# 课外阅读资料之四---Malloc() 和 free()函数的使用

黄永峰整理

## 一、malloc()和 free()的基本概念以及基本用法：

1、函数原型及说明：

void *malloc(long NumBytes)：该函数分配了 NumBytes 个字节，并返回了指向这块内存的指针。如果分配失败，则返回一个空指针（NULL）。关于分配失败的原因，应该有多种，比如说空间不足就是一种。void free(void *FirstByte)：该函数是将之前用 malloc 分配的空间还给程序或者是操作系统，也就是释放了这块内存，让它重新得到自由。

2、函数的用法：

其实这两个函数用起来倒不是很难，也就是 malloc()之后觉得用够了就甩了它把它给 free()了，举个简单例子：

程序代码：

```
// Code
char *Ptr=NULL ;
Ptr = (char *)malloc(100 * sizeof(char));
if (NULL== Ptr)
{exit (1);
}
gets(Ptr);
// code
free(Ptr);
Ptr = NULL;
// code...
```

就是这样！当然，具体情况要具体分析以及具体解决。比如说，你定义了一个指针，在一个函数里申请了一块内存然后通过函数返回传递给这个指针，那么也许释放这块内存这项工作就应该留给其他函数了。

3、关于函数使用需要注意的一些地方：

A、申请了内存空间后，必须检查是否分配成功。

B、当不需要再使用申请的内存时，记得释放；释放后应该把指向这块内存的指针指向 NULL，防止程序后面不小心使用了它。

C、这两个函数应该是配对。如果申请后不释放就是内存泄露；如果无故释放那就是什么也没有做。释放只能一次，如果释放两次及两次以上会出现错误（释放空指针例外，释放空指针其实也等于啥也没做，所以释放空指针释放多少次都没有问题）。

D、虽然 malloc()函数的类型是(void *),任何类型的指针都可以转换成(void *),但是最好还是在前面进行强制类型转换，因为这样可以躲过一些编译器的检查。

**如果你只想学好使用，看到此就可以。如果你喜欢问什么？就继续看。**

**二、malloc()到底从哪里得来了内存空间：**

1、malloc()到底从哪里得到了内存空间？答案是从堆里面获得空间。也就是说函数返回的指针是指向堆里面的一块内存。操作系统中有一个记录空闲内存地址的链表。当操作系统收到程序的申请时，就会遍历该链表，然后就寻找第一个空间大于所申请空间的堆结点，然后就将该结点从空闲结点链表中删除，并将该结点的空间分配给程序。就是这样！

说到这里，不得不另外插入一个小话题，相信大家也知道是什么话题了。什么是堆？说到堆，又忍不住说到了栈！什么是栈？下面就另外开个小部分专门而又简单地说一下这个题外话：

2、什么是堆：堆是大家共有的空间，分全局堆和局部堆。全局堆就是所有没有分配的空间，局部堆就是用户分配的空间。堆在操作系统对进程 初始化的时候分配，运行过程中也可以向系统要额外的堆，但是记得用完了要还给操作系统，要不然就是内存泄漏。

什么是栈：栈是线程独有的，保存其运行状态和局部自动变量的。栈在线程开始的时候初始化，每个线程的栈互相独立。每个函数都有自己的栈，栈被用来在函数之间传递参数。操作系统在切换线程的时候会自动的切换栈，就是切换 SS/ESP 寄存器。栈空间不需要在高级语言里面显式的分配和释放。

通过上面对概念的描述，可以知道：栈是由编译器自动分配释放，存放函数的参数值、局部变量的值等。操作方式类似于数据结构中的栈。

堆一般由程序员分配释放，若不释放，程序结束时可能由 OS 回收。注意这

里说是可能，并非一定。所以我想再强调一次，记得要释放！

注意它与数据结构中的堆是两回事，分配方式倒是类似于链表。（这点我上面稍微提过）所以，举个例子，如果你在函数上面定义了一个指针变量，然后在这个函数里申请了一块内存让指针指向它。实际上，这个指针的地址是在栈上，但是它所指向的内容却是在堆上面的！这一点要注意！所以，再想想，在一个函数里申请了空间后，比如说下面这个函数：

程序代码：

```
// code
void Function(void)
{char *p = (char *)malloc(100 * sizeof(char));
}
```

就这个例子，千万不要认为函数返回，函数所在的栈被销毁指针也跟着销毁，申请的内存也就一样跟着销毁了！这绝对是错误的！因为申请的内存在堆上，而函数所在的栈被销毁跟堆完全没有啥关系。

**所以，还是那句话：记得释放！**

3、free()到底释放了什么

这个问题比较简单，其实我是想和第二大部分的题目相呼应而已！哈哈！free()释放的是指针指向的内存！注意！释放的是内存，不是指针！这点非常非常重要！指针是一个变量，只有程序结束时才被销毁。释放了内存空间后，原来指向这块空间的指针还是存在！只不过现在指针指向的内容的垃圾，是未定义的，所以说是垃圾。因此，前面我已经说过了，释放内存后把指针指向 NULL，防止指针在后面不小心又被解引用了。非常重要啊这一点！。