ИТМО

Лабораторная работа №3

по дисциплине: Низкоуровневое программирование

Вариант: 3 (Protocol Buffers)

Выполнил: Неграш Андрей, P33301

Преподаватель: Кореньков Юрий Дмитриевич

Санкт-Петербург, 2023

Задание

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 п.1
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 п.2
   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 п.2
   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:
   a. В части 3 привести пример сеанса работы разработанных программ
   b. В части 4 описать решение, реализованное в соответствии с пп.2-4
   c. D часть 5 включить составленную схему п.2а

Описание работы

Код лабораторной работы доступен по ссылке:
https://github.com/ANegrash/LLP_lab3

Программа лабораторной собрана из двух предыдущих работ, которые находятся в папках “server” и “gremlin” соответственно. Помимо переноса в один проект двух работ, был создан код для взаимодействия серверной и клиентской части с помощью Protocol Buffers. Собственно, именно это и было основной частью данной лабораторной работы.

Всё, что связано непосредственно с частью о Protocol Buffers, лежит в директории “protobuffer”.

Реализация

Внутри вышеупомянутой директории “protobuffer” есть вложенная директория “nanopb”. Там хранится библиотека, которая реализует работу Protocol Buffers на языке C.

Структура передаваемых данных лежит внутри специального файла message.proto, и она абсолютно идентична структуре из лабораторной работы 2, за исключением одного поля – Response. Данное поле содержит формат ответа сервера, в котором мы максимально просто возвращаем строку. Не важно, ошибка это, какое-то тело ответа или что-либо ещё.

Сама передача данных по сети организована при помощи сетевых сокетов API ОС. Просмотреть код работы можно в файлах server.c и client.c соответственно, там же находятся алгоритмы кодирования и декодирования.

Результаты

В качестве доказательства работоспособности программы прикладываю скриншоты, которые являются частью индивидуального задания, полученного 04.04.2024.

```
negrash@negrash-vb:~/llp_lab3$ ./LLP3_client 127.0.0.1 3939
client: connecting...
g.Entity.has(name,"Deutsche Bank AG",=).Country().get()
Found 9 node(s):
---- NODE 0 ----
id: 5059
country: ""
name: "Deutsche Bank AG"
srid: 4326
x: -74
y: 41
local_id: "deutsche-bank-ag"

Relations count: 1
Relations [ 1 ]
```

```
negrash@negrash-vb:~/llp_lab3$ ./LLP3_client 127.0.0.1 3939
client: connecting...
g.Entity.has(name,"Deutsche Bank AG",=).Country.has(ge,"S",=).get()
Found 4 node(s):
---- NODE 0 ----
id: 5059
country: ""
name: "Deutsche Bank AG"
srid: 4326
x: -74
y: 41
local_id: "deutsche-bank-ag"

Relations count: 1
Relations [ 1 ]
---- NODE 1 ----
id: 534
code: "GBR"
name: "United Kingdom"
srid: 4326
x: -2
y: 54
td: "GB"
```

```
negrash@negrash-vb:~/llp_lab3$ ./LLP3_client 127.0.0.1 3939
client: connecting...
g.Entity.has(name,"Deutsche Bank AG",=).Country.has(ge,"G",=).and().has(le,"I",=).get()
Found 4 node(s):
---- NODE 0 ----
id: 5059
country: ""
name: "Deutsche Bank AG"
srid: 4326
x: -74
y: 41
local_id: "deutsche-bank-ag"

Relations count: 1
Relations [ 1 ]
---- NODE 1 ----
id: 394
code: "HKG"
name: "Hong Kong"
srid: 4326
x: 114
y: 22
tld: "HK"
```

Вывод

Итак, во время выполнения данной лабораторной работы я ознакомился с тем, что такое Protocol Buffers (и, к слову, удивился очень хорошему и качественному решению, которое впоследствии планирую использовать в своих работах), но в данной задаче пришлось несколько переформатировать структуры для верного взаимодействия. А также поработал непосредственно с библиотекой nanopb, которая отвечает за реализацию Protocol Buffers на языке C.