北 京 林 业 大 学

2019 学年—2020 学年第 2 学期计算机体系结构实验报告书

专业：计算机科学与技术(创新实验班)　班级：计创 18

姓　　名：连月菡　　学　　号：181002222

实验地点：家　　任课教师：蔡娟

实验题目：实验 1 存储管理----主存空间的分配与回收

实验环境：Visual Studio 2019 Community

一、 实验目的

1. 存储管理方式及其实现过程的理解；

2．理解计算机系统的资源如何组织，操作系统如何有效地管理这些系统资源，对于训练学生掌握程序设计、熟悉上机操作和程序调试技术都有重要作用。重点培养学生的思维能力、设计能力、创新能力和排错能力。

二、 实验内容

（1）使用高级程序设计语言(C 语言或 C++或 Java)完成存储器动态分区分配算法的模拟实现，包括分配，回收，分区碎片整理等过程。

- ✓ 初始化功能：内存状态设置为初始状态。
- ✓ 分配功能：要求使用两种算法首次(适应算法和最佳适应算法)。
- ✓ 回收功能：
- ✓ 空闲块的合并：即紧凑功能，用以消除碎片。当做碎片整理时，需要跟踪分配的空间，修改其引用以保证引用的正确性。
- ✓ 可以使用表来显示当前内存的使用状态。

三、 实验要求

1．基本思想

现代的操作系统支持多用户程序在主存同时执行，能满足多道程序同时运行，需要存储管理技术是分区方式的，有固定分区和可变分区之分。可变分区的分配（算法包括最先分配算法，最佳分配算法等）。

现代计算机支持虚拟存储系统。最常用的虚拟存储系统是页式虚拟存储管理，其允许把一个程序的多个页面存放到若干不相邻的主存页面中。

动态分区是指在作业执行前并不建立分区，分区的建立是在作业处理过程中进行的。且其大小可随作业或进程对内存的要求而改变，使得为作业或进程分配的分区大小恰好等于该程序的需求量。显然动态分区有较大的灵活性，较之固定分区能获得好的内存利用率。

注意：在实际的算法实现当中还应考虑在回收用户释放的内存块后对内存空间的重新组织，合并连续的内存块，即“节点合并”的问题。因为系统在不断的分配和回收过程中，使得大的空闲块逐渐被分割成小的占用块，然后又重新成为空闲块被系统回收，如此反复，地址相邻的两块空闲块却分别作为链表中的节点存在，导致出现较大的用户“请求”时，虽然有足够的空闲空间，但是他们分布在多个空闲块内，对系统是透明的，分配无法进行。为了使内存空间得到更有效的利用，就要求在回收内存块后对内存空间进行重新组织，合并连续的内存空间，使至最大限度的满足将来的分配需求。

2 实验的具体内容

假设初始状态下，可用的内存空间为 640KB，并有下列的一个作业申请队列以及作业完成后的释放顺序，实现主存的分配和回收。

作业 1 申请 130KB
作业 2 申请 60KB
作业 3 申请 100KB
作业 2 释放 60KB
作业 4 申请 200KB
作业 3 释放 100KB
作业 1 释放 130KB
作业 5 申请 140KB
作业 6 申请 60KB
作业 7 申请 50KB
作业 6 释放 60KB
请分别采用首次适应算法和最佳适应算法进行内存的分配，要求每次分配和回收后显示出空闲内存分区表的情况以及各作业的申请、释放情况显示、打印出来。

四、 源程序及注释

初始化内存空间和作业队列

```
Status Initblock()//开创带头结点的内存空间链表
{
    block_first = (DuLinkList)malloc(sizeof(DuLNode));
    block_last = (DuLinkList)malloc(sizeof(DuLNode));
    block_first->prior = NULL;
    block_first->next = block_last;
    block_last->prior = block_first;
    block_last->next = NULL;
    block_last->data.address = 0;
    block_last->data.size = MAX_length;
    block_last->data.state = Free;
    return OK;
}

Status Initwork()
{
    work[0].address = 1; work[0].size = 130; work[0].state = 1;//申请
    work[1].address = 2; work[1].size = 60; work[1].state = 1;//申请
    work[2].address = 3; work[2].size = 100; work[2].state = 1;//申请
    work[3].address = 2; work[3].size = 60; work[3].state = 0;//释放
    work[4].address = 4; work[4].size = 200; work[4].state = 1;//申请
    work[5].address = 3; work[5].size = 100; work[5].state = 0;//释放
    work[6].address = 1; work[6].size = 130; work[6].state = 0;//释放
    work[7].address = 5; work[7].size = 140; work[7].state = 1;//申请
    work[8].address = 6; work[8].size = 60; work[8].state = 1;//申请
    work[9].address = 7; work[9].size = 50; work[9].state = 1;//申请
    work[10].address = 6; work[10].size = 60; work[10].state = 0;//释放
    return OK;
}
```

头文件,结构体,以及函数声明

```
#include<iostream>
#include<stdlib.h>
using namespace std;
#define Free 0 //空闲状态
#define Busy 1 //已用状态
#define OK 1    //完成
#define ERROR 0 //出错
#define MAX_length 640  //定义最大主存信息640KB
typedef int Status;
int flag;//标志位  0为空闲区     1为已分配的工作区

typedef struct FreAarea//定义一个空闲区说明表结构
{
    long size;   //分区大小
    long address; //分区地址
    int state;   //状态
}ElemType;


typedef struct DuLNode// 线性表的双向链表存储结构
{
    ElemType data;
    struct DuLNode* prior; //前趋指针
    struct DuLNode* next;  //后继指针
}


DuLNode, * DuLinkList;
DuLinkList block_first; //头结点
DuLinkList block_last;  //尾结点
ElemType work[11];//作业队列
Status Alloc(int ch , int request);//内存分配
Status free(int); //内存回收
Status First_fit(int);//首次适应算法
Status Best_fit(int); //最佳适应算法
//Status Worst_fit(int); //最差适应算法
void show();//查看分配
Status Initblock();//开创空间表
Status Initwork();//初始化作业队列
```

分配主存

```
Status Alloc(int ch, int request)//分配主存
{
    if (request < 0 || request == 0)
    {
        cout << "分配大小不合适，请重试！" << endl;
        return ERROR;
    }

    if (ch == 2) //选择最佳适应算法
    {
        if (Best_fit(request) == OK) cout << "分配成功！" << endl;
        else cout << "内存不足，分配失败！" << endl;
        return OK;
    }
    else //默认首次适应算法
    {
        if (First_fit(request) == OK) cout << "分配成功！" << endl;
        else cout << "内存不足，分配失败！" << endl;
        return OK;
    }
}
```

首次适应算法

```
Status First_fit(int request)//首次适应算法
{
    //为申请作业开辟新空间且初始化
    DuLinkList temp = (DuLinkList)malloc(sizeof(DuLNode));
    temp->data.size = request;
    temp->data.state = Busy;

    DuLNode* p = block_first->next;
    while (p)
    {
        if (p->data.state == Free && p->data.size == request)
        {//有大小恰好合适的空闲块
            p->data.state = Busy;
            return OK;
            break;
        }
        if (p->data.state == Free && p->data.size > request)
        {//有空闲块能满足需求且有剩余
            temp->prior = p->prior;
            temp->next = p;
            temp->data.address = p->data.address;
            p->prior->next = temp;
            p->prior = temp;
            p->data.address = temp->data.address + temp->data.size;
            p->data.size -= request;
            return OK;
            break;
        }
        p = p->next;
    }
    return ERROR;
}
```

最佳适应算法

```
Status Best_fit(int request)//最佳适应算法
{
    int ch; //记录最小剩余空间
    DuLinkList temp = (DuLinkList)malloc(sizeof(DuLNode));
    temp->data.size = request;
    temp->data.state = Busy;
    DuLNode* p = block_first->next;
    DuLNode* q = NULL; //记录最佳插入位置

    while (p) //初始化最小空间和最佳位置
    {
        if (p->data.state == Free && (p->data.size >= request))
        {
            if (q == NULL)
            {
                q = p;
                ch = p->data.size - request;
            }
            else if (q->data.size > p->data.size)
            {
                q = p;
                ch = p->data.size - request;
            }
        }
        p = p->next;
    }

    if (q == NULL) return ERROR;//没有找到空闲块
    else if (q->data.size == request)
    {
        q->data.state = Busy;
        return OK;
    }
    else
    {
        temp->prior = q->prior;
        temp->next = q;
        temp->data.address = q->data.address;
        q->prior->next = temp;
        q->prior = temp;
        q->data.address += request;
        q->data.size = ch;
        return OK;
    }
    return OK;
```

主存回收与显示主存使用情况

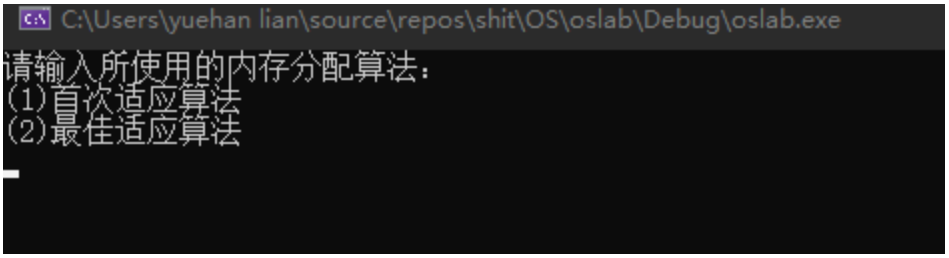
```
Status free(int flag)//主存回收
{
    DuLNode* p = block_first;
    while (p != NULL) {
        p = p->next;
        if (p->data.size == flag)
            break;
    }
    p->data.state = Free;
    if (p->prior != block_first && p->prior->data.state == Free)//与前面的空闲块相连
    {
        p->prior->data.size += p->data.size;//空间扩充,合并为一个
        p->prior->next = p->next;//去掉原来被合并的p
        p->next->prior = p->prior;
        p = p->prior;
    }
    if (p->next != block_last && p->next->data.state == Free)//与后面的空闲块相连
    {
        p->data.size += p->next->data.size;//空间扩充,合并为一个
        p->next->next->prior = p;
        p->next = p->next->next;
    }
    if (p->next == block_last && p->next->data.state == Free)//与最后的空闲块相连
    {
        p->data.size += p->next->data.size;
        p->next = NULL;
    }

    return OK;
}

void show()//显示主存分配情况
{
    int flag = 0;
    cout << "\n主存分配情况:\n";
    cout << "+++++++++++++++++++++++++++++++++++++++++++++++\n\n";
    DuLNode* p = block_first->next;
    cout << "分区号\t起始地址\t分区大小\t状态\n\n";
    while (p)
    {
        cout << "  " << flag++ << "\t";
        cout << "  " << p->data.address << "\t\t";
        cout << " " << p->data.size << "KB\t\t";
        if (p->data.state == Free) cout << "空闲\n\n";
        else cout << "已分配\n\n";
        p = p->next;
    }
    cout << "+++++++++++++++++++++++++++++++++++++++++++++++\n\n";
}
```

主函数

```
int main()//主函数
{
    int ch;//算法选择标记
    cout << "请输入所使用的内存分配算法：\n";
    cout << "(1)首次适应算法\n(2)最佳适应算法\n";

    cin >> ch;
    while (ch < 1 || ch>2)
    {
        cout << "输入错误，请重新输入所使用的内存分配算法：\n";
        cin >> ch;
    }

    Initblock(); //开创空间表
    int choice;  //操作选择标记
    Initwork(); //开创空间表
    for(int i=0;i<11;++i)
    {
        show();
        cout << endl << "======================" << endl;
        choice = work[i].state;
        if (choice == 1) {

            Alloc(ch,work[i].size); // 分配内存
            cout << "作业" << work[i].address << "分配了" << work[i].size << "KB" << endl;
        }
        else  // 内存回收
        {
            int flag;
            flag = work[i].size;
            free(flag);
            cout << "作业" << work[i].address << "释放了" << work[i].size << "KB" << endl;
        }
    }
    cout <<endl<< "本次程序已结束,以下是最后的内存分配情况！" << endl;
    show();
}
```

五、 运行效果

单击运行



选择自己要使用的算法

输入 1，首次适应算法

```
主存分配情况:
+++++++++++++++++++++++++++++++++++++++++++++++
分区号   起始地址         分区大小         状态
  0        0               640KB           空闲
+++++++++++++++++++++++++++++++++++++++++++++++


======================
分配成功!
作业1分配了130KB
主存分配情况:
+++++++++++++++++++++++++++++++++++++++++++++++
分区号   起始地址         分区大小         状态
  0        0               130KB           已分配
  1        130             510KB           空闲
+++++++++++++++++++++++++++++++++++++++++++++++


======================
分配成功!
作业2分配了60KB
主存分配情况:
+++++++++++++++++++++++++++++++++++++++++++++++
分区号   起始地址         分区大小         状态
  0        0               130KB           已分配
  1        130             60KB            已分配
  2        190             450KB           空闲
+++++++++++++++++++++++++++++++++++++++++++++++


======================
分配成功!
作业3分配了100KB
主存分配情况:
+++++++++++++++++++++++++++++++++++++++++++++++
分区号   起始地址         分区大小         状态
  0        0               130KB           已分配
  1        130             60KB            已分配
  2        190             100KB           已分配
  3        290             350KB           空闲
+++++++++++++++++++++++++++++++++++++++++++++++


======================
作业2释放了60KB
主存分配情况:
+++++++++++++++++++++++++++++++++++++++++++++++
分区号   起始地址         分区大小         状态
  0        0               130KB           已分配
  1        130             60KB            空闲
  2        190             100KB           已分配
  3        290             350KB           空闲
+++++++++++++++++++++++++++++++++++++++++++++++


======================
分配成功!
作业4分配了200KB
主存分配情况:
+++++++++++++++++++++++++++++++++++++++++++++++
分区号   起始地址         分区大小         状态
  0        0               130KB           已分配
  1        130             60KB            空闲
  2        190             100KB           已分配
  3        290             200KB           已分配
  4        490             150KB           空闲
+++++++++++++++++++++++++++++++++++++++++++++++


======================
作业3释放了100KB
```

```
======================
作业3释放了100KB

主存分配情况:
++++++++++++++++++++++++++++++++++++++++++++++++++
分区号   起始地址         分区大小         状态
  0        0               130KB           已分配
  1        130             160KB           空闲
  2        290             200KB           已分配
  3        490             150KB           空闲
++++++++++++++++++++++++++++++++++++++++++++++++++


======================
作业1释放了130KB

主存分配情况:
++++++++++++++++++++++++++++++++++++++++++++++++++
分区号   起始地址         分区大小         状态
  0        0               290KB           空闲
  1        290             200KB           已分配
  2        490             150KB           空闲
++++++++++++++++++++++++++++++++++++++++++++++++++


======================
分配成功!
作业5分配了140KB

主存分配情况:
++++++++++++++++++++++++++++++++++++++++++++++++++
分区号   起始地址         分区大小         状态
  0        0               140KB           已分配
  1        140             150KB           空闲
  2        290             200KB           已分配
  3        490             150KB           空闲
++++++++++++++++++++++++++++++++++++++++++++++++++


======================
分配成功!
作业6分配了60KB

主存分配情况:
++++++++++++++++++++++++++++++++++++++++++++++++++
分区号   起始地址         分区大小         状态
  0        0               140KB           已分配
  1        140             60KB            已分配
  2        200             90KB            空闲
  3        290             200KB           已分配
  4        490             150KB           空闲
++++++++++++++++++++++++++++++++++++++++++++++++++


======================
分配成功!
作业7分配了50KB

主存分配情况:
++++++++++++++++++++++++++++++++++++++++++++++++++
分区号   起始地址         分区大小         状态
  0        0               140KB           已分配
  1        140             60KB            已分配
  2        200             50KB            已分配
  3        250             40KB            空闲
  4        290             200KB           已分配
  5        490             150KB           空闲
++++++++++++++++++++++++++++++++++++++++++++++++++


======================
作业6释放了60KB

本次程序已结束,以下是最后的内存分配情况!

主存分配情况:
++++++++++++++++++++++++++++++++++++++++++++++++++
分区号   起始地址         分区大小         状态
  0        0               140KB           已分配
  1        140             60KB            空闲
  2        200             50KB            已分配
  3        250             40KB            空闲
  4        290             200KB           已分配
  5        490             150KB           空闲
++++++++++++++++++++++++++++++++++++++++++++++++++
```

输入 2，最佳适应算法

```
2
主存分配情况:
++++++++++++++++++++++++++++++++++++++++++++++++
分区号  起始地址        分区大小        状态
  0       0              640KB          空闲
++++++++++++++++++++++++++++++++++++++++++++++++


======================
分配成功!
作业1分配了130KB
主存分配情况:
++++++++++++++++++++++++++++++++++++++++++++++++
分区号  起始地址        分区大小        状态
  0       0              130KB          已分配
  1       130            510KB          空闲
++++++++++++++++++++++++++++++++++++++++++++++++


======================
分配成功!
作业2分配了60KB
主存分配情况:
++++++++++++++++++++++++++++++++++++++++++++++++
分区号  起始地址        分区大小        状态
  0       0              130KB          已分配
  1       130            60KB           已分配
  2       190            450KB          空闲
++++++++++++++++++++++++++++++++++++++++++++++++


======================
分配成功!
作业3分配了100KB
主存分配情况:
++++++++++++++++++++++++++++++++++++++++++++++++
分区号  起始地址        分区大小        状态
  0       0              130KB          已分配
  1       130            60KB           已分配
  2       190            100KB          已分配
  3       290            350KB          空闲
++++++++++++++++++++++++++++++++++++++++++++++++


======================
作业2释放了60KB
主存分配情况:
++++++++++++++++++++++++++++++++++++++++++++++++
分区号  起始地址        分区大小        状态
  0       0              130KB          已分配
  1       130            60KB           空闲
  2       190            100KB          已分配
  3       290            350KB          空闲
++++++++++++++++++++++++++++++++++++++++++++++++


======================
分配成功!
作业4分配了200KB
主存分配情况:
++++++++++++++++++++++++++++++++++++++++++++++++
分区号  起始地址        分区大小        状态
  0       0              130KB          已分配
  1       130            60KB           空闲
  2       190            100KB          已分配
  3       290            200KB          已分配
  4       490            150KB          空闲
++++++++++++++++++++++++++++++++++++++++++++++++
```

```
======================
作业3释放了100KB

主存分配情况:
++++++++++++++++++++++++++++++++++++++++++++++++
分区号   起始地址         分区大小         状态
  0        0              130KB          已分配
  1        130            160KB          空闲
  2        290            200KB          已分配
  3        490            150KB          空闲
++++++++++++++++++++++++++++++++++++++++++++++++


======================
作业1释放了130KB

主存分配情况:
++++++++++++++++++++++++++++++++++++++++++++++++
分区号   起始地址         分区大小         状态
  0        0              290KB          空闲
  1        290            200KB          已分配
  2        490            150KB          空闲
++++++++++++++++++++++++++++++++++++++++++++++++


======================
分配成功!
作业5分配了140KB

主存分配情况:
++++++++++++++++++++++++++++++++++++++++++++++++
分区号   起始地址         分区大小         状态
  0        0              290KB          空闲
  1        290            200KB          已分配
  2        490            140KB          已分配
  3        630            10KB           空闲
++++++++++++++++++++++++++++++++++++++++++++++++


======================
分配成功!
作业6分配了60KB

主存分配情况:
++++++++++++++++++++++++++++++++++++++++++++++++
分区号   起始地址         分区大小         状态
  0        0              60KB           已分配
  1        60             230KB          空闲
  2        290            200KB          已分配
  3        490            140KB          已分配
  4        630            10KB           空闲
++++++++++++++++++++++++++++++++++++++++++++++++


======================
分配成功!
作业7分配了50KB

主存分配情况:
++++++++++++++++++++++++++++++++++++++++++++++++
分区号   起始地址         分区大小         状态
  0        0              60KB           已分配
  1        60             50KB           已分配
  2        110            180KB          空闲
  3        290            200KB          已分配
  4        490            140KB          已分配
  5        630            10KB           空闲
++++++++++++++++++++++++++++++++++++++++++++++++


======================
作业6释放了60KB

本次程序已结束,以下是最后的内存分配情况!

主存分配情况:
++++++++++++++++++++++++++++++++++++++++++++++++
分区号   起始地址         分区大小         状态
  0        0              60KB           空闲
  1        60             50KB           已分配
  2        110            180KB          空闲
  3        290            200KB          已分配
  4        490            140KB          已分配
```

六、 实验心得

通过这次实验，让我对首次适应算法和最佳适应算法有了更深刻的认识，提高了 C++代码的熟练程度，以及对内存的分配方法有了具象的理解。明白了书本和 PPT 中的知识如果转化成自身实践的代码形式，会有不一样的印象，跳跃出平面之外，有了立体生动的程序来重新叙述这个过程。