Министерство науки и высшего образования Российской Федерации

Ульяновский государственный технический университет

Кафедра «Вычислительная техника»

Дисциплина «Операционные системы»

**Лабораторная работа №2**

**«****Алгоритмы планирования»**

**Выполнил**:

студент гр. ИВТАПбд-31

Сокольский Р. С.

**Проверил работу:**

Беляев К. С.

Ульяновск 2025

## Цель работы.

В ходе работы были реализованы и протестированы различные алгоритмы планирования задач, что позволило выявить их сильные и слабые стороны. Алгоритм FCFS обеспечивает простоту реализации, но неэффективен при наличии долгих задач. Метод SJF минимизирует среднее время ожидания, однако может привести к голоданию долгих процессов. Приоритетное планирование позволяет учитывать важность задач, но также может создавать условия, при которых низкоприоритетные процессы остаются без выполнения. Циклическое планирование RR демонстрирует справедливое распределение ресурсов, но увеличивает накладные расходы на переключение контекста.

**Ход работы.**

Работа началась с изучения основных алгоритмов планирования процессов: FCFS (первый пришел – первый обслужен), SJF (кратчайшая задача первой), приоритетное планирование, циклическое планирование (Round Robin) и циклическое планирование с приоритетом. Основной задачей было реализовать каждый из этих алгоритмов в консоли Linux, обеспечивая корректное управление задачами и измерение параметров их выполнения (время отклика, время ожидания, время оборота).

На первом этапе были определены основные структуры данных. Каждая задача представлена структурой Task, содержащей название, приоритет и время выполнения (burst). Для хранения задач использовался связанный список node, где каждая запись содержала указатель на Task и ссылку на следующий элемент списка. Были реализованы базовые операции: добавление задачи в список (add()), выбор следующей задачи (pickNextTask()) и удаление завершённых задач (delete()).

Реализация алгоритма FCFS (First-Come, First-Served) потребовала организации списка задач в порядке их поступления. Планировщик проходил по списку, выбирал первую задачу и выполнял её до завершения. После выполнения задачи она удалялась, и выполнение переходило к следующей. Основным недостатком алгоритма является высокая вероятность проблемы «голодания» для долгих задач, если впереди них находится задача с продолжительным временем выполнения.

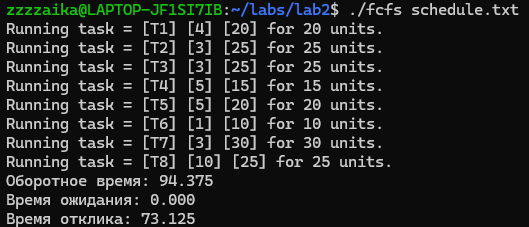


Рис. 1 вывод команды ./fcfs schedule.txt.

Вторым реализованным алгоритмом был SJF (Shortest Job First). Основное отличие от FCFS заключалось в том, что перед выполнением задачи сортировались по времени burst, и всегда выбиралась задача с наименьшей продолжительностью выполнения. Это позволило минимизировать среднее время ожидания, однако алгоритм страдал от проблемы голодания, поскольку короткие задачи могли бесконечно опережать длинные.

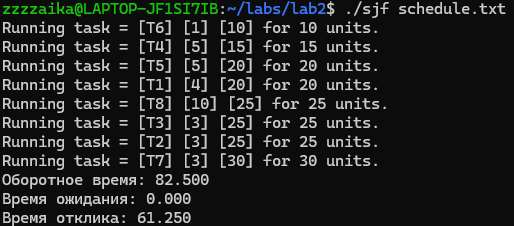


Рис. 2 вывод команды ./sjf schedule.txt.

Для реализации приоритетного планирования задачи перед выполнением сортировались по убыванию приоритетов. Если несколько задач имели одинаковый приоритет, они выполнялись в порядке поступления (аналогично FCFS). Приоритетность позволяла управлять распределением ресурсов между процессами, но могла привести к голоданию задач с низким приоритетом, если в системе постоянно появлялись более важные задачи.

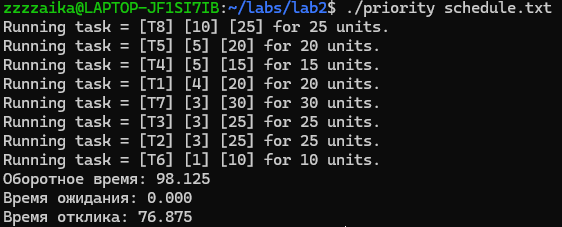


Рис. 3 вывод команды ./priority schedule.txt.

Алгоритм Round Robin (RR) основывался на циклическом обслуживании задач. Каждая задача получала фиксированное время выполнения (квант), после чего, если она не завершилась, переносилась в конец очереди. Это обеспечивало равномерное распределение процессорного времени между всеми задачами. Для реализации использовалась переменная QUANT, определяющая длину кванта. Если время выполнения задачи превышало QUANT, задача выполнялась частично, после чего оставшееся время уменьшалось, и задача вновь ставилась в очередь. Этот алгоритм обеспечивал хорошую отзывчивость системы, но мог приводить к большому количеству переключений контекста.

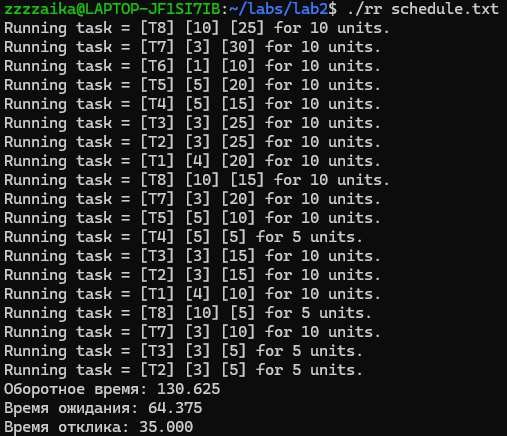


Рис. 4 вывод команды ./rr schedule.txt.

Наконец, был реализован алгоритм циклического планирования с приоритетами. Он сочетал два предыдущих подхода: задачи сначала сортировались по приоритету, после чего те, что имели одинаковый приоритет, выполнялись по принципу Round Robin. Это позволило обеспечить баланс между приоритетами и справедливым распределением процессорного времени между задачами одного уровня.

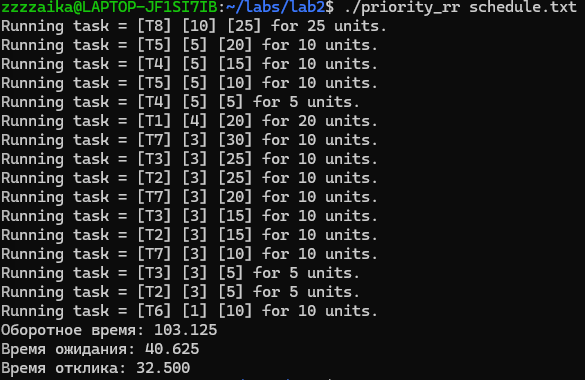


Рис. 5 вывод команды ./priority\_rr schedule.txt.

После реализации всех алгоритмов была проведена проверка их работы на тестовых данных. В качестве входных данных использовались задачи:

* T1, приоритет 4, burst 20
* T2, приоритет 3, burst 25
* T3, приоритет 3, burst 25
* T4, приоритет 5, burst 15
* T5, приоритет 5, burst 20
* T6, приоритет 1, burst 10
* T7, приоритет 3, burst 30
* T8, приоритет 10, burst 25

Программа корректно отрабатывала все алгоритмы, измеряя и выводя показатели их эффективности, такие как среднее время ожидания, среднее время оборота и среднее время отклика. По результатам тестирования было подтверждено, что SJF минимизирует среднее время ожидания, RR обеспечивает справедливое распределение ресурсов, а приоритетное планирование эффективно выделяет процессорное время критически важным задачам.

**Вывод.**

В ходе работы были реализованы и протестированы различные алгоритмы планирования задач, что позволило выявить их сильные и слабые стороны. Алгоритм FCFS обеспечивает простоту реализации, но неэффективен при наличии долгих задач. Метод SJF минимизирует среднее время ожидания, однако может привести к голоданию долгих процессов. Приоритетное планирование позволяет учитывать важность задач, но также может создавать условия, при которых низкоприоритетные процессы остаются без выполнения. Циклическое планирование RR демонстрирует справедливое распределение ресурсов, но увеличивает накладные расходы на переключение контекста. Комбинированный подход, сочетающий приоритетное планирование с циклическим, показал себя как сбалансированный метод, позволяющий учитывать приоритеты, но не допускать блокировки низкоприоритетных задач. Проведённый анализ подтвердил, что выбор подходящего алгоритма зависит от требований системы и характера выполняемых процессов.