Министерство науки и образования Российской Федерации

ФГБОУ ВО «Ульяновский государственный технический университет»

Кафедра «Вычислительная техника»

**Лабораторная работа №4**

Дисциплина: «Операционные системы»

«Проектирование пула потоков»

Выполнил студент  
группы ИВТбд-23

Волков Р.Р

Проверил:  
преподаватель кафедры ВТ  
Беляев К. С.

Ульяновск, 2024

**Постановка задачи**

При использовании пулов потоков задача передается в пул и выполняется потоком из пула. Работа передаётся в пул с помощью очереди, а доступный поток удаляет работу из очереди. Если доступных потоков нет, работа остается в очереди до тех пор, пока один из них не станет доступным. Если работы нет, потоки ожидают уведомления, пока задача не станет доступной.

Этот проект включает в себя создание пула потоков и управление им. Его нужно выполнить с использованием PThreads и Posix синхронизации. Ниже приводится необходимая информация для реализации проекта.

Реализация пула потоков:

1. Функция pool\_init() создаст потоки при запуске, а также инициализирует блокировки взаимного исключения(mutex) и семафоры.
2. Функция pool\_submit() частично реализована и в настоящее время помещает выполняемую функцию, а также ее данные, в структуру задачи. Структура задачи представляет работу, которая будет выполнена потоком в пуле. pool\_submit() добавит эти задачи в очередь, вызывая функцию enqueue(), а рабочие потоки вызовут dequeue() для получения работы из очереди. Очередь может быть реализована статически (с использованием массивов) или динамически (с использованием связанного списка). Функция pool\_init() имеет возвращаемое значение int, которое используется для указания того, была ли задача успешно отправлена в пул (0 указывает на успех, 1 указывает на неудачу). Если очередь реализована с использованием массивов, pool\_init() вернет 1, если будет попытка отправить работу и очередь заполнена. Если очередь реализована как связанный список, pool\_init() пула всегда должен возвращать 0, если только не произойдет ошибка выделения памяти.
3. Функция worker() выполняется каждым потоком в пуле, где каждый поток будет ожидать доступной работы. Как только работа станет доступной, поток удалит ее из очереди и вызовет метод execute() для запуска указанной функции. Семафор можно использовать для уведомления ожидающего потока, когда работа передается в пул потоков. Могут использоваться как именованные, так и безымянные семафоры.
4. Блокировка мьютекса необходима во избежание состояний гонки при доступе или изменении очереди.
5. Функция pool\_shutdown() отменит каждый рабочий поток, а затем будет ждать завершения каждого потока, вызывая pthread join(). (Операция семафора sem\_wait() — это точка отмены, которая позволяет отменить поток, ожидающий семафора.)

**Пример работы программы**

Входные данные программы показаны на рисунке 1, вывод программы показан на рисунке 2.

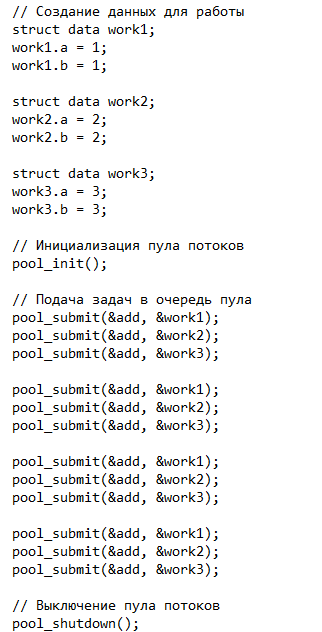


Рисунок 1. Входные данные

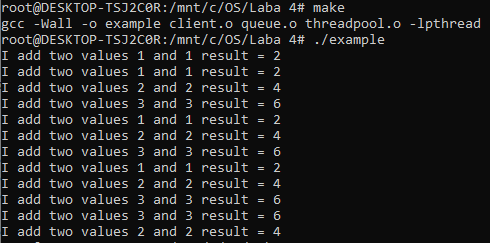


Рисунок 2. Вывод программы

**Вывод**

При выполнении данной лабораторной работы было реализовано себя создание пула потоков и управление им. Выполнение было с использованием PThreads и Posix синхронизации. Реализация пула потоков состояла из:

1. Функция pool\_init() создаст потоки при запуске, а также инициализирует блокировки взаимного исключения(mutex) и семафоры.

2. Функция pool\_submit() частично реализована и в настоящее время помещает выполняемую функцию, а также ее данные, в структуру задачи. Структура задачи представляет работу, которая будет выполнена потоком в пуле. pool\_submit() добавит эти задачи в очередь, вызывая функцию enqueue(), а рабочие потоки вызовут dequeue() для получения работы из очереди. Очередь может быть реализована статически (с использованием массивов) или динамически (с использованием связанного списка). Функция pool\_init() имеет возвращаемое значение int, которое используется для указания того, была ли задача успешно отправлена в пул (0 указывает на успех, 1 указывает на неудачу). Если очередь реализована с использованием массивов, pool\_init() вернет 1, если будет попытка отправить работу и очередь заполнена. Если очередь реализована как связанный список, pool\_init() пула всегда должен возвращать 0, если только не произойдет ошибка выделения памяти.

3. Функция worker() выполняется каждым потоком в пуле, где каждый поток будет ожидать доступной работы. Как только работа станет доступной, поток удалит ее из очереди и вызовет метод execute() для запуска указанной функции. Семафор можно использовать для уведомления ожидающего потока, когда работа передается в пул потоков. Могут использоваться как именованные, так и безымянные семафоры.

4. Блокировка мьютекса необходима во избежание состояний гонки при доступе или изменении очереди.

5. Функция pool\_shutdown() отменит каждый рабочий поток, а затем будет ждать завершения каждого потока, вызывая pthread join(). (Операция семафора sem\_wait() — это точка отмены, которая позволяет отменить поток, ожидающий семафора.)