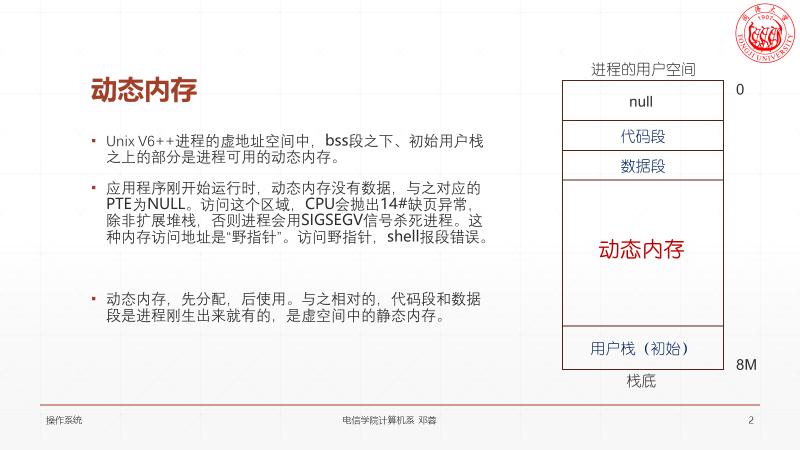
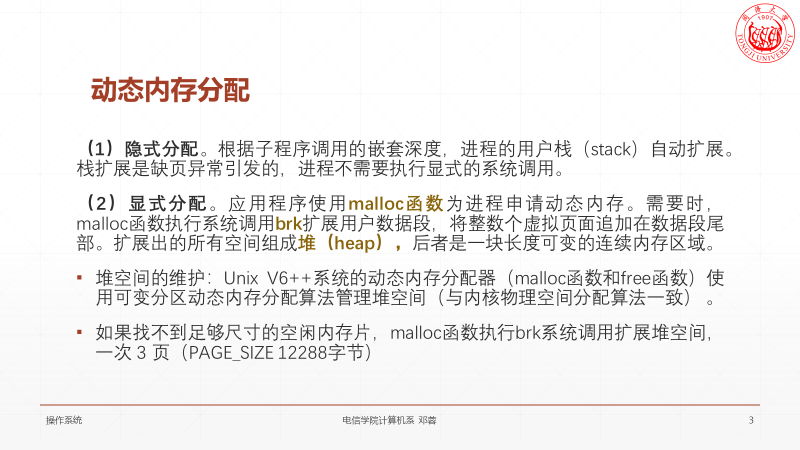
# 进程的动态内存及其管理策略

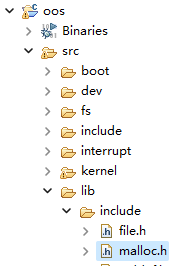
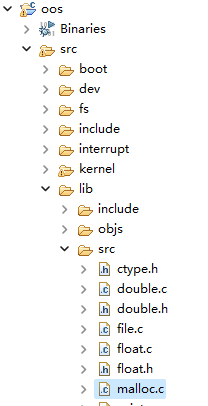
同济大学计算机系 操作系统课程 讲稿 2023-12-07

一、



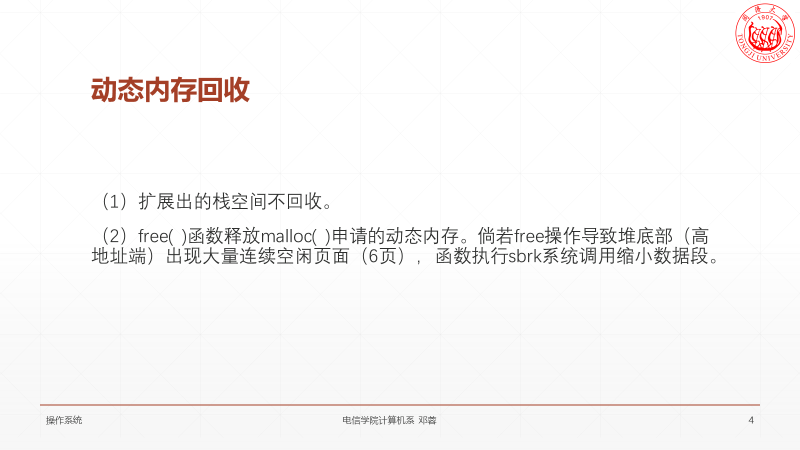
二、



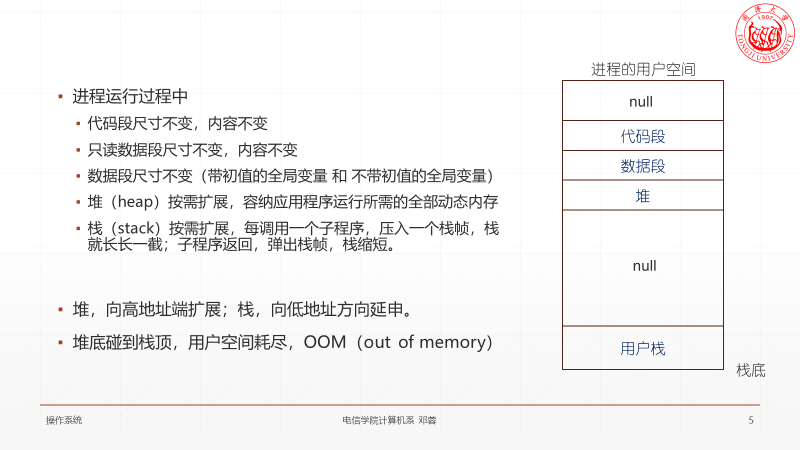


Unix V6++的动态内存分配器在这里，包括malloc函数，free函数和所有的全局变量

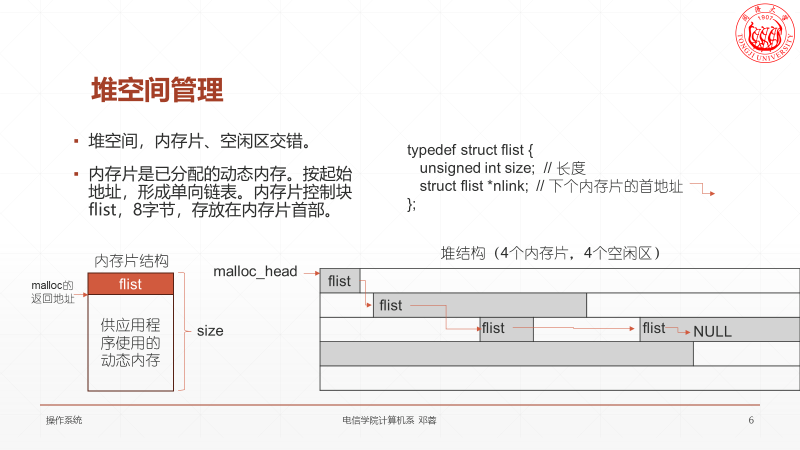
三、



四、



五、

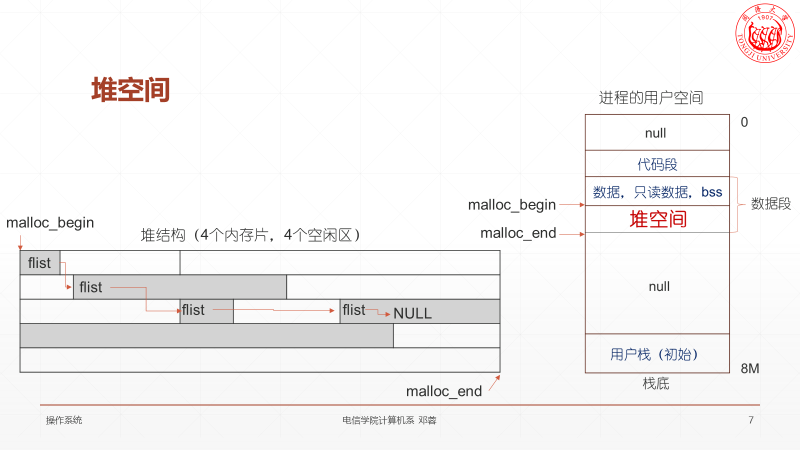


flist是内存片控制块，存放在其管理的内存片首部，占用8字节。size是该内存片的长度，nlist是下一个内存片的首地址。flist链表中的内存片，按起始地址排序。

图中有4个内存片：1个哑元 加 3个有数据的内存片。

malloc\_head是flist链表的首部。

六、



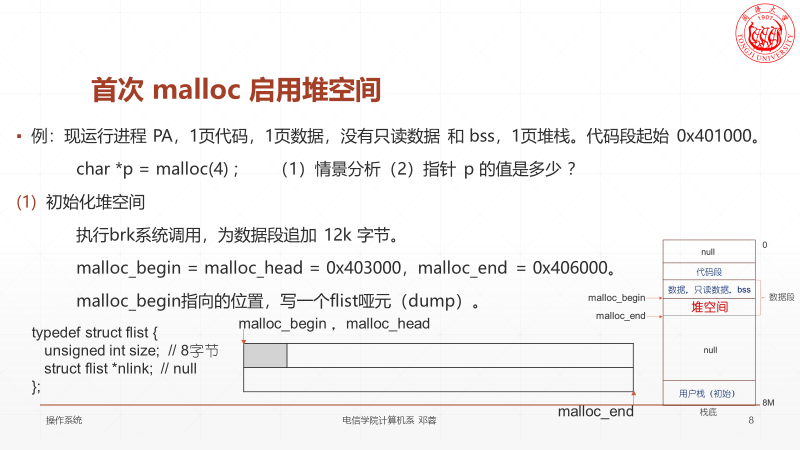
堆空间在进程虚地址空间中的位置。

malloc\_end之后的地址空间尚未分配，与之对应的PTE是空的，引用会引发缺页异常（下节课说）

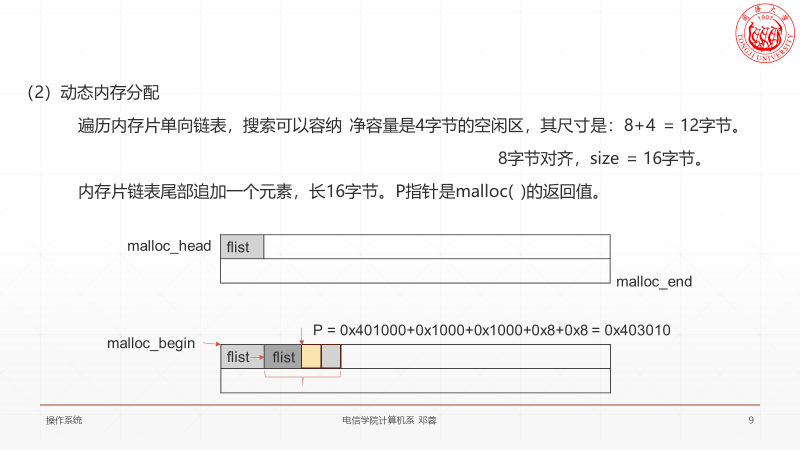
七、malloc( )函数的执行过程

1. 首次malloc的执行过程

第一步，初始化堆空间



第二步，动态内存分配

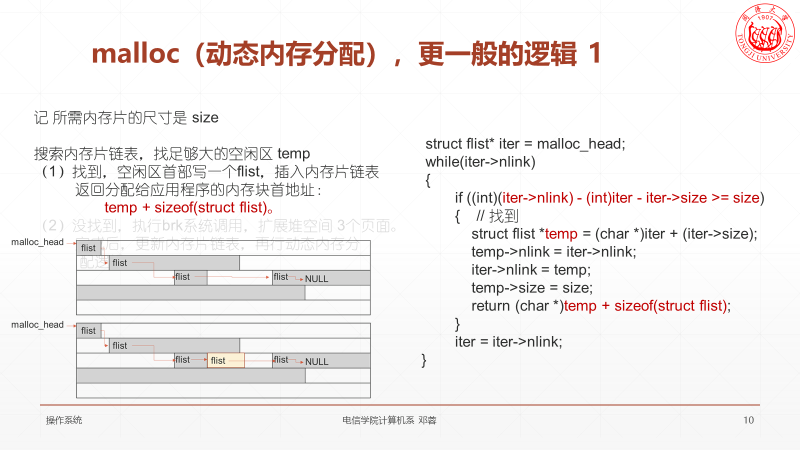


本例，内存片链表只有一个哑元，不存在2个已分配内存片之间的空间区可以容纳新内存片的可能。所以，我们在内存片链表尾部（哑元的后面）追加一个长度是16字节的新内存片。malloc函数的返回值p是新内存片数据区的首部0x403010。黄色部分4字节，是应用程序得到的4字节的动态内存块。因为需要8字节对齐（所有flist的首地址必须是8的整数倍），所以新内存片的尾部浪费4字节。 （（（完成）））

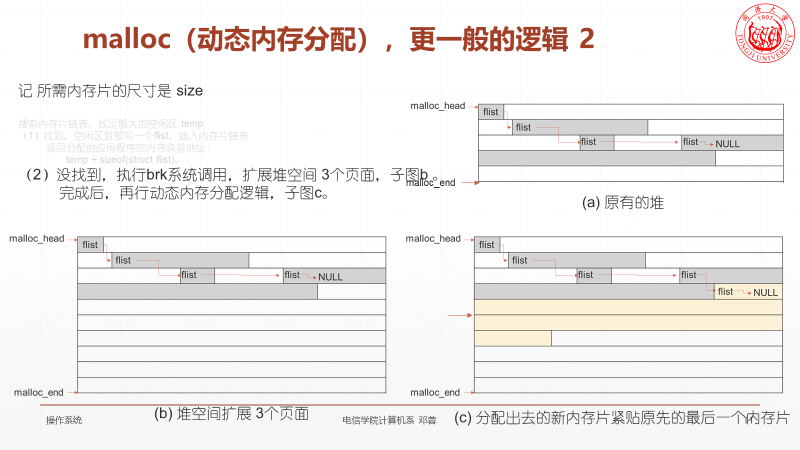
1. 动态内存分配过程，记所需内存片的尺寸是size。

搜索内存片链表，找足够大的空闲区 temp

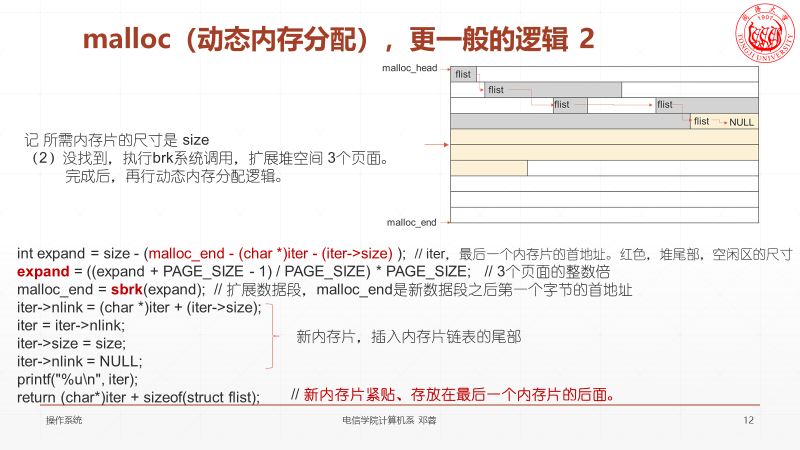
1. 找到，空闲区首部写一个flist，插入内存片链表。返回分配给应用程序的内存块首地址： temp + sizeof(struct flist)。



1. 没找到，执行brk系统调用，扩展堆空间 3个页面。完成后，更新内存片链表，再行动态内存分配逻辑。



代码：



八、free函数的执行过程

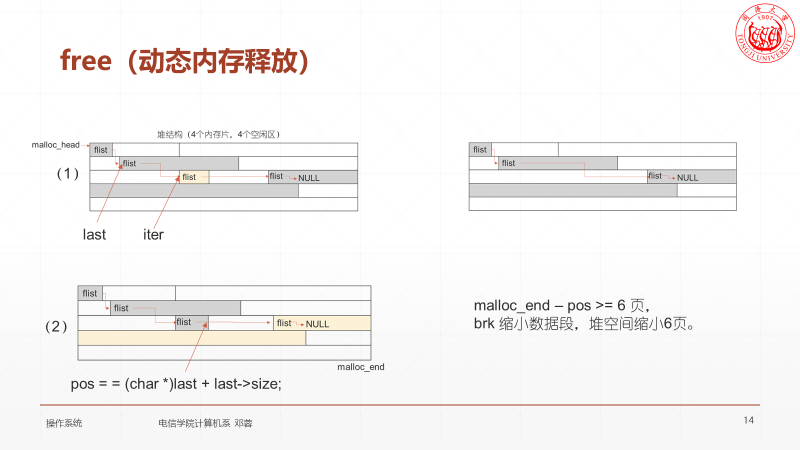
应用程序执行free( )，回收动态内存。入口参数addr是回收的动态内存块的首地址。

第一步：计算动态内存所在的内存片的首地址real\_addr。



第二步：动态内存回收操作

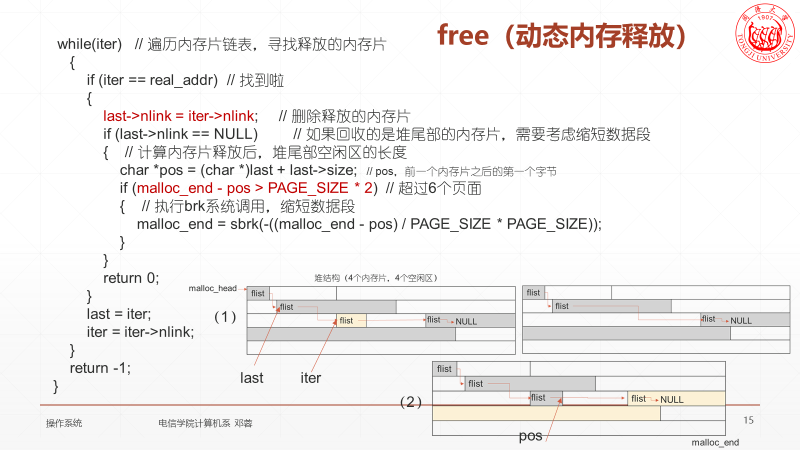
很简单，用iter的nlist字段赋值 last 的nlist字段。iter是回收的内存片，last是上一个内存片。



子图1，回收的内存片不是堆中的最后一个内存片；子图2，回收最后一个内存片。

回收最后一个内存片时，需要考虑是否需要缩小堆的尺寸。考量依据，见图2。

动态内存回收操作，源代码：



阅读源码。