Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет Программной инженерии и компьютерной техники

Отчёт к практическому заданию №3

по «Низкоуровнему программированию»

<u>Выполнил</u>: Группа P33312 Хайкин О.И.

<u>Преподаватель</u>:

Кореньков Ю.Д.

Санкт-Петербург, 2023

Цели

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.

Вариант:

Формат транспортного протокола: Apache Thrift

Задачи

Новые модули в репозитории

Первым делом были созданы 3 новых модуля: transfer, transfer_client и transfer_server.

transfer отвечает за формат протокола передачи - в этом модуле происходит генерация файлов из .trhift-файла.

transfer_client и transfer_server, соответственно, отвечают за клиентскую и серверную часть приложения

Подключение библиотек

Помимо самого Apache Thrift пришлось подключить к разрабатываемым модулям библиотеки glib и gobject. Необходимость вызвана тем, что Apache Thrift не умеет генерировать код для “чистого” C, вместо этого базируясь на Glib.

Имплементация клиента и сервера

Наконец, были имплементированы сами клиент и сервер.

Описание работы

Имплементация и клиента и сервера состоит из двух частей: модуля для подключения и модуля-маппера между структурами (классами) из thrift-формата и структурами из лабораторных № 1 и 2.

Аспекты реализации

Сервер

Сервер принимает 2 аргумента командной строки: путь к БД-файлу и порт. Для сервера используется предлагаемый trhift'ом сервер, использующий сокет.

Создание сервера:

```
handler = database_service_handler_new(database_manager);
processor =
   g_object_new(TYPE_DATABASE_SERVICE_PROCESSOR, "handler", handler, NULL);
server_transport =
   g_object_new(THRIFT_TYPE_SERVER_SOCKET, "port", port, NULL);
transport_factory =
   g_object_new(THRIFT_TYPE_BUFFERED_TRANSPORT_FACTORY, NULL);
protocol_factory = g_object_new(THRIFT_TYPE_BINARY_PROTOCOL_FACTORY, NULL);
server = g_object_new(
   THRIFT_TYPE_SIMPLE_SERVER, "processor", processor, "server_transport",
   server_transport, "input_transport_factory", transport_factory,
   "output_transport_factory", transport_factory, "input_protocol_factory",
   protocol_factory, "output_protocol_factory", protocol_factory, NULL);
```

Также настроено поведение корректного выключения сервера по сигналу SIGINT.

Обработка SIGINT:

```
static void sigint_handler(int signal_number) {
 THRIFT_UNUSED_VAR(signal_number);
 sigint_received = TRUE;

 if (server != NULL)
   thrift_server_stop(server);
}
```

Клиент

Клиент принимает 2 аргумент командной строки: хост и порт сервера. Клиент читает запросы пользователя через stdin, мапит их в формат для передачи и вызывает сервер. Клиент прлодолжает принятие запросов до получения ошибки связи с сервером или до остановки пользователем.

Основной цикл:

```
TRY(parse_stdin(&tree));
CATCH(err, {
 handle_error(err);
 continue;
})

ast_node_print(tree);

TRY(map_stmt(tree, &statement));
CATCH(err, {
 handle_error(err);
 continue;
})

g_result = g_object_new(TYPE_STATEMENT_RESULT, NULL);

if (!g_err && database_service_if_execute(client, &g_result, statement,
                                          &database_error, &g_err)) {
 result = map_result(g_result);
 printf("Successfully executed statement\n");
 print_result(result);
}
```

Результаты

Артефакты

В результате сборки программы создаются следующие артифакты:

- Файл transfer-библиотеки
- Исполняемый файл сервера, линкующийся с файлом библиотеки
- Исполняемый файл клиента, линкующийся с файлом библиотеки

Пример работы

```
> CREATE TABLE amogus (
      name varchar,
      imposter bool
);
>
Successfully executed statement

> INSERT INTO amogus (name, imposter)
```

```
VALUES ("aboba", TRUE), ("bogus", FALSE);
>
Successfully executed statement

> SELECT FROM amogus WHERE amogus.status = TRUE;
>
aboba       TRUE
Successfully executed statement
```

Выводы

В результате выполнения задания проект пополнился тремя модулями - библиотекой для трансфера и парой клиент-сервер, обеспечивающих полноценную работу с базой данных через ввод пользователем SQL-запросов.