

## 知识点1【动态内存申请的API】(重要)

### 1、malloc申请 堆区空间

特点：malloc申请的堆区空间 不自动清0

案例1：申请int空间

### 2、free释放堆区空间

### 3、memset内存设置函数

案例2：动态数组

### 4、calloc 函数

案例1、简易 的动态数组

### 5、realloc函数 追加空间

## 知识点2【内存泄漏】(了解)

案例1：

案例2：

## 知识点3【内存的回顾】(了解)

# 知识点1【动态内存申请的API】(重要)

## 1、malloc申请 堆区空间

```
1 #include <stdlib.h>
2 void *malloc(size_t size);
3 size表示申请的空间字节数
4 函数的返回值：
5 成功：返回值空间起始地址
6 失败：NULL
```

特点：malloc申请的堆区空间 不自动清0

案例1：申请int空间

```

#include <stdio.h>
#include <stdlib.h>
void test01()
{
    //空间申请
    int *p = NULL;
    p = (int *)malloc(sizeof(int));
    if (p == NULL)
    {
        return;
    }

    *p = 100;

    printf("*p = %d\n", *p);

    //空间释放
    free(p);
}

```

```

edu@edu: ~/work/c/day06
edu@edu: ~/work/c/day06$ ls
00_code.c
edu@edu: ~/work/c/day06$ sudo gcc 00_code.c
[sudo] edu 的密码:
edu@edu: ~/work/c/day06$ ./a.out
*p = 100
edu@edu: ~/work/c/day06$ _

```

## 2、free释放堆区空间

```

1  #include <stdlib.h>
2  void free(void *ptr);
3  ptr需要释放的堆区空间的起始地址

```

## 3、memset内存设置函数

```

1  #include <string.h>
2  void *memset(void *s, int c, size_t n);
3  s就是空间的起始地址
4  n就是空间的字节宽度
5  c空间中每个字节 填充的值

```

### 案例2：动态数组

```

1  #include <string.h>
2  void test02()
3  {
4      int n = 0;
5      printf("请输入int元素的个数:");
6      scanf("%d", &n);
7
8      //根据元素的个数申请空间
9      int *p = NULL;
10     p = (int *)malloc(n * sizeof(int));

```

```

11     if (p == NULL)
12     {
13         return;
14     }
15     //将堆区空间清0
16     memset(p, 0, n * sizeof(int));
17
18     //获取键盘输入
19     int i = 0;
20     for (i = 0; i < n; i++)
21     {
22         scanf("%d", p + i);
23     }
24
25     //遍历数组元素
26     for (i = 0; i < n; i++)
27     {
28         printf("%d ", p[i]); /*(p+i)
29     }
30     printf("\n");
31
32     //释放空间
33     free(p);
34
35     return;
36 }

```

```

C:\ edu@edu: ~/work/c/day06
edu@edu: ~/work/c/day06$ sudo gcc 00_code.c
edu@edu: ~/work/c/day06$ ./a.out
请输入int元素的个数:5
1 2 3 4 5
1 2 3 4 5
edu@edu: ~/work/c/day06$ _

```

## 4、calloc 函数

```

1 #include <stdlib.h>

```

```
2 void *calloc(size_t nmemb, size_t size);
3 nmemb: 内存的块数
4 size: 每一块的字节数
5 返回值: 成功为堆区空间起始地址 失败为NULL
6 calloc会对申请的空间 自动清0
```

```
1 malloc(n*sizeof(int));
2 calloc(n, sizeof(int));
```

## 案例1、简易 的动态数组

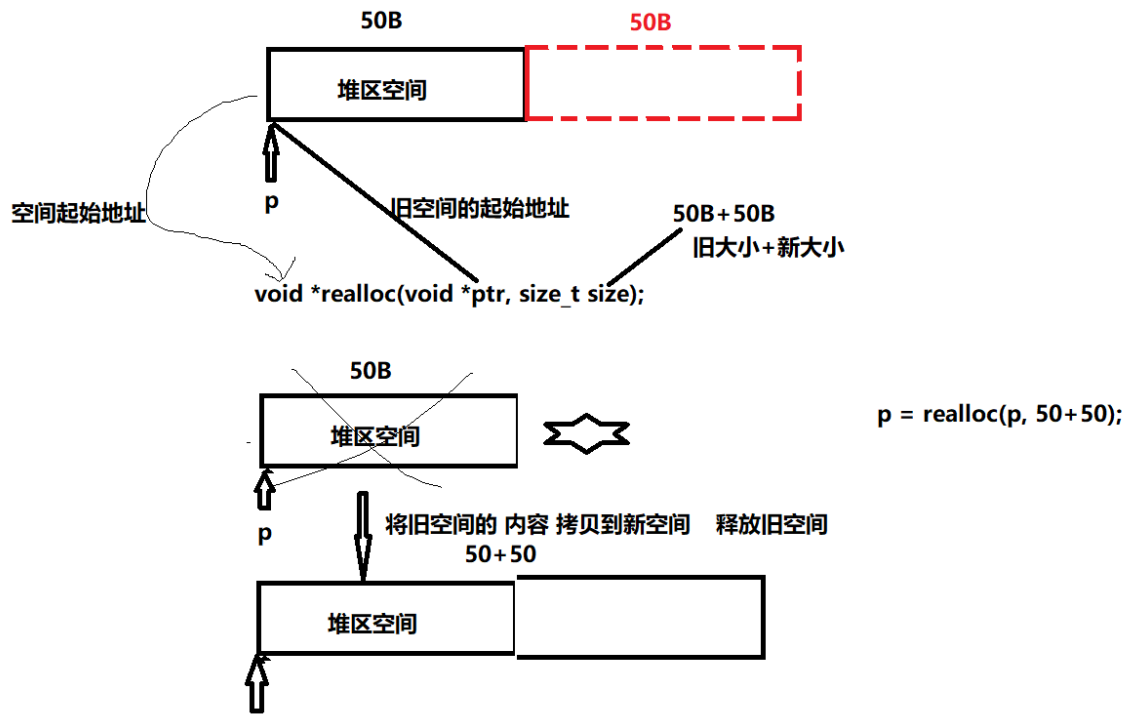
```
1 void *get_addr(int n, int elem_size)
2 {
3     return calloc(n, elem_size);
4 }
5 void input_int_array(int *p, int n)
6 {
7     int i = 0;
8     for (i = 0; i < n; i++)
9     {
10         scanf("%d", p + i);
11     }
12 }
13 void print_int_array(int *p, int n)
14 {
15     int i = 0;
16     for (i = 0; i < n; i++)
17     {
18         printf("%d ", p[i]);
19     }
20     printf("\n");
21 }
22 void test03()
23 {
24     int n = 0;
25     printf("请输入int元素的个数:");
26     scanf("%d", &n);
27
28     //根据元素个数 申请空间
29     int *p = NULL;
30     p = (int *)get_addr(n, sizeof(int));
31     if (p == NULL)
```

```
32     {
33         return;
34     }
35
36     //获取键盘输入
37     input_int_array(p, n);
38
39     //遍历
40     print_int_array(p, n);
41
42     //释放空间
43     free(p);
44 }
```

```
edu@edu: ~/work/c/day06
edu@edu: ~/work/c/day06$ sudo gcc 00_code.c
[sudo] edu 的密码:
edu@edu: ~/work/c/day06$ ./a.out
请输入int元素的个数:5
1 2 3 4 5 6 7 8 9
1 2 3 4 5
edu@edu: ~/work/c/day06$
```

## 5、realloc函数 追加空间

```
1 #include <stdlib.h>
2 void *realloc(void *ptr, size_t size);
```



必须使用指针变量 获取realloc的返回值.

```

1 void test04()
2 {
3     int n = 0;
4     printf("请输入int元素的个数:");
5     scanf("%d", &n);
6
7     //根据元素的个数申请空间
8     int *p = NULL;
9     p = (int *)calloc(n, sizeof(int));
10    if (p == NULL)
11    {
12        return;
13    }
14
15    //获取键盘输入
16    int i = 0;
17    for (i = 0; i < n; i++)
18    {
19        scanf("%d", p + i);
20    }
21
22    //遍历数组元素
23    for (i = 0; i < n; i++)
24    {

```

```
25     printf("%d ", p[i]); /*(p+i)
26 }
27 printf("\n");
28
29 printf("请输入你要新增的元素个数:");
30 int new_n = 0;
31 scanf("%d", &new_n);
32
33 //追加空间
34 p = (int *)realloc(p, (n + new_n) * sizeof(int));
35
36 printf("请输入%d个新增元素:", new_n);
37 //请输入新增的元素
38 for (i = n; i < n + new_n; i++)
39 {
40     scanf("%d", p + i);
41 }
42
43 //遍历数组元素
44 for (i = 0; i < n + new_n; i++)
45 {
46     printf("%d ", p[i]); /*(p+i)
47 }
48 printf("\n");
49
50 free(p);
51 }
```

edu@edu: ~/work/c/day06

```
edu@edu:~/work/c/day06$ sudo gcc 00_code.c
[sudo] edu 的密码:
edu@edu:~/work/c/day06$ ./a.out
请输入int元素的个数:5
1 2 3 4 5
1 2 3 4 5
请输入你要新增的元素个数:3
请输入3个新增元素:100 200 300
1 2 3 4 5 100 200 300
edu@edu:~/work/c/day06$ ./a.out
请输入int元素的个数:5
1 2 3 4 5
1 2 3 4 5
请输入你要新增的元素个数:-2
请输入-2个新增元素:1 2 3
edu@edu:~/work/c/day06$
```

## 知识点2 【内存泄漏】(了解)

申请的内存，首地址丢了，找不了，再也无法使用了，也没法释放了，这块内存就被泄露了。

### 案例1:

```
1 char *p;
2 p=(char *)malloc(100);
3 //接下来，可以用 p 指向的内存了
4 p="hello world";//p 指向别的地方了
5 //从此以后，再也找不到你申请的 100 个字节了。则动态申请的 100 个字节就被泄露了
```

### 案例2:



```
void fun()
{
    char *p;
    p=(char *)malloc(100);
    //接下来，可以用 p 指向的内存了
    ;
    ;
}

int main()
{
    fun();
    fun();
    return 0;
}
```

### 知识点3 【内存的回顾】（了解）

```
1 //在32位平台
2 void test05()
3 {
4     //字符数组 在栈区 开辟12字节 存放"hello world"
5     char str1[] = "hello world";
6
7     //str2字符指针变量 在栈区 4B 保存文字常量区"hello world"的首元素地址
8     char *str2 = "hello world";
9
10    //str3字符指针变量 在栈区 4B 保存堆区128B空间的起始地址
```

```

11     char *str3 = (char *)calloc(128);
12     //将字符串"hello world"拷贝到str3指向的堆区空间
13     strcpy(str3, "hello world");
14
15 }

```

```

//在32位平台
char a;
int b;
//void fun(char str4[128])
void fun(char *str4)
{
    printf("%lu\n", sizeof(str4)); //4
}
void test06()
{
    char str1[] = "hello";
    char *str2 = "hello";
    char *str3 = (char *)malloc(128);
    printf("%lu\n", sizeof(a)); //1
    printf("%lu\n", sizeof(b)); //4
    printf("%lu\n", sizeof(str1)); //6
    printf("%lu\n", sizeof(str2)); //4
    printf("%lu\n", sizeof(str3)); //4
}

```

## 知识点4 【标准的空间释放】（了解）

```

1 void test07()
2 {
3     int *p = NULL;
4
5     p = (int *)calloc(1, sizeof(int));
6
7     *p = 100;
8     printf("*p = %d\n", *p);
9
10    if (p != NULL)
11    {

```

```
12     free(p);
13     p = NULL;
14 }
15
16 if (p != NULL)
17 {
18     free(p);
19     p = NULL;
20 }
21
22 return;
23 }
```

释放free本质功能：回收空间权限

```
7     return;
8 }
9
10 void test08()
11 {
12     char *p = (char *)calloc(1, 128);
13     strcpy(p, "hello world");
14
15     free(p);
16     printf("%s\n", p); // 不确定
17 }
18 }
```



```
edu@edu: ~/work/c/day06
edu@edu: ~/work/c/day06$ sudo gcc 00_code.c
edu@edu: ~/work/c/day06$ ./a.out
hello world
edu@edu: ~/work/c/day06$
```