学号 P21714001 专业 计算机英才班 姓名 刘峰

实验日期 **2019.12.8**  教师签字 成绩

实验报告

【实验名称】  **首次适应和最佳适应算法**

【实验目的】

掌握主存的分配与回收

【实验原理】

首次适应算法从空闲分区链首开始查找，直至找到一个能满足其大小要求的空闲分区为止。然后再按照作业的大小，从该分区中划出一块内存分配给请求者，余下的空闲分区仍留在空闲分区链中。

 特点：该算法倾向于使用内存中低地址部分的空闲区，在高地址部分的空闲区很少被利用，从而保留了高地址部分的大空闲区。显然为以后到达的大作业分配大的内存空间创造了条件。

  缺点：低地址部分不断被划分，留下许多难以利用、很小的空闲区，而每次查找又都从低地址部分开始，会增加查找的开销。

最佳适应算法总是把既能满足要求，又是最小的空闲分区分配给作业。为了加速查找，该算法要求将所有的空闲区按其大小排序后，以递增顺序形成一个空白链。这样每次找到的第一个满足要求的空闲区，必然是最优的。孤立地看，该算法似乎是最优的，但事实上并不一定。因为每次分配后剩余的空间一定是最小的，在存储器中将留下许多难以利用的小空闲区。同时每次分配后必须重新排序，这也带来了一定的开销。

特点：每次分配给文件的都是最合适该文件大小的分区.

 缺点：内存中留下许多难以利用的小的空闲区。

【实验内容】

#include <iostream>

#include <fstream>

#include <iomanip>

using namespace std;

#define MAXNUMBER 100

static int PartitionNum; //内存中空闲分区的个数

static int ProcessNum; //需要分配的进程个数

static int FreePartition[MAXNUMBER]; //空闲分区对应的内存

static int ProcessNeed[MAXNUMBER]; //需要分配的进程大小

static int LeftFreePartition[MAXNUMBER];

static int LeftProcessNeed[MAXNUMBER];

static char ProcessName[MAXNUMBER];

static char NameProcessToPartition[MAXNUMBER][MAXNUMBER];

typedef struct

{

int partitionSize;

int id;

}sortNeed;

void readDataFunction();

void input();

void display();

void FirstFit();

void NextFit();

void BestFit();

void WorstFit();

void selectAlgorithm(int chooceAlgorithm);

void display();

void readDataFunction()

{

ifstream readData;

readData.open("data.txt");

readData>>PartitionNum;

for (int i=0;i<PartitionNum;i++)

{

readData>>FreePartition[i];

}

readData>>ProcessNum;

for (int i=0;i<ProcessNum;i++)

{

readData>>ProcessName[i];

}

for (int i=0;i<ProcessNum;i++)

{

readData>>ProcessNeed[i];

}

}

void input()

{

cout<<"请选择实现的算法："<<endl;

cout<<"\*\*\*\*\* 1 - 首次适应算法 \*\*\*\*\*"<<endl;

cout<<"\*\*\*\*\* 2 - 最佳适应算法 \*\*\*\*\*"<<endl;

cout<<"\*\*\*\*\* 0 - 结束 \*\*\*\*\*"<<endl;

int chooseAlgorithm = 5;

while(chooseAlgorithm)

{

cout<<"chooseAlgorithm = ";

cin>>chooseAlgorithm;

selectAlgorithm(chooseAlgorithm);

}

}

void initial()

{

readDataFunction(); //读取原始数据

for (int i=0;i<PartitionNum;i++)

{

LeftFreePartition[i] = FreePartition[i];

for (int j =0;j<ProcessNum;j++)

{

NameProcessToPartition[i][j] =NULL;

}

}

for (int i = 0;i<ProcessNum;i++)

{

LeftProcessNeed[i] = ProcessNeed[i];

}

}

void FirstFit()

{

cout<<"\*\*\*\*\*\*\*\*\*\*\*首次适应算法\*\*\*\*\*\*\*\*\*\*\*"<<endl;

initial();

int i,j;

for (i = 0;i<ProcessNum;i++) //逐个遍历每个进程

{

for (j = 0;j<PartitionNum;j++)

{

if (LeftProcessNeed[i] <= LeftFreePartition[j] && LeftFreePartition!=0) //当系统内存分区足够大的时候，即分配给进程资源

{

LeftFreePartition[j] -= LeftProcessNeed[i]; //扣除分配给进程的资源

LeftProcessNeed[i] = 0; //当且仅当系统内存分区足够时才执行，即当前进程大小置0

NameProcessToPartition[i][j] = ProcessName[i]; //存储各个进程所在的分区位置

break; //很重要，一个进程分区完后，应该立即break，进行下一个进程的判断

}

}

}

display();

}

void BestFit()

{

cout<<"\*\*\*\*\*\*\*\*\*\*\*最佳适应算法\*\*\*\*\*\*\*\*\*\*\*"<<endl;

initial();

int i,j,temp,tempID;

sortNeed best[MAXNUMBER];

for (i = 0;i<PartitionNum;i++)

{

//初始化结构体

best[i].partitionSize = FreePartition[i];

best[i].id = i;

}

for (i = 0;i<ProcessNum;i++)

{

for (int s = PartitionNum - 1;s > 0;s--) //冒泡排序(每次分配完一个进程后，都需要重新排序)

{

for (int t = 0;t < s;t++)

{

if (best[s].partitionSize < best[t].partitionSize)

{

temp = best[s].partitionSize;

best[s].partitionSize = best[t].partitionSize;

best[t].partitionSize = temp;

tempID = best[s].id;

best[s].id = best[t].id;

best[t].id = tempID;

}

}

}

for (j = 0;j<PartitionNum;j++)

{

if (LeftProcessNeed[i] <= best[j].partitionSize)

{

best[j].partitionSize -= LeftProcessNeed[i];

LeftProcessNeed[i] = 0;

NameProcessToPartition[i][best[j].id] = ProcessName[i];

break;

}

}

LeftFreePartition[best[j].id] = best[j].partitionSize;

}

display();

}

void selectAlgorithm(int chooseAlgorithm)

{

switch(chooseAlgorithm)

{

case 0:break;

case 1:FirstFit();break;

case 2:BestFit();break;

default:cout<<"请输入正确的序号："<<endl;

}

}

void display()

{

int i;

cout<<"需要分配内存的进程名："<<setw(10);

for (i = 0;i<ProcessNum;i++)

{

cout<<ProcessName[i]<<setw(6);

}

cout<<endl;

cout<<"需要分配内存的进程分区大小："<<setw(4);

for (i = 0;i<ProcessNum;i++)

{

cout<<ProcessNeed[i]<<setw(6);

}

cout<<endl;

cout<<"分配结果："<<endl;

cout<<"分区序号:";

for (i = 0;i<PartitionNum;i++)

{

cout<<"分区"<<i+1<<" ";

}

cout<<endl<<"分区大小:";

for (i = 0;i<PartitionNum;i++)

{

cout<<FreePartition[i]<<" ";

}

cout<<endl<<"剩余大小:";

for (i = 0;i<PartitionNum;i++)

{

cout<<LeftFreePartition[i]<<" ";

}

cout<<endl<<"分配进程情况:"<<endl;

for (i = 0;i<PartitionNum;i++)

{

for (int j = 0;j<ProcessNum;j++)

{

if (NameProcessToPartition[j][i]!=NULL)

{

cout<<NameProcessToPartition[j][i]<<": (分区"<<i+1<<")"<<endl;

}

}

//cout<<" ";

}

cout<<endl<<"\*\*\*\*\*\*\*\*结束\*\*\*\*\*\*\*\*\*\*"<<endl;

}

int main()

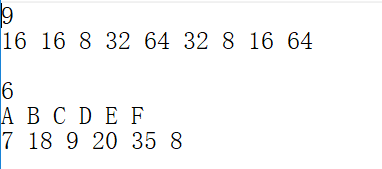
{

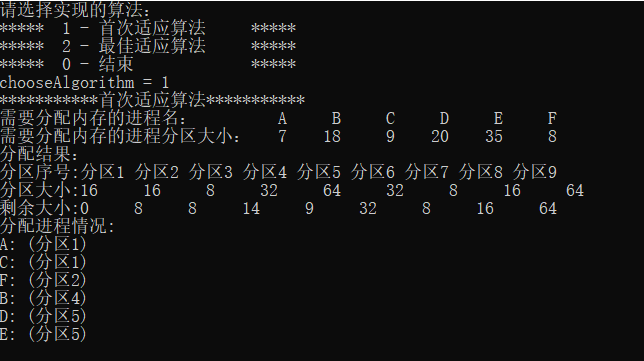
input();

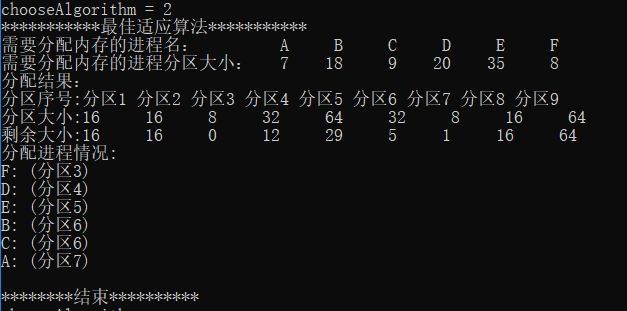
return 0;

}

初始化：







【实验总结】

通过本次实验更加深入了解了首次适应算法和最佳适应算法算法工作以及具体实现过程，增加了我对虚拟内存管理的认识。实验实现过程也加深对所学数据结构以及相关算法的理解与回顾。