

江蘇大學京江學院

JIANGSU UNIVERSITY JINGJIANG COLLEGE



操作系统实验报告

| | |
|---------|------------------|
| 题 目： | 实验二 处理器调度 |
| 上机时间： | 2023.11.23 |
| 授课教师： | 潘雨青 |
| 姓 名： | 马云骥 |
| 学 号： | 4211153047 |
| 专 业： | 软件工程 |
| 班 级： | J软件(嵌入)(专转本)2102 |
| 日 期： | 2023.11.30 |

实验二 处理器调度

1 实验目的

1. 熟悉典型的处理器调度算法
2. 熟悉在编程中使用时钟定时器
3. 能够仿真实现典型的处理器调度算法

2 实验设计

原始参考的案例代码均由  C# 编写，这里本人使用  Python 实现，力求达到同样的效果。

开发环境:  Windows 11 Pro 23H2

 Python 3.12

 PyCharm 2023.2.5

实验指导书实验内容

1. 运行实例中的 ProcessScheduling 项目，了解其程序结构，并理解其中的处理器调度方法是采用了何种调度算法，并在实验报告中回答。
2. 在实例中的 ProcessScheduling 项目实现按优先级调度，以及按优先级的时间片调度。

先回答指导书中的问题，原实例模拟了两种处理器调度算法，一种是 FIFO（先进先出），另一种是 FIFO + 时间片轮转的调度算法。

下面是我的 Python 实现：

```
1  import random
2  import time
3
4  class PCB:
5      def __init__(self, name, run_duration, priority):
6          self.name = name # 进程名称
7          self.run_duration = run_duration # 运行时间
8          self.priority = priority # 优先级
9          self.status = "就绪" # 初始状态设置为"就绪"
10
11     def display(self):
12         # 打印进程的状态信息
13         print(f"Process {self.name} is {self.status} and prio is {self.priority},
RunDuration is {self.run_duration}")
14
15
16 class ReadyList:
17     def __init__(self):
```

```

18         self.plist = [] # 准备队列
19
20     def push(self, pcb):
21         # 将进程添加到队列中
22         self.plist.append(pcb)
23
24     def pop(self):
25         # 从队列中取出一个进程
26         return self.plist.pop(0) if self.plist else None
27
28     def sort_by_priority(self):
29         # 按优先级对队列进行排序
30         self.plist.sort(key=lambda pcb: pcb.priority)
31
32     def is_empty(self):
33         # 检查队列是否为空
34         return len(self.plist) == 0
35
36     def display(self):
37         # 打印队列中所有进程的状态
38         for pcb in self.plist:
39             pcb.display()
40
41
42 def FIFO_scheduling(ready_list):
43     # 先进先出调度算法
44     while not ready_list.is_empty():
45         current_pcb = ready_list.pop()
46         while current_pcb.run_duration > 0:
47             print("=====")
48             current_pcb.status = "运行"
49             current_pcb.run_duration -= 1
50             current_pcb.display()
51             ready_list.display()
52             print("*****")
53             time.sleep(1)
54         current_pcb.status = "完成"
55         current_pcb.display()
56
57
58 def FIFO_time_slice(ready_list, time_slice):
59     # 带时间片的先进先出调度算法
60     while not ready_list.is_empty():
61         current_pcb = ready_list.pop()
62         for _ in range(time_slice):
63             if current_pcb.run_duration == 0:
64                 break
65             print("=====")
66             current_pcb.status = "运行"
67             current_pcb.run_duration -= 1
68             current_pcb.display()
69             ready_list.display()
70             print("*****")
71             time.sleep(1)
72         if current_pcb.run_duration > 0:
73             current_pcb.status = "就绪"

```

```

74         ready_list.push(current_pcb)
75     else:
76         current_pcb.status = "完成"
77         current_pcb.display()
78
79
80 def priority_scheduling(ready_list):
81     # 优先级调度算法
82     while not ready_list.is_empty():
83         ready_list.sort_by_priority()
84         current_pcb = ready_list.pop()
85         while current_pcb.run_duration > 0:
86             print("=====")
87             current_pcb.status = "运行"
88             current_pcb.run_duration -= 1
89             current_pcb.display()
90             ready_list.display()
91             print("*****")
92             time.sleep(1)
93         current_pcb.status = "完成"
94         current_pcb.display()
95
96
97 def priority_time_slice(ready_list, time_slice):
98     # 带时间片的优先级调度算法
99     while not ready_list.is_empty():
100         ready_list.sort_by_priority()
101         current_pcb = ready_list.pop()
102         for _ in range(time_slice):
103             if current_pcb.run_duration == 0:
104                 break
105             print("=====")
106             current_pcb.status = "运行"
107             current_pcb.run_duration -= 1
108             current_pcb.display()
109             ready_list.display()
110             print("*****")
111             time.sleep(1)
112             if current_pcb.run_duration > 0:
113                 current_pcb.status = "就绪"
114                 ready_list.push(current_pcb)
115             else:
116                 current_pcb.status = "完成"
117                 current_pcb.display()
118
119
120 def main():
121     algorithms = {
122         "1": FIFO_scheduling,
123         "2": lambda rl: FIFO_time_slice(rl, 1),
124         "3": priority_scheduling,
125         "4": lambda rl: priority_time_slice(rl, 1)
126     }
127
128     print("选择处理器调度算法")
129     print("1: 先进先出")

```

```

130     print("2: 先进先出+时间片轮转")
131     print("3: 优先级")
132     print("4: 优先级+时间片轮转")
133     choice = input("请输入你的选择: ")
134     ready_list = ReadyList()
135
136     # 生成5个随机PCB
137     for i in range(1, 6):
138         pcb = PCB(f"PCB{i}", random.randint(3, 6), random.randint(1, 3)) # 运行时间3-6,
优先级1-3
139         ready_list.push(pcb)
140
141     print("开始模拟:")
142     ready_list.display()
143     print("*****")
144
145     # 运行调度算法
146     if choice in algorithms:
147         algorithms[choice](ready_list)
148     else:
149         print("无效的选择! ")
150
151
152 if __name__ == "__main__":
153     main()
154

```

我将本次实验需要模拟的四种处理器调度算法实现在了一个 Python 程序中，程序运行后可以根据需要，选择使用对应的算法进行模拟。程序定义了几个类和函数来模拟进程的创建、管理和调度。下面是各部分的详细解释：

2.1 类 PCB（进程控制块）

- `__init__(self, name, run_duration, priority)`：初始化函数，用于创建一个新的进程控制块（PCB）。它接受进程名称 `name`、运行时间 `run_duration` 和优先级 `priority` 作为参数。
- `display(self)`：用于打印当前进程的状态、优先级和剩余运行时间。

2.2 类 ReadyList

- `__init__(self)`：初始化函数，创建一个空的就绪队列。
- `push(self, pcb)`：将一个进程（PCB）添加到就绪队列中。
- `pop(self)`：从就绪队列中取出（并删除）第一个进程。
- `sort_by_priority(self)`：根据优先级对就绪队列中的进程进行排序。
- `is_empty(self)`：检查就绪队列是否为空。
- `display(self)`：打印就绪队列中所有进程的信息。

2.3 调度算法函数

- `FIFO_scheduling(ready_list)`: 实现先进先出 (FIFO) 调度算法。不断从就绪队列中取出进程并运行, 直到队列为空。
- `FIFO_time_slice(ready_list, time_slice)`: 实现带时间片的FIFO调度。每个进程运行固定的时间片后, 重新回到队列末尾 (如果还有剩余运行时间)。
- `priority_scheduling(ready_list)`: 实现基于优先级的调度。总是选取优先级最高的进程运行。
- `priority_time_slice(ready_list, time_slice)`: 结合优先级和时间片的调度算法。优先级高的进程先运行, 每个进程的运行时间受时间片限制。

2.4 主函数 `main`

- 提供一个菜单供用户选择不同的调度算法。
- 随机生成5个进程, 并将它们添加到就绪队列。
- 根据用户的选择执行相应的调度算法。

2.5 运行流程

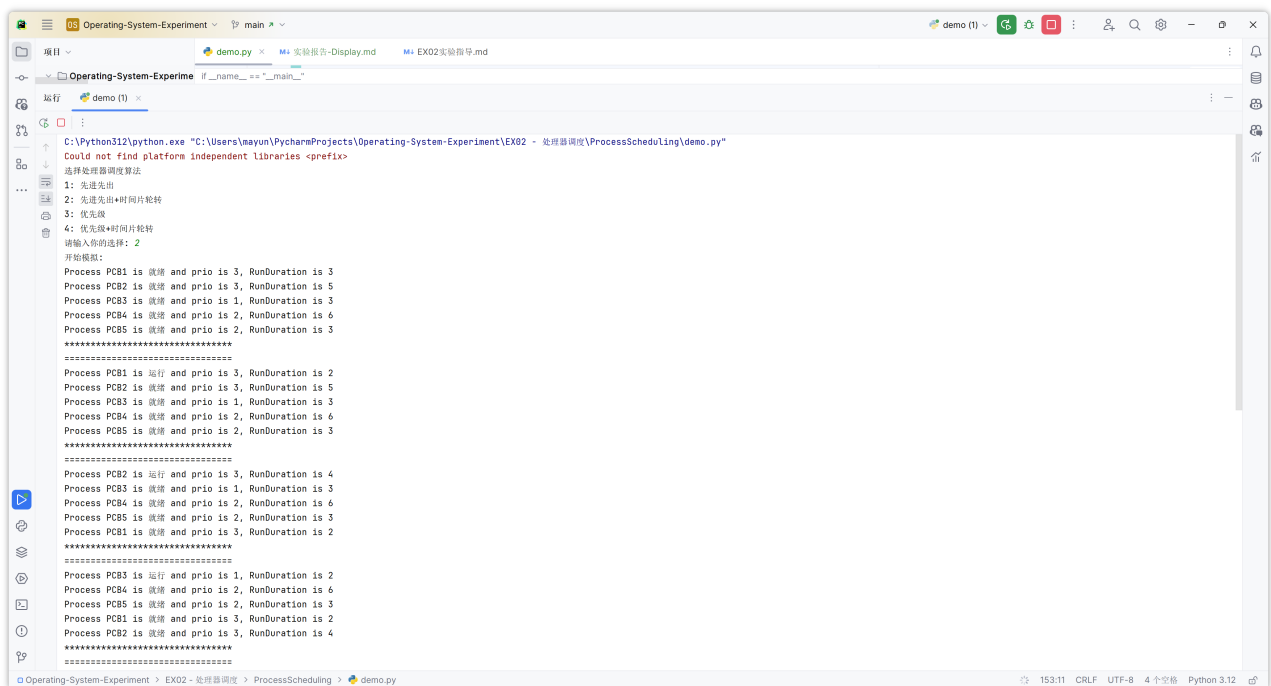
- 程序开始时, 用户选择一个调度算法。
- 程序创建5个具有随机运行时间和优先级的进程。
- 根据所选算法, 程序模拟进程调度的过程, 显示每个进程的状态和就绪队列的变化。

2.6 其他

- 程序使用了 `time.sleep(1)` 来模拟进程运行, 使得每次状态变更之间有一秒的间隔。
- 这种模拟有助于理解不同调度算法的工作原理和区别。

3 实验结果

3.1 FIFO (先进先出)



3.3 优先级

The screenshot displays a code editor window titled "Operating-System-Experiment" with a file named "demo.py". The script implements a process scheduling simulation using a priority queue.

Script Content:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
__author__ = 'mayun'
if __name__ == '__main__':
    import sys
    from collections import defaultdict
    from heapq import heappop, heappush

    # 选择处理调度算法
    print('1: 先进先出')
    print('2: 先来先出+时间片轮转')
    print('3: 优先级')
    print('4: 优先级+时间片轮转')
    choice = input('请输入你的选择: ')

    # 开始模拟
    processes = [
        ('PCB1', 3), ('PCB2', 6), ('PCB3', 5), ('PCB4', 4), ('PCB5', 6)
    ]

    # 按照优先级排序
    processes.sort(key=lambda x: x[1])

    # 运行过程
    while True:
        # 打印当前队列中的进程信息
        print('=====' + '\n'.join('%s %s %s' % (p[0], p[1], p[2]) for p in processes))

        # 取出优先级最高的进程
        p = heappop(processes)

        # 运行该进程
        p[2] -= 1
        if p[2] > 0:
            # 将进程放回队列中
            heappush(processes, p)
        else:
            # 进程完成，输出结果并退出
            print(p[0] + ' 运行结束')
            break
```

Output:

```
C:\Python312\python.exe "C:/Users/mayun/PycharmProjects/Operating-System-Experiment/EX02 - 处理器调度/ProcessScheduling/demo.py"
Could not find platform independent libraries <prefix>
选择处理调度算法
1: 先进先出
2: 先来先出+时间片轮转
3: 优先级
4: 优先级+时间片轮转
请输入你的选择: 3
开始模拟：
Process PCB1 is 就绪 and prio is 3, RunDuration is 6
Process PCB2 is 就绪 and prio is 1, RunDuration is 6
Process PCB3 is 就绪 and prio is 1, RunDuration is 5
Process PCB4 is 就绪 and prio is 2, RunDuration is 4
Process PCB5 is 就绪 and prio is 3, RunDuration is 6
=====
Process PCB2 is 运行 and prio is 1, RunDuration is 5
Process PCB3 is 就绪 and prio is 1, RunDuration is 5
Process PCB4 is 就绪 and prio is 2, RunDuration is 4
Process PCB1 is 就绪 and prio is 3, RunDuration is 6
Process PCB5 is 就绪 and prio is 3, RunDuration is 6
=====
Process PCB2 is 运行 and prio is 1, RunDuration is 4
Process PCB3 is 就绪 and prio is 1, RunDuration is 5
Process PCB4 is 就绪 and prio is 2, RunDuration is 4
Process PCB1 is 就绪 and prio is 3, RunDuration is 6
Process PCB5 is 就绪 and prio is 3, RunDuration is 6
=====
Process PCB2 is 运行 and prio is 1, RunDuration is 3
Process PCB3 is 就绪 and prio is 1, RunDuration is 5
Process PCB4 is 就绪 and prio is 2, RunDuration is 4
Process PCB1 is 就绪 and prio is 3, RunDuration is 6
Process PCB5 is 就绪 and prio is 3, RunDuration is 6
=====
Process PCB2 is 运行结束
```

The bottom status bar shows the file path: "Operating-System-Experiment > EX02 - 处理器调度 > ProcessScheduling > demo.py".

3.4 优先级 + 时间片轮转

Operating-System-Experiment

main

项目

demo.py

实验报告-Display.md

EX02实验指导.md

Operating-System-Experiment

if __name__ == "__main__":

运行 demo (1)

C:\Python312\python.exe "C:\Users\mayun\PycharmProjects\Operating-System-Experiment\EX02 - 处理器调度\ProcessScheduling\demo.py"

Could not find platform independent libraries <prefix>

选择处理器调度算法

1: 先进先出

2: 先进先出+时间片轮转

3: 优先调度

4: 优先调度+时间片轮转

请输入你的选择: 4

开始模拟:

Process PCB1 is 就绪 and prio is 3, RunDuration is 3

Process PCB2 is 就绪 and prio is 1, RunDuration is 3

Process PCB3 is 就绪 and prio is 2, RunDuration is 4

Process PCB4 is 就绪 and prio is 3, RunDuration is 5

Process PCB5 is 就绪 and prio is 3, RunDuration is 4

Process PCB2 is 运行 and prio is 1, RunDuration is 2

Process PCB3 is 就绪 and prio is 2, RunDuration is 4

Process PCB1 is 就绪 and prio is 3, RunDuration is 3

Process PCB4 is 就绪 and prio is 3, RunDuration is 5

Process PCB5 is 就绪 and prio is 3, RunDuration is 4

Process PCB2 is 运行 and prio is 1, RunDuration is 1

Process PCB3 is 就绪 and prio is 2, RunDuration is 4

Process PCB1 is 就绪 and prio is 3, RunDuration is 3

Process PCB4 is 就绪 and prio is 3, RunDuration is 5

Process PCB5 is 就绪 and prio is 3, RunDuration is 4

Process PCB2 is 运行 and prio is 1, RunDuration is 0

Process PCB3 is 就绪 and prio is 2, RunDuration is 4

Process PCB1 is 就绪 and prio is 3, RunDuration is 3

Process PCB4 is 就绪 and prio is 3, RunDuration is 5

Process PCB5 is 就绪 and prio is 3, RunDuration is 4

Process PCB2 is 完成 and prio is 1, RunDuration is 0

Operating-System-Experiment > EX02 - 处理器调度 > ProcessScheduling > demo.py

153/31 CRLF UTF-8 4 个空格 Python 3.12

4 实验总结

通过这次实验，我不仅加深了对处理器调度算法的理解，也提高了我的编程能力。特别是在调度算法的设计和实现过程中，我学会了如何将理论知识应用到实际问题中。此外，这次实验也让我意识到，在设计操作系统或类似系统时，算法的选择对系统性能有着重要影响。在未来的学习和工作中，我将更加注重理论与实践的结合，不断提高自己解决复杂问题的能力。