Университет ИТМО

Факультет программной инженерии и компьютерной техники
Направление подготовки 09.03.01 Информатика и вычислительная техника

Лабораторная работа №3

по дисциплине «Низкоуровневое программирование»
Вариант №1 (XML)

Выполнил:
Студент группы P33302
Иванов Н.Д.

Преподаватель:
Кореньков Юрий Дмитриевич

г. Санкт-Петербург
2023

Содержание

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.

Задачи

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

Описание работы:

Были созданы клиентский и серверный модули

Для парсинга была использована библиотека libxml2.

Основной цикл клиентского модуля

```c
int main(int argc, char *argv[]) {
    if (argc != 3) {
        fprintf( stream: stderr,  format: "Использование: %s <server_ip> <server_port>\n", argv[0]);
        return 1;
    }

    const char *server_ip = argv[1];
    int server_port = atoi( nptr: argv[2]);

    int client_socket = create_client_socket(server_ip, server_port);

    char buffer[MAX_BUFFER_SIZE];

    int times = 5;
    while (times) {
        times--;
        printf( format: "Enter a message to send to the server: ");
        input();
        char * xml = to_xml();

        strncpy( dest: buffer,  src: xml,  n: sizeof(buffer));
        freeAstTree();
        send_data( socket_fd: client_socket,  data: buffer);
        receive_data( socket_fd: client_socket, buffer);
    }
    close_socket( socket_fd: client_socket);


    return 0;
}
```

Была написана небольшая обертка над сокетами со следующим интерфейсом:

```c
int create_client_socket(const char *server_ip, int server_port);
void send_data(int socket_fd, const char *data);
void receive_data(int socket_fd, char *buffer);
void close_socket(int socket_fd);

#endif // SOCKET_WRAPPER_H

```

В ней 1 метод, который требует внимания:

```c
void receive_data(int socket_fd, char *buffer) {
    memset( s: buffer,  c: 0,  n: MAX_BUFFER_SIZE);
    recv( fd: socket_fd,  buf: buffer,  n: MAX_BUFFER_SIZE,  flags: 0);
    char *message = parseResponse(buffer);
    while (trimmed_strcmp(message, "End") != 0) {
        printf( format: "%s\n", message);
        memset( s: buffer,  c: 0,  n: MAX_BUFFER_SIZE);
        recv( fd: socket_fd,  buf: buffer,  n: MAX_BUFFER_SIZE,  flags: 0);
        message = parseResponse(buffer);
    }
}
```

Клиент ожидает сообщения пока сервер не отправит сообщение с ключевым словом “End”.
Это сделано потому что данные от сервера отсылаются пачками на случай если их будет много после какого-нибудь условного select-а.

Серверный модуль

Для сервера тоже была создана небольшая обертка над сокетами со следующим интерфейсом:

```c

int create_server_socket();
void bind_socket(int socket_fd, int server_port);
void listen_for_connections(int socket_fd);
int accept_connection(int socket_fd);
void send_data(int socket_fd, const char *data);
void receive_data(int socket_fd, char *buffer);
void close_socket(int socket_fd);

```

Основной цикл серверный:

```c
int main(int argc, char *argv[]) {
    if (argc != 3) {
        fprintf( stream: stderr,  format: "Использование: %s <server_ip> <server_port>\n", argv[0]);
        return 1;
    }
    const char *filePath = argv[1];
    int server_port = atoi( nptr: argv[2]);
    FILE *file = fopen( filename: filePath,  modes: "rb+");


    int server_socket = create_server_socket();
    bind_socket( socket_fd: server_socket, server_port);
    listen_for_connections( socket_fd: server_socket);

    printf( format: "Server waiting for connections...\n");

    int client_socket = accept_connection( socket_fd: server_socket);
    printf( format: "Connection established with client.\n");

    int times = 10;
    while (times) {
        times--;
        char buffer[MAX_BUFFER_SIZE];
        receive_data( socket_fd: client_socket, buffer);

        printf( format: "Received from client: %s\n", buffer);

        char *xml = buffer;
        from_xml(xml, file, client_socket);
    }

    close_socket( socket_fd: client_socket);
    close_socket( socket_fd: server_socket);

    return 0;
}
```

вся логика находится в метод from_xml, там просходит весь парсинг и создание структру данных для взаимодействия с бд. Пример:

```c
void *handleDelete(xmlNodePtr root, FILE *file, int client_socket) {
    char *tableName;
    for (xmlNodePtr node = root->children; node; node = node->next) {
        if (xmlStrEqual( str1: node->name,  str2: BAD_CAST "tableName")) {
            tableName = (char *) xmlNodeGetContent( cur: node);
        } else if (xmlStrEqual( str1: node->name,  str2: BAD_CAST "filter")) {
            PredMass *predMass = goDepth( node: node->children[0].next);
            deleteRecordFromTable(file, tableName, predMass->predicate, predMass->predicateNumber);
        }
    }
    send_data( socket_fd: client_socket,  data: create_xml_document( message_content: "Your entity successfully deleted"));
    send_data( socket_fd: client_socket,  data: create_xml_document( message_content: "End"));
}
```

сначала мы пытаемся вытащить имя таблицы, затем пытаемся распарсить условия если они есть, создать по ним массив предикатов обходя рекурсивно дерево вглубь. И вызываем метод для удаления сущности из таблицы учитывая условия.

Ну и для примера предлагаю рассмотреть парсинг условий:

```
PredMass *mulCond(PredMass *predMass1, PredMass *predMass2) {
    int newSize = predMass1->predicateNumber + predMass2->predicateNumber;

    Predicate *mergedArray = (Predicate *) malloc( size: newSize * sizeof(Predicate));
    if (mergedArray == NULL) {
        return NULL;
    }

    for (int i = 0; i < predMass1->predicateNumber; ++i) {
        mergedArray[i] = predMass1->predicate[i];
    }

    for (int i = 0; i < predMass2->predicateNumber; ++i) {
        mergedArray[predMass1->predicateNumber + i] = predMass2->predicate[i];
    }
    PredMass *predMass = malloc( size: sizeof(predMass));
    predMass->predicate = mergedArray;
    predMass->predicateNumber = newSize;
    return predMass;
}

PredMass *goDepth(xmlNodePtr node) {
    if (xmlStrEqual( str1: node->name,  str2: BAD_CAST "AND") || xmlStrEqual( str1: node->name,  str2: BAD_CAST "OR")) {
        return mulCond( predMass1: goDepth( node: traverseChildren(node,  childNumber: 0)),  predMass2: goDepth( node: traverseChildren(node,  childNumber: 1)));
    }
    return parseCondition( condition: node);
}
```

У нас получается, что мы рекурсивно обходим наш xml документ, складывая условия в массив предикатов.

Схемы:
Была создана функция, которая принимает в себя строчное представление xml-файла и путь до файла со схемой, и возвращает то соответствует ли переданный xml схеме.

```
int validateXmlAgainstSchemaFile(const char *xmlString, const char
*schemaFilePath) {
    LIBXML_TEST_VERSION

    xmlDocPtr doc = xmlReadMemory(xmlString, strlen(xmlString),
"noname.xml", NULL, 0);
    if (doc == NULL) {
        fprintf(stderr, "Failed to parse the input XML.\n");
        return -1;
    }

    xmlSchemaParserCtxtPtr parserCtxt =
xmlSchemaNewParserCtxt(schemaFilePath);
    if (parserCtxt == NULL) {
        fprintf(stderr, "Failed to parse the input XML schema.\n");
        xmlFreeDoc(doc);
        return -2;
    }

    xmlSchemaPtr schema = xmlSchemaParse(parserCtxt);
    if (schema == NULL) {
        fprintf(stderr, "Failed to parse the input XML schema.\n");
        xmlSchemaFreeParserCtxt(parserCtxt);
        xmlFreeDoc(doc);
        return -2;
    }
```

```
    xmlSchemaValidCtxtPtr validCtxt =
xmlSchemaNewValidCtxt(schema);
    if (validCtxt == NULL) {
        fprintf(stderr, "Failed to create a validation
context.\n");
        xmlSchemaFree(schema);
        xmlSchemaFreeParserCtxt(parserCtxt);
        xmlFreeDoc(doc);
        return -3;
    }
    int isValid = xmlSchemaValidateDoc(validCtxt, doc);

    xmlSchemaFreeValidCtxt(validCtxt);
    xmlSchemaFree(schema);
    xmlSchemaFreeParserCtxt(parserCtxt);
    xmlFreeDoc(doc);
    xmlCleanupParser();

    return isValid; // Возвращаем 0, если XML соответствует схеме,
и отрицательное число в противном случае
}
```

Простейшая схема для Response-а:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="Root">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="Message" type="xs:string"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Для response-а я решил использовать простую схему, потому что я по факту передаю лишь 1 сообщение строчное, которое уже заранее в удобочитаемом виде само по себе.

Схема для request-а:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="Root">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="requestType">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="SELECT_QUERY"/>
                            <xs:enumeration value="INSERT_QUERY"/>
                            <xs:enumeration value="DELETE_QUERY"/>
                            <xs:enumeration value="UPDATE_QUERY"/>
```

```
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="tableName" type="xs:string"/>
            <xs:choice>
                <!-- If requestType is UPDATE -->
                <xs:sequence>
                    <xs:element name="updateField">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element
name="updateFieldName" type="xs:string"/>
                                <xs:element
name="updateFieldValue" type="xs:string"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="filter" minOccurs="0">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="filter"
maxOccurs="unbounded">
                                    <xs:complexType>
                                        <xs:sequence>
                                            <xs:element
name="leftOp">

<xs:complexType>

<xs:sequence>

<xs:element name="isColumnName" type="xs:boolean"/>

<xs:element name="value" minOccurs="0">

<xs:simpleType>

<xs:union

memberTypes="xs:string xs:int xs:double xs:boolean"/>

</xs:simpleType>

</xs:element>

</xs:sequence>

</xs:complexType>
```

```
                                                    </xs:element>
                                                    <xs:element
name="operator">
                                                        <xs:simpleType>

<xs:restriction base="xs:string">

<xs:enumeration value="=="/>

<xs:enumeration value="&lt;"/>

<xs:enumeration value="&gt;"/>

</xs:restriction>

</xs:simpleType>
                                                    </xs:element>
                                                    <xs:element
name="rightOp">

<xs:complexType>

<xs:sequence>

<xs:element name="isColumnName" type="xs:boolean"/>

<xs:element name="value" minOccurs="0">

<xs:simpleType>

<xs:union

memberTypes="xs:string xs:int xs:double xs:boolean"/>

</xs:simpleType>

</xs:element>

</xs:sequence>

</xs:complexType>
                                                    </xs:element>
                                                </xs:sequence>
                                            </xs:complexType>
                                        </xs:element>
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
```

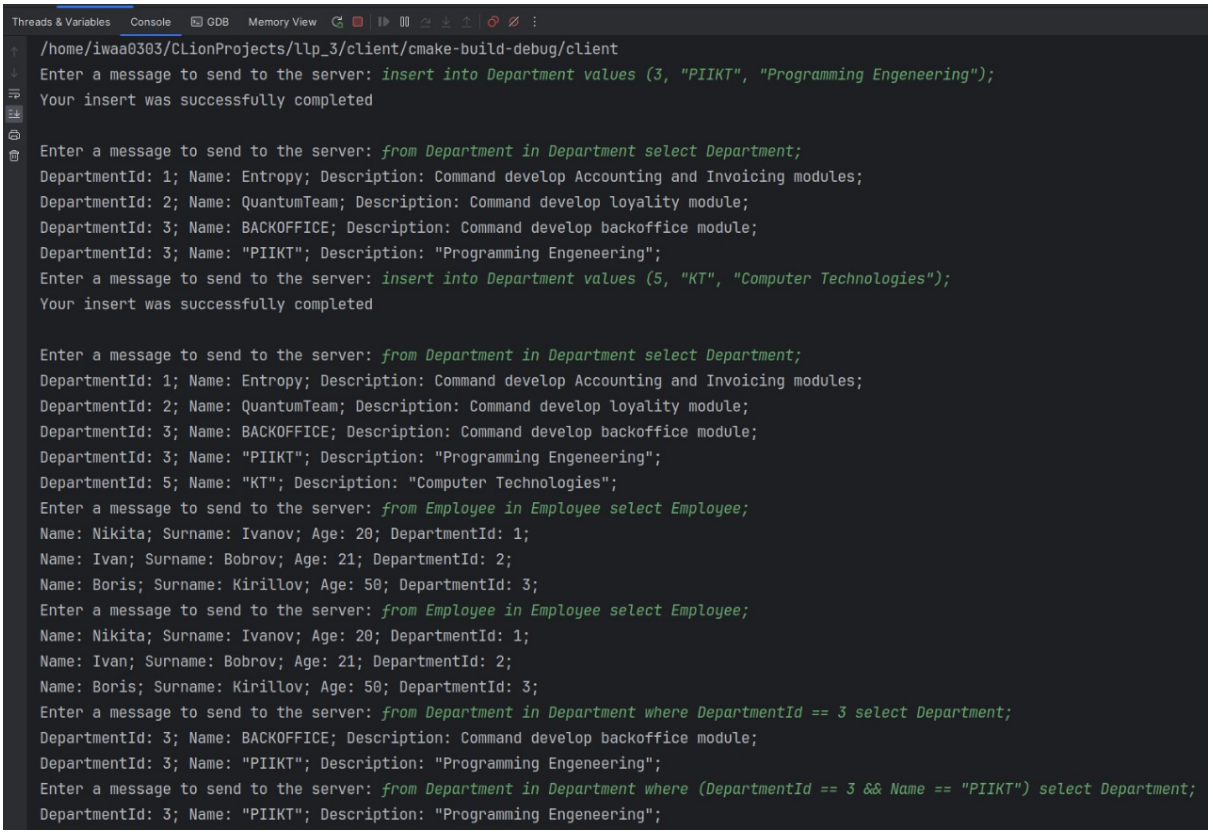
```
                    </xs:sequence>

                    <!-- If requestType is DELETE or SELECT -->
                    <xs:sequence>
                        <xs:element name="filter" minOccurs="0">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element name="filter"
maxOccurs="unbounded">
                                        <xs:complexType>
                                            <xs:sequence>
                                                <xs:element
name="leftOp">

<xs:complexType>

<xs:sequence>

<xs:element name="isColumnName" type="xs:boolean"/>

<xs:element name="value" minOccurs="0">

<xs:simpleType>

<xs:union

memberTypes="xs:string xs:int xs:double xs:boolean"/>

</xs:simpleType>

</xs:element>

</xs:sequence>

</xs:complexType>
                                                </xs:element>
                                                <xs:element
name="operator">
                                                    <xs:simpleType>

<xs:restriction base="xs:string">

<xs:enumeration value="=="/>

<xs:enumeration value="&lt;"/>

<xs:enumeration value="&gt;"/>
```

```
</xs:restriction>

</xs:simpleType>
                                                    </xs:element>
                                                    <xs:element
name="rightOp">

<xs:complexType>

<xs:sequence>

<xs:element name="isColumnName" type="xs:boolean"/>

<xs:element name="value" minOccurs="0">

<xs:simpleType>

<xs:union

memberTypes="xs:string xs:int xs:double xs:boolean"/>

</xs:simpleType>

</xs:element>

</xs:sequence>

</xs:complexType>
                                                        </xs:element>
                                                    </xs:sequence>
                                                </xs:complexType>
                                            </xs:element>
                                        </xs:sequence>
                                    </xs:complexType>
                                </xs:element>
                                <xs:element name="join" minOccurs="0">
                                    <xs:complexType>
                                        <xs:sequence>
                                            <xs:element name="leftOperand">
                                                <xs:complexType>
                                                    <xs:sequence>
                                                        <xs:element
name="leftTable" type="xs:string"/>
                                                        <xs:element
name="leftField" type="xs:string"/>
                                                    </xs:sequence>
                                                </xs:complexType>
```

```
                                    </xs:element>
                                    <xs:element
name="rightOperand">
                                        <xs:complexType>
                                            <xs:sequence>
                                                <xs:element
name="rightTable" type="xs:string"/>
                                                <xs:element
name="rightField" type="xs:string"/>
                                            </xs:sequence>
                                        </xs:complexType>
                                    </xs:element>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                        <xs:element name="selectedVal"
minOccurs="0">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element name="entity"
type="xs:string" minOccurs="0"/>
                                    <xs:element name="field"
type="xs:string" minOccurs="0"/>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                    <!-- If requestType is INSERT -->
                    <xs:element name="insertValues" minOccurs="0">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="field"
maxOccurs="unbounded">
                                    <xs:complexType>
                                        <xs:sequence>
                                            <xs:element name="type"
type="Literal"/>
                                            <xs:element
name="value" type="xs:string"/>
                                        </xs:sequence>
                                    </xs:complexType>
                                </xs:element>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:choice>
            </xs:sequence>
        </xs:complexType>
```

```
    </xs:element>
    <xs:simpleType name="Literal">
        <xs:restriction base="xs:string">
            <xs:enumeration value="STRING_LITERAL"/>
            <xs:enumeration value="INTEGER_LITERAL"/>
            <xs:enumeration value="DOUBLE_LITERAL"/>
            <xs:enumeration value="BOOL_LITERAL"/>
        </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

Пример клиентской сессии:



1) Добавили 1 сущность в таблицу Department
2) Посмотрели содержимое таблицы Department
3) Добавили еще сущность в таблицу Department
4) Посмотрели содержимое таблицы Department (убедились, что у нас все сохранилось)
5) Далее пару раз посмотрели содержимое таблицы Employee
6) Сделали выборку из таблицы Department с условиями (проверили работу conditions)

Выводы:

В ходе выполнения данной работы были соединены предыдущие работы воедино при помощи протокола xml. Это помогло во-первых осознать мотивацию того, что происходило, а именно понять как должна взаимодействовать система на разных уровнях и что от чего зависит, какие подводные камни могут встретиться из-за мелких недочетов на предыдущих шагах. В целом получился боле менее готовый продукт, который уже представляет собой нормальное работающее приложение.