

Heap Basic

김우종

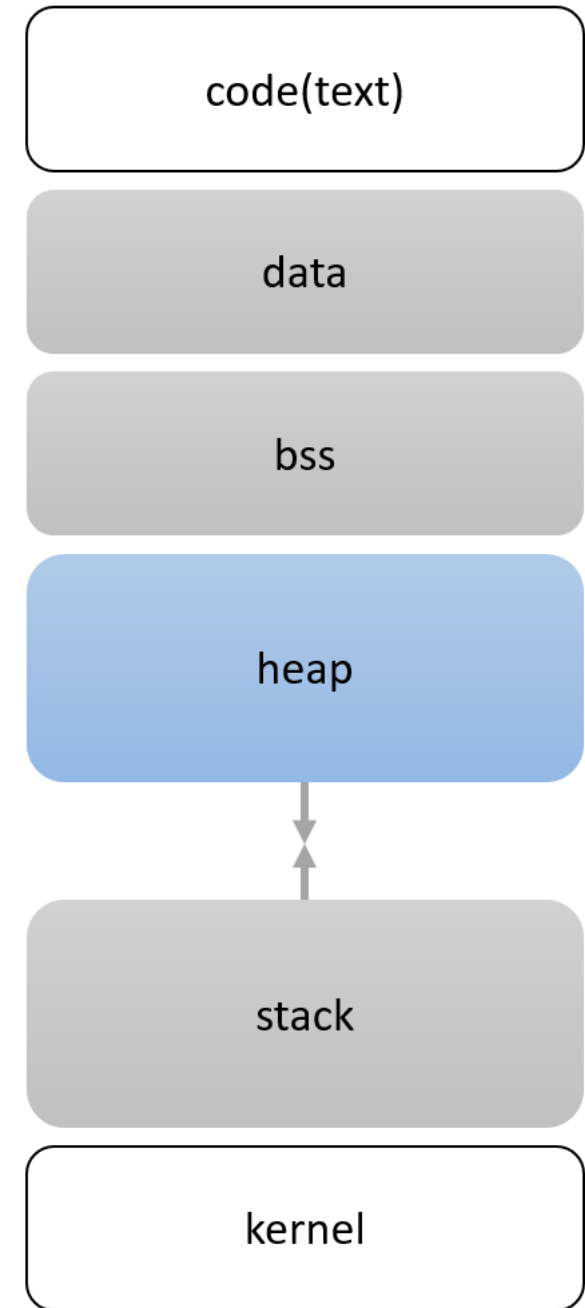
목차

- Heap이란?
- 동적 메모리 할당자
- Heap 할당/해제
- chunk란?
- Allocated Chunk
- Freed Chunk
- Top chunk

Heap 이란?

- 메모리 구조에는 여러가지 영역이 존재
 - 그 중 Heap에 대한 것을 발표
- Heap이란?
 - Heap이란 프로그램이 실행되는 도중 동적으로 할당하고 해제하여 사용하는 메모리 영역을 의미
 - 대표적인 메모리 할당/해제 함수
 - malloc()
 - free()

Low address



High add

동적 메모리 할당자

- 프로그램이 실행되면 다양한 크기의 데이터가 들어감으로 다양한 크기의 동적 영역이 필요하다.
- 이에 따라 메모리 할당자도 여러가지가 존재한다.
- 할당자의 종류는 다음과 같다
 - tcmalloc
 - Libumem
 - Jemalloc
 - dlmalloc
 - ptmalloc2

Heap 할당/해제

- 메모리 할당 전

```
gdb-peda$ info proc map
process 141
Mapped address spaces:

    Start Addr           End Addr       Size     Offset objfile
    -----
    0x400000             0x401000       0x1000        0x0  /home/test
    0x600000             0x601000       0x1000        0x0  /home/test
    0x601000             0x602000       0x1000       0x1000 /home/test
    0x7f894bf79000        0x7f894c139000 0x1c0000        0x0  /lib/x86_64-linux-gnu/libc-2.23.so
    0x7f894c139000        0x7f894c339000 0x200000     0x1c0000 /lib/x86_64-linux-gnu/libc-2.23.so
    0x7f894c339000        0x7f894c33d000   0x4000     0x1c0000 /lib/x86_64-linux-gnu/libc-2.23.so
    0x7f894c33d000        0x7f894c33f000   0x2000     0x1c4000 /lib/x86_64-linux-gnu/libc-2.23.so
    0x7f894c33f000        0x7f894c343000   0x4000        0x0  /lib/x86_64-linux-gnu/libc-2.23.so
    0x7f894c343000        0x7f894c369000  0x26000        0x0  /lib/x86_64-linux-gnu/ld-2.23.so
    0x7f894c560000        0x7f894c563000   0x3000        0x0  /lib/x86_64-linux-gnu/ld-2.23.so
    0x7f894c568000        0x7f894c569000   0x1000     0x25000 /lib/x86_64-linux-gnu/ld-2.23.so
    0x7f894c569000        0x7f894c56a000   0x1000     0x26000 /lib/x86_64-linux-gnu/ld-2.23.so
    0x7f894c56a000        0x7f894c56b000   0x1000        0x0  /lib/x86_64-linux-gnu/ld-2.23.so
    0x7ffe86c12000        0x7ffe86c33000  0x21000        0x0  [stack]
    0x7ffe86c68000        0x7ffe86c6b000   0x3000        0x0  [vvar]
    0x7ffe86c6b000        0x7ffe86c6c000   0x1000        0x0  [vdso]
    0xffffffff600000      0xffffffff601000 0x1000        0x0  [vsyscall]
gdb-peda$
```

Heap 할당/해제

- 메모리 할당 후
 - `char *p = malloc(size)`

```
gdb-peda$ info proc map
process 145
Mapped address spaces:

Trash
Start Addr      End Addr      Size          Offset objfile
0x400000        0x401000      0x1000        0x0  /home/test
0x600000        0x601000      0x1000        0x0  /home/test
0x601000        0x602000      0x1000        0x1000 /home/test
0x7fc000        0x81d000      0x21000       0x0  [heap]
0x7fd1e5064000  0x7fd1e5224000 0x1c0000      0x0  /lib/x86_64-linux-gnu/libc-2.23.so
0x7fd1e5224000  0x7fd1e5424000 0x200000      0x1c0000 /lib/x86_64-linux-gnu/libc-2.23.so
0x7fd1e5424000  0x7fd1e5428000 0x4000        0x1c0000 /lib/x86_64-linux-gnu/libc-2.23.so
0x7fd1e5428000  0x7fd1e542a000 0x2000        0x1c4000 /lib/x86_64-linux-gnu/libc-2.23.so
0x7fd1e542a000  0x7fd1e542e000 0x4000        0x0
0x7fd1e542e000  0x7fd1e5454000 0x26000       0x0  /lib/x86_64-linux-gnu/ld-2.23.so
0x7fd1e564b000  0x7fd1e564e000 0x3000        0x0
0x7fd1e5653000  0x7fd1e5654000 0x1000        0x25000 /lib/x86_64-linux-gnu/ld-2.23.so
0x7fd1e5654000  0x7fd1e5655000 0x1000        0x26000 /lib/x86_64-linux-gnu/ld-2.23.so
0x7fd1e5655000  0x7fd1e5656000 0x1000        0x0
0x7fff64ea7000  0x7fff64ec8000 0x21000       0x0  [stack]
0x7fff64f74000  0x7fff64f77000 0x3000        0x0  [vvar]
0x7fff64f77000  0x7fff64f78000 0x1000        0x0  [vdso]
0xffffffff600000 0xffffffff601000 0x1000        0x0  [syscall]

gdb-peda$
```


Heap 할당/해제

- free() 호출 후
 - free(p)

```
gdb-peda$ info proc map
process 145
Mapped address spaces:

Trash
Start Addr      End Addr       Size           Offset objfile
0x400000        0x401000       0x1000         0x0  /home/test
0x600000        0x601000       0x1000         0x0  /home/test
0x601000        0x602000       0x1000         0x1000 /home/test
0x7fc000        0x81d000       0x21000        0x0  [heap]
0x7fd1e5064000  0x7fd1e5224000 0x1c0000       0x0  /lib/x86_64-linux-gnu/libc-2.23.so
0x7fd1e5224000  0x7fd1e5424000 0x200000       0x1c0000 /lib/x86_64-linux-gnu/libc-2.23.so
0x7fd1e5424000  0x7fd1e5428000 0x4000         0x1c0000 /lib/x86_64-linux-gnu/libc-2.23.so
0x7fd1e5428000  0x7fd1e542a000 0x2000         0x1c4000 /lib/x86_64-linux-gnu/libc-2.23.so
0x7fd1e542a000  0x7fd1e542e000 0x4000         0x0
0x7fd1e542e000  0x7fd1e5454000 0x26000        0x0  /lib/x86_64-linux-gnu/ld-2.23.so
0x7fd1e564b000  0x7fd1e564e000 0x3000         0x0
0x7fd1e5653000  0x7fd1e5654000 0x1000         0x25000 /lib/x86_64-linux-gnu/ld-2.23.so
0x7fd1e5654000  0x7fd1e5655000 0x1000         0x26000 /lib/x86_64-linux-gnu/ld-2.23.so
0x7fd1e5655000  0x7fd1e5656000 0x1000         0x0
0x7fff64ea7000  0x7fff64ec8000 0x21000        0x0  [stack]
0x7fff64f74000  0x7fff64f77000 0x3000         0x0  [vvar]
0x7fff64f77000  0x7fff64f78000 0x1000         0x0  [vdso]
0xffffffff600000 0xffffffff601000 0x1000         0x0  [syscall]

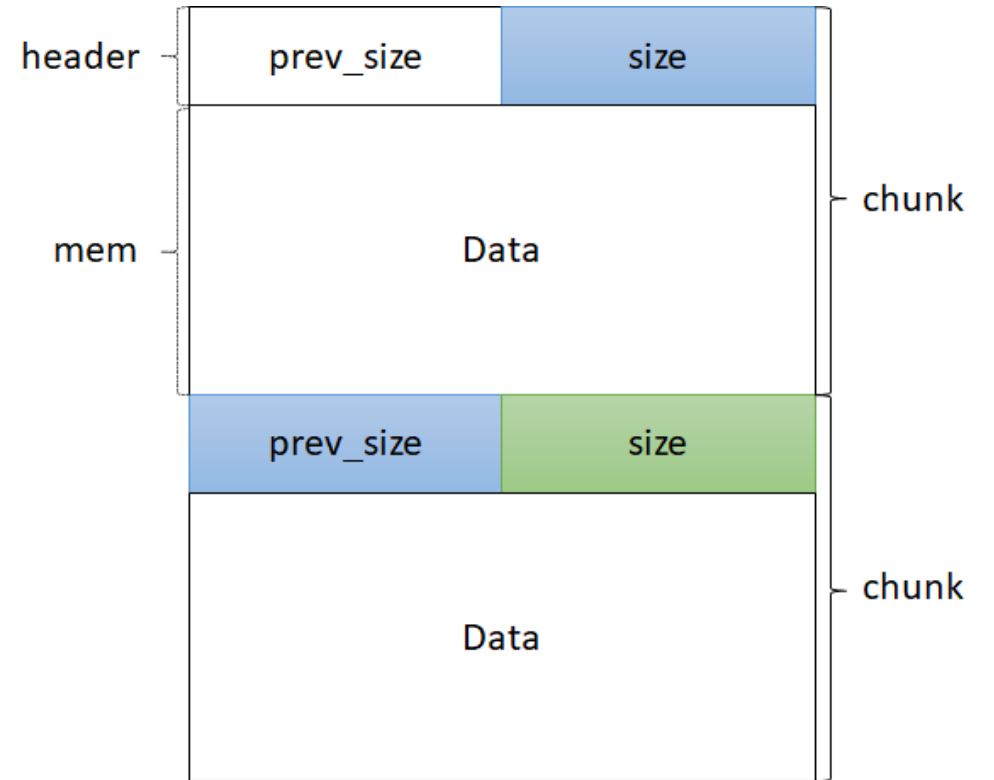
gdb-peda$
```

Chunk란 ?

[illegible]

Chunk란?

- mem
 - malloc()으로 할당 받은 부분
- chunk
 - mem에서 header를 포함한 부분
- size
 - chunk의 크기를 나타냄
 - 32bit : 8byte 정렬
 - 64bit : 16byte 정렬
- prev_size
 - 인접한 앞쪽 chunk의 크기를 나타냄



size

- size

- size는 chunk의 크기 뿐만 아니라 3가지 flag 또한 가지고 있음
- why?
 - 32bit환경에서는 8byte 단위 정렬
 - 64bit환경에서는 16byte 단위 정렬
 - 하위 3byte 사용 X

$$01000(2) = 8$$

$$10000(2) = 16$$

$$11000(2) = 24$$

flag

prev_inuse bit

000(2)

이전 chunk 할당 시 1

이전 chunk 해제 시 0

is_mmaped bit

000(2)

mmap으로 할당 시 1

mmap으로 할당 아닐 시 0

non_main_arena bit

000(2)

thread arena에 속하면 1

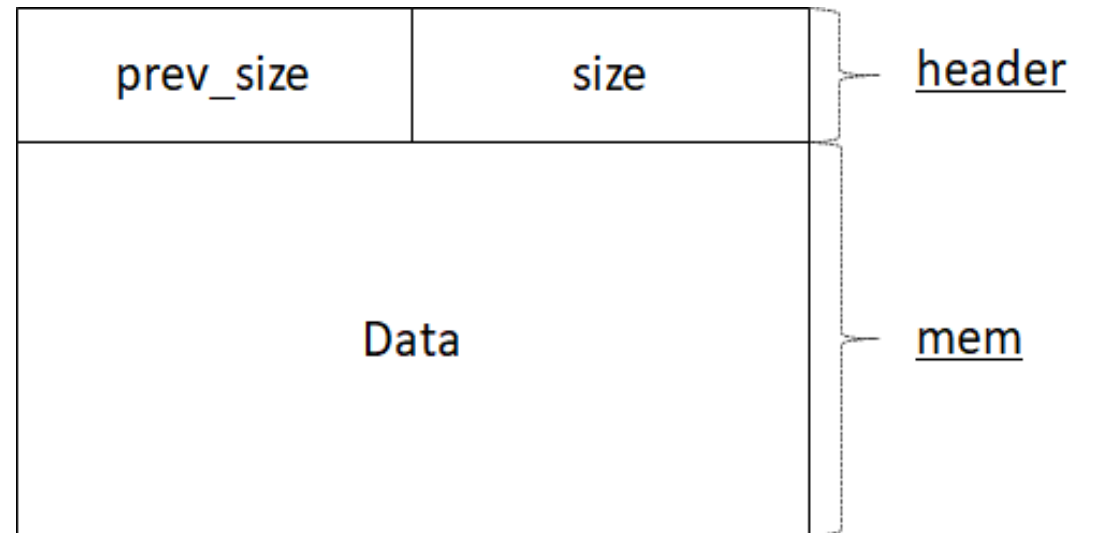
thread arena에 속하지 않음 0

Chunk 의 종류

- Chunk의 종류는 크게 3가지로 나뉜다.
 - Allocated chunk
 - malloc()을 호출할 때 생기는 chunk
 - free된 chunk와 구분하는 용도로 Allocated chunk라고 부른다.
 - Freed chunk
 - free()를 호출할 경우 할당 해제된 chunk
 - 할당 해제된 chunk들을 효율적으로 관리하기 위해 linked list로 관리한다.
 - Top chunk
 - 힙 영역에 가장 마지막에 위치한다.
 - 새롭게 메모리를 할당하면 Top chunk에서 분리해서 반환한다.
 - 인접한 chunk가 할당 해제되면 Top chunk에 병합한다.

Allocated chunk

- **Allocated chunk**
 - malloc()을 호출할 때 생기는 chunk
 - free된 chunk와 구분하는 용도로 Allocated chunk라고 부른다.



Allocated chunk

Allocated chunk

- Allocated chunk
 - malloc()을 호출할 때 생기는 chunk
 - free된 chunk와 구분하는 용도로 Allocated chunk라고 부른다.

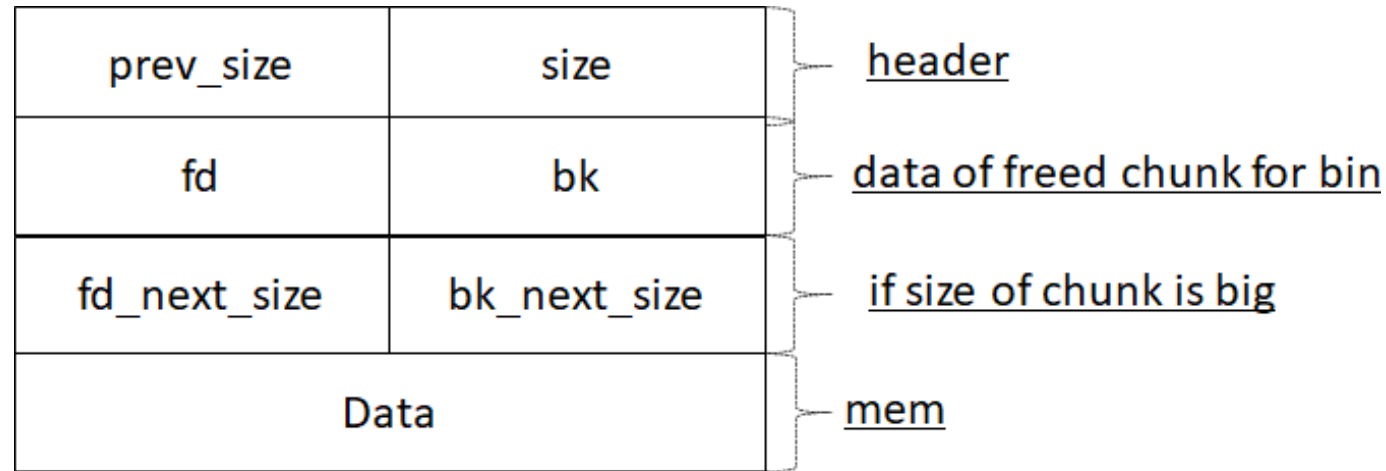
```
gdb-peda$ x/32gx $rax - 0x10
0x405290: 0x0000000000000000 0x0000000000000021
0x4052a0: 0x0000000000000000 0x0000000000000000
```

header
mem

 : size : prev_size

Freed chunk

- Freed chunk
 - 할당 해제된 chunk를 의미
 - linked list로 관리됨
 - fd : 인접한 앞 chunk
 - bk : 인접한 뒷 chunk
 - 512byte 보다 큰 chunk들은 large chunk라고 함
 - fd_next_size
 - bk_next_size
 - 를 이용해 함께 관리







Freed chunk

Freed chunk

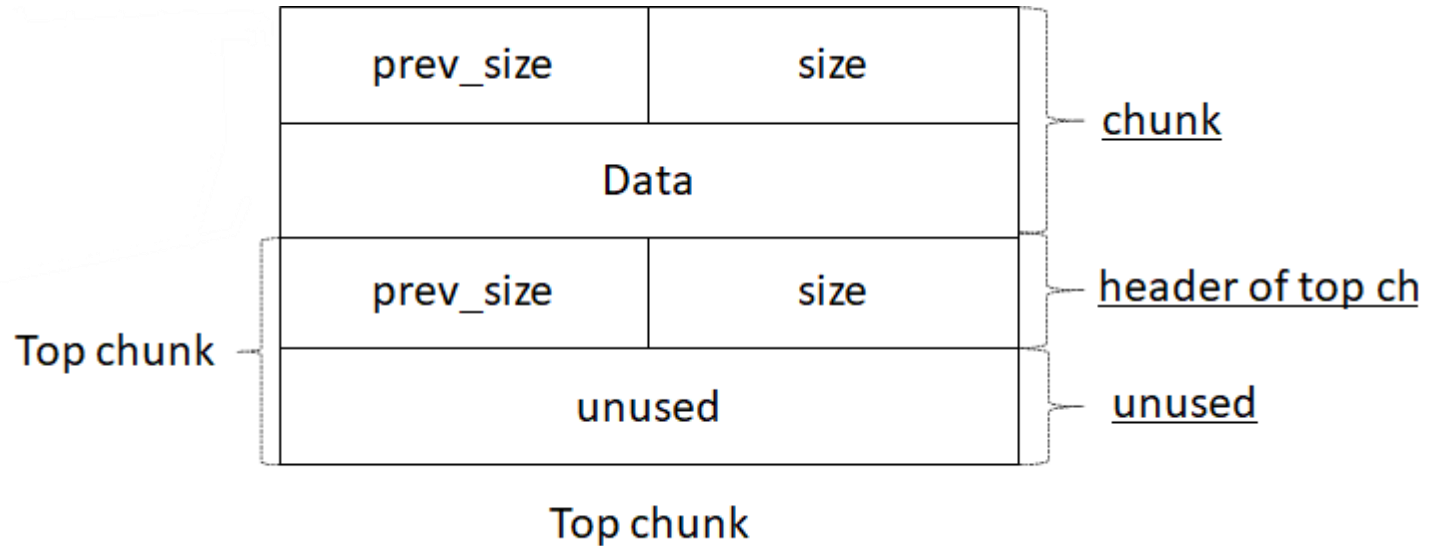
- Freed chunk
 - 할당 해제된 chunk를 의미
 - linked list로 관리됨
 - fd : 인접한 앞 chunk
 - bk : 인접한 뒷 chunk

```
gdb-peda$ x/32gx 0x22a2000
0x22a2000: 0x0000000000000000 0x00000000000000a1
0x22a2010: 0x00007f9e0c2a1b78 0x00007f9e0c2a1b78
```

	: heap1_size		: heap1_prev_size
	: fd		: bk

Top chunk

- Top chunk
 - 힙 영역에 가장 마지막에 위치한다.
 - 새롭게 메모리를 할당하면 Top chunk에서 분리해서 반환한다.
 - 인접한 chunk가 할당 해제되면 Top chunk에 병합한다.



Top chunk

- Top chunk
 - 힙 영역에 가장 마지막에 위치한다.
 - 새롭게 메모리를 할당하면 Top chunk에서 분리해서 반환한다.
 - 인접한 chunk가 할당 해제되면 Top chunk에 병합한다.

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

void main(){
    char *heap1 = malloc(0x90);
    char *heap2 = malloc(0x90);

    free(heap2);
    free(heap1);
}
```

Top chunk

- Top chunk
 - 힙 영역에 가장 마지막에 위치한다.
 - 새롭게 메모리를 할당하면 Top chunk에서 분리해서 반환한다.
 - 인접한 chunk가 할당 해제되면 Top chunk에 병합한다.

```
gdb-peda$ x/64gx 0x11a8000
0x11a8000: 0x0000000000000000 0x00000000000000a1
0x11a8010: 0x0000000000000000 0x0000000000000000
0x11a8020: 0x0000000000000000 0x0000000000000000
0x11a8030: 0x0000000000000000 0x0000000000000000
0x11a8040: 0x0000000000000000 0x0000000000000000
0x11a8050: 0x0000000000000000 0x0000000000000000
0x11a8060: 0x0000000000000000 0x0000000000000000
0x11a8070: 0x0000000000000000 0x0000000000000000
0x11a8080: 0x0000000000000000 0x0000000000000000
0x11a8090: 0x0000000000000000 0x0000000000000000
0x11a80a0: 0x0000000000000000 0x00000000000000a1
0x11a80b0: 0x0000000000000000 0x0000000000000000
0x11a80c0: 0x0000000000000000 0x0000000000000000
0x11a80d0: 0x0000000000000000 0x0000000000000000
0x11a80e0: 0x0000000000000000 0x0000000000000000
0x11a80f0: 0x0000000000000000 0x0000000000000000
0x11a8100: 0x0000000000000000 0x0000000000000000
0x11a8110: 0x0000000000000000 0x0000000000000000
0x11a8120: 0x0000000000000000 0x0000000000000000
0x11a8130: 0x0000000000000000 0x0000000000000000
0x11a8140: 0x0000000000000000 0x00000000000020ec1
```

chunk

chunk

top chunk

0x0000000000000000 : prev_size 0x00000000000020ec1 : size

Top chunk

- Top chunk
 - 힙 영역에 가장 마지막에 위치한다.
 - 새롭게 메모리를 할당하면 Top chunk에서 분리해서 반환한다.
 - 인접한 chunk가 할당 해제되면 Top chunk에 병합한다.

```
gdb-peda$ x/64gx 0x11a8000
0x11a8000: 0x0000000000000000 0x00000000000000a1
0x11a8010: 0x0000000000000000 0x0000000000000000
0x11a8020: 0x0000000000000000 0x0000000000000000
0x11a8030: 0x0000000000000000 0x0000000000000000
0x11a8040: 0x0000000000000000 0x0000000000000000
0x11a8050: 0x0000000000000000 0x0000000000000000
0x11a8060: 0x0000000000000000 0x0000000000000000
0x11a8070: 0x0000000000000000 0x0000000000000000
0x11a8080: 0x0000000000000000 0x0000000000000000
0x11a8090: 0x0000000000000000 0x0000000000000000
0x11a80a0: 0x0000000000000000 0x00000000000020f61
0x11a80b0: 0x0000000000000000 0x0000000000000000
0x11a80c0: 0x0000000000000000 0x0000000000000000
0x11a80d0: 0x0000000000000000 0x0000000000000000
0x11a80e0: 0x0000000000000000 0x0000000000000000
0x11a80f0: 0x0000000000000000 0x0000000000000000
0x11a8100: 0x0000000000000000 0x0000000000000000
0x11a8110: 0x0000000000000000 0x0000000000000000
0x11a8120: 0x0000000000000000 0x0000000000000000
0x11a8130: 0x0000000000000000 0x0000000000000000
0x11a8140: 0x0000000000000000 0x00000000000020ec1
```

chunk

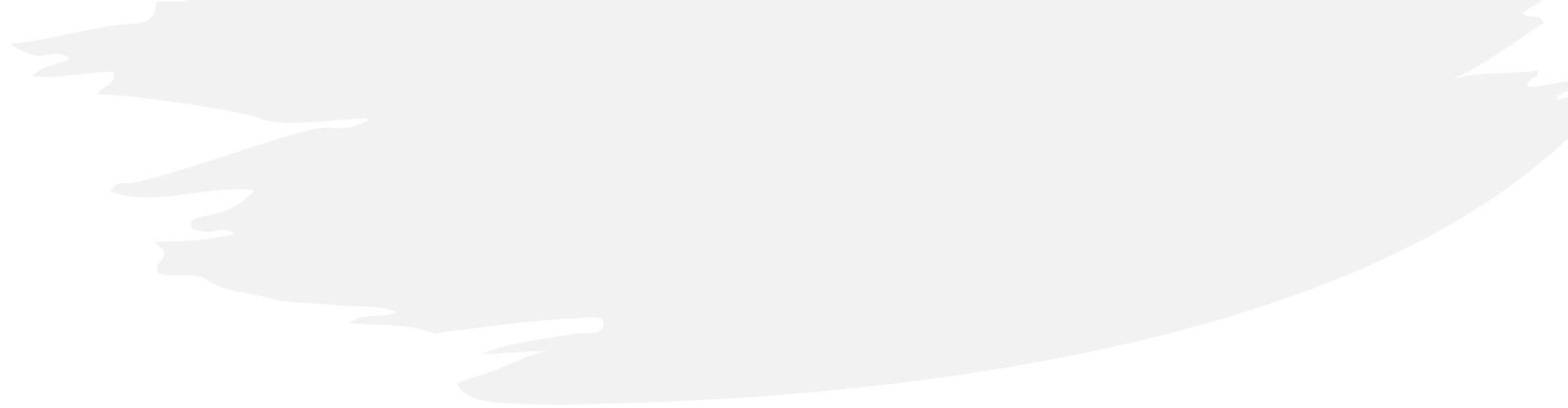
top chunk

unused

 : prev_size : size

삽질





QnA