中南大學
CENTRAL SOUTH UNIVERSITY

操作系统原理实验报告

学生姓名	张文睿
学　　号	8207211004
专业班级	计算机科学与技术 2101
指导教师	沈海澜
学　　院	计算机学院
完成时间	2023.6.2

计算机学院

2023 年 6 月

目录

实验 进程管理与内存分配模拟程序

一、【实验目的】

(1) 通过模拟实现进程的创建、阻塞、唤醒、撤销等进程控制原语，加深操作系统进程控制原语主要任务和过程的理解。

(2) 进一步熟悉进程调度和内存分配的基本策略，加深操作系统以进程为核心的完整架构的理解。

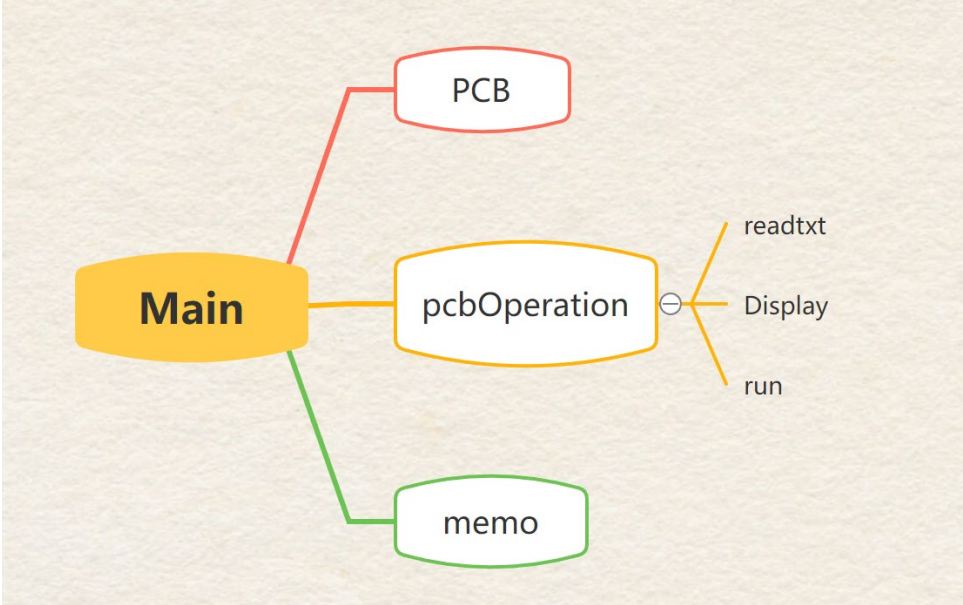
(3) 提高综合性实验的分析、设计及编程实现能力。

二、【实验内容】

（1）设计一个 OS 进程管理模拟程序，模拟实现 OS 创建进程原语、阻塞进程原语、唤醒进程原语，终止进程原语、调度进程原语等功能；每个进程用一个 PCB 表示，其内容可根据具体情况设置。进程调度算法可选择 FCFS、RR、SJF 或优先级中任意一种。内存分配可采用可变分区策略或页式内存分配方案中任意一种，进程创建时需为进程分配内存空间，进程终止时需要回收进程的内存空间。

（2）程序在运行过程中能提供前端交互界面，用于提交作业基本信息以及显示或打印各进程的状态及有关参数的变化情况（类似于 Linux 中的 ps 命令），以便观察诸进程的运行情况。

三、【实验设计】



本程序除 Main 类外，共三个类：Memory，PCB 和 PCBOperations。

其中 Memory 类表示内存块，包含了内存块的地址和大小属性；PCB 类表示进程控制块，包含了描述进程的一系列属性，如到达时间，服务时间，进程状态，所需内存大小等等。

PCBOperations 类是本程序的关键类，用于处理 PCB 的一系列操作，如读取进程信息，显示当前进程，增加进程，阻塞进程，唤醒进程，终止进程，模拟进程运行等。

算法方面，在进程调度上使用了先入先出的的算法，内存分配采用的是动态分配的首次适应算法。

四、程序介绍

① 增加进程：

当用户选择为增加进程时，进入增加进程功能。

首先让用户输入进程名称，并遍历判断名称是否重复，若重复则增加失败，若不重复，则分别让用户输入进程的到达时间，运行时长，所占内存大小等。在输入完毕后，根据进程到达时间，将新创建的进程插入到对应位置。若进程到达时间一致，则根据创建的顺序排列。

② 阻塞进程

当用户选择为阻塞进程时，进入阻塞进程功能。

首先让用户输入拟阻塞的进程名称，后遍历找到待阻塞进程，将其移出进程列表 pcb，加入到阻塞进程列表 temp。调整完毕后，更新当前存储空间容量。

③ 唤醒进程

当用户选择为唤醒进程时，进入唤醒进程功能。

首先判断阻塞进程列表的大小，若其大小为 0，即无进程处于阻塞状态，提示后退出。

接下来遍历阻塞进程列表，找到拟唤醒的进程，将其移出列表，再根据其到达时间的大小，按照从小到大的顺序插入到进程列表的对应位置，并更新当前的存储情况。

④ 移除进程

当用户选择为移除进程时，进入移除进程功能。

遍历进程列表，找到拟移除的进程，直接删除当前进程即可。

⑤ 启动进程

当用户选择进行下一步时，则默认进入调度进程功能。

首先初始化进程的开始运行时间，完成时间为 0，进程队列为空，创建 Memory 对象 me，存储块起始地址为 0。接下来遍历进程列表，将其依次添加到进程队列中。

然后循环判断，取队列头的元素，更新该进程的实际开始时间和完成时间。然后根据进程的标志 flag 进行判断，若 flag 为 false，即该进程尚未分配内存，因而进行内存分配，并更新总内存和对应起始地址信息。

遍历完所有进程队列中的进程后，开始遍历阻塞进程队列中的进程。首先让用户选择是否释放这些进程，若选择释放这一系列进程，则在原先内存分配和进程调度的基础上，按照相同方式调度这一系列进程。

若过程中出现总剩余内存小于 0 的情况，意味着内存不足，内存分配失败，因而给出相应提示信息后返回。

五、代码及运行结果

Main.java

```
import java.io.BufferedReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class Main {
    public static void main(String[] args) throws IOException {
        List<PCB> pcb = new ArrayList<>();
        pcbOperation pcbOperation = new pcbOperation();

        pcbOperation.readtxt(pcb);
    }
}
```

pcbOperation.java

```
import java.io.*;
import java.util.*;

public class pcbOperation {
    int totalMemory=1600;
    List<PCB> temp = new ArrayList<>();

    public void readtxt(List<PCB> pcb) throws IOException {
        BufferedReader br = null;
        Reader fr = null;
        try {
            fr = new FileReader("D:\\javacode\\os\\table.txt");
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
        br = new BufferedReader(fr);
        String str;
        while ((str = br.readLine()) != null) {
            int at;
            int nt;
            int me;
            String na;
```

```
                String[] nowEntry;
                if (str.startsWith("")) {
                    str = str.substring(1);
                }//排除异常
                if (str.endsWith(" ")) {
                    for (int i = str.length() - 1; i >= 0; i--) {
                        if (str.charAt(i) == ' ') {
                            str = str.substring(0, i);
                        } else {
                            break;
                        }
                    }
                }
                nowEntry = str.split(" {2}", 4);
                at = Integer.valueOf(nowEntry[0]);
                nt = Integer.valueOf(nowEntry[1]);
                na = nowEntry[3];
                me = Integer.valueOf(nowEntry[2]);
                PCB p = new PCB(at,me,nt,na,false,null);
                pcb.add(p);
            }
            Display(pcb);
        }
        public void Display(List<PCB> pcb){
            System.out.println("名称   到达时间   服务时长   所占内存");
            for(PCB p:pcb){
                System.out.println(p.name + "     "+p.arriveTime + "              " +
p.needTime + "              " + p.memory);
            }
            System.out.println("目前存在的进程如上所示，请输入您的选择：");
            while (true){
                System.out.println("0.无操作进入下一步");
                System.out.println("1.增加进程");
                System.out.println("2.阻塞进程");
                System.out.println("3.唤醒进程");
                System.out.println("4.终止进程");
                Scanner sc = new Scanner(System.in);
                String s = sc.next();
                switch (s){
                    case "0":break;
                    case "1":
                        System.out.println("请您按要求添加进程的所有信息");
                        System.out.println("请输入进程名称：");
```

```
                String sname = sc.next();
                System.out.println("请输入进程到达时间：");
                int at = sc.nextInt();
                boolean a=false;
                for (PCB b:pcb) {
                    if(b.arriveTime == at){
                        System.out.println("输入进程时间重复，请重新
输入！");
                        a=true;
                        break;
                    }
                }
                if(a==true){
                    break;
                }
                System.out.println("请输入进程运行时长：");
                int nt = sc.nextInt();
                System.out.println("请输入进程所占内存大小：");
                int me = sc.nextInt();
                PCB pcb1 = new PCB(at,me,nt,sname,false,null);
                boolean flag=false;
                for(int i=0;i<pcb.size();i++){

if(pcb1.arriveTime>=pcb.get(i).arriveTime&&pcb1.arriveTime<pcb.get(i+1).arriv
eTime){
                        pcb.add(i+1,pcb1);
                        flag=true;
                        break;
                    }
                }
                if(!flag){
                    if(pcb1.arriveTime<pcb.get(0).arriveTime){
                        pcb.add(0,pcb1);
                    }
                    else{
                        pcb.add(pcb1);
                    }
                }
                System.out.println("添加成功，以下是目前进程的情况：
");
                for(PCB p:pcb){
                    System.out.println(p.name + "     "+p.arriveTime + "
" + p.needTime + "          " + p.memory);
```

```
                }
                break;
            case "2":
                System.out.println("请输入你要阻塞的进程");
                String sn = sc.next();
                PCB t = null;
                for (int i=0;i<pcb.size();i++){
                    if(pcb.get(i).name.equals(sn)){
                        t=pcb.remove(i);
                        temp.add(t);
                        break;
                    }
                }
                if(t==null){
                    System.out.println("输入有误，请重新输入！");
                    break;
                }
                totalMemory -= t.memory;
                System.out.println(t.name + "已阻塞，目前就绪队列中的
进程如下:");
                for(PCB p:pcb){
                    System.out.println(p.name + "     "+p.arriveTime + "
" + p.needTime + "          " + p.memory);
                }
                break;
            case "3":
                if(temp.size()==0){
                    System.out.println("当前不存在就绪进程！");
                    break;
                }
                System.out.println("请输入想要唤醒的进程：");
                String snn = sc.next();
                PCB t2 = null;
                for (int i = 0; i < temp.size(); i++) {
                    if(temp.get(i).name.equals(snn)){
                        t2 = temp.remove(i);
                        break;
                    }
                }
                if(t2==null){
                    System.out.println("输入有误，请重新输入！");
                    break;
                }
```

```
                    boolean flag1=false;
                    for(int i=0;i<pcb.size()-1;i++){

if(t2.arriveTime>=pcb.get(i).arriveTime&&t2.arriveTime<pcb.get(i+1).arriveTim
e){
                            pcb.add(i+1,t2);
                            flag1=true;
                            break;
                        }
                    }
                    if(!flag1){
                        if(t2.arriveTime<pcb.get(0).arriveTime){
                            pcb.add(0,t2);
                        }
                        else{
                            pcb.add(t2);
                        }
                    }
                    totalMemory += t2.memory;
                    System.out.println("唤醒成功，下面是目前进程情况：");
                    for(PCB p:pcb){
                        System.out.println(p.name + "     "+p.arriveTime + "
" + p.needTime + "          " + p.memory);
                    }
                    break;
                case "4":
                    System.out.println("请输入你要移除的进程：");
                    Scanner scanner4 = new Scanner(System.in);
                    String ss = scanner4.next();
                    boolean f = false;
                    for (PCB c:pcb){
                        if (c.name.equals(ss)){
                            pcb.remove(c);
                            f = true;
                            break;
                        }
                    }
                    if(f==false){
                        System.out.println("输入有误，请重新输入！");
                        break;
                    }
                    System.out.println("移除成功，以下是目前进程的情况：
");
```

```
                    for(PCB p:pcb){
                        System.out.println(p.name + "      "+p.arriveTime + "
" + p.needTime + "            " + p.memory);
                    }
                }
                if(s.equals("0")){
                    break;
                }
            }
            while (true) {
                System.out.println("是否启动文件中的进程（y/n）");
                Scanner sc = new Scanner(System.in);
                String s = sc.next();
                if(s.equals("y")||s.equals("Y")){
                    run(pcb);
                    break;
                }else if(s.equals("n")||s.equals("N")){
                    return;
                }else {
                    System.out.println("输入有误！");
                }
            }
        }
        void run(List<PCB> pcb){
            int arriveTime=0;
            int endTime=0;
            Deque<PCB> queue = new ArrayDeque<>();
            for(PCB p:pcb){
                queue.addLast(p);
            }
            PCB p;
            int address=0;
            memo me = new memo();
            while (queue.peek()!=null){
                p=queue.pollFirst();
                arriveTime = Math.max(p.arriveTime,endTime);
                endTime = arriveTime+p.needTime;
                if(p.flag == false){
                    totalMemory -= p.memory;
                    address+=p.memory;
                    me.address = address;
                    me.memory = p.memory;
                    p.me = me;
```

```
                }
                p.flag = true;
                for(PCB b:queue){
                    if(endTime > b.arriveTime&&b.flag == false){
                        memo m = new memo();
                        b.flag = true;
                        totalMemory-=b.memory;
                        address+=b.memory;
                        m.address=address;
                        m.memory=b.memory;
                        b.me=m;
                    }
                }
                if(totalMemory<0){
                    System.out.println("内存不够");
                    return;
                }

                System.out.println(p.name+"到达时间："+p.arriveTime+"
"+p.name+"终止时间："+endTime+"  运行过程中剩余内存"+totalMemory + "
地址："+p.me.address);
                totalMemory+=p.memory;
            }
            while (true) {
                if(temp.size()!=0){
                    System.out.println("目前还有进程处在阻塞状态，是否释放
(y/n)");
                    Scanner scanner = new Scanner(System.in);
                    String s = scanner.next();
                    if(s.equals("y")||s.equals("Y")){
                        for (PCB c:temp){
                            arriveTime = Math.max(c.arriveTime,endTime);
                            endTime = arriveTime+c.needTime;
                            System.out.println(c.name+"到达时间：
"+c.arriveTime+"  "+c.name+"终止时间："+endTime+"  运行过程中剩余内
存"+totalMemory+ "  地址："+me.address);
                            totalMemory+=c.memory;
                        }
                        return;
                    }else if(s.equals("n")||s.equals("N")){
                        System.out.println("程序结束");
                        return;
                    }else {
```

```
                        System.out.println("输入有误，请重新输入！");
                    }
                }
                return;
            }
        }
    }
```

memo.java

```
public class memo {
    int address;
    int memory;

    public memo(int address, int memory) {
        this.address = address;
        this.memory = memory;
    }

    public memo() {
    }

    public int getAddress() {
        return address;
    }

    public void setAddress(int address) {
        this.address = address;
    }

    public int getMemory() {
        return memory;
    }

    public void setMemory(int memory) {
        this.memory = memory;
    }
}
```

PCB.java

```
public class PCB {
    public int arriveTime;
    public int memory;
    public int needTime;
    public String name;
    public boolean flag;
    public memo me = new memo();

    public PCB(int arriveTime, int memory, int needTime,String name,boolean
flag,memo me) {
        this.arriveTime = arriveTime;
        this.memory = memory;
        this.needTime = needTime;
        this.name = name;
        this.flag = flag;
        this.me = me;
    }

    public PCB() {
    }

    public int getArriveTime() {
        return arriveTime;
    }

    public void setArriveTime(int arriveTime) {
        this.arriveTime = arriveTime;
    }

    public int getMemory() {
        return memory;
    }

    public void setMemory(int memory) {
        this.memory = memory;
    }

    public int getNeedTime() {
        return needTime;
    }

    public void setNeedTime(int needTime) {
        this.needTime = needTime;
```

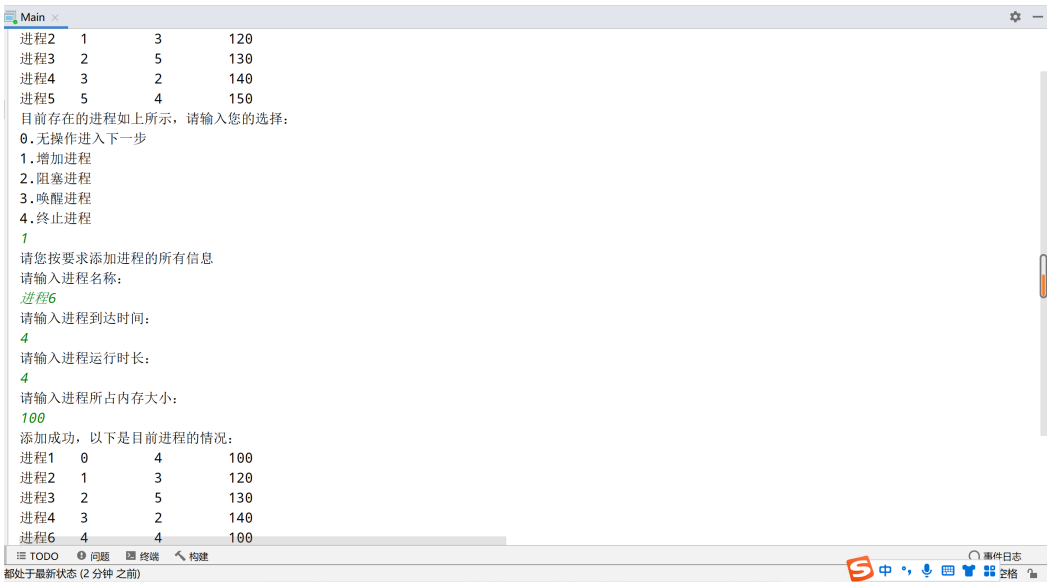
```
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

运行结果

1.添加进程



2.阻塞进程

```
0.无操作进入下一步
1.增加进程
2.阻塞进程
3.唤醒进程
4.终止进程
2
请输入你要阻塞的进程
进程6
进程6已阻塞，目前就绪队列中的进程如下:
进程1    0          4          100
进程2    1          3          120
进程3    2          5          130
进程4    3          2          140
进程5    5          4          150
```

3.唤醒进程

```
0.无操作进入下一步
1.增加进程
2.阻塞进程
3.唤醒进程
4.终止进程
3
请输入想要唤醒的进程:
进程6
唤醒成功，下面是目前进程情况:
进程1    0          4          100
进程2    1          3          120
进程3    2          5          130
进程4    3          2          140
进程6    4          4          100
进程5    5          4          150
```

4.终止进程

0.无操作进入下一步
1.增加进程
2.阻塞进程
3.唤醒进程
4.终止进程
4
请输入你要移除的进程：
进程6
移除成功，以下是目前进程的情况：
进程1　0　4　100
进程2　1　3　120
进程3　2　5　130
进程4　3　2　140
进程5　5　4　150

5.运行进程

```
进程1    0          4          100
进程2    1          3          120
进程3    2          5          130
进程4    3          2          140
进程5    5          4          150
0.无操作进入下一步
1.增加进程
2.阻塞进程
3.唤醒进程
4.终止进程
0
是否启动文件中的进程（y/n）
y
进程1到达时间：0   进程1终止时间：4   运行过程中剩余内存1110   地址：100
进程2到达时间：1   进程2终止时间：7   运行过程中剩余内存1060   地址：220
进程3到达时间：2   进程3终止时间：12   运行过程中剩余内存1180   地址：350
进程4到达时间：3   进程4终止时间：14   运行过程中剩余内存1310   地址：490
进程5到达时间：5   进程5终止时间：18   运行过程中剩余内存1450   地址：640
```

六、【总结与收获】

通过这次实验，我巩固了操作系统进程调度的几个算法，这也算是完成了一个简易的操作系统，感受到了底层的精妙性，debug 的过程中也收益良多，另外，还巩固了 java 的一些好用的 API。