操作系统存储管理实验

一、实验目的

掌握存储管理机制

二、实验内容

1、采用链表进行空闲区和已分配空间的管理，该存储为连续空间上的存储

2、初始的内存大小为100000字节

3、采用最佳适应分配算法分配内存

三、实验源代码

#include <iostream.h>

#include <stdlib.h>

struct AllocatedSpace //已分配内存空间结构体

{

int start\_address;

int length;

char job;

struct AllocatedSpace \*next;

};

struct FreeSpace //未分配内存空间结构体

{

int start\_address;

int length;

struct FreeSpace \*next;

};

struct AllocatedSpace \*allocated\_header;

struct FreeSpace \*free\_header;

struct FreeSpace \*freenext;

void allocatememory() //分配内存函数

{

char jobname;

int joblength;

int min,l=0;

cout<<"请输入作业名（一个字母）:\n";

cin>>jobname;

cout<<"请输入作业长度（整数）:\n";

cin>>joblength;

struct FreeSpace \*w=free\_header;

struct FreeSpace \*p=free\_header; //指向第一个可以分配的空闲结点

p=p->next;

struct FreeSpace \*s=free\_header; //指向p的前一个结点

struct AllocatedSpace \*t=allocated\_header;

while(p->length<joblength) //p指向第一个可以分配的空结点

{

p=p->next;

if(p==NULL) break;

}

if(p==NULL)

cout<<"无法分配!\n";

else

{

w=p;

min=p->length-joblength;

while(p!=NULL) //现在p指向当前要被切割的结点

{

if(p->length>joblength) l=p->length-joblength;

if(l<min)

{

w=p;

min=l;

}

p=p->next;

}

struct AllocatedSpace \*q=new AllocatedSpace;

q->job=jobname;

q->length=joblength;

q->start\_address=w->start\_address;

while(t->next!=NULL) t=t->next;

q->next=t->next;

t->next=q;

while(s->next!=w) s=s->next;

if(w->length==q->length)

{

s->next=w->next;

delete w;

}

else

{

w->start\_address=w->start\_address+q->length;

w->length=w->length-q->length;

}

}

}

void reclaimmemory()

{

char jobname;

cout<<"请输入要回收的作业名:\n";

cin>>jobname;

struct FreeSpace \*p=new struct FreeSpace;//指向当前被创建的空闲结点

struct FreeSpace \*s=free\_header;//指p的前一个结点

struct FreeSpace \*m=free\_header;//指向s的前一个结点

struct AllocatedSpace \*q=allocated\_header;//指向被回收的结点

struct AllocatedSpace \*t=allocated\_header;//指向被回收的结点的前一个结点

while(q->job!=jobname)//找到被回收的结点

q=q->next;

if(q==NULL)

cout<<"回收错误,要回收的结点不存在!\n";

else

{

p->length=q->length;

p->start\_address=q->start\_address;

while(s->start\_address<p->start\_address) s=s->next;

while(m->next!=s) m=m->next;

p->next=s;

m->next=p;

cout<<"回收成功!\n";

}

while(t->next!=q) t=t->next;

t->next=q->next;

delete q;

s=free\_header;

m=free\_header;

s=s->next;

m=m->next;

while(m!=NULL)

{

m=m->next;

if(m==NULL) break;

if(s->start\_address+s->length==m->start\_address)

{

s->length=s->length+m->length;

s->next=m->next;

continue;

}

s=s->next;

}

}

void showmemory()

{

struct FreeSpace \*s=free\_header;//指向FreeSpace的头结点，开始往后遍历

s=s->next;

struct AllocatedSpace \*q=allocated\_header;//指向AllocatedSpace的头结点，开始往后遍历q=q->next;

cout<<"已分配区的情况如下:\n";

while(q!=NULL)

{

cout<<"作业名:"<<q->job<<"开始地址:"<<q->start\_address<<"长度:"<<q->length<<endl;

q=q->next;

}

cout<<"空闲结点的情况如下:\n";

while(s!=NULL)

{

cout<<"开始地址:"<<s->start\_address<<"长度:"<<s->length<<endl;

s=s->next;

}

}

int main()

{

int n;

allocated\_header=new struct AllocatedSpace;

free\_header=new struct FreeSpace;

freenext=new struct FreeSpace;

freenext->length=100000;

freenext->start\_address=0;

freenext->next=NULL;

free\_header->next=freenext;

allocated\_header->next=NULL;

while(1)

{

cout<<"请输入您要进行的操作：0退出1分配内存2 回收内存3 显示内存状态\n";

cin>>n;

switch(n)

{

case 0:exit(0);

case 1:allocatememory();break;

case 2:reclaimmemory();break;

case 3:showmemory();break;

}

}

}备注功能键 定义

a)0 退出

b)1 为作业分配内存

输入：作业名m,要求的内存大小

 执行逻辑：若无满足要求大小的空闲区，则显示无法分配的消息并退出，否则依据最佳适应分配算法找到空闲区，分配给作业（有可能需要对空闲区进行切割），并在已分配链表中进行登记

c)2 回收内存

输入：作业名

  执行逻辑：在已分配链表中找到给定的作业名对应的项，将对应的内存回收，修改已分配链表和空闲区链表（有可能需要在空闲区链表中进行合并）

d)3 显示内存分配情况

已分配内存空间情况（作业名，起始地址, 长度）和空闲区情况（起始地址，长度）

四、实验要求

1、写出各个功能模块的流程图

2、给出运行结果，并分析

3、实验总结

4、有兴趣的同学，采用其它分配算法分配内存