Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное учреждение высшего образования

«Ульяновский государственный технический университет»

Дисциплина «Операционные системы».

**Лабораторная работа №4. Проектирование пула потоков**

**Выполнил**

Студент группы ИВТИИбд-21

Алатырский Даниил Игоревич

**Проверил(а):**

Старший Преподаватель Беляев Константин Сергеевич

Ульяновск 2025

### **1. Цель работы**

Реализовать пул потоков на базе POSIX Threads, обеспечивающий:

* инициализацию фиксированного числа рабочих потоков (pool\_init());
* приём и очередь задач (pool\_submit());
* ожидание задач потоками и их выполнение;
* корректное завершение всех потоков после pool\_shutdown().

### 2. Описание реализации

1. **Структуры данных**
   * thread\_pool\_t pool содержит кольцевую очередь задач размером QUEUE\_SIZE, указатели head/tail, счётчик count, мьютекс и семафор, а также флаг shutdown\_flag.
   * Массив pthread\_t threads[MAX\_THREADS] хранит дескрипторы рабочих потоков.
2. **Инициализация пула** (pool\_init)
   * Обнуляются head, tail, count, сбрасывается shutdown\_flag.
   * Создаётся мьютекс и безымянный семафор (sem\_init с начальным значением 0).
   * Запускаются MAX\_THREADS потоков, каждый из которых выполняет функцию worker().
3. **Очередь задач**
   * **enqueue**: под мьютексом проверяет, не заполнена ли очередь; вставляет задачу в позицию tail, инкрементирует tail и count; после разблокировки выпускает один семафор (sem\_post), сигнализируя о появлении работы.
   * **dequeue**: под мьютексом проверяет, не пуста ли очередь; извлекает задачу из head, инкрементирует head и декрементирует count.
4. **Рабочий поток** (worker)
   * В бесконечном цикле вызывает sem\_wait(), блокируясь до появления задачи или до сигнала завершения.
   * После пробуждения проверяет под мьютексом флаг shutdown\_flag и пустоту очереди — в этом случае выходим из цикла и завершаем поток.
   * Иначе извлекает задачу (dequeue) и выполняет её, вызывая task.function(task.data).
5. **Отправка задачи** (pool\_submit)
   * Формирует task\_t и вызывает enqueue.
   * При успешном добавлении вновь увеличивает семафор (sem\_post), чтобы «разбудить» ожидающий поток.
6. **Завершение работы** (pool\_shutdown)
   * Под мьютексом выставляет shutdown\_flag = 1.
   * Делает sem\_post столько раз, сколько потоков, чтобы все sem\_wait разблокировались.
   * Выходит из критической секции и вызывает pthread\_join для каждого рабочего потока.
   * Уничтожает мьютекс и семафор.

### 3. Результаты

В тестовой программе main() создаётся 20 задач, каждая из которых печатает свой номер и «спит» 1 с. Пул из 10 потоков позволяет выполнять 10 задач одновременно, после чего освобождающиеся потоки забирают следующие.

### **4**. Вывод****

Пул потоков корректно:

* балансирует нагрузку между заранее созданными потоками,
* гарантирует отсутствие состязаний при доступе к очереди (благодаря мьютексу),
* эффективно уведомляет о появлении работы через семафор,
* позволяет «правильно» остановить все потоки после pool\_shutdown().