操作系统实验报告

存储器的分配与回收算法实现

姓　名：_____

学　号：_____

班　级：_____

一、实验名称及要求

1、实验名称：

存储器的分配与回收算法实现

2、实验要求：

学生应正确地设计有关的数据结构与各个功能模块，画出程序的流程图，编写程序，程序执行结果应正确。

3、实验方式：

学生通过实验室的微机上机，实际调试程序。。

4、实验环境：

Windows 操作系统环境下的个人微机

C或C＋＋程序设计语言

二、实验内容

1　本实验是模拟操作系统的主存分配，运用可变分区的存储管理算法设计主存分配和回收程序，并不实际启动装入作业。

2　采用最先适应法、最佳适应法、最坏适应法分配主存空间。

3　当一个新作业要求装入主存时，必须查空闲区表，从中找出一个足够大的空闲区。若找到的空闲区大于作业需要量，这是应把它分成二部分，一部分为占用区，加一部分又成为一个空闲区。

4　当一个作业撤离时，归还的区域如果与其他空闲区相邻，则应合并成一个较大的空闲区，登在空闲区表中。

5　运行所设计的程序，输出有关数据结构表项的变化和内存的当前状态。

三、实验程序

```
#include <iostream.h>
#include <malloc.h>
#include <stdlib.h>

typedef struct FreeLink{//        定义自由链
    struct FreeLink *prior;
    char name;
    int start;
    int size;
```

```
    bool flag;
    struct FreeLink *next;
}* ptr,*head;

head top;
ptr p;

void print(){//      将内存分配情况打印到屏幕上
    p=top;
                                        内    存    分    配    情    况    表
            区号          起始位置        区间长度        区间状态
    do{

                                        空闲
                            已占用
        p=p->next;
    }
    while(p!=NULL);
}

void clear(){//      结束操作时清空“内存”以备其他操作
    do{
        p=top;
        top=top->next;
        free(p);
    }
    while(top!=NULL);
}

void asc(ptr &p){//      最佳适应法的内存分配函数
    int min;
    ptr op;
    FreeLink *fl=(FreeLink *)malloc(sizeof(FreeLink));
            请输入要分配内存的进程名
    cin>>fl->name;
```

```
            请输入要分配内存的大小
cin>>fl->size;
min=256;
fl->flag=true;
do{
      if(p->flag==false&&p->size<=min&&p->size>=fl->size){
            min=p->size;
            op=p;
      }
      p=p->next;
}
while(p!=NULL);
if(op->size>fl->size){
   fl->start=op->start;
   op->start=fl->start+fl->size;
   op->size=op->size-fl->size;
   fl->next=op;
   fl->prior=op->prior;
   op->prior->next=fl;
op->prior=fl;
goto flag1;
}
if(op->size==fl->size){
op->flag=fl->flag;
op->name=fl->name;
      free(fl);
      goto flag1;
      }
                  内存过小，分配失败!
                        分配成功!
flag2:  ;
}

void dec(ptr &p){//      最坏适应法的内存分配函数
      int max;
      ptr op;
```

```
FreeLink *fl=(FreeLink *)malloc(sizeof(FreeLink));
        请输入要分配内存的进程名
cin>>fl->name;
        请输入要分配内存的大小
cin>>fl->size;
max=fl->size;
fl->flag=true;
do{
    if(p->flag==false&&p->size>=max){
        max=p->size;
        op=p;
    }
    p=p->next;
}
while(p!=NULL);
if(op->size>fl->size){
  fl->start=op->start;
  op->start=fl->start+fl->size;
  op->size=op->size-fl->size;
  fl->next=op;
  fl->prior=op->prior;
  op->prior->next=fl;
op->prior=fl;
goto flag3;
}
    if(op->size==fl->size){
    op->flag=fl->flag;
    op->name=fl->name;
      free(fl);
    goto flag3;
    }
            内存过小，分配失败!
                分配成功!
flag4: ;
}
```

```
void splice(ptr &p){//        若被操作的内存有相邻空闲区则将空闲区拼接合并
      int x;
      if(p->prior->flag==false&&p->next->flag==false)x=1;
      if((p->prior->flag==false&&p->next->flag==true)||(p->prior->flag==false&&
p->next==NULL))x=2;
      if((p->prior->flag==true&&p->next->flag==false)||(p->prior==NULL&&p->next
->flag==false))x=3;
      if((p->prior->flag==true&&p->next->flag==true)||(p->prior==NULL&&p->next-
>flag==true)||(p->prior->flag==true&&p->next==NULL))x=4;
   switch(x){
      case 1:p->next->prior=p->prior;
             p->prior->next=p->next;
             p->prior->size=p->prior->size+p->size+p->next->size;
             p->prior->next=p->next->next;
             if(p->next->next!=NULL)p->next->next->prior=p->next->prior;
             free(p->next);
             free(p);
             break;
   case 2:if(p->next==NULL){
                  p->prior->next=p->next;
             }else{
                  p->next->prior=p->prior;
                  p->prior->next=p->next;
               }
             p->prior->size=p->prior->size+p->size;
             free(p);
             break;
  case 3:if(p->prior==NULL){
                   top=p->next;
                   p->next->prior=NULL;
                   p->next->start=p->start;
                p->next->size=p->next->size+p->size;
             }else{
                    p->next->prior=p->prior;
                    p->prior->next=p->next;
                      p->next->start=p->start;
```

```
                p->next->size=p->next->size+p->size;
              }
            free(p);
            break;
      case 4:p->name='@';
            p->flag=false;
            break;
      }
}

void allocate(ptr &p){//        最先适应法的内存分配函数
FreeLink *fl=(FreeLink *)malloc(sizeof(FreeLink));
          请输入要分配内存的进程名
cin>>fl->name;
          请输入要分配内存的大小
cin>>fl->size;
fl->flag=true;
do{
      if(p->flag==false&&p->size>fl->size){
        fl->start=p->start;
        p->start=fl->start+fl->size;
        p->size=p->size-fl->size;
        fl->next=p;
        fl->prior=p->prior;
        p->prior->next=fl;
    p->prior=fl;
    goto a;
        }
        if(p->flag==false&&p->size==fl->size){
    p->flag=fl->flag;
          p->name=fl->name;
          free(fl);
    goto a;
        }
      p=p->next;
}
```

```
    while(p!=NULL);
            内存过小，分配失败！
            分配成功！
b:  ;
}

void recover(ptr &p){//      内存回收函数
    char n = ' ';
            请输入要回收的内存对应的进程名
    cin>>n;
    do{
        if(p->flag==true&&p->name==n){
            splice(p);
         goto c;
          }
        p=p->next;
    }
    while(p!=NULL);
            内存并未分配给对应进程，回收失败！
            内存回收成功！
d:  ;
}

int ffa(){//     最先适应法
    char choice=' ';
    print();
    ptr pcb=(FreeLink *)malloc(sizeof(FreeLink));
    pcb->next=top;
    pcb->prior=top->prior;
  top->prior=pcb;
    pcb->start=top->start;
            请输入要为系统分配的内存块名
    cin>>pcb->name;
            请输入要分配内存的大小
e:          超过内存最大容量请重新输入要分配内存的大小
f:  cin>>pcb->size;
```

```
if(pcb->size>256) goto e;
top->size=top->size-pcb->size;
top=pcb;
top->flag=true;
top->next->start+=top->size;
print();
while(true){
do{
    p=top->next;
            请从下列选项中进行选择
              分配内存
              回收内存
              结束操作
            请输入你的选择
      cin>>choice;
    }
    while(choice!='1'&&choice!='2'&&choice!='3');
    switch(choice){
    case '1':allocate(p);print();break;
    case '2':recover(p);print();break;
    case '3':clear();return 0;break;
      }
    }
}
int bfa(){//     最佳适应法
    char choice=' ';
    print();
    ptr pcb=(FreeLink *)malloc(sizeof(FreeLink));
    pcb->next=top;
    pcb->prior=top->prior;
  top->prior=pcb;
    pcb->start=top->start;
            请输入要为系统分配的内存块名
    cin>>pcb->name;
            请输入要分配内存的大小
```

```
h:   cin>>pcb->size;
     if(pcb->size>256) goto g;
     top->size=top->size-pcb->size;
     top=pcb;
     top->flag=true;
     top->next->start+=top->size;
     print();
     while(true){
     do{
          p=top->next;
                  请从下列选项中进行选择
                    分配内存
                    回收内存
                    结束操作
                  请输入你的选择
       cin>>choice;
}
while(choice!='1'&&choice!='2'&&choice!='3');
switch(choice){
case '1':asc(p);print();break;
case '2':recover(p);print();break;
case '3':clear();return 0;break;
       }
}
}

int wfa(){//     最坏适应法
       char choice=' ';
       print();
       ptr pcb=(FreeLink *)malloc(sizeof(FreeLink));
       pcb->next=top;
       pcb->prior=top->prior;
    top->prior=pcb;
       pcb->start=top->start;
                  请输入要为系统分配的内存块名
```