華中師範大學
CENTRAL CHINA NORMAL UNIVERSITY

操作系统实验报告

第一次实验

姓名 单禹嘉

学号 2023215177

课程 操作系统

学院 计算机学院

2025 年 9 月 25 日

1 实验目的和要求

- 掌握周转时间、等待时间、平均周转时间等概念及其计算方法。
- 理解三种常用的进程调度算法，区分算法之间的差异性，并模拟实现各算法。
- 了解操作系统中高级调度、中级调度和低级调度的区别和联系

2 问题描述

- (1) 在单道环境下，已知 n 个作业的进入时间和估计运行时间（以分钟计），分别求出每一个作业的开始时间、结束时间、周转时间、带权周转时间，以及这些作业的平均周转时间和带权平均周转时间；
- (2) 在多道环境（如 2 道）下，已知 n 个作业的进入时间和估计运行时间（以分钟计），分别求出每一个作业的开始时间、结束时间、周转时间、带权周转时间，以及这些作业的平均周转时间和带权平均周转时间。

3 实验要求

- 分别用先来先服务调度算法（FCFS）、短作业优先调度算法（SJF）、响应比高者优先调度算法（HRRN），求出批作业的平均周转时间和带权平均周转时间；
- 就同一批次作业，分别讨论这些算法的优劣；
- 衡量同一调度算法对不同作业流的性能。

4 实验环境

- 开发工具：VS Code
- 编程语言：Rust

5 设计思想及实验步骤

（包括实验设计原理，分析方法、计算步骤、模块组织，或主要流程图、伪代码等）

5.1 实验设计原理

本实验采用非抢占式批处理作业调度模型，核心指标：周转时间 $T_i = C_i - A_i$、带权周转时间 $W_i = T_i/S_i$，以及其平均值 $\overline{T} = \frac{1}{n}\sum_i T_i$、$\overline{W} = \frac{1}{n}\sum_i W_i$。多道情形抽象为 m 条并行“道”，作业一旦开始在某道上运行至完成。

5.2 分析方法

实现三种典型非抢占调度算法：FCFS（按到达时间先后分配到最早空闲道）、SJF（就绪集合选服务时间最短者）、HRRN（就绪集合按 $R = (t - A + S)/S$ 从大到小选择）。时间推进遵循事件驱动：若有空闲道且就绪非空则立即分配，否则跳至下一个到达或最近空闲时间的较小者。

5.3 计算步骤

1. 构造作业流 (A_i, S_i)，设定道数 m；
2. 维护就绪集合与各道最早空闲时间；
3. 依据所选算法分配作业并记录开始/结束时间；
4. 计算 T_i, W_i 及 $\overline{T}, \overline{W}$；
5. 输出并整理为表格，进行对比分析。

5.4 模块组织

Rust 程序包含：结构体 ‘Job‘；函数 ‘schedule_fcfs‘、‘schedule_sjf‘、‘schedule_hrrn‘；‘print_results‘ 统计与打印；‘main‘ 组装单道/双道与两组作业流实验。

5.5 伪代码

以 SJF 为例（非抢占，多道）：

```
time <- 0; core_free[m] <- 0; ready <- {}
按到达时间排序 all
while 未完成:
  将 arrival <= time 的作业加入 ready
  free_cores <- {k | core_free[k] <= time}
  if free_cores 为空:
    if ready 非空: time <- 最早 core_free
    else: time <- min(下一个到达, 最早 core_free); continue
```

```
  if ready 为空: time <- 下一个到达; continue
  按 service 升序排序 ready
  对每个空闲 core:
    取最短作业 j; start <- time; end <- start + service
    记录 j.start/j.end; core_free[core] <- end; 加入 finished
  time <- min(下一个到达, 最早 core_free)
```

6 实验结果及分析

以下给出程序运行结果（单位：分钟）。

原始终端输出

```
=== FCFS - 单道 ===
id      arr     serv    start   end     turn    wturn
1       0.00    3.00    0.00    3.00    3.00    1.00
2       2.00    6.00    3.00    9.00    7.00    1.17
3       4.00    4.00    9.00    13.00   9.00    2.25
4       6.00    5.00    13.00   18.00   12.00   2.40
5       8.00    2.00    18.00   20.00   12.00   6.00
平均周转时间 = 8.6000
带权平均周转时间 = 2.5633

=== SJF - 单道 ===
id      arr     serv    start   end     turn    wturn
1       0.00    3.00    0.00    3.00    3.00    1.00
2       2.00    6.00    3.00    9.00    7.00    1.17
3       4.00    4.00    11.00   15.00   11.00   2.75
4       6.00    5.00    15.00   20.00   14.00   2.80
5       8.00    2.00    9.00    11.00   3.00    1.50
平均周转时间 = 7.6000
带权平均周转时间 = 1.8433

=== HRRN - 单道 ===
id      arr     serv    start   end     turn    wturn
1       0.00    3.00    0.00    3.00    3.00    1.00
2       2.00    6.00    3.00    9.00    7.00    1.17
3       4.00    4.00    9.00    13.00   9.00    2.25
```

```
4       6.00    5.00    15.00   20.00   14.00   2.80
5       8.00    2.00    13.00   15.00   7.00    3.50
平均周转时间 = 8.0000
带权平均周转时间 = 2.1433

=== FCFS - 双道 ===
id      arr     serv    start   end     turn    wturn
1       0.00    3.00    0.00    3.00    3.00    1.00
2       2.00    6.00    2.00    8.00    6.00    1.00
3       4.00    4.00    4.00    8.00    4.00    1.00
4       6.00    5.00    8.00    13.00   7.00    1.40
5       8.00    2.00    8.00    10.00   2.00    1.00
平均周转时间 = 4.4000
带权平均周转时间 = 1.0800

=== SJF - 双道 ===
id      arr     serv    start   end     turn    wturn
1       0.00    3.00    0.00    3.00    3.00    1.00
2       2.00    6.00    2.00    8.00    6.00    1.00
3       4.00    4.00    4.00    8.00    4.00    1.00
4       6.00    5.00    8.00    13.00   7.00    1.40
5       8.00    2.00    8.00    10.00   2.00    1.00
平均周转时间 = 4.4000
带权平均周转时间 = 1.0800

=== HRRN - 双道 ===
id      arr     serv    start   end     turn    wturn
1       0.00    3.00    0.00    3.00    3.00    1.00
2       2.00    6.00    2.00    8.00    6.00    1.00
3       4.00    4.00    4.00    8.00    4.00    1.00
4       6.00    5.00    8.00    13.00   7.00    1.40
5       8.00    2.00    8.00    10.00   2.00    1.00
平均周转时间 = 4.4000
带权平均周转时间 = 1.0800

=== 同一算法在不同作业流上的比较（示例） ===
```

```
=== Stream A - FCFS - 单道 ===
id      arr     serv    start   end     turn    wturn
1       0.00    3.00    0.00    3.00    3.00    1.00
2       2.00    6.00    3.00    9.00    7.00    1.17
3       4.00    4.00    9.00    13.00   9.00    2.25
4       6.00    5.00    13.00   18.00   12.00   2.40
5       8.00    2.00    18.00   20.00   12.00   6.00
平均周转时间 = 8.6000
带权平均周转时间 = 2.5633

=== Stream B - FCFS - 单道 ===
id      arr     serv    start   end     turn    wturn
1       0.00    8.00    0.00    8.00    8.00    1.00
2       1.00    4.00    8.00    12.00   11.00   2.75
3       2.00    9.00    12.00   21.00   19.00   2.11
4       3.00    5.00    21.00   26.00   23.00   4.60
5       10.00   2.00    26.00   28.00   18.00   9.00
6       10.00   1.00    28.00   29.00   19.00   19.00
平均周转时间 = 16.3333
带权平均周转时间 = 6.4102
(base)   lab1 git:(main)
```

6.1 单道（m=1）

id	arr	serv	start	end	turn	wturn
1	0.00	3.00	0.00	3.00	3.00	1.00
2	2.00	6.00	3.00	9.00	7.00	1.17
3	4.00	4.00	9.00	13.00	9.00	2.25
4	6.00	5.00	13.00	18.00	12.00	2.40
5	8.00	2.00	18.00	20.00	12.00	6.00

平均周转时间 = 8.6000，带权平均周转时间 = 2.5633

FCFS

SJF

HRRN

id	arr	serv	start	end	turn	wturn
1	0.00	3.00	0.00	3.00	3.00	1.00
2	2.00	6.00	3.00	9.00	7.00	1.17
3	4.00	4.00	11.00	15.00	11.00	2.75
4	6.00	5.00	15.00	20.00	14.00	2.80
5	8.00	2.00	9.00	11.00	3.00	1.50

平均周转时间 = 7.6000，带权平均周转时间 = 1.8433

id	arr	serv	start	end	turn	wturn
1	0.00	3.00	0.00	3.00	3.00	1.00
2	2.00	6.00	3.00	9.00	7.00	1.17
3	4.00	4.00	9.00	13.00	9.00	2.25
4	6.00	5.00	15.00	20.00	14.00	2.80
5	8.00	2.00	13.00	15.00	7.00	3.50

平均周转时间 = 8.0000，带权平均周转时间 = 2.1433

6.2 双道（m=2）

三种算法在该作业流下得到相同的调度与指标：

id	arr	serv	start	end	turn	wturn
1	0.00	3.00	0.00	3.00	3.00	1.00
2	2.00	6.00	2.00	8.00	6.00	1.00
3	4.00	4.00	4.00	8.00	4.00	1.00
4	6.00	5.00	8.00	13.00	7.00	1.40
5	8.00	2.00	8.00	10.00	2.00	1.00

平均周转时间 = 4.4000，带权平均周转时间 = 1.0800

6.3 讨论与分析

- 单道下：SJF 的 $\overline{T}, \overline{W}$ 最低；HRRN 介于 FCFS 与 SJF 之间，能够缓解长作业饥饿；
- 双道下：由于到达/服务时间结构，本例三算法输出一致，且显著优于单道；
- 多道并行能降低等待时间；真实系统若考虑 I/O、抢占、优先级等，策略需进一步扩展。

7 附录：部分源代码

```
use std::cmp::Ordering;

#[derive(Clone, Debug)]
struct Job {
    id: usize,
    arrival: f64, // 到达时间，分钟
    service: f64, // 估计运行时间，分钟
    start: Option<f64>,
    end: Option<f64>,
}

impl Job {
    fn new(id: usize, arrival: f64, service: f64) -> Self {
        Self { id, arrival, service, start: None, end: None }
    }

    fn turnaround(&self) -> Option<f64> {
        match (self.end, Some(self.arrival)) {
            (Some(e), Some(a)) => Some(e - a),
            _ => None,
        }
    }

    fn weighted_turnaround(&self) -> Option<f64> {
        match (self.turnaround(), self.service) {
            (Some(t), s) if s > 0.0 => Some(t / s),
            _ => None,
        }
    }
}
```

8 写在最后

8.1 发布地址

- Github: https://github.com/eleliauk/OS-LABS