堆

堆动态分配相关函数: malloc与 free 系列:

在Debug下, 堆空间默认填充 0xcd

在Debug下,释放的空间默认填充 0xdd 或者 0xfeee

堆块在内存中的表现

Debug

在堆块中有控制信息 堆指针减0x20

- 前一个堆的地址,为0表示第一个
- 后一个堆的地址,为0表示最后一个
- 调用堆的文件名称指针
- 代码行数
- 空间大小,不含附加数据 (高版本跟堆类型互换了位置)
- 堆类型 (高版本跟空间大小互换了位置)
- 第多少次申请,从程序开始运行时就计算
- 上溢标志
- 堆数据
- 下溢标志

Release

表一般在堆指针的页面起始位置,即0xxxx00处 在堆块中没有其他额外信息

设计规范

```
malloc source
if malloc is error:
    Error Proc
    goto EXTI
...

EXIT:
    free source
```

分析堆块结构

部分源码

```
char *ptr1 = (char *)_malloc_dbg(5, _NORMAL_BLOCK, __FILE__, __LINE__);
for (int i = 0; i < 5; i++) {
    ptr1[i] = 'a' + i;
}

int *ptr2 = (int *)malloc(sizeof(int) * 6);
for (int i = 0; i < 6; i++) {
    ptr2[i] = i + 1;
}

int *ptr3 = (int *)calloc(6, sizeof(int));
for (int i = 0; i < 6; i++) {
    ptr3[i] = i + 1;
}
...</pre>
```

分析

malloc dbg

红线部分即堆块中的各个成员结构

```
0x005F6080 f8 5f 5f 00 00 00 00 00 30 7b 41 00 07 00 00 00
                                                             ?__....0{A.....
           01 00 00 00 05 00 00 00 4c 00 00 00 fd fd fd fd
                                                             .....L...?????
0x005F60A0
           cd cd cd cd cd fd fd fd 00 63 00 72 00 6f 00
                                                             ????????.c.r.o.
0x005F60B0
           e3 ec 52 b3 b1 40 00 0c 43 00 3a 00 5c 00 57 00
                                                             ??R??@..C.:.\.W.
           69 00 6e 00 64 00 6f 00 77 00
0x005F60C0
                                          73 00 5c
                                                   00
                                                      53 00
                                                             i.n.d.o.w.s.\.S.
           59 00 53 00 54 00 45 00 4d 00 33 00 32 00
0x005F60D0
                                                     5c
                                                        00
                                                             Y.S.T.E.M.3.2.\.
0x005F60E0
           75 00 63 00 72 00 74 00 62 00 61 00 73 00 65 00
                                                             u.c.r.t.b.a.s.e.
           64 00 2e 00 64 00 6c 00 6c 00 00 00 6c 00 00 00
                                                             d...d.l.l...l...
0x005F60F0
            e4 ec 52 b4 bc 40 00 08 80 fe 8e 00 03 00 00 00
                                                             ??R??@..€??.....
```

- 0x005f5ff8
 - 前一个堆块在0x005f5ff8地址处
- 0x00000000 后一个堆块,0表示当前这个堆块就是最后一个
- 0x00417b30 文件名, 标明调用堆分配是哪个文件

```
▼ C 列: 自动
地址: 0x00417B30
x00417B30 67 3a 5c 74 6d 70 5c 70 72 6f 6a 65 63 74 31 5c
                                                             g:\tmp\project1\
0x00417B40 70 72 6f 6a 65 63 74 31 5c d4 b4 2e 63 00 00 00
                                                             project1\??.c...
0x00417B50 00 00 00 00 70 61 75 73 65 00 00 00 18 7c 41 00
                                                               . . . . pause . . . . A.
0x00417B60
            28 7d 41 00 80 7e 41 00 a4 7e 41 00
                                                e4 7e 41 00
                                                              (}A.€~A.?~A.?~A.
0x00417B70
            18 7f
                  41
                     00
                        01 00
                              00 00 00 00
                                          00
                                             00
                                                01 00
                                                      00
                                                          00
                                                              . . A . . . . . . . . . . . . .
                                                              .....Stac
           01 00 00 00 01 00 00 00 01 00 00
                                                    74 61 63
0x00417B80
                                             00
                                                53
0x00417B90
           6b 20 61 72 6f 75 6e 64 20 74 68 65 20 76 61 72
                                                              k around the var
                                                              iable '.' was co
           69 61 62 6c 65 20 27 00 27 20 77 61 73 20 63 6f
0x00417BA0
0x00417BB0
           72 72 75 70 74 65 64 2e 00 00 00 00 54 68 65 20
                                                              rrupted....The
                                                              variable ' 'is
            76 61 72 69 61 62 6c 65 20 27 00 00 27 20 69 73
MYMM417RCM
```

• 0x00000007 行号, 标明改堆分配函数在文件的第几行

```
char *ptr1 = (char *)_malloc_dbg(5, _NORMAL_BLOCK, __FILE__, __LINE__);

for (int i = 0; i < 5; i++) { 巨用时间 <= 1ms
    ptr1[i] = 'a' + i;
}

int *ptr2 = (int *)malloc(sizeof(int) * 6);

for (int i = 0; i < 6; i++) {
    ptr2[i] = i + 1;
}

int *ptr3 = (int *)calloc(6, sizeof(int));

for (int i = 0; i < 6; i++) {
    ptr2[i] = i + 1;
}

ptr2[i] = i + 1;
}
```

• 0x00000001

请求类型,这里的_NORMAL_BLOCK的值为1

0x00000005

请求的内存块的大小,这里申请了5字节

0x0000004c

申请次数

0xfdfdfdfd

上溢标志

0xcd

请求的内存块,接着执行后续的代码可知

```
▼ C 列:自动
地址: 0x005F6080
                                                           ?__....0{A.....
0x005F6080 f8 5f 5f 00 00 00 00 00 30 7b 41 00 07 00 00 00
0x005F6090 01 00 00 00 05 00 00 00 4c 00 00 00 fd fd fd fd
                                                           ....L...?????
0x005F60A0 61 62 63 64 65 fd fd fd fd 00 63 00 72 00 6f 00
                                                          abcde????.c.r.o.
0x005F60B0 e3 ec 52 b3 b1 40 00 0c 43 00 3a 00
                                              5c 00 57 00
                                                           ??R??@..C.:.\.W.
0x005F60C0 69 00 6e 00 64 00 6f 00 77 00 73 00 5c 00 53 00
                                                           i.n.d.o.w.s.\.S.
0x005F60D0 59 00 53 00 54 00 45 00 4d 00 33 00 32 00 5c 00
                                                           Y.S.T.E.M.3.2.\.
0x005F60E0 75 00 63 00 72 00 74 00 62 00 61 00 73 00 65 00
                                                           u.c.r.t.b.a.s.e.
                                                           d...d.l.l...l...
0x005F60F0 64 00 2e 00 64 00 6c 00 6c 00 00 00 6c 00 00 00
0x005F6100 e4 ec 52 b4 bc 40 00 08 80 fe 8e 00 03 00 00 00
                                                            ??R??@..€??.....
MYMMSF611M M1 MM MM MM M3 MM MM MM C2 C2 h9 d2 M5 MM MM MM
                                                                   7777
内存1 内存2 内存3 内存4
```

• Oxfdfdfdfd 下溢标志

malloc

红线部分即堆块中的各个成员结构

```
地址: 0x005F5B20
                                           ▼ 🖒 列: 自动
0x005F5B30 01 00 00 00 18 00 00 00 4d 00 00 00 fd fd fd fd
                                                  ........M...?????
????????????????
0x005F5B50 cd cd cd cd cd cd cd fd fd fd 90 60 14 10
                                                  ???????????.`
0x005F5B60
         e0 ec 53 b1 bt 40 00 00 d8 59 5f 00 90 05 5f 00
                                                  ??5??@..?Y_.?._.
0x005F5B70 02 00 00 00 50 81 07 10 6c 62 14 10 00 40 17 00
                                                  ....P?..lb...@..
0x005F5B80 00 00 00 00 ff ff ff ff 00 00 00 00 75 00 63 00
0x005F5B90 72 00 74 00 62 00 61 00 73 00 65 00 64 00 2e 00
                                                  r.t.b.a.s.e.d...
0x005F5BA0 64 00 6c 00 6c 00 00 00 ff ec 52 af bf 40 00 08
                                                 d.l.l....?R??@...
MxMM5E5RRM 4c Mc a6 77 7M 5a 5f MM 54 Mc a6 77 78 5a 5f MM I 2Wm7 T 2Wx7
内存1 内存2 内存3 内存4
```

0x005f6080

前一个堆块在0x005f6080地址处,刚好是我们上次调用堆分配所得到的堆块地址

0x00000000

后一个堆块,0表示当前这个堆块就是最后一个

0x00000000

文件名, 标明调用堆分配是哪个文件, 可见 malloc 函数默认给空

0x00000000

行号,标明改堆分配函数在文件的第几行,可见 malloc 函数默认给空

• 0x00000001

请求类型

• 0x0000018

请求的内存块的大小,这里申请了24字节

0x0000004d

申请次数,在上次的基础上加了1

0xfdfdfdfd

上溢标志

0xcd

请求的内存块,接着执行后续的代码可知

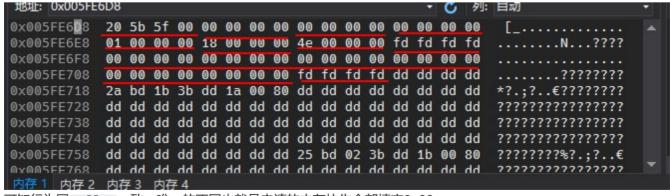
```
▼ C 列: 自动
地址: 0x005F5B20
01 00 00 00 18 00 00 00 4d 00 00 00 fd fd fd fd
0x005F5B30
                                                       .......M...?????
0x005F5B40
          01 00 00 00 02 00 00 00 03 00 00 00 04 00 00 00
0x005F5B50 05 00 00 00 06 00 00 fd fd fd fd 00 60 14 10
                                                         .....?????.`..
0x005F5B60 e0 ec 53 b1 bf 40 00 00 d8 59 5f 00 90 05 5f 00
                                                       ??S??@..?Y_.?._.
0x005F5B70 02 00 00 00 50 81 07 10 6c 62 14 10 00 40 17 00
                                                       ....P?..lb...@..
0x005F5B80 00 00 00 00 ff
                        ff
                           ff ff 00 00 00 00 75 00 63 00
                                                       ....u.c.
0x005F5B90
         72 00 74 00 62 00 61 00 73 00 65 00 64 00
                                                 2e 00
                                                       r.t.b.a.s.e.d...
          64 00 6c 00 6c 00 00 00 ff
                                   ec 52 af
0x005F5BA0
                                           bf
                                              40 00 08
                                                       d.l.l....?R??@..
MxMM5F5RRM 4c Mc a6 77 7M 5a 5f MM 54 Mc a6 77 78 5a 5f MM
                                                       1 7wn7 T 7wx7
```

0xfdfdfdfd

下溢标志

calloc

红线部分即堆块中的各个成员结构



可知行为同 malloc 一致,唯一的不同也就是申请的内存块先全部填充0x00

后一个堆块

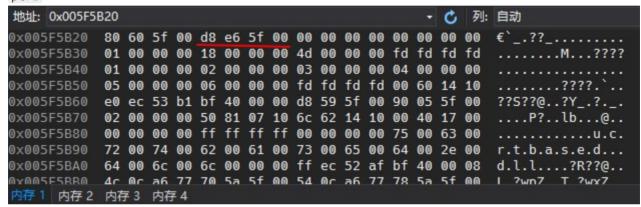
如此,我们三个堆分配函数执行完毕,从中验证了前一个堆块的地址是正确的,接着我们在转而去前面两个堆块看看后继的堆块地址是否正确

• ptr1

```
地址: 0x005F6080
            f8 5f 5f 00 20 5b 5f 00 30 7b 41 00 07 00 00 00
0x005F6080
                                             00
           01 00 00 00 05 00 00 00 4c 00
                                          00
                                                fd fd
0x005F6090
                                                       fd fd
0x005F60A0
           61 62 63 64 65 fd fd fd fd 00
                                          63 00 72 00 6f
                                                         00
0x005F60B0
            e3 ec 52 b3 b1 40
                              00 Oc 43 00
                                          3a 00 5c 00 57 00
0x005F60C0
            69 00 6e 00 64 00
                              6f 00 77
                                       00
                                          73 00
                                                5c 00
0x005F60D0
            59 00 53
                     00 54 00
                              45 00 4d 00
                                          33 00
                                                32 00
            75 00 63
                        72 00
0x005F60E0
                     00
                              74 00 62 00
                                                73 00
                                                      65
                                          61 00
0x005F60F0
           64 00
                  2e
                     00 64 00
                              6c 00
                                    6c 00
                                          00
                                             00
                                                6c
                                                   00
0x005F6100
           e4 ec 52 b4 bc 40 00 08 80 fe 8e 00
                                                03 00 00
0x005F6110
           01 00 00 00 03 00 00 00 c2 c2 b9 d2 05 00 00 00
内存1 内存2
           内存3 内存4
```

可以看出 0x005f5b20 是 ptr2 堆块的地址

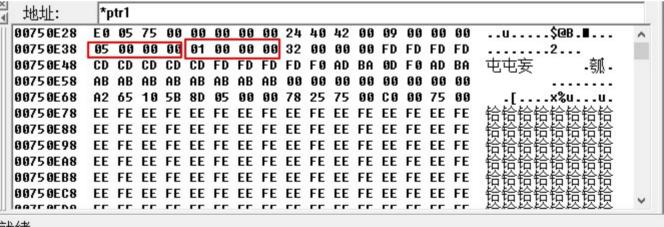
• ptr2



可以看出 0x005f6ed8 是 ptr3 堆块的地址

在VC6下

vc6与高版本的VS的不同之处在于,请求类型和空间大小这两个存储位置是互换的



就绪