广州大学学生实验报告

开课学院及实验室：计算机科学与网络工程学院软件实验室　　2019 年 12 月 23 日

学院	计算机科学与网络工程学院	**年级/专业/班**	软件 171	**姓名**	谢金宏	**学号**	1706300001
实验课程名称	操作系统实验					**成绩**	
实验项目名称	时间片轮转进程调度					**指导老师**	陶文正

课程设计 时间片轮转进程调度

一、 设计内容

设计一个按时间片轮转法进行进程调度的模拟程序。

在宏观上，系统中可以同时启动多个进程，每个进程并行不悖，同时运行。但是在微观上，由于只有一个 CPU，一次只能处理一个进程。如何处理公平？一种方法就是引入“时间片”，每个进程轮流执行。操作系统一般是按照一定策略，定期给每个活动的进程执行的机会，并且每次只执行一个时间片，然后操作系统将当前进程信息压栈，然后开始执行下一个进程。通过这样不断快速的循环切换，每个进程都获得执行。

时间片轮转法（Round-Robin，RR）主要用于分时系统中的进程调度。假设系统有 n 个进程，每个进程用一个进程控制块（PCB）来代表。为了实现轮转调度，系统把所有就绪进程按先入先出的原则排成一个队列。新来的进程加到就绪队列末尾。每当执行进程调度时，进程调度程序总是选出就绪队列的队首进程，让它在 CPU 上运行一个时间片的时间。当进程用完分给它的时间片后，系统的计时器发出时钟中断，调度程序便停止该进程的运行，把它放入就绪队列的末尾；然后，把 CPU 分给就绪队列的队首进程，同样也让它运行一个时间片，如此往复，直到 n 个进程都运行完毕。

二、 软件运行环境

软件需要在安装有 Visual Studio C++编程软件的计算机上的编译和运行，也可略经改动后在类 Linux 环境中使用 G++编译和运行。

三、 数据结构和主要符号说明

这是一个模拟程序，每个进程用一个进程控制块 PCB 表示。每个 PCB 中包含进程的名称、进程的到达时间和估计运行时间。PCB 中没有记录进程的状态，因为这是一个模拟程序，进程的状态隐含在 PCB 所处的容器中。

```
// 进程控制块
struct PCB {
    string name;  // 进程名称
    int arrivedAt;  // 到达时间
    int remainTime;  // 估计运行时间
};
```

程序中有三个容纳 PCB 的容器。其一是保存系统中已安排但是还没到达的进程 PCB 的容器 processToRun，这个容器中的进程可以视为阻塞状态。其二是，模拟就绪队列的容器 processQueue，这个容器中的进程处于就绪状态。第三个容器保存已完成进程 PCB 的 finishedProcess，此容器中的进程处于完成状态。

```
vector<PCB> processToRun; // 系统中的所有已安排且尚未到达的进程
deque<PCB> processQueue; // 进程队列
vector<PCB> finishedProcess;  // 完成运行的进程
```

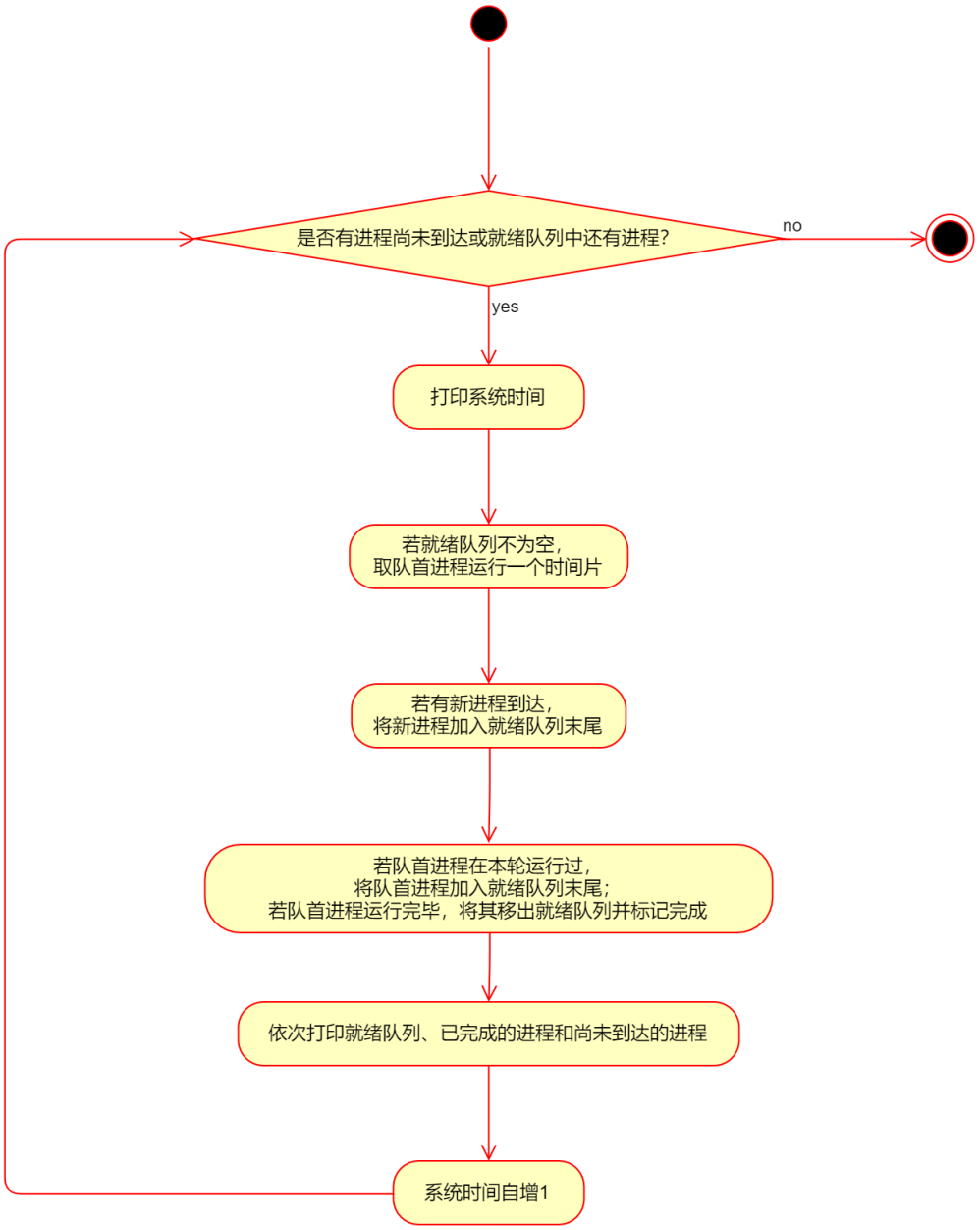
为了模拟时间的推移，程序中还设置了全局变量 systemClock 表示系统时钟。

```
int systemClock;  // 系统时钟
```

四、 程序运行流程

程序运行时，首先要从用户确认输入，然后才进入程序的主要部分。程序先询用户要模拟几个进程运行，接下来询问用户是手动输入测试数据还是随机产生测试数据。对于用户不正确或不合理的输入，要提示是用户重新输入合理的数据。若是随机产生测试数据，则生成进程的名称为 P1~Pn，随机生成[0, 10]范围内的进程到达时间和[1, 11]范围内的运行时间。

从用户处确认信息后，程序进入主要部分。程序主要部分的流程图如下：



图表 1 程序主要部分流程图

程序的主要运行逻辑如图所示。程序先判断系统中是否还有进程未到达（即判断 processToRun 是否为空）或就绪队列中是否还有进程（即判断 processQueue 是否为空）。若有，则按下面的步骤继续进行循环；若无，则停止模拟，程序运行完毕。

在一轮循环中，程序先打印系统时间 systemClock。接下来判断就绪队列 processQueue 队首是否为空，若不为空，取队首进程运行模拟运行一个时间片，即将队首进程的剩余运行时间减一。然后判断此时是否有新的进程到达，判断依据是 processToRun 中进程的到达时间是否与此时的系统时间相等。若有新进程

到达，则将新到达的进程按顺序加入就绪队列末尾。接下来，若就绪队列队首的进程在本轮循环中运行过，则判断它的剩余运行时间是否为 0。若为 0，则将它标记为完成，即加入已完成进程的容器 finishedProcess 中，否则将它移动到就绪队列 processQueue 末尾。然后依次输出就绪队列、已完成的进程和尚未到达的进程的信息。最后，系统时间自增 1，并进入下一轮循环。

五、 程序运行结果和结果分析

采用《操作系统》教材 P123 习题 8 中的数据为输入数据，将数据输入程序中：

进程名称	到达时间	运行时间
1	0	10
2	1	1
3	2	2
4	3	1
5	4	5

程序在处理用户输入时的输出如下：

```
模拟时间片轮转进程调度法
==================
请输入测试进程的个数>> n
进程的个数必须大于 0，请重新输入。
请输入测试进程的个数>> 5
数据是随机生成还是手动输入？ [y/n] >> n
正在输入第 1 个进程的信息:
进程名称>> 1
到达时间>> 0
预计运行时间>> 10

……

正在输入第 5 个进程的信息:
进程名称>> 5
到达时间>> 4
预计运行时间>> 5
========
已安排的进程、到达时间和预计持续时间:
1       0       10
2       1       1
3       2       2
4       3       1
5       4       5
========
```

图表 2 程序处理输入

然后获得程序的输出数据：

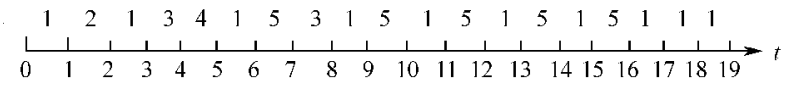
```
========
系统时间：0
系统空转。
进程 1 到达，预计运行时间：10。
就绪队列中的进程和预计运行时间：1(10)
尚未到达的进程的进程名和预计到达时间：2(1) 3(2) 4(3) 5(4)
========
系统时间：1
进程 1 运行，剩余运行时间：9。
进程 2 到达，预计运行时间：1。
就绪队列中的进程和预计运行时间：2(1) 1(9)
尚未到达的进程的进程名和预计到达时间：3(2) 4(3) 5(4)
========
系统时间：2
进程 2 运行完毕。
进程 3 到达，预计运行时间：2。
就绪队列中的进程和预计运行时间：1(9) 3(2)
已完成的进程：2
尚未到达的进程的进程名和预计到达时间：4(3) 5(4)
========

……

========
系统时间：18
进程 1 运行，剩余运行时间：1。
就绪队列中的进程和预计运行时间：1(1)
已完成的进程：2 4 3 5
========
系统时间：19
进程 1 运行完毕。
就绪队列为空。
已完成的进程：2 4 3 5 1
========
```

图表 3 程序的输出

将程序的输出数据绘制在时间轴上，如图所示：



图表 4 将模拟程序的输出绘制于时间轴上

比对得知程序运行结果与教材给出的参考答案相符。

经测试，程序处理其他输入的运行结果也与理论相符。

六、 实验收获和体会

在本次实验中，我设计了一个按时间片轮转法进行进程调度的模拟程序。实话说，程序的流程并不复杂，编写的难度也不大。在本次实验中的收获主要是重温了数据结构的知识，以及增强了对操作系统进程调度的理解。

时间片轮转法有一处容易“出错”的地方：新到达进程加入就绪队列的时机。不同的处理方法会导致程序出现不同的运行结果。若一个新的进程到达时，就绪队列队首的进程恰好运行完一个时间片的时间。此时，应该先将新到达的进程加入就绪队列的末尾，再将运行完一个时间片的队首进程加入就绪队列的末尾。如果，先将运行完一个时间片的队首进程加入就绪队列的末尾，再将新到达的进程加入就绪队列末尾，按照这种逻辑，若原本整个就绪队列中只有队首进程，那么就会出现队首进程连续运行两次，而不是队首进程运行一次，新加入的进程运行一次的现象。这种现象与标准答案不符，相应地，这样的程序逻辑应该是错误的。

七、 实验数据及源代码

实验数据及源代码见“1706300001_谢金宏_课程设计.zip”。