**安徽科技学院**

**软件类实验报告**

|  |  |
| --- | --- |
| 课程名称： | 操作系统 |
| 学 号： | 2701200421 |
| 姓 名： | 乔梁 |
| 班 级： | 计算机科学与技术204班 |
| 指导教师： | 李文才 |
| 开课学期： | 2022-2023-1 |
| 学 院： | 信息与网络工程学院 |

安徽科技学院教务处

|  |  |  |  |
| --- | --- | --- | --- |
| **实验名称** | **进程调度算法** | **实验学时** | 4 |
| **实验场所** | **软件实验室九** | **实验性质** | **验证** |
| **一、实验目标**：  通过对进程调度算法的模拟加深对进程概念和进程调度算法的理解，了解进程调度算法的具过程和方式，对其有更深的认识。 | | | |
| **二、实验原理：**  使用不同的进程调度算法可以实现操作系统的各种目标和要求，譬如想要高相应度就要使用时间片轮转算法，而追求最小平均周转时间就可以使用短进程优先算法，所以集合很多算法为一体的操作系统也是非常有必要的。 | | | |
| **三、实验要求：**  1.设计进程控制块PCB表结构适用于优先权调度算法。  2.建立进程就绪队列。  3.编制时间片轮转调度算法。  4.编制先来先服务算法。  5.编制短进程优先算法。  6.调制进程的各项其他应用和辅助。 | | | |
| **四、实验步骤： （包括详细实验操作步骤、流程图、代码及重要语句注释）**  在原有的算法基础上添加**短进程**优先算法，只要调整好开始的准备队列顺序，随后调用开始的FCFS算法就可以完成短进程优先代码。  //短进程优先算法 **public void SJF() {  System.out.println("当前系统时间：" + systemTime);  newSequence.sort(new Comparator<PCB>() {  @Override  public int compare(PCB o1, PCB o2) {  return o1.needTime - o2.needTime;  }  });  FCFS(); //调制好顺序后使用FCFS即可。 }**  **case 3:  Found Process3 = new Found();  Process3.SFS(); //短进程优先算法  print();**  **完整代码**  package ch1;  import java.util.ArrayList; import java.util.Comparator;  public class Found {  ArrayList<PCB> newSequence;   PCB pcb[] = new PCB[5];   int systemTime = (int) (Math.random() \* 3) + 1;   public Found() {  newSequence = new ArrayList<PCB>();  for (int i = 0; i < 5; i++) {  pcb[i] = new PCB();  pcb[i].pcbId = i + 1;  newSequence.add(pcb[i]);  }  }   // 先来先服务算法  public void FCFS() {  PCB exeRun = null;  while (newSequence.size() > 0)  {  exeRun = getPcb();  System.out.println("当前系统时间：" + systemTime);  systemTime += exeRun.useTime;  ShowMessages(exeRun);  }  }   // 时间片轮换算法  public void RR() {  int time = systemTime;  while (newSequence.size() > 0) {  rrMethode(time);  }  }   //短进程优先算法  public void SJF() {  System.out.println("当前系统时间：" + systemTime);  newSequence.sort(new Comparator<PCB>() {  @Override  public int compare(PCB o1, PCB o2) {  return o1.needTime - o2.needTime;  }  });  FCFS(); //调制好顺序后使用FCFS即可。  }  //输出  public void ShowMessages(PCB p) {  System.out.println("当前运行进程：" + p.pcbId +  " " + "服务时间：" + p.useTime +  " " + "需要时间：" + p.needTime +  " " + "优先级：" + p.perior +  " " + "状态：" + p.status);  if (newSequence.size() > 0) {  System.out.println("当前就绪进程：");  for (PCB p1 : newSequence) {  System.out.println("进程编号：" + p1.pcbId +  " " + "服务时间：" + p1.useTime +  " " + "需要时间：" + p1.needTime +  " " + "优先级：" + p1.perior +  " " + "状态：" + p1.status);  System.out.println("--------------------------------------------------------------------------");  }  } else {  System.out.println("当前系统中已经没有就绪进程！");  }  System.out.println('\n');  }  private PCB getPcb() {  PCB exeRun;  exeRun = newSequence.remove(0);  exeRun.useTime = exeRun.needTime;  exeRun.needTime = 0;  exeRun.perior = 0;  return exeRun;  }  private void rrMethode(int time) {  PCB exeRun;  System.out.println("当前系统时间：" + systemTime);  exeRun = newSequence.remove(0);   //时间片用完结束  if (exeRun.needTime <= time) {  exeRun.useTime = exeRun.needTime;  exeRun.needTime = 0;  exeRun.perior = 0;  exeRun.status = "Finish";  systemTime += exeRun.useTime;  } else {  exeRun.useTime += time;  exeRun.needTime -= time;  exeRun.perior--;  exeRun.status = "Ready";  newSequence.add(exeRun);  systemTime += time;  }  ShowMessages(exeRun);  }  }  package ch1;  import java.util.Scanner;  //主界面菜单 public class Menu {  Scanner sc = new Scanner(System.in);   public int print() {  System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  System.out.println("进程调度算法演示");  System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  System.out.println(" 1.先来先服务（FCFS）算法");  System.out.println(" 2.时间片轮换（RR）算法");  System.out.println(" 3.短进程优先（SJF）算法");  System.out.println(" 4.退出该程序");  System.out.print("请选择所要采用的算法：");  int flag = sc.nextInt();  return flag;  }   public void select() {  int flag = print();  switch (flag) {  case 1:  Found Process1 = new Found();  Process1.FCFS(); // 先来先服务算法  print();  case 2:  Found Process2 = new Found();  Process2.RR(); // 时间片轮换算法  print();  case 3:  Found Process3 = new Found();  Process3.SJF(); //短进程优先算法  print();  case 4:  System.exit(0);  default:  break;  }  } }  package ch1;  public class PCB {  int pcbId;  int useTime;  int needTime;  int perior;  String status;  public PCB() {  pcbId++;  useTime = 0;  needTime = (int) Math.round(Math.random() \* 5) + 1;  perior = (int) Math.round(Math.random() \* 5) + 1;  status = "Ready";  } }  package ch1;  public class ProcessControl {  public static void main(String args[]) {  Menu application = new Menu();  application.select();  } } | | | |
| **五、实验结果：** | | | |
| **六、实验小结：**  本次实验让我对调度算法有了更深的认识和感受，对算法中的细节方面有更深的认识，算法上对其有更好的把握进行计算。自己实现调度算法在细节的计算中会更好的计算和细节。 | | | |