Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное учреждение высшего образования

«Ульяновский государственный технический университет»

Дисциплина «Операционные системы».

**Лабораторная работа №2. Алгоритмы планирования**

**Выполнил**

Студент группы ИВТИИбд-21

Алатырский Даниил Игоревич

**Проверил(а):**

Старший Преподаватель Беляев Константин Сергеевич

Ульяновск 2025

### **1. Цель работы**

Реализовать в C следующие алгоритмы планирования процессов в заранее заданном наборе задач (все задачи поступают одновременно, время прибытия = 0 мс):

* FCFS (First-Come, First-Served)
* SJF (Shortest-Job-First), непредвзятый
* Приоритетное планирование (непредвзятое)
* RR (Round-Robin) с квантом 10 мс
* Приоритетное RR: задачи выбираются по убыванию приоритета, для одинаковых приоритетов – RR с квантами 10 мс

Каждая задача характеризуется именем, приоритетом (1–10, большее значение – более высокий приоритет) и требуемым временем работы на ЦПУ (burst).

В дополнительном задании необходимо было подсчитать для каждого алгоритма:

* **Время отклика** (Response Time)
* **Время ожидания** (Waiting Time)
* **Время оборота** (Turnaround Time)

### 2. Принцип реализации

1. **Добавление задач**  
   — все задачи помещаются в единый односвязный список; порядок вставки зависит от задачи:
   * FCFS, SJF, RR — append (в конец списка)
   * Приоритетное — insert, поддерживающий отсортированный по убыванию приоритета список
   * Приоритетное RR — аналогично приоритетному, но при реинсёрте в список приоритет остаётся ключом
2. **Выбор следующей задачи**
   * FCFS: берётся голова списка
   * SJF: линейный поиск задачи с минимальным burst
   * Приоритетное: голова отсортированного по приоритету списка
   * RR: голова списка, затем — реинсёрт в хвост (если осталось > кванта)
   * Приоритетное RR: поиск задачи с максимальным приоритетом, выполнение квант-ом, реинсёрт в список (с учётом приоритета)
3. **Вызов выполнения**  
   Для каждой «выбранной» задачи вызывается run(task, t), где t = burst (для непредвзятых) или t = min(burst, QUANTUM) (для RR-алгоритмов). После полного выполнения задачи (burst = 0) удаляем её окончательно, иначе обновляем burst и возвращаем в список.
4. **Сбор статистики**  
   При первом запуске задачи фиксируется время начала — это и будет её временем отклика. Время завершения вычисляется по глобальному «часовому» счётчику, увеличивающемуся на каждое run(...).
   * **Response = start\_time − arrival\_time**
   * **Turnaround = finish\_time − arrival\_time**
   * **Waiting = Turnaround − оригинальный\_burst**

### **4. Вывод**

* **SJF** даёт наименьшие средние времена ожидания и оборота, но не гарантирует справедливость.
* **FCFS** проста в реализации, но страдает «эффектом длинных задач».
* **Приоритетное** ускоряет важные задачи, но в базовой форме может приводить к «голоданию» низкоприоритетных.
* **RR** гарантирует регулярное обслуживание всех задач (низкое время отклика), но увеличивает общее время оборота из-за частых переключений.
* **Приоритетное RR** объединяет преимущества приоритетов и квантования, снижая время ожидания для высокоприоритетных задач, одновременно ограничивая «голодание» для остальных.