**内存 -- calloc realloc**

1. Calloc（会有初始化默认值）
   1. Calloc(个数， 字节数)
   2. 代码，比如我们现在要开辟十个 int 数组

void test01(){

int \*p = calloc(10, sizeof(int)); //现在已经开辟完了

for(int i = 0; i < 10; i++){

p[i] = i; //连续的内存都可以进行下标赋值

}

//打印出来，看可不可以

for(int i = 0; i < 10; i++){

printf("%d ", \*p++);

}

if(p != NULL){ //申请完要释放

free(p);

p = NULL;

}

}

1. Malloc（不能初始化默认值 这与calloc的区别）
   1. Malloc（字符类型 \* 个数）
   2. 代码（用法差不多）

void test01(){

int \*p = malloc(sizeof(int)\*10); //现在已经开辟完了

for(int i = 0; i < 10; i++){

p[i] = i; //连续的内存都可以进行下标赋值

}

//打印出来，看可不可以

for(int i = 0; i < 10; i++){

printf("%d ", \*p++);

}

if(p != NULL){ //申请完要释放

free(p);

p = NULL;

}

}

1. Realloc用法
   1. Realloc（指针， 个数）
   2. 工作原理
      1. 其实就是重新申请一块内存空间
      2. 把原来的内存里面的值copy到新的空间去
      3. 释放原来的空间
      4. 把新的空间的地址返回去
   3. 代码 （当用\*p++进行输出时，记得指针还得重新指向回去）

void test01(){

int \*p = malloc(sizeof(int)\*10); //现在已经开辟完了

for(int i = 0; i < 10; i++){

p[i] = i; //连续的内存都可以进行下标赋值

}

//打印出来，看可不可以

for(int i = 0; i < 10; i++){

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

}

printf("\n");

p = realloc(p, sizeof(int) \* 20);

//打印出来，看可不可以

for(int i = 0; i < 20; i++){

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

}

if(p != NULL){ //申请完要释放

free(p);

p = NULL;

}

}