实验一、进程调度试验

[目的要求 ]

用高级语言编写和调试一个进程调度程序，以加深对进程的概念及进程调度算法的 理解．

[准备知识 ]

一、基本概念

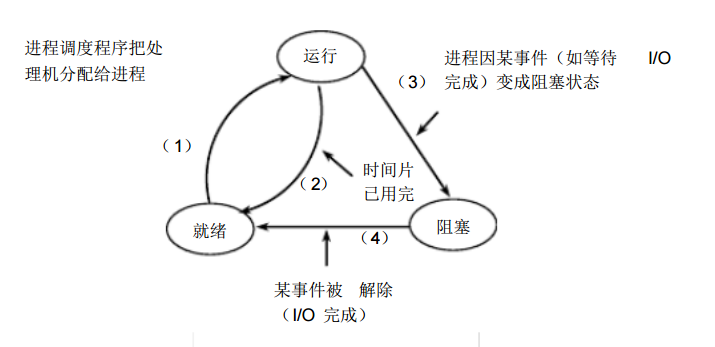
1、进程的概念；

2、进程的状态和进程控制块；

3、进程调度算法；

二、进程调度

1、进程的状态



2、进程的结构—— PCB

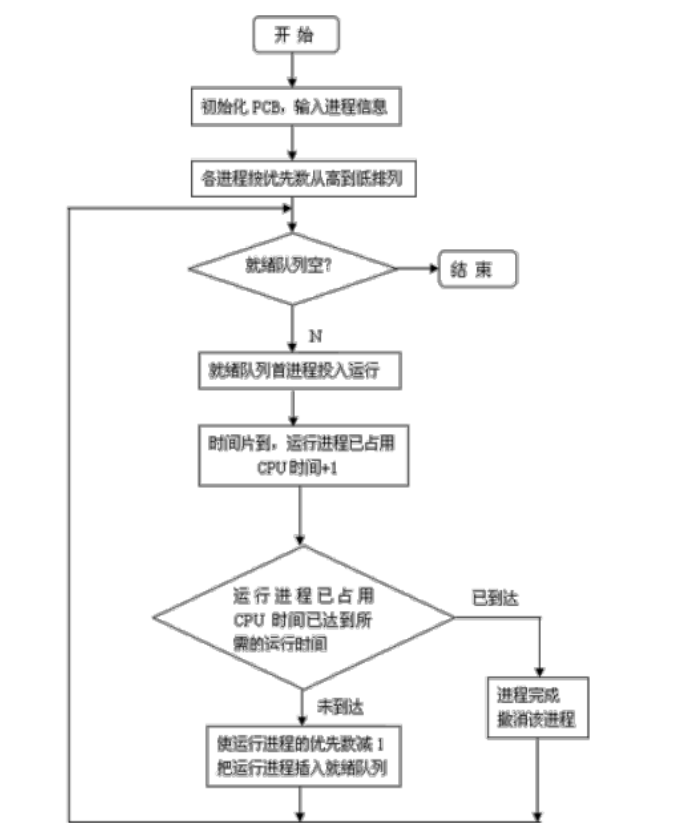
进程都是由一系列操作 (动作 )所组成，通过这些操作来完成其任务。因此，不同的进 程，其内部操作也不相同。在操作系统中，描述一个进程除了需要程序和私有数据之外， 最主要的是需要一个与动态过程相联系的数据结构， 该数据结构用来描述进程的外部特性 (名字、状态等 )以及与其它进程的联系 (通信关系 )等信息，该数据结构称为进程控制块 (PCB，Process Control Block) 。

进程控制块 PCB 与进程一一对应， PCB 中记录了系统所需的全部信息、用于描述进 程情况所需的全部信息和控制进程运行所需的全部信息。 因此，系统可以通过进程的 PCB 来对进程进行管理。

[试验内容 ]

设计一个有 N 个进程共行的进程调度程序。

进程调度算法： 采用最高优先数优先的调度算法 （即把处理机分配给优先数最高的进 程）和先来先服务算法。每个进程有一个进程控制块（ PCB）表示。进程控制块可以包 含如下信息：进程名、优先数、到达时间、需要运行时间、已用 CPU 时间、进程状态等 等。 进程的优先数及需要的运行时间可以事先人为地指定 （也可以由随机数产生） 。进程 的到达时间为进程输入的时间。 进程的运行时间以时间片为单位进行计算。 每个进程的状 态可以是就绪 W（Wait）、运行 R（Run）、或完成 F（ Finish）三种状态之一。就绪进程 获得 CPU 后都只能运行一个时间片。用已占用 CPU 时间加 1 来表示。如果运行一个时 间片后，进程的已占用 CPU 时间已达到所需要的运行时间，则撤消该进程，如果运行一 运行 就绪 阻塞 进程因某事件（如等待 I/O 完成）变成阻塞状态 某事件被 解除 （I/O 完成） 时间片 已用完 进程调度程序把处 理机分配给进程 （1） （2） （3） （4） 计算机操作系统教程（第三版） 个时间片后进程的已占用 CPU 时间还未达所需要的运行时间，也就是进程还需要继续运 行，此时应将进程的优先数减 1（即降低一级） ，然后把它插入就绪队列等待 CPU。每进 行一次调度程序都打印一次运行进程、就绪队列、 以及各个进程的 PCB，以便进行检查。 重复以上过程，直到所要进程都完成为止。 调度算法的流程图如下 :



进程调度源程序如下：

//App.java

**import** java.util.ArrayList;

**import** java.util.Comparator;

**import** java.util.Iterator;

**import** java.util.List;

**import** java.util.Random;

**import** java.util.Scanner;

**public** **class** App {

**public** **static** **void** main(String[] args) {

ArrayList<Pcb> arrayList = **new** ArrayList<Pcb>();

**int** len = Math.*abs*(**new** Random().nextInt(10) );

**if**(len==0)

len = 5;

System.***out***.println("系统随机创建了"+len+"个线程");

*create*(len, arrayList);

*printAllInfo*(arrayList);

**while**(arrayList.size()>0){

Pcb pcb = App.*Schedule*(arrayList);

**boolean** over = pcb.run();

**if**(over){

arrayList.remove(pcb);

}

*printAllInfo*(arrayList);

}

System.***out***.println("所有线程运行结束");

}

**public** **static** **void** printAllInfo(List<Pcb> list) {

**if**(list.size()>0)

System.***out***.println("线程名 ||到达时间 ||需要运行时间 ||已运行时间|| 级别|| 状态");

**for** (Iterator<Pcb> iterator = list.iterator(); iterator.hasNext();) {

Pcb pcbInfo = (Pcb) iterator.next();

pcbInfo.info();

}

}

@SuppressWarnings("resource")

**public** **static** **void** create(Integer num,List<Pcb> arrayList) {

**for**(**int** i = 0;i < num;i++){

Pcb pcb = **new** Pcb();

//设置进程名字

pcb.setCourSeName("进程 "+(i+1));

System.***out***.println("请输入进程"+(i+1)+"的到达时间（int）");

**int** arriveTime = 0 ;

**while**(arriveTime<=0){

**try**{

arriveTime = **new** Scanner(System.***in***).nextInt();

}**catch**(Exception e){

e.printStackTrace();

}

**if**(arriveTime<=0)

System.***out***.println("到达时间应该大于o");

}

pcb.setArriveTime(arriveTime);

**int** needTime = Math.*abs*(**new** Random().nextInt(5));

pcb.setNeedTime(needTime);

**int** grade =Math.*abs*( **new** Random().nextInt(10));

pcb.setGreade(grade);

pcb.setState("就绪");

arrayList.add(pcb );

**try** {

Thread.*sleep*(100);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

}

**public** **static** Pcb Schedule(List<Pcb> list){

list.sort(**new** Comparator<Pcb>() {

@Override

**public** **int** compare(Pcb o1, Pcb o2) {

**if**(o1.getGreade()==o2.getGreade()){

**return** o1.getArriveTime()-o2.getArriveTime();

}**else**{

**return** o2.getGreade()-o1.getGreade();

}

}

});

**return** list.get(0);

}

}

//Pcb.java

**public** **class** Pcb {

/\*\*

\* 进程名

\*/

**private** String courSeName;

/\*\*

\* 到达时间

\*/

**private** Integer arriveTime;

/\*\*

\* 需要时间

\*/

**private** Integer needTime;

/\*\*

\* 状态

\*/

**private** String state;

/\*\*

\* 已用CPU 时间

\*/

**private** Integer runTime =0;

/\*\*

\* 优先级

\*/

**private** Integer greade;

**public** String getCourSeName() {

**return** courSeName;

}

**public** **void** setCourSeName(String courSeName) {

**this**.courSeName = courSeName;

}

**public** String getState() {

**return** state;

}

**public** **void** setState(String state) {

**this**.state = state;

}

**public** **boolean** run(){

System.***out***.println(**this**.courSeName+"正在运行.........");

**try** {

Thread.*sleep*(500);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

**this**.runTime+= 1;

**this**.greade-=1;

**this**.state="运行";

**if**(**this**.runTime>=**this**.needTime){

System.***out***.println(**this**.courSeName+"运行结束!!!!!!!");

**this**.state="结束";

**return** **true**;

}

**return** **false**;

}

**public** Integer getArriveTime() {

**return** arriveTime;

}

**public** **void** setArriveTime(Integer arriveTime) {

**this**.arriveTime = arriveTime;

}

**public** Integer getNeedTime() {

**return** needTime;

}

**public** **void** setNeedTime(Integer needTime) {

**this**.needTime = needTime;

}

**public** Integer getRunTime() {

**return** runTime;

}

**public** **void** setRunTime(Integer runTime) {

**this**.runTime = runTime;

}

**public** Integer getGreade() {

**return** greade;

}

**public** **void** setGreade(Integer greade) {

**this**.greade = greade;

}

**public** **void** info(){

System.***out***.println(**this**.courSeName+" || " +

**this**.arriveTime +" ||"+

**this**.needTime +" ||"+ **this**.runTime+" ||"+

**this**.greade+" ||"+**this**.state);

}

}

实验结果截图：

