Федеральное государственное автономное образовательное
учреждение высшего образования
Университет ИТМО

Факультет программной инженерии и компьютерной техники

Отчёт по лабораторной работе №3 по дисциплине «Низкоуровневое программирование»

Вариант: Protocol Buffers

Выполнил:
Деев Роман Александрович

Группа: **P33102**

Преподаватели:
Кореньков Ю. Д.

Санкт-Петербург 2023 г.

Задание:

Задание 3

На базе данного транспортного формата описать схему протокола обмена информацией и воспользоваться существующей библиотекой по выбору для реализации модуля, обеспечивающего его функционирование. Протокол должен включать представление информации о командах создания, выборки, модификации и удаления данных в соответствии с данной формой, и результатах их выполнения.

Используя созданные в результате выполнения заданий модули, разработать в виде консольного приложения две программы: клиентскую и серверную части. Серверная часть – получающая по сети запросы и операции описанного формата и последовательно выполняющая их над файлом данных с помощью модуля из первого задания. Имя фала данных для работы получать с аргументами командной строки, создавать новый в случае его отсутствия. Клиентская часть – в цикле получающая на стандартный ввод текст команд, извлекающая из него информацию о запрашиваемой операции с помощью модуля из второго задания и пересылающая её на сервер с помощью модуля для обмена информацией, получающая ответ и выводящая его в человеко-понятном виде в стандартный вывод.

Порядок выполнения:

1. Изучить выбранную библиотеку
    a. Библиотека должна обеспечивать сериализацию и десериализацию с валидацией в соответствии со схемой
    b. Предпочтителен выбор библиотек, поддерживающих кодогенерацию на основе схемы
    c. Библиотека может поддерживать передачу данных посредством TCP соединения
        - Иначе, использовать сетевые сокеты посредством API ОС
    d. Библиотека может обеспечивать диспетчеризацию удалённых вызовов
        - Иначе, реализовать диспетчеризацию вызовов на основе информации о виде команды
2. На основе существующей библиотеки реализовать модуль, обеспечивающий взаимодействие
    a. Описать схему протокола в поддерживаемом библиотекой формате
        - Описание должно включать информацию о командах, их аргументах и результатах
        - Схема может включать дополнительные сущности (например, для итератора)
    b. Подключить библиотеку к проекту и сформировать публичный интерфейс модуля с использованием встроенных или сгенерированных структур данных используемой библиотеки
        - Поддержать установление соединения, отправку команд и получение их результатов
        - Поддержать приём входящих соединений, приём команд и отправку их результатов
    c. Реализовать публичный интерфейс посредством библиотеки в соответствии с п1
3. Реализовать серверную часть в виде консольного приложения
    a. В качестве аргументов командной строки приложение принимает:
        - Адрес локальной конечной точки для прослушивания входящих соединений
        - Имя файла данных, который необходимо открыть, если он существует, иначе создать
    b. Работает с файлом данных посредством модуля из задания 1
    c. Принимает входящие соединения и взаимодействует с клиентами посредством модуля из п2
    d. Поступающая информация о запрашиваемых операциях преобразуется из структур данных модуля взаимодействия к структурам данных модуля управления данными и наоборот
4. Реализовать клиентскую часть в виде консольного приложения
    a. В качестве аргументов командной строки приложение принимает адрес конечной точки для подключения
    b. Подключается к серверу и взаимодействует с ним посредством модуля из п2
    c. Читает со стандартного ввода текст команд и анализирует их посредством модуля из задания 2
    d. Преобразует результат разбора команды к структурам данных модуля из п2, передаёт их для обработки на сервер, возвращаемые результаты выводит в стандартный поток вывода
5. Результаты тестирования представить в виде отчёта, в который включить:
    d. В части 3 привести пример сеанса работы разработанных программ
    e. В части 4 описать решение, реализованное в соответствии с пп.2-4
    f. В часть 5 включить составленную схему п.2а

Цель

Используя библиотеку для работы с выданным форматом сериализации данных реализовать:

- сериализацию структур из второй лабораторной

- отправку этих структур с клиента на сервер

- сервер должен десериализовать структуры и преобразовать их в публичный интерфейс реализованный в первой лабораторной

Аспекты реализации

Для реализации использована библиотека gPRC. Плюсом библиотеки является уже реализованный клиент-сервер для передачи структур, что избавило от необходимости использовать API системных сокетов. А также опыт работы с ней на Python.

Структуры выдаваемые парсером, указатели на другие структуры пришлось дополнительно дообработать: списки преобразовать в массивы, дерево условий выборки записать в массив, а ссылки на дочерние элементы сделать индексами в массиве. Также добавлено высвобождение памяти выделяемой парсером. В базе данных пришлось улучшить выборку по условиям для поддержки вложенных условий.

Результаты

Исходный код: https://github.com/deevroman/low-level-programming-labs

Код клиента/сервера лежит в папке client/server. Схема передаваемых данных в папке proto

Кодогенерация и сборка примера с помощью cmake.

Для сборки у себя:

Установить библиотеку gPRC https://grpc.io/docs/languages/cpp/quickstart/

```
git clone https://github.com/deevroman/low-level-programming-labs.git && cd
low-level-programming-labs
# сборка сервера
cmake -B server/build -DCMAKE_BUILD_TYPE=Release server
cmake --build sever/build --config Release
# сборка клиента
cmake -B client/build -DCMAKE_BUILD_TYPE=Release client
cmake --build client/build --config Release
# запуск сервера
./server/build/bin/*/*
# запуск клиента
./client/build/bin/*/*
```

```
При успешной сборке запущенные бинарники покажут сообщение о том, с какими
аргументаи их нужно запустить
```

Примеры сеанса работы:

```
>db.insert(0, "SCHEMA", {pole:"takoe", vtoroe_pole: True})
Ok: True
Inserted id: 4216
>db.insert(0, "SCHEMA", {pole:"takoe", vtoroe_pole: True})
Ok: False
Error: Multiple tree roots
>db.insert(4216, "SCHEMA", {pole:"takoe", vtoroe_pole: True})
Ok: True
Inserted id: 4344
>db.find("SCHEMA", {pole:"takoe", vtoroe_pole: True})
Ok: True
Count elements: 2
id: 4344
schema: SCHEMA
pole: takoe
vtoroe_pole: True

id: 4216
schema: SCHEMA
pole: takoe
vtoroe_pole: True

>db.update("SCHEMA", {vtoroe_pole: True}, {$set:{vtoroe_pole: False}})
Ok: True
>db.find("SCHEMA", {vtoroe_pole: False})
Ok: True
Count elements: 2
id: 4344
schema: SCHEMA
pole: takoe
vtoroe_pole: False

id: 4216
schema: SCHEMA
pole: takoe
vtoroe_pole: False

>db.delete("SCHEMA", {vtoroe_pole: False})
Ok: True
>db.find("SCHEMA", {pole:"takoe"})
Ok: True
Count elements: 0
>
```

Схема данных

```
syntax = "proto3";
package proto_query;

enum CommandType {
  CMD_INSERT = 0;
  CMD_FIND = 1;
  CMD_UPDATE = 2;
  CMD_DELETE = 3;
}

enum ValueType {
  DB_INT32 = 0;
  DB_DOUBLE = 1;
  DB_STRING = 2;
  DB_BOOL = 3;
};

enum Comparator {
  OP_LT = 0;
  OP_LTE = 1;
  OP_GT = 2;
  OP_GTE = 3;
  OP_NE = 4;
  OP_REGEX = 5;
};

enum ASTType {
  OP_AND = 0;
  OP_OR = 1;
  OP_KEY_VALUE = 2;
  OP_COMP = 3;
};

message field_key_value {
  string key = 1;
  value value = 2;
}

message filter {
  int64 left_node = 1;
  int64 right_node = 2;
  field_key_value key_value = 3;
  Comparator comp = 4;
  ASTType op = 5;
}

message value {
  ValueType value_type = 1;
  oneof data {
    int32  int_value = 2;
    double double_value = 3;
    bool bool_value = 4;
```

```
    string str_value = 5;
  };
}

message Query {
  CommandType command = 1;
  int64 parent = 2;
  string schema = 3;
  repeated filter cond = 4;
  repeated field_key_value new_fields = 5;
}

message Element {
  int64 id = 1;
  string schema = 2;
  repeated field_key_value key_values = 3;
}

message QueryResponse {
  bool ok = 1;
  string error_message = 2;
  repeated Element elements = 3;
}

service QueryService {
  rpc GetQuery(Query) returns (QueryResponse) {}
}
```

Выводы:

- Научился работать с gRPC в C++
- Лучше разобрался в cmake и научился подключать более сложные библиотеки
- ~~Возненавидел C++ за зоопарк конструкторов.~~