内存管理API使用说明

一、文件说明

共两个文件mymalloc\_t.c和mymalloc\_t.h，两个文件需放在同一个目录下。

1 .h文件中常量定义

（1）大内存块和小内存块的大小、数量（用户可以根据需求对数值大小进行修改，并重新编译.c文件）。

|  |  |  |
| --- | --- | --- |
| 常量名 | 默认值 | 含义 |
| SMALLSIZE | 50 | 小内存块数据域的大小（单位：字节） |
| SMALLNUM | 20 | 小内存块的个数 |
| BIGSIZE | 1000 | 大内存块数据域的大小（单位：字节） |
| BIGNUM | 4 | 大内存块的个数 |

1. 申请内存块类型（根据需要的内存空间，判断分配类型）

|  |  |  |
| --- | --- | --- |
| 常量名 | 值 | 使用场景（申请一块m字节的内存时） |
| SMALLAREA | 1 | 从已申请的一整块内存中申请一个小内存块 |
| BIGAREA | 2 | 从已申请的一整块内存中申请一个大内存块 |
| HEAPAREA | 3 | 从heap中申请内存 |

1. （各）内存块的状态

|  |  |  |
| --- | --- | --- |
| 常量名 | 值 | 含义 |
| BLOCKAVAILABLE | 0 | 该内存块可用 |
| BLOCKBUSY | 1 | 该内存块已被占用 |

1. 内存占用状态

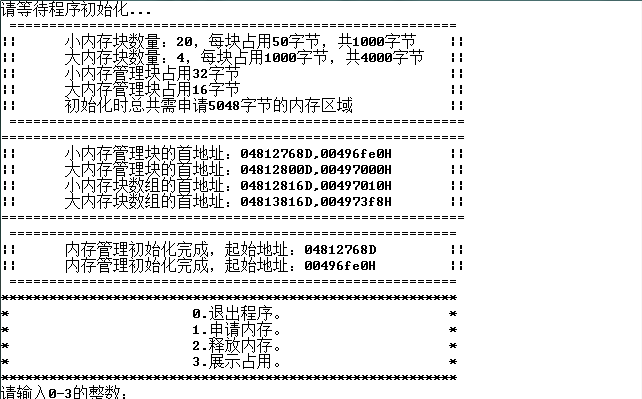
|  |  |  |
| --- | --- | --- |
| 常量名 | 值 | 含义 |
| SMALL\_FREE | 0 | 有可用的小内存块 |
| SMALL\_FULL | 1 | 小内存块全部被占用 |
| BIG\_FULL | 2 | 大内存块全部被占用 |

1. h文件中数据结构定义

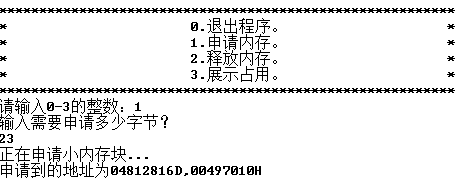
共定义了小内存块结构、大内存块结构、小内存块管理结构和大内存块管理结构、从堆申请的内存信息结构体，共五种结构体。

详细注释见mymalloc\_t.h文件。

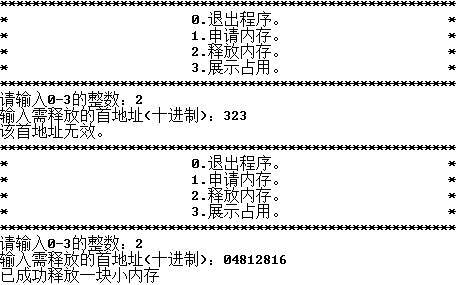
二、使用样例



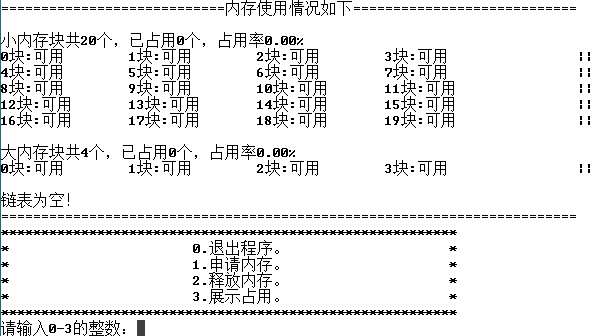
1. 用户输入0则退出程序。
2. 用户输入1，提示用户输入“需要多少字节的内存”，并进行申请，返回申请到的内存的首地址。



3. 用户输入2，进行内存释放，输入要释放的内存的首地址。程序进行校验，如果合法则释放，提示用户释放成功，否则提示地址不合法。



4.用户输入3，展示内存占用情况（如下图）



其中链表为记录从heap申请的内存块的首地址的链表。

三、函数说明

1. void main(void) /\*主函数：用户循环输入控制等；\*/

2. int\* mem\_init(void) /\*内存空间初始化，返回内存空间首地址\*/

3. void calculate\_address(int\* head\_add)/\*计算每一块内存块的首地址，head\_add为申请的一大块内存的首地址,其中head\_add即传入mem\_init的返回值\*/

4. int\* malloc\_by\_type(int type,int size) /\*申请不同类型的内存块，根据type区分，size为需要的内存大小，该函数由mem\_malloc进行调用，返回申请到内存块的首地址\*/

5. int\* mem\_malloc(int size)/\*计算申请内存块的类型，计算好后调用malloc\_by\_type，返回malloc\_by\_type的返回值\*/

6. void mem\_free(int free\_add)/\*释放一个内存块，free\_add即为要释放的首地址\*/

7. void traverse\_list()/\*遍历保存heap申请到的内存信息的链表\*/

8. void free\_list()/\*释放保存heap申请到的内存信息的链表\*/

9. void mem\_show()/\*展示内存占用情况\*/

10. void print\_menu()/\*打印用户菜单\*/

1. 注意事项

在Tornado上运行时，输入中一定不要点击Backspace按键，否则会陷入死循环。