Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №3

по «Низкоуровневое программирование»

Вариант XML

Выполнил:

Студент группы P33301

Акимов Роман Иванович

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.

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

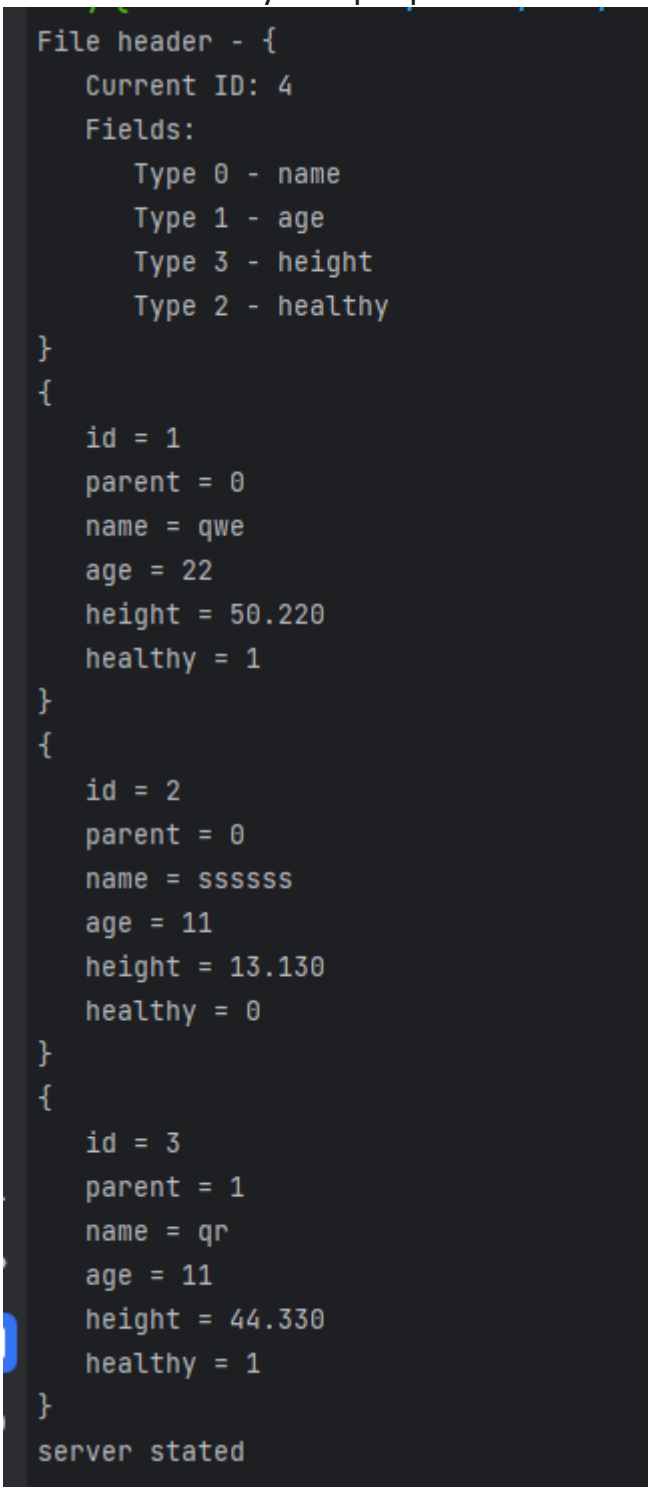
f. В часть 5 включить составленную схему п.2a

Ход работы

Клиент для передачи запроса по сети перед отправкой упаковывает его в формат XML. Библиотека libxml для упаковки. Сервер после получения запроса преобразует его в XML и на его основе выбирает нужную crud функцию, по итогам операции посылает ответ.

Пример работы программы

Запуск сервера



Добавление элемента

Клиент

```
enter your request
add/2/[name=qerere][age=22][height=14.33][healthy=1]
<?xml version="1.0"?>
<add><tuple id="2"><tuple name="qerere" operand_1="=" age="22" operand_2="=" height="14.33" operand_3="=" healthy="1" operand_4="="/></tuple></add>

added!

enter your request
```

Сервер

```
server stated
{
   id = 1
   parent = 0
   name = qwe
   age = 22
   height = 50.220
   healthy = 1
}
{
   id = 2
   parent = 0
   name = ssssss
   age = 11
   height = 13.130
   healthy = 0
}
{
   id = 3
   parent = 1
   name = qr
   age = 11
   height = 44.330
   healthy = 1
}
{
   id = 4
   parent = 2
   name = qerere
   age = 22
   height = 14.330
   healthy = 1
}
```

Поиск по id

Клиент

```
enter your request
find/2
<?xml version="1.0"?>
<find><tuple id="2"/></find>

{
        id - 2
        parent_id - 0
        name - ssssss
        age - 11
        height - 13.130
        healthy - 0
}

enter your request
```

Поиск по родителю
Клиент

```
enter your request
find/1/*
<?xml version="1.0"?>
<find><tuple id="1"><tuple id="*" operand_1=""/></tuple></find>

{
        id - 3
        parent_id - 1
        name - qr
        age - 11
        height - 44.330
        healthy - 1
}

enter your request
```

Найти все
Клиент

```
enter your request
find/*
<?xml version="1.0"?>
<find><tuple id="*"/></find>

{
        id - 1
        parent_id - 0
        name - qwe
        age - 22
        height - 50.220
        healthy - 1
}
{
        id - 2
        parent_id - 0
        name - ssssss
        age - 11
        height - 13.130
        healthy - 0
}
{
        id - 3
        parent_id - 1
        name - qr
        age - 11
        height - 44.330
        healthy - 1
}
{
        id - 4
        parent_id - 2
        name - qerere
        age - 22
        height - 14.330
        healthy - 1
}
```

Поиск по условию

Клиент

```
enter your request
find/*/[age=22]
<?xml version="1.0"?>
<find><tuple id="*"><tuple age="22" operand_1="="/></tuple></find>

{
        id - 4
        parent_id - 2
        name - qerere
        age - 22
        height - 14.330
        healthy - 1
}
{
        id - 1
        parent_id - 0
        name - qwe
        age - 22
        height - 50.220
        healthy - 1
}
```

Обновление по id

Клиент

```
enter your request
update/2/[age=33]
<?xml version="1.0"?>
<update><tuple id="2"><tuple age="33" operand_1="="/></tuple></update>

updated
```

Сервер

```
{
   id = 1
   parent = 0
   name = qwe
   age = 22
   height = 50.220
   healthy = 1
}
{
   id = 2
   parent = 0
   name = ssssss
   age = 33
   height = 13.130
   healthy = 0
}
{
   id = 3
   parent = 1
   name = qr
   age = 11
   height = 44.330
   healthy = 1
}
{
   id = 4
   parent = 2
   name = qerere
   age = 22
   height = 14.330
   healthy = 1
}
```

Удаление по id
Клиент

```
enter your request
remove/1
<?xml version="1.0"?>
<remove><tuple id="1"/></remove>

deleted!
```

Сервер

```
{
   id = 2
   parent = 0
   name = ssssss
   age = 33
   height = 13.130
   healthy = 0
}
{
   id = 4
   parent = 2
   name = qerere
   age = 22
   height = 14.330
   healthy = 1
}
```

Аспекты реализации

Упаковка запроса происходит в документ формата XML.

Пример упакованных запросов

```
enter your request
find/1/*
<?xml version="1.0"?>
<find><tuple id="1"><tuple id="*" operand_1=""/></tuple></find>
```

```
enter your request
add/1/[name=qwe][age=22][height=50.22][healthy=1]
<?xml version="1.0"?>
<add><tuple id="1"><tuple name="qwe" operand_1="=" age="22" operand_2="=" height="50.22" operand_3="=" healthy="1" operand_4="="/></tuple></add>
```

```
enter your request
update/2/[age=22]
<?xml version="1.0"?>
<update><tuple id="2"><tuple age="22" operand_1="="/></tuple></update>
```

```
enter your request
find/*
<?xml version="1.0"?>
<find><tuple id="*"/></find>
```

Передача по сети происходит через API ОС.

```c
int Socket(int domain, int type, int protocol) {
    int res = socket(domain, type, protocol);
    if (res == -1) {
        perror( s: "socket failed");
        exit( status: EXIT_FAILURE);
    }
    return res;
}

void Bind(int sock_fd, const struct sockaddr *addr, socklen_t addr_len) {
    int res = bind( fd: sock_fd, addr,  len: addr_len);
    if (res == -1) {
        perror( s: "bind failed");
        exit( status: EXIT_FAILURE);
    }
}

void Listen(int sock_fd, int back_log) {
    int res = listen( fd: sock_fd,  n: back_log);
    if (res == -1) {
        perror( s: "listen failed");
        exit( status: EXIT_FAILURE);
    }
}

int Accept(int sock_fd, struct sockaddr *addr, socklen_t *addr_len) {
    int res = accept( fd: sock_fd, addr, addr_len);
    if (res == -1) {
        perror( s: "accept failed");
        exit( status: EXIT_FAILURE);
    }
    return res;
}
```

```c
void Connect(int sock_fd, const struct sockaddr *addr, socklen_t addr_len) {
    int res = connect( fd: sock_fd, addr,  len: addr_len);
    if (res == -1) {
        perror( s: "connect failed");
        exit( status: EXIT_FAILURE);
    }
}

void Inet_pton(int af, const char *src, void *dst) {
    int res = inet_pton(af,  cp: src,  buf: dst);
    if (res == 0) {
        printf( format: "inet error\n");
        exit( status: EXIT_FAILURE);
    }
    if (res == -1) {
        perror( s: "inet failed");
        exit( status: EXIT_FAILURE);
    }
}
```

Вывод:

Изучил библиотеку libxml, а также разработал клиент-серверное приложение с передачей по сети через API ОС.