实验报告一

袁子华 14122260

一, 实验名称：操作系统的进程调度

二、目的与要求

1、目的

进程是操作系统最重要的概念之一,进程调度又是操作系统核心的主要内容。本实习要求学生独立地用高级语言编

写和调试一个简单的进程调度程序。调度算法可任意选择或自行设计。例如,简单轮转法和优先数法等。本实习可加深

对于进程调度和各种调度算法的理解。

2、 要求

(1) 设计一个有 n 个进程工行的进程调度程序。每个进程由一个进程控制块(PCB)表示。进程控制块通常应包含

下述信息:进程名、进程优先数、进程需要运行的时间、占用 CPU 的时间以及进程的状态等,且可按调度算法

的不同而增删。

(2) 调度程序应包含 2~3 种不同的调度算法,运行时可任意选一种,以利于各种算法的分析比较。

(3) 系统应能显示或打印各进程状态和参数的变化情况,便于观察诸进程的调度过程

操作系统进程调度有多种算法，时间片轮转算法是一个经典的调度算法，此次上机我完成了模拟操作系统调度算法中的时间片轮转算法，下面是是我上机实验的主要操作步骤：

1. 时间片轮转算法的主要思想是，把处理机平均分配为时间片，每次然就绪队列中每个进程运行一个时间片，如果在该时间片内该进程已经完成，则把该进程从就绪队列中删除，反之把该进程插入就绪队列的末尾。
2. 用Java 写一个PCB 类，该类的数据成员如下

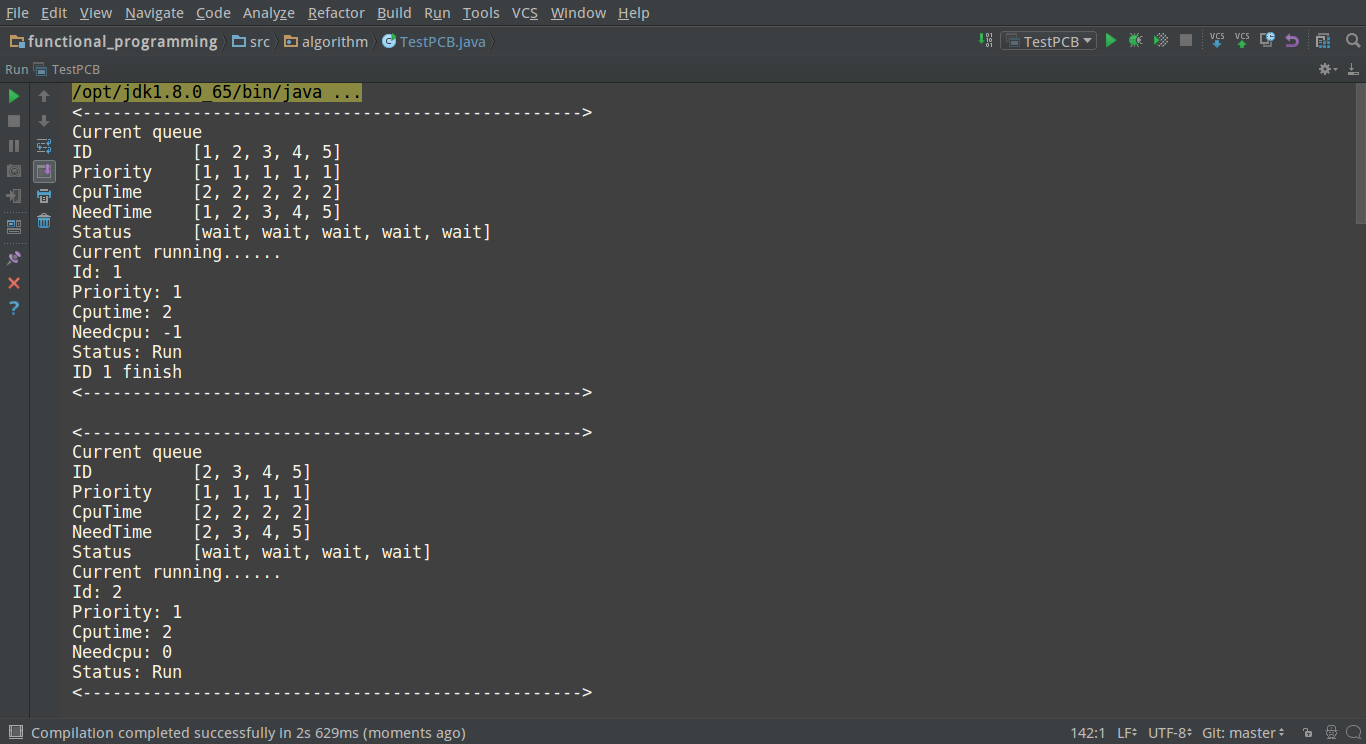
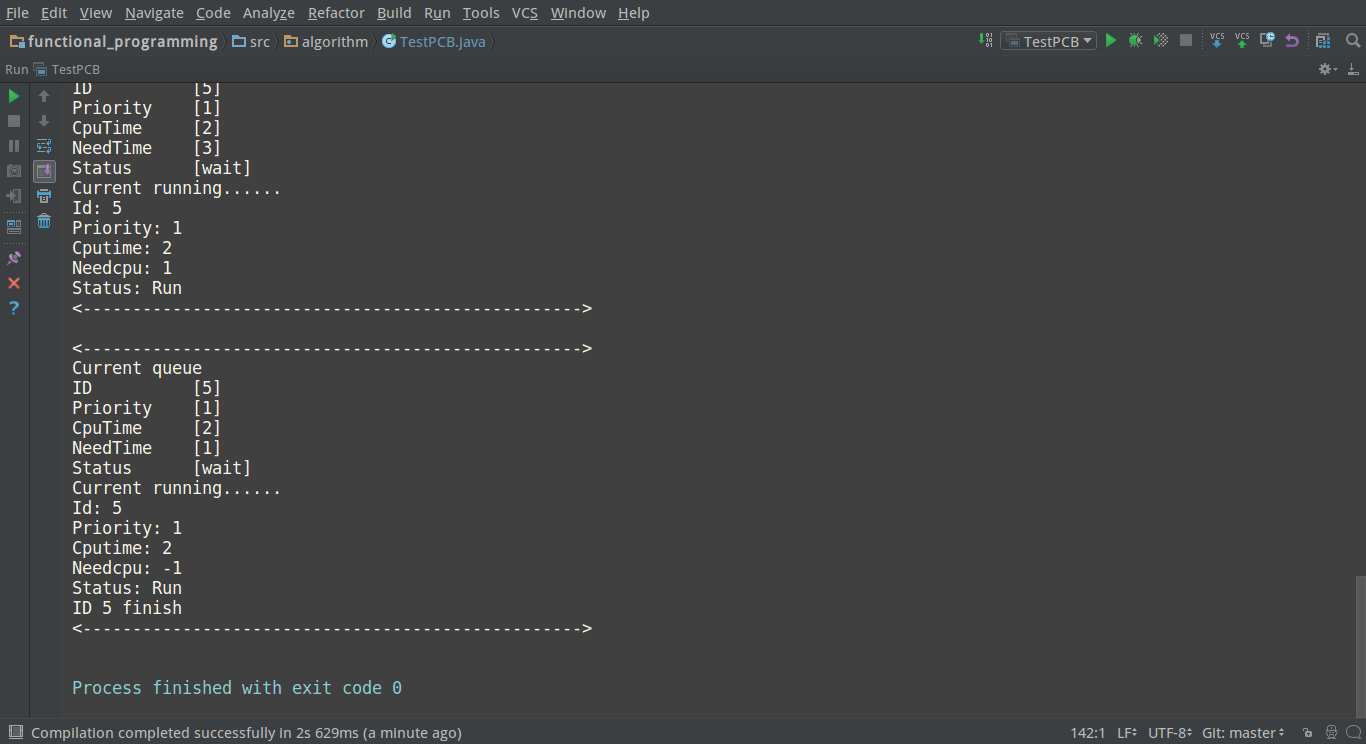
class PCB implements Comparable<PCB>{  
 private int **flag**; **//进程标识** private int **priority**; **//优先级** private int **cpuTime**; **//当前获得的时间片** private int **needCp**; **//还需要多少个时间片** private String **status**; **//状态** public PCB(){}  
 public PCB(int flag,int priority,int cupTime,int needCp,String status){}  
 public int getFlag(){}  
 public int getPriority(){}  
 public int getCpuTime(){}  
 public int getNeedCp(){}  
 public String getStatus(){}  
 public void setStatus(String status){}  
 public void setNeedCp(){}  
 public void setNeedCp(int t){}  
 public void setCpuTime(int cpuTime){}  
 public void setPriority(int priority){}  
 public void setPriority(){}  
 @Override  
 public int compareTo(PCB o) {} **//在时间片轮转算法中用不到** @Override  
 public String toString(){} **//转为字符串 方便输出**}

C,上面步骤写完了一个 PCB 类，我们知道操作系统调度是一CPB 为单位的，每次调度都是从就绪队列中，得到一个就绪的进程，然后给改进程分配。下面程序是 模拟时间片轮转算法：

**//queue队列 模拟就绪队列**while(!queue.isEmpty()){ **//队列不为空表示，还有进程** Queue<PCB>queue1=new LinkedList<>(queue);**//临时的一个队列，只是为了看各个进程的状态，和该算法无关** show(queue1); **//查看就绪队列中各个进程的状态** PCB t=queue.poll();  
 exe(t,cpuTime); **//模拟 处理机 给t进程分配时间片，并运行其** if(t.getNeedCp()>0){ **//如果t进程还没有完成，那么把该进程插入就绪队列** t.setStatus("wait");  
 queue.offer(t); **//插入就绪队列** }  
}

D.通过上面几个步骤就可以成功的模拟操作系统进行时间片轮转调度算法了

结果如下



以上是我本次实验的具体实现过程，具体代码 如下

package algorithm;  
  
import java.util.Comparator;  
import java.util.LinkedList;  
import java.util.Queue;  
import java.util.Vector;  
  
  
/\*\*  
 \* Created by zihua on 16-11-29.  
 \*/  
public class TestPCB {  
 public static void main(String[]args){  
 int cpuTime=2;  
 PCB p1=new PCB(1,1,cpuTime,1,"wait");  
 PCB p2=new PCB(2,1,cpuTime,2,"wait");  
 PCB p3=new PCB(3,1,cpuTime,3,"wait");  
 PCB p4=new PCB(4,1,cpuTime,4,"wait");  
 PCB p5=new PCB(5,1,cpuTime,5,"wait");  
  
 Queue<PCB>queue=new LinkedList<>();  
 queue.add(p1);  
 queue.add(p2);  
 queue.add(p3);  
 queue.add(p4);  
 queue.add(p5);  
 while(!queue.isEmpty()){  
 Queue<PCB>queue1=new LinkedList<>(queue);  
 show(queue1);   
 PCB t=queue.poll();  
 exe(t,cpuTime);  
 if(t.getNeedCp()>0){  
 t.setStatus("wait");  
 queue.offer(t);  
 }  
 }  
  
 }  
  
 public static void exe(PCB a,int cpu){  
 if(a.getNeedCp()>0){  
 a.setStatus("Run");  
 a.setNeedCp(cpu);  
 a.setCpuTime(cpu);  
 System.***out***.println("Current running......");  
 System.***out***.println("Id: "+a.getFlag());  
 System.***out***.println("Priority: "+a.getPriority());  
 System.***out***.println("Cputime: "+a.getCpuTime());  
 System.***out***.println("Needcpu: "+a.getNeedCp());  
 System.***out***.println("Status: "+a.getStatus());  
 }  
  
 if(a.getNeedCp()<0){  
 System.***out***.println("ID "+a.getFlag()+" finish");  
 }  
 System.***out***.println("<-------------------------------------------------->\n");  
  
 }  
  
 public static void show(Queue a){  
 Vector<Integer>id=new Vector<>();  
 Vector<Integer>priority=new Vector<>();  
 Vector<Integer>cputTime=new Vector<>();  
 Vector<Integer>needTime=new Vector<>();  
 Vector<String>status=new Vector<>();  
 while(!a.isEmpty()){  
 PCB t=(PCB)a.poll();  
 id.add(t.getFlag());  
 priority.add(t.getPriority());  
 cputTime.add(t.getCpuTime());  
 needTime.add(t.getNeedCp());  
 status.add(t.getStatus());  
 }  
 if(id.size()>0){  
 System.***out***.println("<-------------------------------------------------->");  
 System.***out***.println("Current queue ");  
 System.***out***.println("ID "+id);  
 System.***out***.println("Priority "+priority);  
 System.***out***.println("CpuTime "+cputTime);  
 System.***out***.println("NeedTime "+needTime);  
 System.***out***.println("Status "+status);  
  
 }  
  
 }  
}  
  
class PCB implements Comparable<PCB>{  
 private int **flag**;  
 private int **priority**;  
 private int **cpuTime**;  
 private int **needCp**;  
 private String **status**;  
  
 public PCB(){}  
 public PCB(int flag,int priority,int cupTime,int needCp,String status){  
 this.**flag**=flag;  
 this.**priority**=priority;  
 this.**cpuTime**=cupTime;  
 this.**needCp**=needCp;  
 this.**status**=status;  
 }  
 public int getFlag(){  
 return **flag**;  
 }  
 public int getPriority(){  
 return **priority**;  
 }  
 public int getCpuTime(){  
 return **cpuTime**;  
 }  
 public int getNeedCp(){  
 return **needCp**;  
 }  
 public String getStatus(){  
 return **status**;  
 }  
 public void setStatus(String status){  
 this.**status**=status;  
 }  
 public void setNeedCp(){  
 **needCp**-=**cpuTime**;  
 }  
 public void setNeedCp(int t){  
 **needCp**-=t;  
 }  
 public void setCpuTime(int cpuTime){  
 this.**cpuTime**=cpuTime;  
 }  
 public void setPriority(int priority){  
 this.**priority**=priority;  
 }  
 public void setPriority(){  
 **priority**-=1;  
 }  
 @Override  
 public int compareTo(PCB o) {  
 if(**priority**>o.**priority**)return 1;  
 else if(**priority**<o.**priority**)return -1;  
 return 0;  
 }  
 @Override  
 public String toString(){  
 return **flag**+" "+**priority**+" "+**cpuTime**+" "+**needCp**+" "+**status**;  
 }  
}

体会：

通过用Java模拟操作系统的调度算法，让我对操作系统调度过程有了更深刻的理解，同时我也深刻的体会到算法的重要性，可以说没有算法，就没有我们今天的社会。