x804a000:      0x00000000      0x00000011      0x0804a010      0x00000000
0x804a010:      0x00000000      0x00000011      0x0804a000      0x00000000
0x804a020:      0x00000000      0x00000011      0x00000000      0x00000000
0x804a030:      0x00000000      0x00020fd1      0x00000000      0x00000000
0x804a040:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a050:      0x00000000      0x00000000      0x00000000      0x00000000
(gdb)
```

fastbin영역에는 최근에 해제된 heap영역이 등록됨.



fastbin_dup_into_stack.c

```c
#include<stdio.h>
#include<stdlib.h>

int main()
{
    unsigned long long stack_var;

    int *a = malloc(8);
    int *b = malloc(8);
    int *c = malloc(8);

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

    unsigned long long *d = malloc(8);

    malloc(8);

    stack_var = 0x20;

    *d = (unsigned long long ) (((char *)&stack_var) - sizeof(d));

    malloc(8);

    printf("malloc(8): %p\n",malloc(8));

}
```

```
(gdb) p main_arena.fastbinsY
$14 = {0x804a000, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0}
(gdb) x/24wx 0x0804a000
0x804a000:      0x00000000      0x00000011      0xbffffc84      0xffffffff
0x804a010:      0x00000000      0x00000011      0x0804a000      0x00000000
0x804a020:      0x00000000      0x00000011      0x00000000      0x00000000
0x804a030:      0x00000000      0x00020fd1      0x00000000      0x00000000
0x804a040:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a050:      0x00000000      0x00000000      0x00000000      0x00000000
(gdb)
```

fastbin영역에는 최근에 해제된 heap영역이 등록됨.

prev_size
size
0xbffffc84 0xffffffff
prev_size
size
fd=0x084a000 bk

fastbin_dup_into_stack.c

```c
#include<stdio.h>
#include<stdlib.h>

int main()
{
    unsigned long long stack_var;

    int *a = malloc(8);
    int *b = malloc(8);
    int *c = malloc(8);

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

    unsigned long long *d = malloc(8);

    malloc(8);

    stack_var = 0x20;

    *d = (unsigned long long ) (((char *)&stack_var) - sizeof(d));

    malloc(8);

    printf("malloc(8): %p\n",malloc(8));

}
```

```
(gdb) p main_arena.fastbinsY
$15 = {0xbffffc84, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0}
(gdb) x/24wx 0x0804a000
0x804a000:      0x00000000      0x00000011      0xbffffc84      0xffffffff
0x804a010:      0x00000000      0x00000011      0x0804a000      0x00000000
0x804a020:      0x00000000      0x00000011      0x00000000      0x00000000
0x804a030:      0x00000000      0x00020fd1      0x00000000      0x00000000
0x804a040:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a050:      0x00000000      0x00000000      0x00000000      0x00000000
(gdb)
```

fastbin영역에는 최근에 해제된 heap영역이 등록됨.

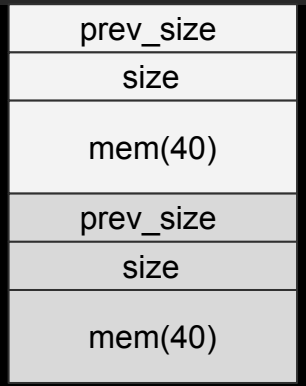
fastbin_dup_into_stack.c

```c
#include<stdio.h>
#include<stdlib.h>

int main()
{
    unsigned long long stack_var;

    int *a = malloc(8);
    int *b = malloc(8);
    int *c = malloc(8);

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

    unsigned long long *d = malloc(8);

    malloc(8);

    stack_var = 0x20;

    *d = (unsigned long long ) (((char *)&stack_var) - sizeof(d));

    malloc(8);

    printf("malloc(8): %p\n",malloc(8));

}
```

fastbin_dup_consolidate.c

```c
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>

int main() {
  void* p1 = malloc(40);
  void* p2 = malloc(40);
  fprintf(stderr, "Allocated two fastbins: p1=%p p2=%p\n", p1, p2);
  fprintf(stderr, "Now free p1!\n");
  free(p1);

  void* p3 = malloc(0x400);
  fprintf(stderr, "Allocated large bin to trigger malloc_consolidate(): p3=%p\n", p3);
  fprintf(stderr, "In malloc_consolidate(), p1 is moved to the unsorted bin.\n");
  free(p1);
  fprintf(stderr, "Trigger the double free vulnerability!\n");
  fprintf(stderr, "We can pass the check in malloc() since p1 is not fast top.\n");
  fprintf(stderr, "Now p1 is in unsorted bin and fast bin. So we'will get it twice: %p %p\n", malloc(40), malloc(40));
}
```

```
(gdb) x/80wx 0x0804a000
0x804a000:      0x00000000      0x00000031      0x00000000      0x00000000
0x804a010:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a020:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a030:      0x00000000      0x00000031      0x00000000      0x00000000
0x804a040:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a050:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a060:      0x00000000      0x00020fa1      0x00000000      0x00000000
0x804a070:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a080:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a090:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0a0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0b0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0c0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0d0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0e0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0f0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a100:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a110:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a120:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a130:      0x00000000      0x00000000      0x00000000      0x00000000
(gdb)
```



fastbin_dup_consolidate.c

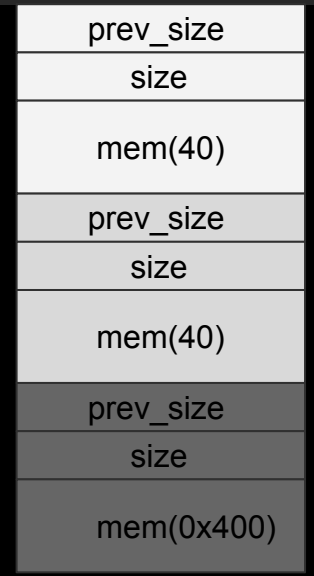
```c
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>

int main() {
  void* p1 = malloc(40);
  void* p2 = malloc(40);
  fprintf(stderr, "Allocated two fastbins: p1=%p p2=%p\n", p1, p2);
  fprintf(stderr, "Now free p1!\n");
  free(p1);

  void* p3 = malloc(0x400);
  fprintf(stderr, "Allocated large bin to trigger malloc_consolidate(): p3=%p\n", p3);
  fprintf(stderr, "In malloc_consolidate(), p1 is moved to the unsorted bin.\n");
  free(p1);
  fprintf(stderr, "Trigger the double free vulnerability!\n");
  fprintf(stderr, "We can pass the check in malloc() since p1 is not fast top.\n");
  fprintf(stderr, "Now p1 is in unsorted bin and fast bin. So we'will get it twice: %p %p\n", malloc(40), malloc(40));
}
```

```
(gdb) p main_arena.fastbinsY
$13 = {0x0, 0x0, 0x0, 0x0, 0x804a000, 0x0, 0x0, 0x0, 0x0, 0x0}
(gdb) x/80wx 0x0804a000
0x804a000:      0x00000000      0x00000031      0x00000000      0x00000000
0x804a010:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a020:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a030:      0x00000000      0x00000031      0x00000000      0x00000000
0x804a040:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a050:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a060:      0x00000000      0x00020fa1      0x00000000      0x00000000
0x804a070:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a080:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a090:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0a0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0b0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0c0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0d0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0e0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0f0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a100:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a110:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a120:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a130:      0x00000000      0x00000000      0x00000000      0x00000000
(gdb)
```

fastbin_dup_consolidate.c

```c
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>

int main() {
  void* p1 = malloc(40);
  void* p2 = malloc(40);
  fprintf(stderr, "Allocated two fastbins: p1=%p p2=%p\n", p1, p2);
  fprintf(stderr, "Now free p1!\n");
  free(p1);

  void* p3 = malloc(0x400);
  fprintf(stderr, "Allocated large bin to trigger malloc_consolidate(): p3=%p\n", p3);
  fprintf(stderr, "In malloc_consolidate(), p1 is moved to the unsorted bin.\n");
  free(p1);
  fprintf(stderr, "Trigger the double free vulnerability!\n");
  fprintf(stderr, "We can pass the check in malloc() since p1 is not fast top.\n");
  fprintf(stderr, "Now p1 is in unsorted bin and fast bin. So we'will get it twice: %p %p\n", malloc(40), malloc(40));
}
```

```
(gdb) p main_arena.fastbinsY
$14 = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0}
(gdb) x/80wx 0x0804a000
0x804a000:      0x00000000      0x00000031      0xb7fd93f8      0xb7fd93f8
0x804a010:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a020:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a030:      0x00000030      0x00000030      0x00000000      0x00000000
0x804a040:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a050:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a060:      0x00000000      0x00000409      0x00000000      0x00000000
0x804a070:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a080:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a090:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0a0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0b0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0c0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0d0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0e0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0f0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a100:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a110:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a120:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a130:      0x00000000      0x00000000      0x00000000      0x00000000
(gdb)
```



```
(gdb) p main_arena.bins
$15 = {0xb7fd93d0, 0xb7fd93d0, 0xb7fd93d8, 0xb7fd93d8, 0xb7fd93e0, 0xb7fd93e0, 0xb7fd93e8, 0xb7fd93e8, 0xb7fd9
3f0, 0xb7fd93f0, 0x804a000, 0x804a000, 0xb7fd9400, 0xb7fd9400, 0xb7fd9408, 0xb7fd9408, 0xb7fd9410,
  0xb7fd9410, 0xb7fd9418, 0xb7fd9418, 0xb7fd9420, 0xb7fd9420, 0xb7fd9428, 0xb7fd9428, 0xb7fd9430, 0xb7fd9430,
0xb7fd9438, 0xb7fd9438, 0xb7fd9440, 0xb7fd9440, 0xb7fd9448, 0xb7fd9448, 0xb7fd9450, 0xb7fd9450,
  0xb7fd9458, 0xb7fd9458, 0xb7fd9460, 0xb7fd9460, 0xb7fd9468, 0xb7fd9468, 0xb7fd9470, 0xb7fd9470, 0xb7fd9478,
0xb7fd9478, 0xb7fd9480, 0xb7fd9480, 0xb7fd9488, 0xb7fd9488, 0xb7fd9490, 0xb7fd9490, 0xb7fd9498,
  0xb7fd9498, 0xb7fd94a0, 0xb7fd94a0, 0xb7fd94a8, 0xb7fd94a8, 0xb7fd94b0, 0xb7fd94b0, 0xb7fd94b8, 0xb7fd94b8,
0xb7fd94c0, 0xb7fd94c0, 0xb7fd94c8, 0xb7fd94c8, 0xb7fd94d0, 0xb7fd94d0, 0xb7fd94d8, 0xb7fd94d8,
  0xb7fd94e0, 0xb7fd94e0, 0xb7fd94e8, 0xb7fd94e8, 0xb7fd94f0, 0xb7fd94f0, 0xb7fd94f8, 0xb7fd94f8, 0xb7fd9500,
0xb7fd9500, 0xb7fd9508, 0xb7fd9508, 0xb7fd9510, 0xb7fd9510, 0xb7fd9518, 0xb7fd9518, 0xb7fd9520,
  0xb7fd9520, 0xb7fd9528, 0xb7fd9528, 0xb7fd9530, 0xb7fd9530, 0xb7fd9538, 0xb7fd9538, 0xb7fd9540, 0xb7fd9540,
0xb7fd9548, 0xb7fd9548, 0xb7fd9550, 0xb7fd9550, 0xb7fd9558, 0xb7fd9558, 0xb7fd9560, 0xb7fd9560,
  0xb7fd9568, 0xb7fd9568, 0xb7fd9570, 0xb7fd9570, 0xb7fd9578, 0xb7fd9578, 0xb7fd9580, 0xb7fd9580, 0xb7fd9588,
0xb7fd9588, 0xb7fd9590, 0xb7fd9590, 0xb7fd9598, 0xb7fd9598, 0xb7fd95a0, 0xb7fd95a0, 0xb7fd95a8,
  0xb7fd95a8, 0xb7fd95b0, 0xb7fd95b0, 0xb7fd95b8, 0xb7fd95b8, 0xb7fd95c0, 0xb7fd95c0, 0xb7fd95c8, 0xb7fd95c8,
0xb7fd95d0, 0xb7fd95d0, 0xb7fd95d8, 0xb7fd95d8, 0xb7fd95e0, 0xb7fd95e0, 0xb7fd95e8, 0xb7fd95e8,
  0xb7fd95f0, 0xb7fd95f0, 0xb7fd95f8, 0xb7fd95f8, 0xb7fd9600, 0xb7fd9600, 0xb7fd9608, 0xb7fd9608, 0xb7fd9610,
0xb7fd9610, 0xb7fd9618, 0xb7fd9618, 0xb7fd9620, 0xb7fd9620, 0xb7fd9628, 0xb7fd9628, 0xb7fd9630,
  0xb7fd9630, 0xb7fd9638, 0xb7fd9638, 0xb7fd9640, 0xb7fd9640, 0xb7fd9648, 0xb7fd9648, 0xb7fd9650, 0xb7fd9650,
0xb7fd9658, 0xb7fd9658, 0xb7fd9660, 0xb7fd9660, 0xb7fd9668, 0xb7fd9668, 0xb7fd9670, 0xb7fd9670,
  0xb7fd9678, 0xb7fd9678, 0xb7fd9680, 0xb7fd9680, 0xb7fd9688, 0xb7fd9688, 0xb7fd9690, 0xb7fd9690, 0xb7fd9698,
0xb7fd9698, 0xb7fd96a0, 0xb7fd96a0, 0xb7fd96a8, 0xb7fd96a8, 0xb7fd96b0, 0xb7fd96b0, 0xb7fd96b8,
  0xb7fd96b8, 0xb7fd96c0, 0xb7fd96c0, 0xb7fd96c8, 0xb7fd96c8, 0xb7fd96d0, 0xb7fd96d0, 0xb7fd96d8, 0xb7fd96d8,
0xb7fd96e0, 0xb7fd96e0, 0xb7fd96e8, 0xb7fd96e8...}
(gdb)
```

fastbin_dup_consolidate.c

```c
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>

int main() {
  void* p1 = malloc(40);
  void* p2 = malloc(40);
  fprintf(stderr, "Allocated two fastbins: p1=%p p2=%p\n", p1, p2);
  fprintf(stderr, "Now free p1!\n");
  free(p1);

  void* p3 = malloc(0x400);
  fprintf(stderr, "Allocated large bin to trigger malloc_consolidate(): p3=%p\n", p3);
  fprintf(stderr, "In malloc_consolidate(), p1 is moved to the unsorted bin.\n");
  free(p1);
  fprintf(stderr, "Trigger the double free vulnerability!\n");
  fprintf(stderr, "We can pass the check in malloc() since p1 is not fast top.\n");
  fprintf(stderr, "Now p1 is in unsorted bin and fast bin. So we'will get it twice: %p %p\n", malloc(40), malloc(40));
}
```

```
(gdb) p main_arena.fastbinsY
$14 = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0}
(gdb) x/80wx 0x0804a000
0x804a000:      0x00000000      0x00000031      0xb7fd93f8      0xb7fd93f8
0x804a010:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a020:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a030:      0x00000030      0x00000030      0x00000000      0x00000000
0x804a040:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a050:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a060:      0x00000000      0x00000409      0x00000000      0x00000000
0x804a070:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a080:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a090:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0a0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0b0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0c0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0d0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0e0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0f0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a100:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a110:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a120:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a130:      0x00000000      0x00000000      0x00000000      0x00000000
(gdb)
```

fastbin_dup_consolidate.c

```c
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>

int main() {
  void* p1 = malloc(40);
  void* p2 = malloc(40);
  fprintf(stderr, "Allocated two fastbins: p1=%p p2=%p\n", p1, p2);
  fprintf(stderr, "Now free p1!\n");
  free(p1);

  void* p3 = malloc(0x400);
  fprintf(stderr, "Allocated large bin to trigger malloc_consolidate(): p3=%p\n", p3);
  fprintf(stderr, "In malloc_consolidate(), p1 is moved to the unsorted bin.\n");
  free(p1);
  fprintf(stderr, "Trigger the double free vulnerability!\n");
  fprintf(stderr, "We can pass the check in malloc() since p1 is not fast top.\n");
  fprintf(stderr, "Now p1 is in unsorted bin and fast bin. So we'will get it twice: %p %p\n", malloc(40), malloc(40));
}
```

```
(gdb) p main_arena.fastbinsY
$17 = {0x0, 0x0, 0x0, 0x0, 0x804a000, 0x0, 0x0, 0x0, 0x0, 0x0}
(gdb) x/80wx 0x0804a000
0x804a000:      0x00000000      0x00000031      0x00000000      0xb7fd93f8
0x804a010:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a020:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a030:      0x00000030      0x00000030      0x00000000      0x00000000
0x804a040:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a050:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a060:      0x00000000      0x00000409      0x00000000      0x00000000
0x804a070:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a080:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a090:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0a0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0b0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0c0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0d0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0e0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0f0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a100:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a110:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a120:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a130:      0x00000000      0x00000000      0x00000000      0x00000000
(gdb)
```

fastbin_dup_consolidate.c

```c
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>

int main() {
  void* p1 = malloc(40);
  void* p2 = malloc(40);
  fprintf(stderr, "Allocated two fastbins: p1=%p p2=%p\n", p1, p2);
  fprintf(stderr, "Now free p1!\n");
  free(p1);

  void* p3 = malloc(0x400);
  fprintf(stderr, "Allocated large bin to trigger malloc_consolidate(): p3=%p\n", p3);
  fprintf(stderr, "In malloc_consolidate(), p1 is moved to the unsorted bin.\n");
  free(p1);
  fprintf(stderr, "Trigger the double free vulnerability!\n");
  fprintf(stderr, "We can pass the check in malloc() since p1 is not fast top.\n");
  fprintf(stderr, "Now p1 is in unsorted bin and fast bin. So we'will get it twice: %p %p\n", malloc(40), malloc(40));
}
```

```
(gdb) p main_arena.fastbinsY
$17 = {0x(
(gdb) x/8(
0x804a000:
0x804a010:
0x804a020:
0x804a030:
0x804a040:
0x804a050:
0x804a060:
0x804a070:
0x804a080:
0x804a090:
0x804a0a0:
0x804a0b0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0c0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0d0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0e0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0f0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a100:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a110:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a120:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a130:      0x00000000      0x00000000      0x00000000      0x00000000
(gdb)
```

```
root@protostar:/opt/how2heap# ./test_2
Allocated two fastbins: p1=0x804a008 p2=0x804a038
Now free p1!
Allocated large bin to trigger malloc_consolidate(): p3=0x804a068
In malloc_consolidate(), p1 is moved to the unsorted bin.
Trigger the double free vulnerability!
We can pass the check in malloc() since p1 is not fast top.
Now p1 is in unsorted bin and fast bin. So we'will get it twice: 0x804a008 0x804a008
root@protostar:/opt/how2heap#
```

```
(gdb) p main_arena.bins
$23 = {0xb7fd93d0, 0xb7fd93d0, 0xb7fd93d8, 0xb7fd93d8, 0xb7fd93e0, 0xb7fd93e0, 0xb7fd93e8, 0xb7fd93e8, 0xb7fd93f0, 0xb7fd93f0, 0xb7fd93f8, 0xb7fd93f8, 0xb7fd9400, 0xb7fd9400, 0xb7fd9408, 0xb7fd9408,
  0xb7fd9410, 0xb7fd9410, 0xb7fd9418, 0xb7fd9418, 0xb7fd9420, 0xb7fd9420, 0xb7fd9428, 0xb7fd9428, 0xb7fd9430, 0xb7fd9430, 0xb7fd9438, 0xb7fd9438, 0xb7fd9440, 0xb7fd9440, 0xb7fd9448, 0xb7fd9448, 0xb7fd9450,
  0xb7fd9450, 0xb7fd9458, 0xb7fd9458, 0xb7fd9460, 0xb7fd9460, 0xb7fd9468, 0xb7fd9468, 0xb7fd9470, 0xb7fd9470, 0xb7fd9478, 0xb7fd9478, 0xb7fd9480, 0xb7fd9480, 0xb7fd9488, 0xb7fd9488, 0xb7fd9490, 0xb7fd9490,
  0xb7fd9498, 0xb7fd9498, 0xb7fd94a0, 0xb7fd94a0, 0xb7fd94a8, 0xb7fd94a8, 0xb7fd94b0, 0xb7fd94b0, 0xb7fd94b8, 0xb7fd94b8, 0xb7fd94c0, 0xb7fd94c0, 0xb7fd94c8, 0xb7fd94c8, 0xb7fd94d0, 0xb7fd94d0, 0xb7fd94d8,
```

```
(gdb) p main_arena.fastbinsY
$19 = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0}
(gdb) x/80wx 0x0804a000
0x804a000:      0x00000000      0x00000031      0x00000000      0xb7fd93f8
0x804a010:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a020:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a030:      0x00000030      0x00000030      0x00000000      0x00000000
0x804a040:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a050:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a060:      0x00000000      0x00000409      0x00000000      0x00000000
0x804a070:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a080:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a090:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0a0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0b0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0c0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0d0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0e0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a0f0:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a100:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a110:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a120:      0x00000000      0x00000000      0x00000000      0x00000000
0x804a130:      0x00000000      0x00000000      0x00000000      0x00000000
(gdb)
```

e.N.d