**郑州轻工业大学**

**《操作系统》课程****实验报告**

实验名称：基本分页存储管理

姓 名： 原彬贺

学 号： 542001020223

地 点： 实验楼302

指导老师： 吴庆岗

专业班级： 计算机科学与技术20-02班

实验成绩：

1. 实验目的：

熟悉并掌握基本分页存储管理的思想及其实现方法，熟悉并掌握基本分页存储管理的分配方式和回收方式。

二、实验内容：

模拟实现基本分页存储管理方式下内存空间的分配和回收。实验内容如下：

内存空间的初始化——可以由用户输入初始内存空间各个物理块情况。（用二维矩阵的方式按物理块号，逐行给出每个物理块的状态，1——表示已分配，0——表示未分配，并能够将行标、列标转换为对应的物理块号，以查看或修改每一个块的状态，要求：初始时部分物理块已分配）。

基本分页的分配过程：由用户输入作业号和作业的大小（这里的大小是逻辑页面数），实现分配过程：空间充足，分配，修改状态矩阵的相应位置的值（值由0转变为1），并用专门的数据结构记录下该作业占用的物理块的块号，以备删除作业时回收空间。

作业空间的回收：用户输入作业号，实现分区回收（通过相应的数据结构找到该作业占有的物理块号，将块号转变成对应的行标、列标，将对应位置的值由1转变成0就完成了回收）。

分区的显示：任何时刻，可以查看当前内存的情况（显示记录内存情况的矩阵的值）。

实验要求如下：

（1）内存空间不足的情况，要有相应的显示；

（2）作业不能同名，但是删除后可以再用这个名字；

（3）作业空间回收时输入作业名，回收相应的空间，如果这个作业名不存在，也要有相应的提示。

三、实验代码

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <time.h>

#define PAGE 100 //内存总量100块

int memory[PAGE];

int process[PAGE][PAGE + 1]; //进程数组

int use\_num; //已经使用内存

int process\_num; //进程数目

void check()

{

printf("内存总量: %d块，已用空间: %d块，剩余空间: %d块，进程总数: %d个\n", PAGE, use\_num, PAGE - use\_num, process\_num);

if (use\_num < PAGE) {

printf("已使用的内存块(%d):\n", use\_num);

int tmp = 1; //用于换行

for (int i = 0; i < PAGE; i++) {

if (memory[i] == 1) {

if (tmp == 15) {

printf("%d\n", i);

tmp = 1;

}

printf("%d ", i);

tmp++;

}

}

}

if (process\_num > 0) {

printf("\n各进程详细使用情况如下: \n");

for (int i = 0; i < PAGE; i++) {

if (process[i][0] > 0) {

printf("进程号：%d \n占用内存块 (%d块)：", i, process[i][0]);

int tmp = 1;

for (int j = 1; j <= process[i][0]; j++)

{

if (tmp == 15) {

printf("%d\n", process[i][j]);

tmp = 1;

}

printf("%d ", process[i][j]);

tmp++;

}

}

}

}

else {

printf("\n当前内存无进程块!\n");

}

}

void init\_memory() //随机占用18块内存块

{

memset(memory, 0, sizeof(memory)); //将所有内存块置为0，表示未使用

const int count = 18;

srand((unsigned)time(NULL)); //初始化随机数

for (int i = 0; i < count; i++) {

memory[rand() % PAGE] = 1; //产生0~99的随机数

}

use\_num = count; //初始化已经使用的内存

for (int i = 0; i < PAGE; i++) { //初始化进程数组

process[i][0] = 0;

for (int j = 1; j < PAGE; j++)

process[i][j] = -1;

}

process\_num = 0; //初始化进程数

printf("初始化结果如下:\n");

check();

}

void build\_process() //创建进程

{

int num, size;

printf("请输入进程号（小于%d）和所需页面: ", PAGE);

scanf("%d%d", &num, &size);

if (num >= PAGE) {

printf("输出错误，进程号越界!\n");

return;

}

if (PAGE - use\_num < size) {

printf("内存空间不足\n");

return;

}

use\_num += size;

process[num][0] = size;

int k = 0;

for (int i = 1; i <= size; i++) { //检查是否存在相同进程号的进程

while (memory[k] == 1 && k < PAGE) {

k++;

}

process[num][i] = k;

memory[k++] = 1;

}

process\_num++;

printf("创建新进程成功!\n");

}

void end\_process() //结束进程

{

int num, size;

if (process\_num < 1) {

printf("当前内存没有进程!\n");

return;

}

printf("请输入进程号（小于%d）: ", PAGE);

scanf("%d", &num);

if (process[num][0] == 0) {

printf("该进程不存在!\n");

return;

}

size = process[num][0];

for (int j = 1; j < size; j++)

{

memory[process[num][j]] = 0;

process[num][j] = -1;

}

process[num][0] = 0;

process\_num--;

use\_num -= size;

printf("结束进程成功!\n");

}

void menu() //菜单

{

printf("\n操作菜单:\n");

printf("1、创建进程\n");

printf("2、结束进程\n");

printf("3、查看内存\n");

printf("0、退出程序\n");

printf("请输入您要进行的操作: ");

}

int main(int argc, char\* argv[])

{

init\_memory(); /\*初始化内存空间\*/

int choice;

menu();

while (scanf("%d", &choice) != 0) {

switch (choice)

{

case 1:

build\_process();

break;

case 2:

end\_process();

break;

case 3:

check();

break;

case 0:

printf("程序已经退出!\n");

exit(0);

default:

printf("输入错误，请重新输入\n");

break;

}

menu();

}

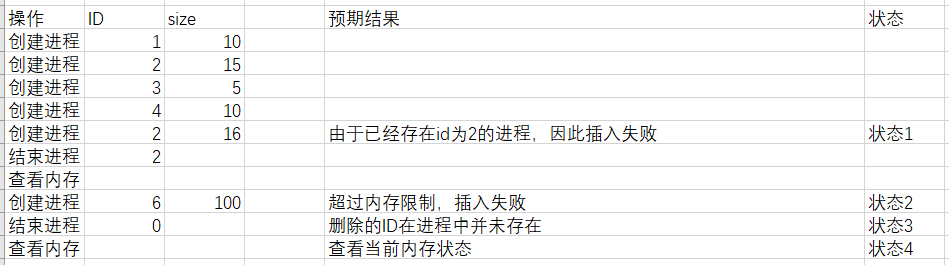
system("pause");

return 0;

}

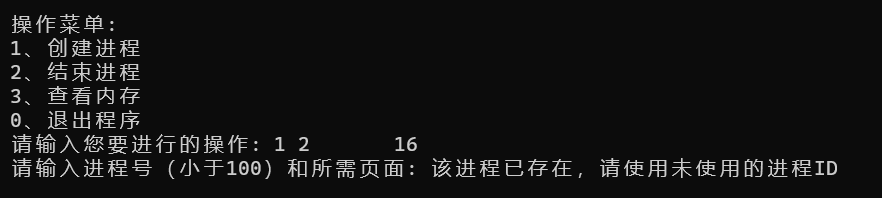
四、实验结果

输入测试：



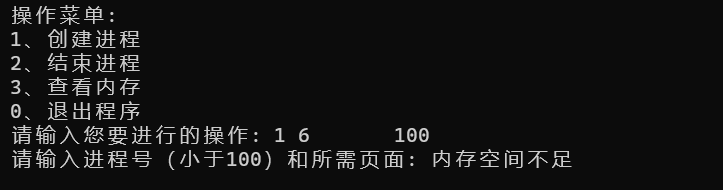
输出结果

状态1：由于已经存在id为2的进程，因此插入失败



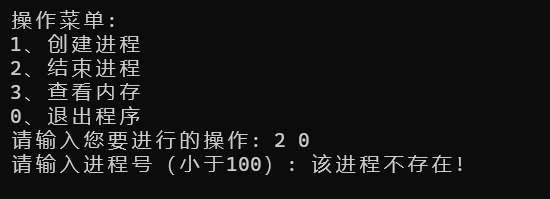
图表 1插入相同id

状态2：超过内存限制，插入失败



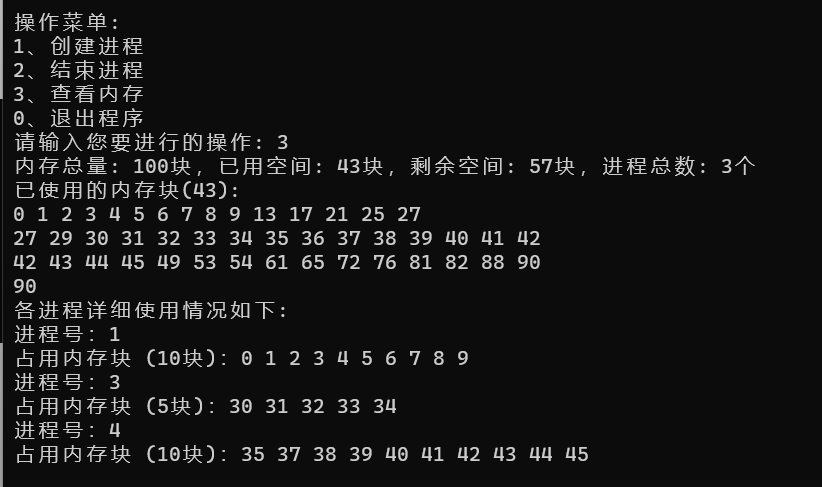
图表 2超过内存限制

状态3: 删除的ID在进程中并未存在



图表 3删除进程不存在

状态4：查看当前内存状态



图表 4查看进程

五、实验总结

本次实验通过对进程进行分配内存块，同时支持对进程内存块的回收。本次实验我通过编写程序，自己亲身体会进程内存块的分配，更加深入理解了操作系统中进程内存块分配的基本原理。同时本次实验也让我对编程更加熟悉，锻炼了我的动手能力。使我收获很多，让我能够在接下来对操作系统的理解更加深入，带给了我一个深入了解操作系统的窗口。

|  |
| --- |
| **评语**（不要删除，指导老师填写评语，放置在实验报告后面单独成页） |
|  |