南京信息工程大学　　实验（实习）报告

实验名称 多级反馈调度算法 日期 2023.11.30 指导教师 赵晓平

专业信息安全年级班级 21奇安信姓名朱宸扬学号 202183760012

一.　实验目的

掌握处理器调度算法中的多级反馈调度算法

二.　实验内容

本系统采用多级反馈调度算法，模拟操作系统处理器调度的过程。

三.　实验原理

多级反馈调度队列算法是一种操作系统中用于进程调度的算法，其核心思想是为不同优先级的进程定义多个队列，并通过动态调整进程的优先级来实现不同程度的公平性和响应性。这种调度算法通常用于分时系统，其中多个进程共享系统资源，需要公平地分配CPU时间

以下是多级反馈调度队列算法的基本原理：

多级队列： 系统维护多个就绪队列，每个队列对应一个不同的优先级。通常，初始时将进程放入最高优先级的队列。

调度策略： 调度器根据某种策略从最高优先级的非空队列中选择一个进程执行。一旦一个进程的时间片用完，它将被移到下一个较低优先级的队列，以便为其他进程让出CPU资源。

时间片轮转： 每个队列都可以使用时间片轮转调度算法，确保每个进程在一个时间片内得到执行机会。当进程在当前队列用完时间片后，它将被移到下一个较低优先级的队列。

优先级提升： 如果一个进程等待了足够长的时间而没有得到执行，系统可

以提升它的优先级，以确保等待时间过长的进程有更大的机会获得 CPU 执行时间。

优先级降低： 如果一个进程在其队列内执行了一段时间，而仍然需要 CPU 执行时间，那么它的优先级可能会降低，以便给其他进程更多的机会。

四. 实验设计及编码

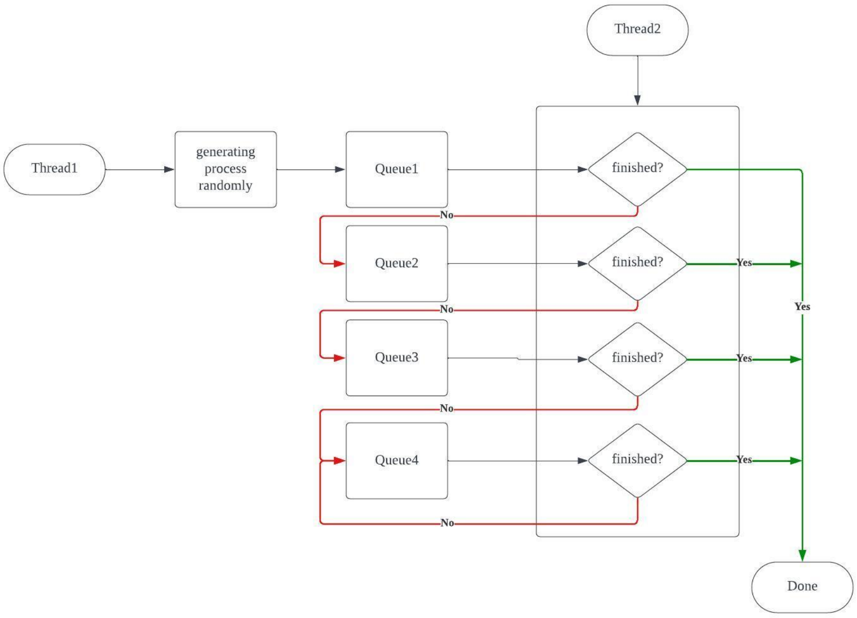
1. 模块分析

进程有序号和工作量大小两个属性

列表选用了队列这种数据结构

采用多线程模拟进程不断进入 1 号队列和处理器处理进程两个任务同步进行

2. 流程图



3. 代码实现

```
import random
import time
from queue import Queue


class Process:
    def __init__(self, name, job_size):
        self.name = name
        self.job_size = job_size
```

```
# 创建四个队列，时间片分别为1、2、4、8
queues = [Queue() for _ in range(4)]
time_slices = [1, 2, 4, 8]

# 创建十个进程，每个进程具有随机的作业量
processes = [Process(f"Process-{i}", random.randint(1, 50)) for i in
range(1, 11)]

# 放入第一个队列（最高优先级队列）
for process in processes:
    print(f"Running {process.name} :(Job Size: {process.job_size})")
    queues[0].put(process)

# 模拟调度器
while not all(queue.empty() for queue in queues):
    for i in range(len(queues)):
        while not queues[i].empty():
            process = queues[i].get()
            temp = process.job_size
            process.job_size -= time_slices[i]



            if(process.job_size<0):
                time.sleep(temp/5)
                process.job_size = 0
            else:
                time.sleep(time_slices[i]/5)
            print(f"Running {process.name} (Time Slice:
{time_slices[i]}) (Job Size: {process.job_size})")
            if process.job_size > 0:
                if i == 3:
                    queues[3].put(process)
                else:
                    queues[i + 1].put(process)  # Move the process to the next queue
            else:
                print(f" {process.name} 运行结束")
```

4. 结果及其相关分析（结果必须是图示）

```
Running Process-1 :(Job Size: 36)
Running Process-2 :(Job Size: 35)
Running Process-3 :(Job Size: 3)
Running Process-4 :(Job Size: 38)
Running Process-5 :(Job Size: 50)
Running Process-6 :(Job Size: 49)
Running Process-7 :(Job Size: 3)
Running Process-8 :(Job Size: 28)
Running Process-9 :(Job Size: 11)
Running Process-10 :(Job Size: 10)
Running Process-1 (Time Slice: 1) (Job Size: 35)
Running Process-2 (Time Slice: 1) (Job Size: 34)
Running Process-3 (Time Slice: 1) (Job Size: 2)
Running Process-4 (Time Slice: 1) (Job Size: 37)
Running Process-5 (Time Slice: 1) (Job Size: 49)
Running Process-6 (Time Slice: 1) (Job Size: 48)
Running Process-7 (Time Slice: 1) (Job Size: 2)
Running Process-8 (Time Slice: 1) (Job Size: 27)
Running Process-9 (Time Slice: 1) (Job Size: 10)
Running Process-10 (Time Slice: 1) (Job Size: 9)
Running Process-1 (Time Slice: 2) (Job Size: 33)
Running Process-2 (Time Slice: 2) (Job Size: 32)
Running Process-3 (Time Slice: 2) (Job Size: 0)
 Process-3 运行结束
```

```
Running Process-4 (Time Slice: 2) (Job Size: 35)
Running Process-5 (Time Slice: 2) (Job Size: 47)
Running Process-6 (Time Slice: 2) (Job Size: 46)
Running Process-7 (Time Slice: 2) (Job Size: 0)
 Process-7 运行结束
Running Process-8 (Time Slice: 2) (Job Size: 25)
Running Process-9 (Time Slice: 2) (Job Size: 8)
Running Process-10 (Time Slice: 2) (Job Size: 7)
Running Process-1 (Time Slice: 4) (Job Size: 29)
Running Process-2 (Time Slice: 4) (Job Size: 28)
Running Process-4 (Time Slice: 4) (Job Size: 31)
Running Process-5 (Time Slice: 4) (Job Size: 43)
Running Process-6 (Time Slice: 4) (Job Size: 42)
Running Process-8 (Time Slice: 4) (Job Size: 21)
Running Process-9 (Time Slice: 4) (Job Size: 4)
Running Process-10 (Time Slice: 4) (Job Size: 3)
Running Process-1 (Time Slice: 8) (Job Size: 21)
Running Process-2 (Time Slice: 8) (Job Size: 20)
Running Process-4 (Time Slice: 8) (Job Size: 23)
Running Process-5 (Time Slice: 8) (Job Size: 35)
Running Process-6 (Time Slice: 8) (Job Size: 34)
Running Process-8 (Time Slice: 8) (Job Size: 13)
Running Process-9 (Time Slice: 8) (Job Size: 0)
 Process-9 运行结束
Running Process-10 (Time Slice: 8) (Job Size: 0)
 Process-10 运行结束
Running Process-1 (Time Slice: 8) (Job Size: 13)
Running Process-2 (Time Slice: 8) (Job Size: 12)
Running Process-4 (Time Slice: 8) (Job Size: 15)
Running Process-5 (Time Slice: 8) (Job Size: 27)
Running Process-6 (Time Slice: 8) (Job Size: 26)
Running Process-8 (Time Slice: 8) (Job Size: 5)
Running Process-1 (Time Slice: 8) (Job Size: 5)
Running Process-2 (Time Slice: 8) (Job Size: 4)
Running Process-4 (Time Slice: 8) (Job Size: 7)
Running Process-5 (Time Slice: 8) (Job Size: 19)
Running Process-6 (Time Slice: 8) (Job Size: 18)
Running Process-8 (Time Slice: 8) (Job Size: 0)
 Process-8 运行结束
Running Process-1 (Time Slice: 8) (Job Size: 0)
 Process-1 运行结束
Running Process-2 (Time Slice: 8) (Job Size: 0)
 Process-2 运行结束
Running Process-4 (Time Slice: 8) (Job Size: 0)
 Process-4 运行结束
Running Process-5 (Time Slice: 8) (Job Size: 11)
Running Process-6 (Time Slice: 8) (Job Size: 10)
Running Process-5 (Time Slice: 8) (Job Size: 3)
Running Process-6 (Time Slice: 8) (Job Size: 2)
Running Process-5 (Time Slice: 8) (Job Size: 0)
 Process-5 运行结束
Running Process-6 (Time Slice: 8) (Job Size: 0)
 Process-6 运行结束
```

五. 实验小结

通过本次实验，我们深入了解了多级反馈调度算法的机制和运作方式。这不仅有助于我们理解操作系统中的进程调度，还为我们进一步研究和优化调度算法提供了基础。

实验的完成使我们更好地理解了理论知识，并通过实际操作加深了对多级反馈调度算法的认识。这将有助于我们更好地应用和理解操作系统的相关概念。