Задание 9

Основных целей две, нужно переработать код из задания про пакетную обработку команд (**Задание 6**) таким образом, чтобы:

1. команды обрабатывались многопоточно
2. ввод данных управлялся внешним кодом.

1. Многопоточная обработка команд

При старте программы должно быть создано в дополнение к существующему основному потоку ещё три дополнительных. Дадим им условные имена:

- **main** – основной поток (в пользовательском приложении)
- **log** – поток для вывода данных в консоль
- **file1** – первый поток для вывода в файл
- **file2** – второй поток для вывода в файл

Основная логика обработки меняется таким образом, что блок команд после своего формирования должен быть отправлен в консоль (потоком **log**) и сразу в файл (одним из потоков **file1** или **file2**). При этом отправка блока в файл распределяется между файловыми потоками.

Можно напрямую отправлять, например, чётные команды через поток **file1**, а нечётные – через **file2**. Но лучшим решение станет использование единой очереди команд, которую будут обрабатывать оба файловых потока одновременно.

Следует обратить внимание на недостаточную точность часов для формирования уникального имени. Необходимо, сохранив timestamp в имени, добавить дополнительный постфикс, который будет гарантированно отличаться у файловых потоков.

2. Управление вводом данных внешним кодом

В рамках этой части работы нужно выполнить код в виде библиотеки с описанным в файле **async.h** внешним интерфейсом.

Инициатором обмена будет выступать внешний код. Вместо привычной точки входа в приложение **main()** будут три внешних функции **connect()**, **receive()** и **disconnect()**.

Порядок вызовов следующий:

- вызывается **connect()** с передачей размера блока команд, сохраняется значение возврата. Значение никак не интерпретируется вызывающим кодом и служит только в качестве контекста для функций **receive()** и **disconnect()**.
- вызывается **receive()** с передачей указателя на начало буфера, его размера, а также контекста. Вызов повторяемый – вызывающий код может использовать его для передачи нескольких команд подряд.
- вызывается **disconnect()** с передачей контекста. Вызов разрушает контекст полностью. С точки зрения логики обработки команд этот вызов считается завершением текущего блока команд.

Необходимо реализовать эти функции так, чтобы сохранить прежнюю функционально проекта.

Реализация должна допускать множественные вызовы **connect()**. Вызовы **receive()** с разными контекстами не должны мешать друг другу. Вызовы могут осуществляться из разных потоков, однако вызовы с одинаковым контекстом всегда выполняются из одного и того же потока.

Опционально реализовать возможность вызывать все функции из любых потоков.

Требования к реализации

Результатом работы должна стать библиотека, устанавливаемая по стандартному пути. Библиотека должна называться **libasync.so** и находиться в пакете **async**.

Для проверки работоспособности следует реализовать также исполняемый файл, который использует библиотеку с демонстрацией всех вызовов. Исполняемый файл также должен быть добавлен в пакет.

Проверка

Задание считается выполненным успешно, если после установки пакета, линковки с тестовым кодом (пример в **main.cpp**) и запуска с тестовыми данными вывод соответствует описанию **Задания 6**. Данные подаются порциями в разных контекстах в большом объёме без пауз.