定义

[折叠](http://baike.so.com/doc/6701356-6915295.html)原型

extern void \*malloc(unsigned int num\_bytes);

[折叠](http://baike.so.com/doc/6701356-6915295.html)功能

分配长度为num\_bytes字节的[内存](http://baike.so.com/doc/176561-186541.html)块

[折叠](http://baike.so.com/doc/6701356-6915295.html)返回值

如果分配成功则返回指向被分配内存的[指针](http://baike.so.com/doc/1043844-1104112.html)(此存储区中的初始值不确定)，否则返回空指针[NULL](http://baike.so.com/doc/3409199-3588404.html)。当内存不再使用时，应使用free()函数将内存块释放。函数返回的[指针](http://baike.so.com/doc/1043844-1104112.html)一定要适当对齐，使其可以用于任何数据对象。

[折叠](http://baike.so.com/doc/6701356-6915295.html)说明

关于该函数的原型，在以前malloc返回的是[char](http://baike.so.com/doc/19102-19824.html)型[指针](http://baike.so.com/doc/1043844-1104112.html)，新的ANSIC标准规定，该函数返回为void型指针，因此必要时要进行类型转换。

[折叠](http://baike.so.com/doc/6701356-6915295.html)名称解释

malloc的全称是memory allocation，中文叫[动态内存](http://baike.so.com/doc/7863225-8137320.html)分配，当无法知道内存具体位置的时候，想要绑定真正的内存空间，就需要用到动态的分配内存。

[折叠](http://baike.so.com/doc/6701356-6915295.html)相关函数

[calloc](http://baike.so.com/doc/5579484-5792854.html)、[realloc](http://baike.so.com/doc/7106035-7329073.html)、free、[\_alloca](http://baike.so.com/doc/4192180-4392945.html)

[折叠](http://baike.so.com/doc/6701356-6915295.html)[编辑本段](http://baike.so.com/create/edit/?eid=6701356&sid=6915295&secid=2)函数声明

[折叠](http://baike.so.com/doc/6701356-6915295.html)全名

void \*malloc(size\_t size);

[折叠](http://baike.so.com/doc/6701356-6915295.html)备注

void\* 表示未确定类型的[指针](http://baike.so.com/doc/1043844-1104112.html)，void \*可以指向任何类型的数据，更明确的说是指申请内存空间时还不知道用户是用这段空间来存储什么类型的数据(比如是char还是[int](http://baike.so.com/doc/3405534-3584546.html)或者...)

[折叠](http://baike.so.com/doc/6701356-6915295.html)[编辑本段](http://baike.so.com/create/edit/?eid=6701356&sid=6915295&secid=3)与new的区别

从本质上来说，malloc([Linux](http://baike.so.com/doc/5349227-5584683.html)上具体实现可以参考man malloc，glibc通过brk()&mmap()实现)是libc里面实现的一个函数，如果在source code中没有直接或者间接include过stdlib.h，那么gcc就会报出error:'malloc' was not declared in this scope。如果生成了目标文件(假定动态链接malloc)，如果运行平台上没有libc(Linux平台，手动指定LD\_LIBRARY\_PATH到一个空目录即可)，或者libc中没有malloc函数，那么会在运行时(Run-time)出错。new则不然，是c++的关键字，它本身不是函数。new不依赖于头文件，c++编译器就可以把new编译成目标代码(g++4.6.3会向目标中插入\_Znwm这个函数，另外，编译器还会根据参数的类型，插入相应的构造函数)。

在使用上，malloc 和 new 至少有两个不同: new 返回指定类型的[指针](http://baike.so.com/doc/1043844-1104112.html)，并且可以自动计算所需要大小。比如:

或:

而 malloc 则必须要由我们计算字节数，并且在返回后强行转换为实际类型的[指针](http://baike.so.com/doc/1043844-1104112.html)。

第一、malloc 函数返回的是 void \* 类型。对于C++，如果你写成:p = malloc (sizeof(int)); 则程序无法通过编译，报错:"不能将 void\* 赋值给 int \* 类型变量"。所以必须通过 (int \*) 来将[强制转换](http://baike.so.com/doc/6744020-6958553.html)。而对于C，没有这个要求，但为了使C程序更方便的移植到C++中来，建议养成[强制转换](http://baike.so.com/doc/6744020-6958553.html)的习惯。

第二、函数的[实参](http://baike.so.com/doc/6665407-6879236.html)为 sizeof(int) ，用于指明一个[整型数据](http://baike.so.com/doc/6060849-6273903.html)需要的大小。

在Linux中可以有这样:malloc(0),这是因为Linux中malloc有一个下限值16Bytes，注意malloc(-1)是禁止的;

但是在某些系统中是不允许malloc(0)的。在规范的程序中我们有必要按照这样的格式去使用malloc及free:

malloc 也可以达到 new [] 的效果，申请出一段连续的内存，方法无非是指定你所需要内存大小。

比如想分配100个int类型的空间:

另外有一点不能直接看出的区别是，malloc 只管分配内存，并不能对所得的内存进行初始化，所以得到的一片新内存中，其值将是随机的。

除了分配及最后释放的方法不一样以外，通过malloc或new得到[指针](http://baike.so.com/doc/1043844-1104112.html)，在其它操作上保持一致。

对其做一个特例补充

此时得到的是Got a valid pointer。把0赋给malloc能得到一个合法的[指针](http://baike.so.com/doc/1043844-1104112.html)。

[折叠](http://baike.so.com/doc/6701356-6915295.html)[编辑本段](http://baike.so.com/create/edit/?eid=6701356&sid=6915295&secid=4)工作机制

malloc函数的实质体现在，它有一个将可用的内存块连接为一个长长的列表的所谓空闲[链表](http://baike.so.com/doc/6335578-6549190.html)。调用malloc函数时，它沿[连接表](http://baike.so.com/doc/2152317-2277303.html)寻找一个大到足以满足用户请求所需要的内存块。然后，将该内存块一分为二(一块的大小与用户请求的大小相等，另一块的大小就是剩下的字节)。接下来，将分配给用户的那块内存传给用户，并将剩下的那块(如果有的话)返回到连接表上。调用free函数时，它将用户释放的内存块连接到空闲链上。到最后，空闲链会被切成很多的小内存片段，如果这时用户申请一个大的内存片段，那么空闲链上可能没有可以满足用户要求的片段了。于是，malloc函数请求延时，并开始在空闲链上翻箱倒柜地检查各内存片段，对它们进行整理，将相邻的小空闲块合并成较大的内存块。如果无法获得符合要求的内存块，malloc函数会返回NULL[指针](http://baike.so.com/doc/1043844-1104112.html)，因此在调用malloc动态申请内存块时，一定要进行返回值的判断。

Linux Libc6采用的机制是在free的时候试图整合相邻的碎片，使其合并成为一个较大的free空间。