堆上分配

malloc calloc realloc free

原则：谁申请谁释放

严格要求自己，如果你写的程序没有内存泄漏，没有越界现象，那你就是一个非常优秀的程序员了

malloc

函数原型，void \*malloc(size\_t size);

它的参数是一个size\_t类型的size，size\_t是typedef出来的一个整型。size\_t size就是我想要这么大一块空间，它会在堆上找size个连续的字节空间，然后给你，怎么给你，把起始地址返回给你，所以这个malloc的返回值是一个void \*，是一个地址值，你会发现它给你的其实是最好用的百搭的指针类型，为什么呢，是因为你只跟它说了你要size个字节空间，malloc函数给你找这块空间的时候也不知道你要存放什么样的类型，所以只能用一个void \*给你，你用任何一个类型的指针去接收判断就行

calloc

函数原型，void \*calloc(size\_t nmemb, size\_t size);

我想要nmemb这么多个成员，size\_t size表示一个成员的大小为size，size\_t nmemb表示我想要这么多个成员的空间，实际上calloc实现的是，连续申请n块多大的内存，比如1个学生占40个字节，我现在一下想要10个学生的连续的存储空间，总体来讲我其实是想要400个字节来存放我这10个学生，同样的道理，返回值也是告诉它能否成功

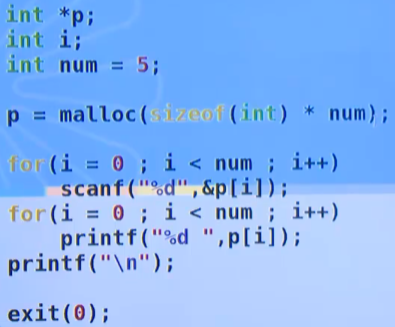
realloc

函数原型，void \*realloc(void \*ptr, size\_t size);

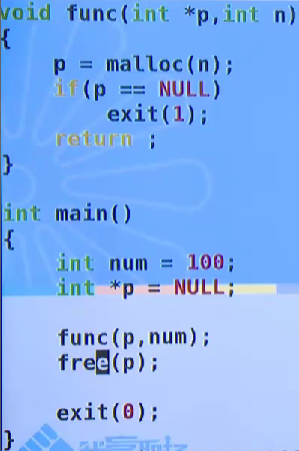
re表示重新怎么怎么样，在这表示重新地为我分配一块动态的内存空间，第一个参数是一个指针，手册上说了，必须是malloc和calloc这两个函数调用之后返回的某一个指针，realloc指的是，我原来可能用malloc和calloc这两个函数申请过内存空间，起始地址是void \*ptr，但是我现在大小不太合适了，有点大了或有点小了，那我需要realloc重新分配，起始位置是void \*ptr，后边的size\_t size就是新的大小，比如说，我一开始用malloc申请了100个字节的空间，然后用着用着发现不够用了，我就说从void \*ptr这块起始地址开始，我要size\_t size（300）个字节的空间，然后把这300个字节空间的起始位置还给你，如果在空间扩展时，发现下面的连续空间被别人占用了，那它就会在别的地方来找，找到一块连续的内存地址是300个字节，然后把内容拿过来，把原来的释放掉，再把新的起始位置给你，相当于重新分配，当然如果你原来要的内存太大，我现在要把它变小，那它就从尾巴开始截，截到满足你最终获取的是size个字节

malloc，calloc，realloc都是，成功返回非空指针，失败返回空指针

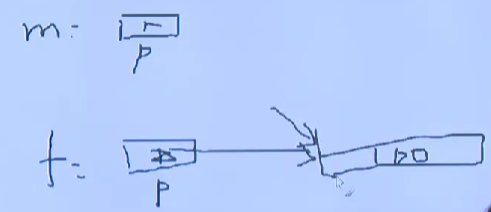
malloc使用



malloc经典面试题，以下程序有没有问题

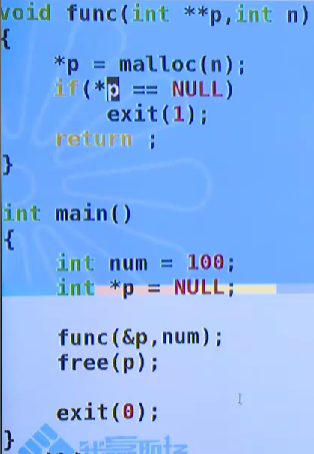


答：有问题，产生了内存泄漏，main函数传个p过去，是值传递，被调函数生成的是一个全新的p，你在被调函数内部分配了空间，但没有释放，等被调函数一结束，被调函数的p直接消亡，你再也找不到那块被分配的空间了

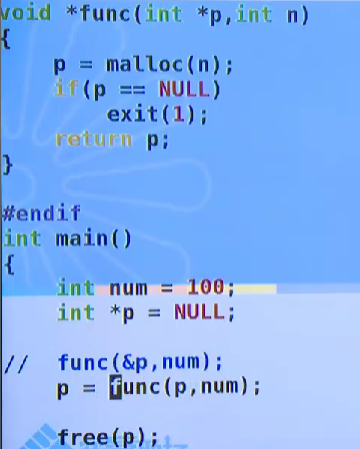


两种办法改进

改进1：修改传参，用一个二级指针



改进2：修改函数返回值



free



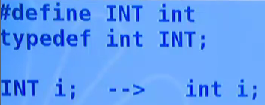
有人认为free掉之后，这块空间还能用，不对，有人认为free掉之后，这块空间没有了，也不对，你的free并没有把原先用的那块内存扣掉，也没有把内容清掉，甚至指针的指向还指向那个位置呢，但它已经是野指针，free代表的是你通过变量p对于那块空间再也没有引用的权限（你退租了，你就不要再回去了，你已经没有权限回去了）

指针被free掉后，给它立即置空，防止它成为野指针，如果后面一不小心还使用到p，那么会报段错误，出现段错误不是啥坏事，这说明你这个错误没有被无限放大

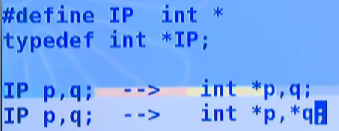
typedef

针对某一个已有的数据类型（包括基本类型，也包括构造类型）来进行改名的这样一个方法

语法结构：typedef 已有的数据类型 新名字;



看起来typedef和宏没啥区别，但还是有的，如下

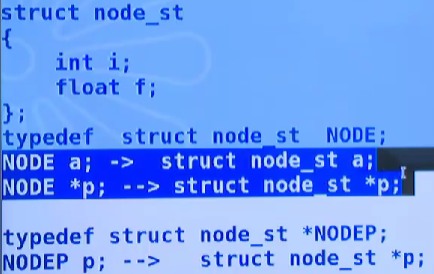


上面用宏，是定义了一个整型的指针变量p和整型变量q

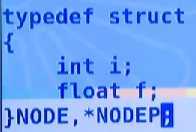
下面用typedef，是定义了两个整型的指针变量p和q



又回到数组的话题了，int [6]才是数组的本质，这不是在给int改名，而是在给int[6]改名



像给结构体进行typedef的话，其实完全没必要写这么复杂



typedef后面这个结构体叫NODE，typedef后面这个结构体的\*叫NODEP

