Лабораторная работа

«изучение менеджера памяти OS/языка»

Выполним следующий код:

*#include* "stdio.h"

*#include* "stdint.h"

*#include* "stdlib.h"

void allocate(uint64\_t *bytes*) {

const int n = 4;

void\* mem[n];

printf("allocate %ld bytes:\n", bytes);

*for* (int i = 0; i < n; i++)

mem[i] = malloc(bytes);

*for* (int i = 0; i < n; i++)

printf("%p ", mem[i]);

printf("\n");

printf("diff: ");

*for* (int i = 1; i < n; i++)

printf("%ld ", mem[i] - mem[i - 1]);

printf("\n");

}

int main() {

allocate(1);

allocate(16);

allocate(32);

allocate(48);

allocate(64);

allocate(1024);

allocate(1024 \* 8);

allocate(1024 \* 128);

allocate(1024 \* 1024);

uint64\_t mb = 1024 \* 1024 \* 1024;

allocate(mb);

allocate(2 \* mb);

*return* 0;

}

Результат:

mvtsarev@dev-vm:~/asm$ ./exe

allocate 1 bytes:

0x5559c60946b0 0x5559c60946d0 0x5559c60946f0 0x5559c6094710

diff: 32 32 32

allocate 16 bytes:

0x5559c6094730 0x5559c6094750 0x5559c6094770 0x5559c6094790

diff: 32 32 32

allocate 32 bytes:

0x5559c60947b0 0x5559c60947e0 0x5559c6094810 0x5559c6094840

diff: 48 48 48

allocate 48 bytes:

0x5559c6094870 0x5559c60948b0 0x5559c60948f0 0x5559c6094930

diff: 64 64 64

allocate 64 bytes:

0x5559c6094970 0x5559c60949c0 0x5559c6094a10 0x5559c6094a60

diff: 80 80 80

allocate 1024 bytes:

0x5559c6094ab0 0x5559c6094ec0 0x5559c60952d0 0x5559c60956e0

diff: 1040 1040 1040

allocate 8192 bytes:

0x5559c6095af0 0x5559c6097b00 0x5559c6099b10 0x5559c609bb20

diff: 8208 8208 8208

allocate 131072 bytes:

0x799c14d9d010 0x799c14d7c010 0x799c14d5b010 0x799c14d3a010

diff: -135168 -135168 -135168

allocate 1048576 bytes:

0x799c14c39010 0x799c148ff010 0x799c147fe010 0x799c146fd010

diff: -3383296 -1052672 -1052672

allocate 1073741824 bytes:

0x799bd4600010 0x799b94400010 0x799b54200010 0x799b14000010

diff: -1075838976 -1075838976 -1075838976

allocate 2147483648 bytes:

0x799a93e00010 0x799a13c00010 0x799993a00010 0x799913800010

diff: -2149580800 -2149580800 -2149580800

mvtsarev@dev-vm:~/asm$

Память выделяется glibc:

**mvtsarev@dev-vm**:**~**$ ldd --version

ldd (Ubuntu GLIBC 2.39-0ubuntu8.3) 2.39

Наблюдаем следующее

1. При выделении 16 байт, выделяется блоками 32 байта; 32 байта – 48; 64 – 80; Из чего делаем вывод, что на каждое выделение добавляется дополнительно 16 байт – значит есть заголовок.

Действительно, в [исходниках](https://elixir.bootlin.com/glibc/glibc-2.39/source/malloc/malloc.c#L1283) (мы находим следующее:  
[1283](https://sourceware.org/git?p=glibc.git;a=blob;f=malloc/malloc.c;h=f16f89d2834f05abc3bc9e45fff182ab70bb86b3;hb=HEAD#l1283) */\* The chunk header is two SIZE\_SZ elements, but this is used widely, so*

[1284](https://sourceware.org/git?p=glibc.git;a=blob;f=malloc/malloc.c;h=f16f89d2834f05abc3bc9e45fff182ab70bb86b3;hb=HEAD#l1284)  *we define it here for clarity later. \*/*

[1285](https://sourceware.org/git?p=glibc.git;a=blob;f=malloc/malloc.c;h=f16f89d2834f05abc3bc9e45fff182ab70bb86b3;hb=HEAD#l1285) #define CHUNK\_HDR\_SZ (2 \* SIZE\_SZ)

[Где SIZE\_SZ](https://elixir.bootlin.com/glibc/glibc-2.40.9000/source/sysdeps/generic/malloc-size.h#L57):

*/\* The corresponding word size. \*/*

#define [**SIZE\_SZ**](https://elixir.bootlin.com/glibc/glibc-2.40.9000/C/ident/SIZE_SZ) (sizeof ([**INTERNAL\_SIZE\_T**](https://elixir.bootlin.com/glibc/glibc-2.40.9000/C/ident/INTERNAL_SIZE_T)))

На x64 системе 2 раза по 8 байт как раз 16 байт.

1. При выделении <= 16 байт, мы получаем блок в 32 байта, 16 байт данных + 16 байт заголовка. При выделении 17 - 32 байт – получаем 32 байта данных + 16 байт заголовка = 48 байт. Следовательно, гранулярность выделения памяти – 16 байт
2. При выделении областей памяти размером менее 128 кб, можно заметить, что все адреса находятся «близко» друг к другу, однако начиная с 128кб, адреса «резко перепрыгивают» в «далекую» область памяти. 128кб, 1мб и 2мб были выделены близко друг к другу, но далеко от меньших объемов. Следовательно, присутствует механизм предотвращения фрагментации, и блоки разных размеров выделяются в разных участках памяти.