# 操作系统实验六

171491223

刘欣月

## 实验目的

1. 理解并掌握操作系统的内存分配。
2. 掌握首次适应算法、最佳适应算法和最坏适应算法。

## 实验内容

1. 将老师给的原始代码进行修改和完善
2. 写出最佳适应算法

**算法思想：**

将内存块中的所有的块按照地址递增的顺序连接成一个链表，每次要将新的作业放入内存的时候就按顺序查找内存块链表，每次都是用找到的可以用的最小的内存块。

链表数据结构：

链表结点共有4个区域和一个下指针构成，四个区域分别记录该内存块的起始地址，该内存块长度，内存块的状态和内存块存放的作业编号（没有作业时存放的是0）。

模拟实现的策略：

1.插入操作时依照地址递增的顺序检查可以装入的第一个内存块；  
若找到，则装入内存中，并且把内部碎片分出来变成新的结点插入链表中；  
若没找到则分配失败；  
2.撤销操作时按照作业编号找到要撤销作业所在内存块；  
若找到，则将状态置位未分配，此时假如前后有可以合并的空内存块则合并；

若没有找到，则返回没有找到该作业信息；

**代码：**

**首次适应法：**

void Allocation(pLNode L, int len, int num) {

pLNode p = L->next;

while (p != NULL) {

if (p->state == 0) {

//表示内存块未分配

if (p->length > len) {

//表示可以分配内存空间

//但是需要把内部碎片分出来，构成新的空闲内存块

//新的内存块的起始地址为p->address + len，长度为p->length-len

pLNode l = (pLNode)malloc(sizeof(LNode));

l->length = p->length - len;

l->address = p->address + len;

l->state = 0;

l->number = 0;

l->next = p->next;

p->next = l;

p->length = len;

p->number = num;

p->state = 1;

printf("内存分配成功！\n");

return;

}

else if (p->length == len) {

//刚刚好够分配，只要修改作业编号和内存块状态即可

p->number = num;

p->state = 1;

printf("内存分配成功！\n");

return;

}

}

p = p->next;

}

printf("内存分配失败，没有找到合适的空闲内存块\n");

}

**最佳适应法：**

修改

void Allocation(pLNode L, int len, int num) {

pLNode p = L->next;

pLNode bestP=NULL;

int minplace=INF;

while (p != NULL) {

if (p->state == 0) {

//表示内存块未分配

/////////////

if(p->length>=len){

if(p->length<minplace){

bestP=p;

minplace=p->length;

}

}

}

p = p->next;

}

if (minplace!=INF&&bestP->length > len) {

//表示可以分配内存空间

//但是需要把内部碎片分出来，构成新的空闲内存块

//新的内存块的起始地址为bestP->address + len，长度为p->length-len

pLNode l = (pLNode)malloc(sizeof(LNode));

l->length = bestP->length - len;

l->address = bestP->address + len;

l->state = 0;

l->number = 0;

l->next = bestP->next;

bestP->next = l;

bestP->length = len;

bestP->number = num;

bestP->state = 1;

printf("内存分配成功！\n");

return;

}

else if (minplace!=INF&&bestP->length == len) {

//刚刚好够分配，只要修改作业编号和内存块状态即可

bestP->number = num;

bestP->state = 1;

printf("内存分配成功！\n");

return;

}

printf("内存分配失败，没有找到合适的空闲内存块\n");

}

运行结果：

