Dynamic memory allocation

Dynamic allocation

```c
#include <stdio.h>

int main(void)
{
    char msg[0x100];
    printf("Input message: ");
    scanf("%s", msg);
    printf("Your message: %s", msg);

    return 0;
}
```

msg 배열의 크기가 0x100으로 고정되어 있음

만약, scanf에서 입력한 문자열의 길이가 0x100보다 크다면? => Stack Overflow

Dynamic allocation

```c
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    unsigned int len;
    printf("Input length of your message: ");
    scanf("%u", len);

    char *msg = (char *)malloc(len);
    printf("Input message: ");
    scanf("%s", msg);
    printf("Your message: %s", msg);

    return 0;
}
```

우선, 입력하고자 하는 길이를 입력 받고
그 길이만큼 malloc을 이용하여 메모리를 할당 받는다.

-> Stack Overflow 방지

만약 입력한 len보다 큰 길이의 문자열을 입력한다면?

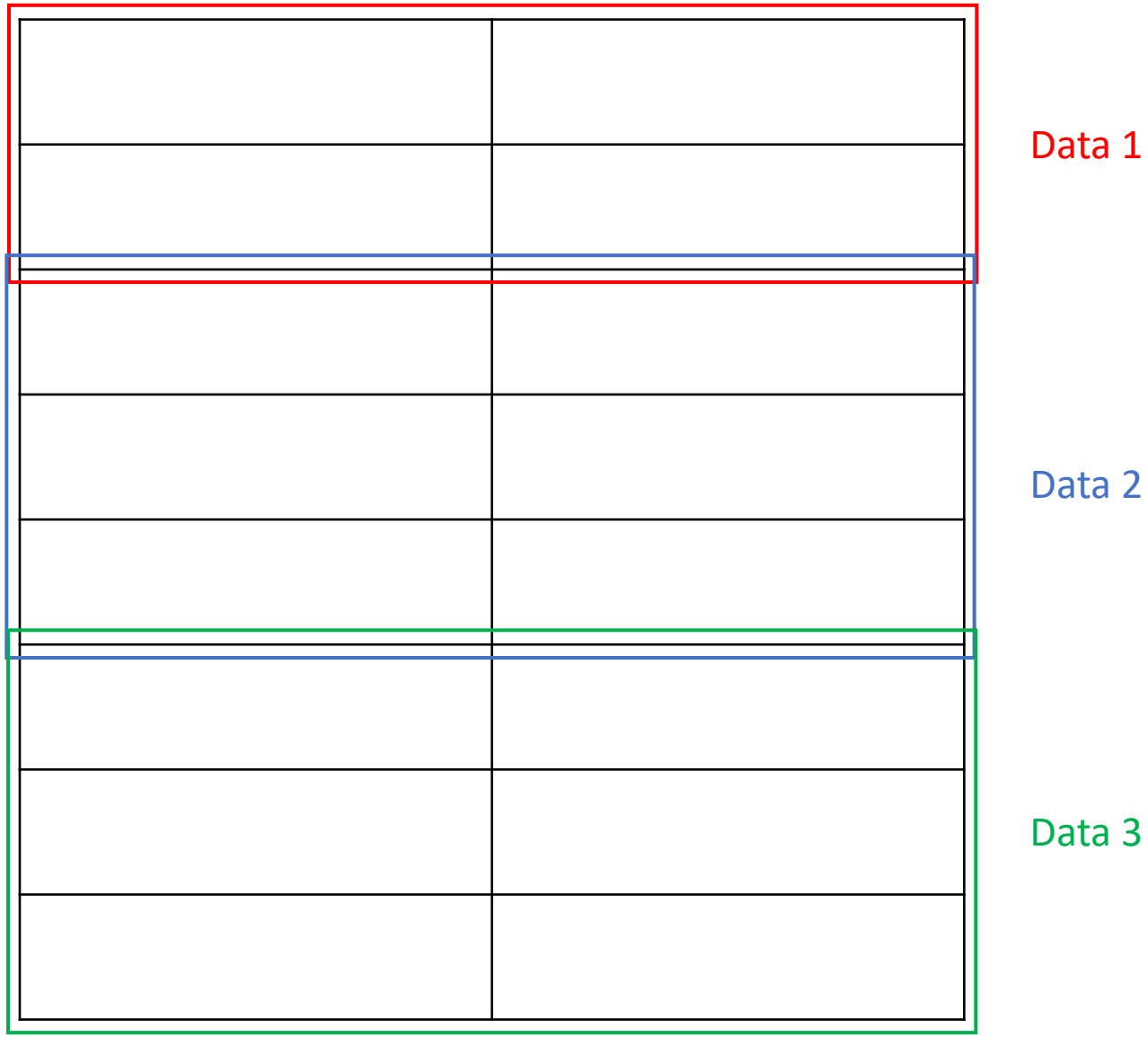
-> Heap Overflow

Heap

방법 1) Heap 영역에 순서대로 데이터를 저장

```c
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    char Data_1 = (char *)malloc(0x20);
    char Data_2 = (char *)malloc(0x30);
    char Data_3 = (char *)malloc(0x30);

    return 0;
}
```



Heap

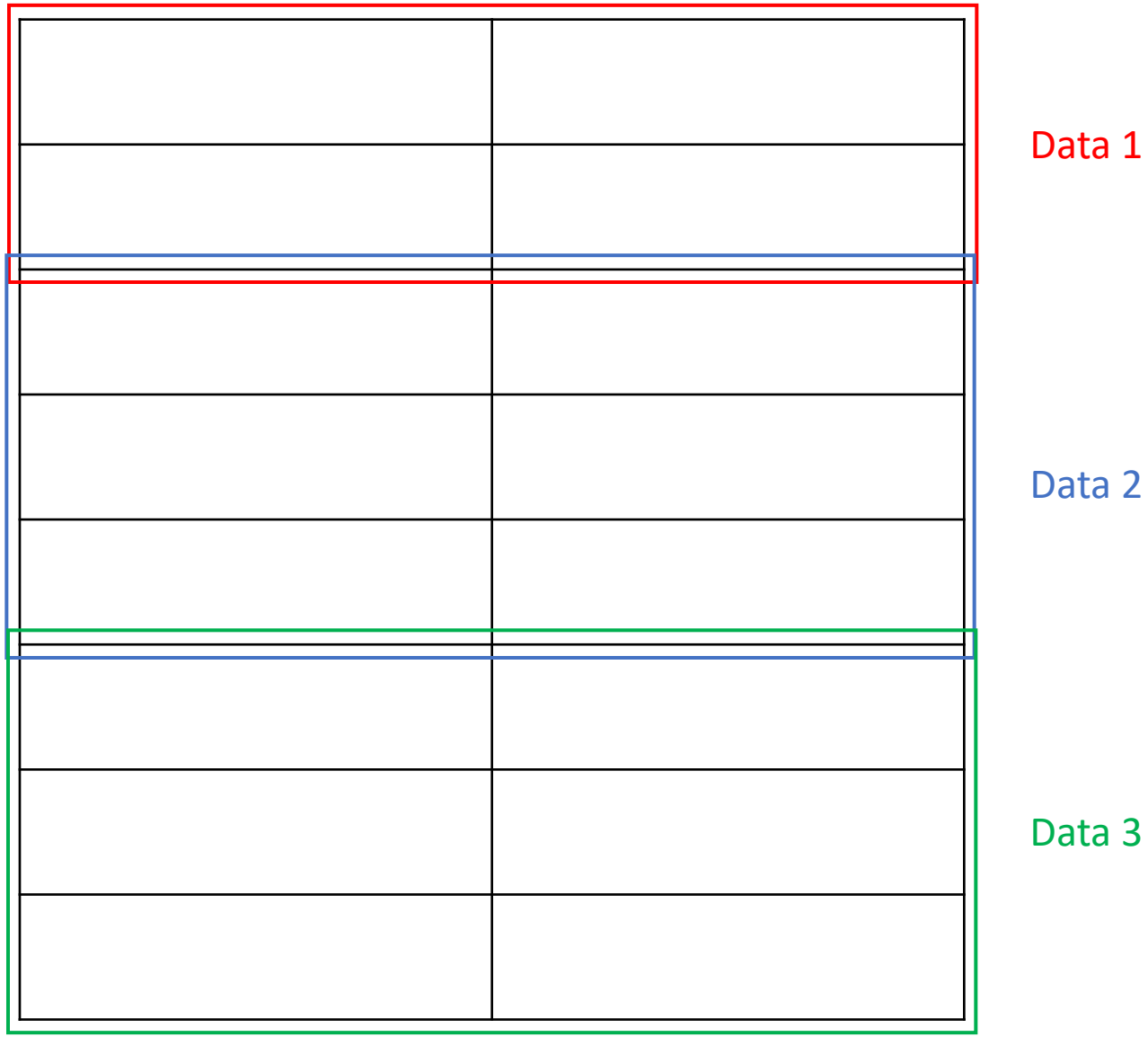
방법 1) Heap 영역에 순서대로 데이터를 저장

```c
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    char Data_1 = (char *)malloc(0x20);
    char Data_2 = (char *)malloc(0x30);
    char Data_3 = (char *)malloc(0x30);

    return 0;
}
```

* 문제점 *

어디서 어디까지가 Data 1인지, Data 2인지
그 경계가 명확하지 않음



Heap

	Data size
Data	Data
Data	Data

동적할당 시,
데이터 바로 앞에 데이터의 사이즈를 저장

-> Data의 범위 확인 가능

Heap

	Data size
Data	Data
Data	Data

Size 이외에 필요한 정보들

- Flag (현재 메모리의 상태 등을 나타내기 위함)
- 내 뒤에 존재하는 메모리 위치
- 내 앞에 존재하는 메모리 위치

Terms

- Arena
- heap
- chunk
- bin

Arena

- 한 Thread에서 동적 할당을 하면, heap 영역을 할당 받음.
- Thread별로 주어지는 heap 영역 -> Arena
- main_arena -> main thread의 heap 영역을 관리하기 위한 구조체

Arena

- malloc 직전

```
gdb-peda$ vmmap
Start              End                Perm      Name
0x0000555555554000 0x0000555555555000 r-xp      /home/jeongsuhwan/Desktop/heap/ex
0x0000555555754000 0x0000555555755000 r--p      /home/jeongsuhwan/Desktop/heap/ex
0x0000555555755000 0x0000555555756000 rw-p      /home/jeongsuhwan/Desktop/heap/ex
0x00007ffff79e4000 0x00007ffff7bcb000 r-xp      /lib/x86_64-linux-gnu/libc-2.27.so
0x00007ffff7bcb000 0x00007ffff7dcb000 ---p      /lib/x86_64-linux-gnu/libc-2.27.so
0x00007ffff7dcb000 0x00007ffff7dcf000 r--p      /lib/x86_64-linux-gnu/libc-2.27.so
0x00007ffff7dcf000 0x00007ffff7dd1000 rw-p      /lib/x86_64-linux-gnu/libc-2.27.so
0x00007ffff7dd1000 0x00007ffff7dd5000 rw-p      mapped
0x00007ffff7dd5000 0x00007ffff7dfc000 r-xp      /lib/x86_64-linux-gnu/ld-2.27.so
0x00007ffff7fdc000 0x00007ffff7fde000 rw-p      mapped
0x00007ffff7ff8000 0x00007ffff7ffb000 r--p      [vvar]
0x00007ffff7ffb000 0x00007ffff7ffc000 r-xp      [vdso]
0x00007ffff7ffc000 0x00007ffff7ffd000 r--p      /lib/x86_64-linux-gnu/ld-2.27.so
0x00007ffff7ffd000 0x00007ffff7ffe000 rw-p      /lib/x86_64-linux-gnu/ld-2.27.so
0x00007ffff7ffe000 0x00007ffff7fff000 rw-p      mapped
0x00007ffffffde000 0x00007ffffffff000 rw-p      [stack]
0xffffffffff600000 0xffffffffff601000 --xp      [vsyscall]
```

- malloc 직후

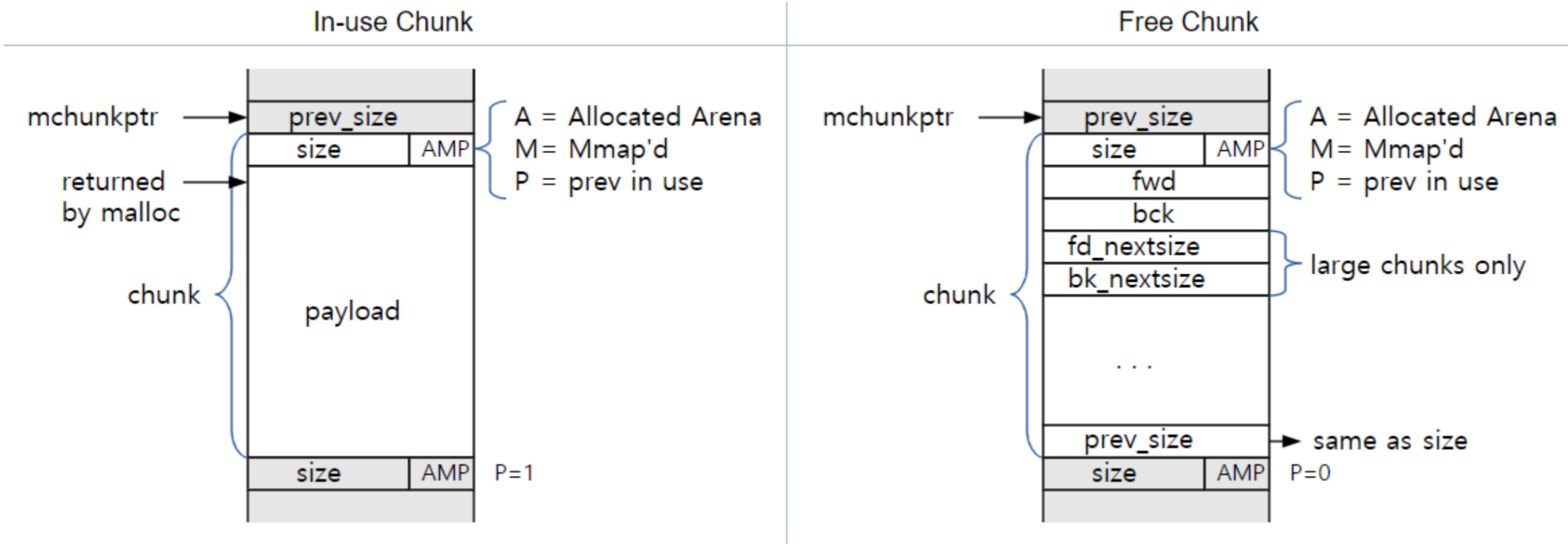
```
gdb-peda$ vmmap
Start              End                Perm      Name
0x0000555555554000 0x0000555555555000 r-xp      /home/jeongsuhwan/Desktop/heap/ex
0x0000555555754000 0x0000555555755000 r--p      /home/jeongsuhwan/Desktop/heap/ex
0x0000555555755000 0x0000555555756000 rw-p      /home/jeongsuhwan/Desktop/heap/ex
0x0000555555756000 0x0000555555777000 rw-p      [heap]
0x00007ffff79e4000 0x00007ffff7bcb000 r-xp      /lib/x86_64-linux-gnu/libc-2.27.so
0x00007ffff7bcb000 0x00007ffff7dcb000 ---p      /lib/x86_64-linux-gnu/libc-2.27.so
0x00007ffff7dcb000 0x00007ffff7dcf000 r--p      /lib/x86_64-linux-gnu/libc-2.27.so
0x00007ffff7dcf000 0x00007ffff7dd1000 rw-p      /lib/x86_64-linux-gnu/libc-2.27.so
0x00007ffff7dd1000 0x00007ffff7dd5000 rw-p      mapped
0x00007ffff7dd5000 0x00007ffff7dfc000 r-xp      /lib/x86_64-linux-gnu/ld-2.27.so
0x00007ffff7fdc000 0x00007ffff7fde000 rw-p      mapped
0x00007ffff7ff8000 0x00007ffff7ffb000 r--p      [vvar]
0x00007ffff7ffb000 0x00007ffff7ffc000 r-xp      [vdso]
0x00007ffff7ffc000 0x00007ffff7ffd000 r--p      /lib/x86_64-linux-gnu/ld-2.27.so
0x00007ffff7ffd000 0x00007ffff7ffe000 rw-p      /lib/x86_64-linux-gnu/ld-2.27.so
0x00007ffff7ffe000 0x00007ffff7fff000 rw-p      mapped
0x00007ffffffde000 0x00007ffffffff000 rw-p      [stack]
0xffffffffff600000 0xffffffffff601000 --xp      [vsyscall]
```

```
gdb-peda$ p main_arena
$1 = {
  mutex = 0x0,
  flags = 0x0,
  have_fastchunks = 0x0,
  fastbinsY = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0},
  top = 0x555555756270,
  last_remainder = 0x0,
```

Chunk

- 동적 할당을 할 때 할당되는 각각의 블록들을 Chunk라고 부름

Chunk


In-use Chunk
mchunkptr
prev_size
size
AMP
returned
by malloc
chunk
payload
size
AMP
P=1
A = Allocated Arena
M = Mmap'd
P = prev in use
Free Chunk
mchunkptr
prev_size
size
AMP
fwd
bck
fd_nextsize
bk_nextsize
chunk
large chunks only
. . .
prev_size
same as size
size
AMP
P=0
A = Allocated Arena
M = Mmap'd
P = prev in use

Free Chunk

- fd : 다음 free chunk를 가리킴
- bk : 이전 free chunk를 가리킴

- fd_nextsize : large bin에서, 현재 청크보다 작은 사이즈의 chunk를 가리킴
- bk_nextsize : large bin에서, 현재 청크보다 큰 사이즈의 chunk를 가리킴

Flag

- A : 현재 청크가 main_arena에서 관리되지 않을 경우 On
- M : mmap으로 할당된 경우 On
- P : 이전 청크가 free되었을 경우 On

Bin

- Free된 chunk들을 관리하기 위한 연결 리스트

- Chunk의 크기에 따라 종류가 있음(fast bin, small bin, large bin …)

- Fastbin : ~0x80 , LIFO , single linked list
- Unsorted bin: 0xc0 ~ , FIFO , double linked list
- small bin : ~0x3f0 , FIFO , double linked list
- large bin : 0x400 ~ , FIFO , double linked list

Tcache

- glibc 2.27 ~ (Ubuntu 18.04 LTS ~)
- single linked list
- size 별로 구분됨
- ~ 0x410 byte
- 각 size마다 7개까지만 저장할 수 있음

pwngdb

- heap 디버깅을 위한 gdb plugin

- $ git clone https://github.com/scwuaptx/Pwngdb.git
- $ cp ~/Pwngdb/.gdbinit ~/

pwngdb

- heapinfo
- parseheap
- magic

example

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

int main(void)
{
    char *chunk[10];
    for (int i = 0; i < 10; i++)
        chunk[i] = (char *)malloc(0x20);

    for (int i = 0; i < 10; i++)
        free(chunk[i]);

    return 0;
}
```

example 2

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

int main(void)
{
    char *a[10];
    for (int i = 0; i < 10; i++)
        a[i] = (char *)malloc(0x20);
    for (int i = 0; i < 9; i++)
        free(a[i]);
    char *c = (char *)malloc(0x500);

    return 0;
}
```

Assignment

- https://sourceware.org/glibc/wiki/MallocInternals 정리하기