와! 이거시 heap이다.

2019-01-14 서동훈

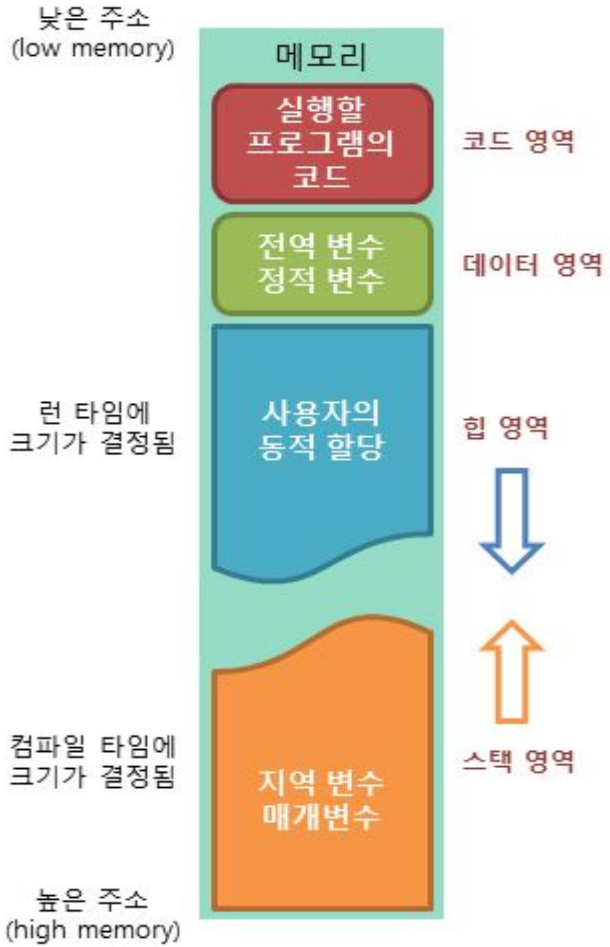
Heap

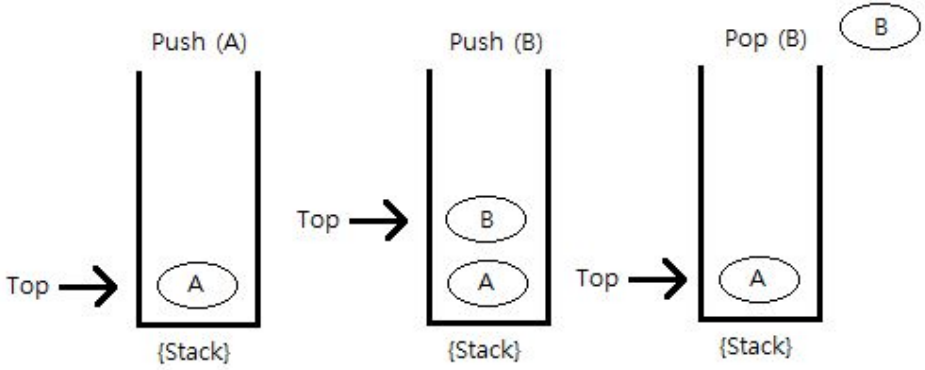
간단하게 프로그램이 실행되는 도중에
동적으로 할당하고 해제하는 메모리
영역이라고 보면 된다.

Heap

간단하게 프로그램이 실행되는 도중에 동적으로 할당하고 해제하는 메모리 영역이라고 보면 된다.

그렇다면 Heap 과 Stack의 차이는 뭘까?





LIFO(Last-IN First-Out)

○dlmalloc - 일반적인 목적의 할당자

○ptmalloc2 - glibc

ptmalloc2 은 dlmalloc에서 나뉘게 된 이후
스레딩 지원 기능이 추가되어,배포된것

○jemalloc - FreeBSD와 Firefox

○tcmalloc - Google(chrome)

○libumem - Solaris

heap의 할당과 반환

```
char *p = malloc(size)

free(p)
```

heap의 할당과 반환

char *p = malloc(size)

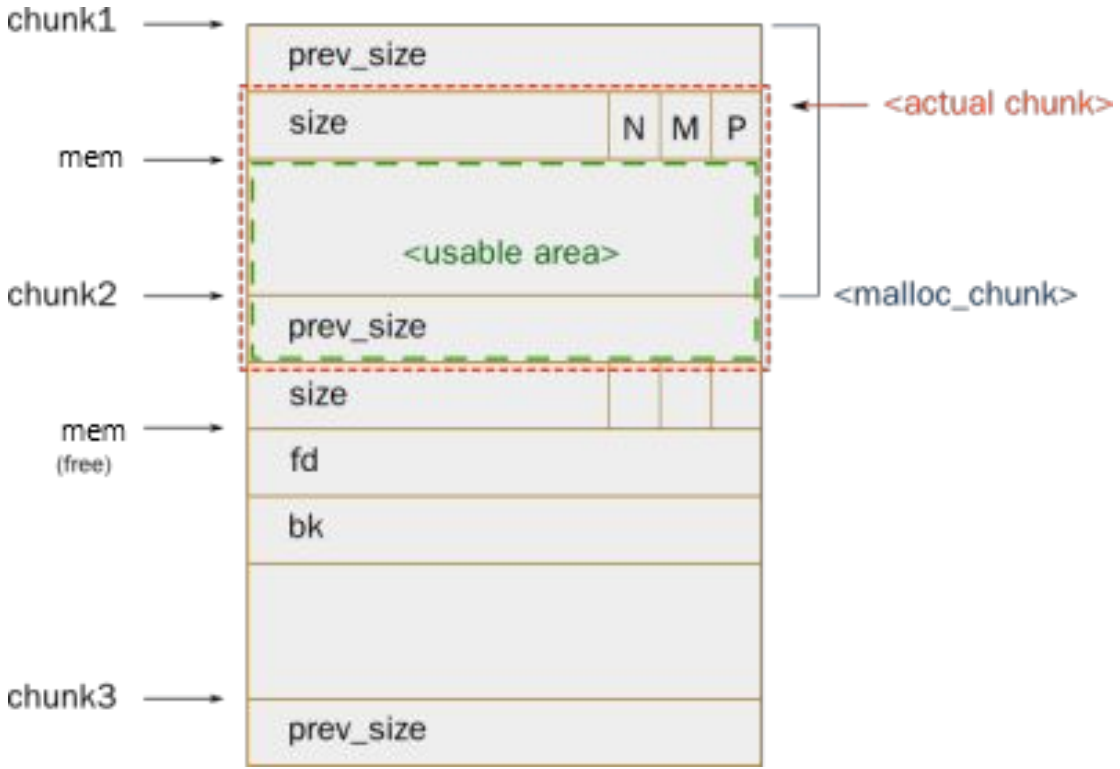
free(p)

```
gdb-peda$ vmmap
Start      End        Perm      Name
0x56555000 0x56556000 r--p      /root/pratice/tt
0x56556000 0x56557000 r-xp      /root/pratice/tt
0x56557000 0x56558000 r--p      /root/pratice/tt
0x56558000 0x56559000 r--p      /root/pratice/tt
0x56559000 0x5655a000 rw-p      /root/pratice/tt
0xf7dce000 0xf7de7000 r--p      /usr/lib32/libc-2.28.so
0xf7de7000 0xf7f35000 r-xp      /usr/lib32/libc-2.28.so
0xf7f35000 0xf7fa5000 r--p      /usr/lib32/libc-2.28.so
0xf7fa5000 0xf7fa6000 ---p      /usr/lib32/libc-2.28.so
0xf7fa6000 0xf7fa8000 r--p      /usr/lib32/libc-2.28.so
0xf7fa8000 0xf7fa9000 rw-p      /usr/lib32/libc-2.28.so
0xf7fa9000 0xf7fac000 rw-p      mapped
0xf7fcd000 0xf7fcf000 rw-p      mapped
0xf7fcf000 0xf7fd2000 r--p      [vvar]
0xf7fd2000 0xf7fd4000 r-xp      [vdso]
0xf7fd4000 0xf7fd5000 r--p      /usr/lib32/ld-2.28.so
0xf7fd5000 0xf7ff1000 r-xp      /usr/lib32/ld-2.28.so
0xf7ff1000 0xf7ffb000 r--p      /usr/lib32/ld-2.28.so
0xf7ffc000 0xf7ffd000 r--p      /usr/lib32/ld-2.28.so
0xf7ffd000 0xf7ffe000 rw-p      /usr/lib32/ld-2.28.so
0xfffdd000 0xffffe000 rw-p      [stack]
```

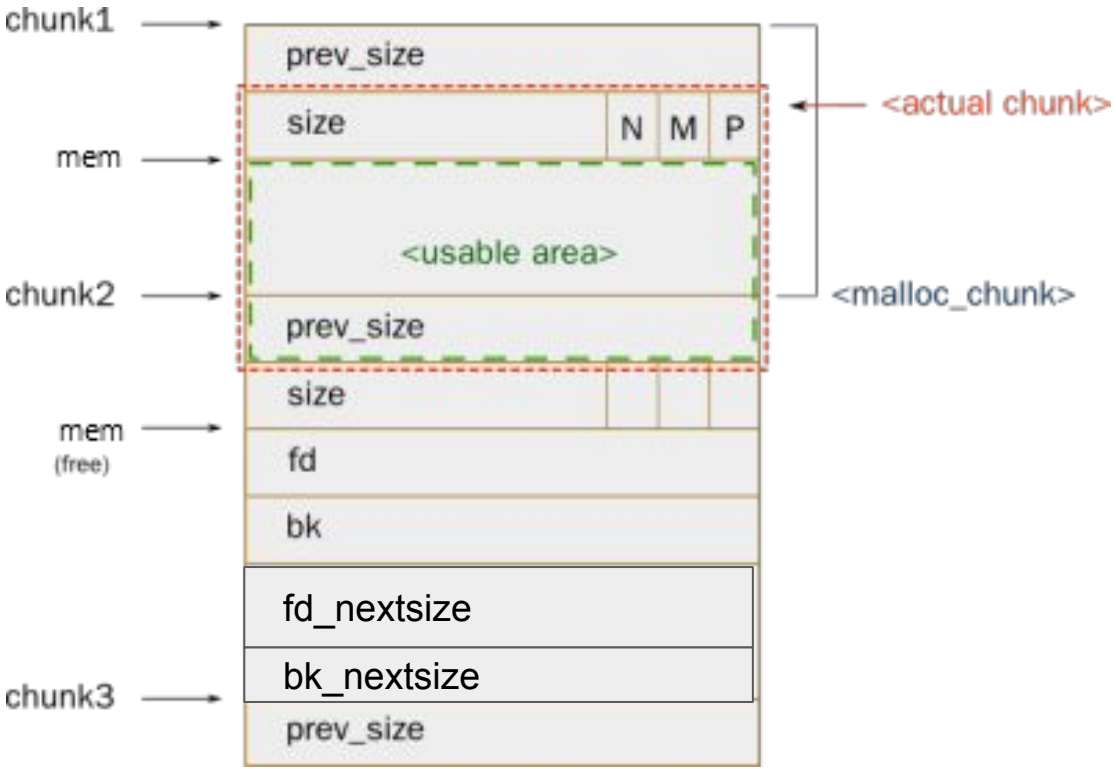
```
Start      End        Perm      Name
0x56555000 0x56556000 r--p      /root/pratice/tt
0x56556000 0x56557000 r-xp      /root/pratice/tt
0x56557000 0x56558000 r--p      /root/pratice/tt
0x56558000 0x56559000 r--p      /root/pratice/tt
0x56559000 0x5655a000 rw-p      /root/pratice/tt
0x5655a000 0x5657c000 rw-p      [heap]
0xf7dce000 0xf7de7000 r--p      /usr/lib32/libc-2.28.so
0xf7de7000 0xf7f35000 r-xp      /usr/lib32/libc-2.28.so
0xf7f35000 0xf7fa5000 r--p      /usr/lib32/libc-2.28.so
0xf7fa5000 0xf7fa6000 ---p      /usr/lib32/libc-2.28.so
0xf7fa6000 0xf7fa8000 r--p      /usr/lib32/libc-2.28.so
0xf7fa8000 0xf7fa9000 rw-p      /usr/lib32/libc-2.28.so
0xf7fa9000 0xf7fac000 rw-p      mapped
0xf7fcd000 0xf7fcf000 rw-p      mapped
0xf7fcf000 0xf7fd2000 r--p      [vvar]
0xf7fd2000 0xf7fd4000 r-xp      [vdso]
0xf7fd4000 0xf7fd5000 r--p      /usr/lib32/ld-2.28.so
0xf7fd5000 0xf7ff1000 r-xp      /usr/lib32/ld-2.28.so
0xf7ff1000 0xf7ffb000 r--p      /usr/lib32/ld-2.28.so
0xf7ffc000 0xf7ffd000 r--p      /usr/lib32/ld-2.28.so
0xf7ffd000 0xf7ffe000 rw-p      /usr/lib32/ld-2.28.so
0xfffdd000 0xffffe000 rw-p      [stack]
```

Chunk

Chunk


chunk1
prev_size
size
N
M
P
<actual chunk>
mem
<usable area>
chunk2
<malloc_chunk>
prev_size
size
mem
(free)
fd
bk
chunk3
prev_size

Chunk


chunk1
prev_size
size
N
M
P
<actual chunk>
mem
<usable area>
chunk2
<malloc_chunk>
prev_size
size
mem
(free)
fd
bk
fd_nextsize
bk_nextsize
chunk3
prev_size

Chunk Size

size= 현재 chunk의 크기를 나타내며 하위 3비트는
특별한 용도로 쓰임

8바이트 단위로 정렬되어 하위 3바이트가 사용되지 않음

01000 = 8
10000 = 16
11000 = 24
8바이트단위 정렬
(2진법)

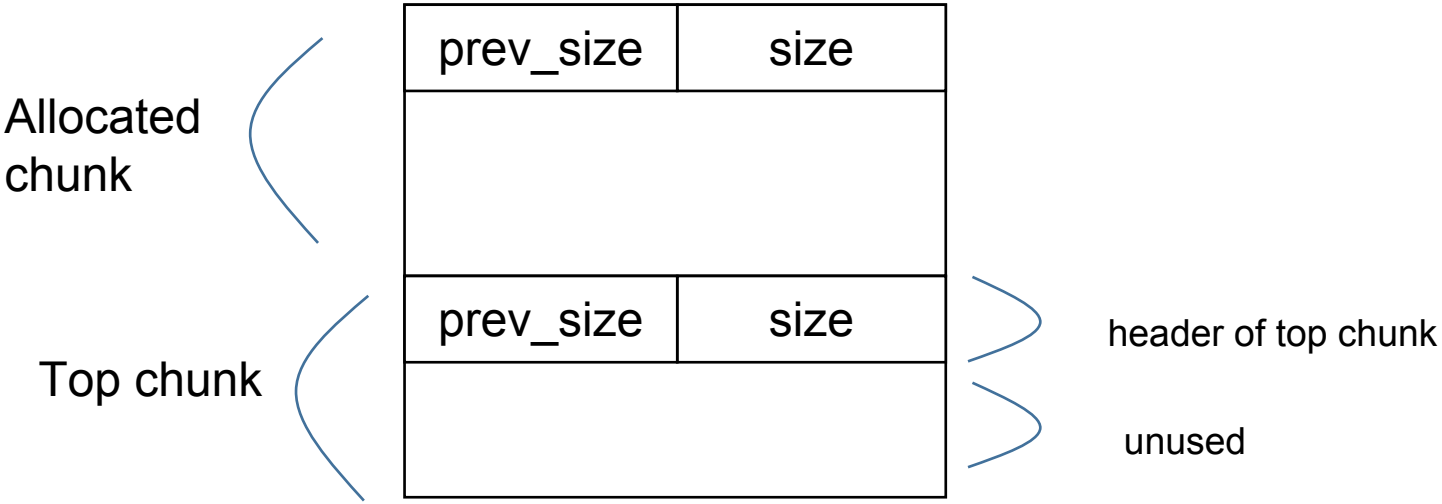
001

prev_inuse bit
1=인접한 이전 chunk가 할당되었다.
0=인접한 이전 chunk가 해제되었다.

NON_MAIN_ARENA
0=main heap(arena)에 속한다
1=속하지 않는다.

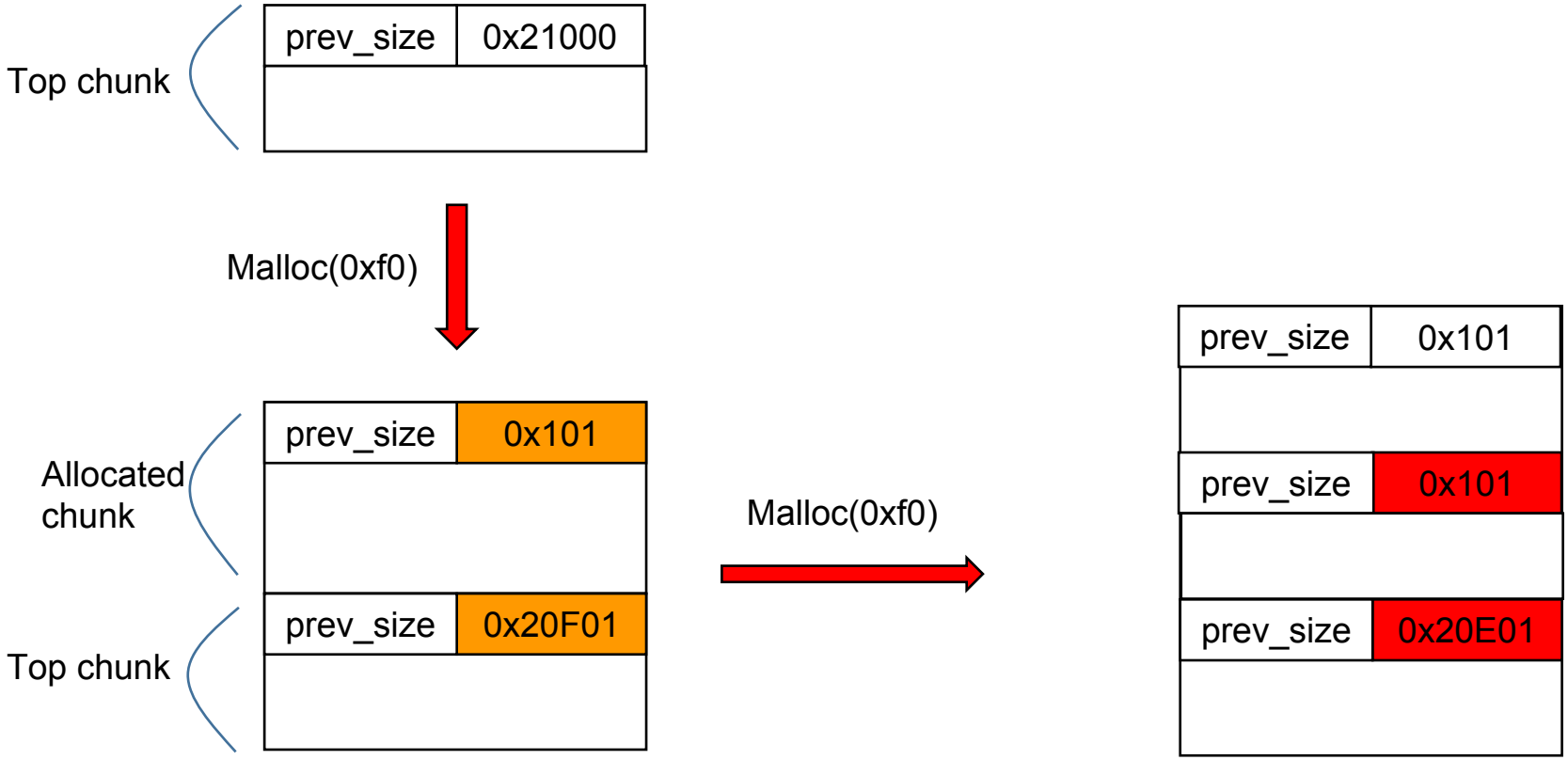
IS_MMAPPED
1=chunk 자체가 단일 mmap()호출로 할당된 영역 전체
0=할당되지 않는다.

Top Chunk(wildness chunk)

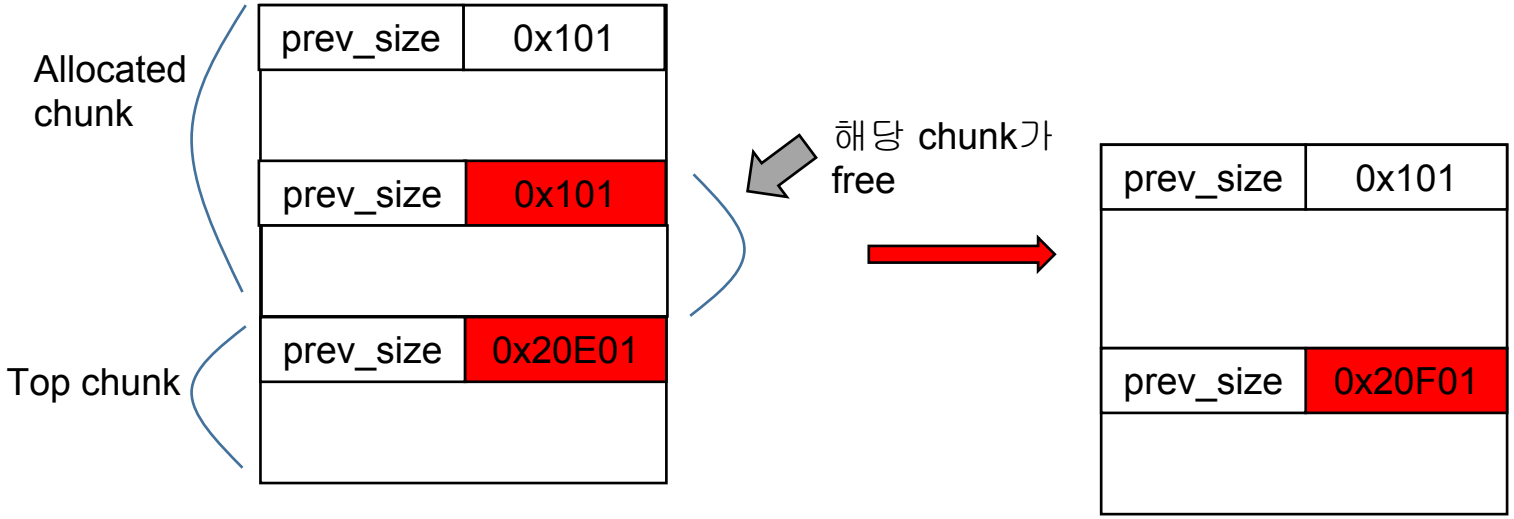


heap 영역의 가장 마지막에 위치하며 새롭게 할당(malloc)되면 top chunk에서 분리해서 반환하고 top chunk에서 인접한 chunk가 free되면 병합한다

Top Chunk(wildness chunk)



Top Chunk(wildness chunk)



```c
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <stdio.h>
int main(int argc, char **argv)
{
        char *a, *b, *c;

        a = malloc(32);
        b = malloc(32);
        c = malloc(32);

        strcpy(a, argv[1]);
        strcpy(b, argv[2]);
        strcpy(c, argv[3]);

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

        return 0;
}
```

```
0x5655a150:     0x00000000      0x00000000      0x00000000      0x00000031
0x5655a160:     0x41414141      0x41414141      0x00000000      0x00000000
0x5655a170:     0x00000000      0x00000000      0x00000000      0x00000000
0x5655a180:     0x00000000      0x00000000      0x00000000      0x00000031
0x5655a190:     0x42424242      0x42424242      0x00000000      0x00000000
0x5655a1a0:     0x00000000      0x00000000      0x00000000      0x00000000
0x5655a1b0:     0x00000000      0x00000000      0x00000000      0x00000031
0x5655a1c0:     0x43434343      0x43434343      0x00000000      0x00000000
0x5655a1d0:     0x00000000      0x00000000      0x00000000      0x00000000
0x5655a1e0:     0x00000000      0x00000000      0x00000000      0x00021e19
0x5655a1f0:     0x00000000      0x00000000      0x00000000      0x00000000
0x5655a200:     0x00000000      0x00000000      0x00000000      0x00000000
0x5655a210:     0x00000000      0x00000000
```

```c
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <stdio.h>
int main(int argc, char **argv)
{
        char *a, *b, *c;

        a = malloc(32);
        b = malloc(32);
        c = malloc(32);

        strcpy(a, argv[1]);
        strcpy(b, argv[2]);
        strcpy(c, argv[3]);

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

        return 0;
}
```

```
0x5655a150:     0x00000000      0x00000000      0x00000000      0x00000031
0x5655a160:     0x41414141      0x41414141      0x00000000      0x00000000
0x5655a170:     0x00000000      0x00000000      0x00000000      0x00000000
0x5655a180:     0x00000000      0x00000000      0x00000000      0x00000031
0x5655a190:     0x42424242      0x42424242      0x00000000      0x00000000
0x5655a1a0:     0x00000000      0x00000000      0x00000000      0x00000000
0x5655a1b0:     0x00000000      0x00000000      0x00000000      0x00000031
0x5655a1c0:     0x43434343      0x43434343      0x00000000      0x00000000
0x5655a1d0:     0x00000000      0x00000000      0x00000000      0x00000000
0x5655a1e0:     0x00000000      0x00000000      0x00000000      0x00021e19
0x5655a1f0:     0x00000000      0x00000000      0x00000000      0x00000000
0x5655a200:     0x00000000      0x00000000      0x00000000      0x00000000
0x5655a210:     0x00000000      0x00000000
```

free(c)

```
0x5655a150:     0x00000000      0x00000000      0x00000000      0x00000031
0x5655a160:     0x41414141      0x41414141      0x00000000      0x00000000
0x5655a170:     0x00000000      0x00000000      0x00000000      0x00000000
0x5655a180:     0x00000000      0x00000000      0x00000000      0x00000031
0x5655a190:     0x42424242      0x42424242      0x00000000      0x00000000
0x5655a1a0:     0x00000000      0x00000000      0x00000000      0x00000000
0x5655a1b0:     0x00000000      0x00000000      0x00000000      0x00000031
0x5655a1c0:     0x00000000      0x5655a010      0x00000000      0x00000000
0x5655a1d0:     0x00000000      0x00000000      0x00000000      0x00000000
0x5655a1e0:     0x00000000      0x00000000      0x00000000      0x00021e19
0x5655a1f0:     0x00000000      0x00000000      0x00000000      0x00000000
0x5655a200:     0x00000000      0x00000000      0x00000000      0x00000000
0x5655a210:     0x00000000      0x00000000
```

```c
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <stdio.h>
int main(int argc, char **argv)
{
        char *a, *b, *c;

        a = malloc(32);
        b = malloc(32);
        c = malloc(32);

        strcpy(a, argv[1]);
        strcpy(b, argv[2]);
        strcpy(c, argv[3]);

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

        return 0;
}
```

```
0x5655a150:     0x00000000      0x00000000      0x00000000      0x00000031
0x5655a160:     0x41414141      0x41414141      0x00000000      0x00000000
0x5655a170:     0x00000000      0x00000000      0x00000000      0x00000000
0x5655a180:     0x00000000      0x00000000      0x00000000      0x00000031
0x5655a190:     0x42424242      0x42424242      0x00000000      0x00000000
0x5655a1a0:     0x00000000      0x00000000      0x00000000      0x00000000
0x5655a1b0:     0x00000000      0x00000000      0x00000000      0x00000031
0x5655a1c0:     0x00000000      0x5655a010      0x00000000      0x00000000
0x5655a1d0:     0x00000000      0x00000000      0x00000000      0x00000000
0x5655a1e0:     0x00000000      0x00000000      0x00000000      0x00021e19
0x5655a1f0:     0x00000000      0x00000000      0x00000000      0x00000000
0x5655a200:     0x00000000      0x00000000      0x00000000      0x00000000
0x5655a210:     0x00000000      0x00000000
```

free(b)

```
0x5655a150:     0x00000000      0x00000000      0x00000000      0x00000031
0x5655a160:     0x41414141      0x41414141      0x00000000      0x00000000
0x5655a170:     0x00000000      0x00000000      0x00000000      0x00000000
0x5655a180:     0x00000000      0x00000000      0x00000000      0x00000031
0x5655a190:     0x5655a1c0      0x5655a010      0x00000000      0x00000000
0x5655a1a0:     0x00000000      0x00000000      0x00000000      0x00000000
0x5655a1b0:     0x00000000      0x00000000      0x00000000      0x00000031
0x5655a1c0:     0x00000000      0x5655a010      0x00000000      0x00000000
0x5655a1d0:     0x00000000      0x00000000      0x00000000      0x00000000
0x5655a1e0:     0x00000000      0x00000000      0x00000000      0x00021e19
0x5655a1f0:     0x00000000      0x00000000      0x00000000      0x00000000
0x5655a200:     0x00000000      0x00000000      0x00000000      0x00000000
0x5655a210:     0x00000000      0x00000000
```

```c
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <stdio.h>
int main(int argc, char **argv)
{
        char *a, *b, *c;

        a = malloc(32);
        b = malloc(32);
        c = malloc(32);

        strcpy(a, argv[1]);
        strcpy(b, argv[2]);
        strcpy(c, argv[3]);

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

        return 0;
}
```

```
0x5655a150:     0x00000000      0x00000000      0x00000000      0x00000031
0x5655a160:     0x41414141      0x41414141      0x00000000      0x00000000
0x5655a170:     0x00000000      0x00000000      0x00000000      0x00000000
0x5655a180:     0x00000000      0x00000000      0x00000000      0x00000031
0x5655a190:     0x5655a1c0      0x5655a010      0x00000000      0x00000000
0x5655a1a0:     0x00000000      0x00000000      0x00000000      0x00000000
0x5655a1b0:     0x00000000      0x00000000      0x00000000      0x00000031
0x5655a1c0:     0x00000000      0x5655a010      0x00000000      0x00000000
0x5655a1d0:     0x00000000      0x00000000      0x00000000      0x00000000
0x5655a1e0:     0x00000000      0x00000000      0x00000000      0x00021e19
0x5655a1f0:     0x00000000      0x00000000      0x00000000      0x00000000
0x5655a200:     0x00000000      0x00000000      0x00000000      0x00000000
0x5655a210:     0x00000000      0x00000000
```

free(a)

```
0x5655a150:     0x00000000      0x00000000      0x00000000      0x00000031
0x5655a160:     0x5655a190      0x5655a010      0x00000000      0x00000000
0x5655a170:     0x00000000      0x00000000      0x00000000      0x00000000
0x5655a180:     0x00000000      0x00000000      0x00000000      0x00000031
0x5655a190:     0x5655a1c0      0x5655a010      0x00000000      0x00000000
0x5655a1a0:     0x00000000      0x00000000      0x00000000      0x00000000
0x5655a1b0:     0x00000000      0x00000000      0x00000000      0x00000031
0x5655a1c0:     0x00000000      0x5655a010      0x00000000      0x00000000
0x5655a1d0:     0x00000000      0x00000000      0x00000000      0x00000000
0x5655a1e0:     0x00000000      0x00000000      0x00000000      0x00021e19
0x5655a1f0:     0x00000000      0x00000000      0x00000000      0x00000000
0x5655a200:     0x00000000      0x00000000      0x00000000      0x00000000
0x5655a210:     0x00000000      0x00000000
```