

Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

Отчет по лабораторной работе №2

По дисциплине «Базы данных»

«Сетевой сервер учета ошибок «Bug tracking»»

Работу выполнили студенты группы №43501/4

Шаляпин Н.С._____

Работу принял преподаватель

Мяснов А.В._____

Санкт-Петербург

2016

Глава 1

Задание

Разработать клиент-серверную систему регистрации, учета и управления ошибками в программных проектах. Система должна состоять из сервера, хранящего репозиторий ошибок и рабочих клиентских мест, позволяющих программистам и тестировщикам управлять ошибками, найденными в программных проектах.

1.1 Функциональные требования

Серверное приложение должно реализовывать следующие функции:

1. Прослушивание определенного порта
2. Обработка запросов на подключение по этому порту от клиентов
3. Поддержка одновременной работы нескольких клиентов через механизм нитей
4. Регистрация подключившегося клиента в качестве тестировщика или разработчика
5. Выдача тестировщику списка исправленных ошибок
6. Выдача тестировщику списка активных (неисправленных) ошибок
7. Прием от тестировщика новой ошибки с указанием проекта, идентификатора ошибки, текста ошибки
8. Прием от тестировщика команды о подтверждении или отклонении исправления ошибки
9. Выдача разработчику списка найденных ошибок: идентификаторов и текстов
10. Прием команды разработчика об исправлении ошибки
11. Обработка запроса на отключение клиента
12. Принудительное отключение клиента

Клиентское приложение должно реализовывать следующие функции:

1. Установление соединения с сервером
2. Посылка регистрационных данных клиента (как тестировщика или как разработчика)
3. Для тестировщика: получение списка активных ошибок
4. Для тестировщика: получение списка исправленных ошибок
5. Для тестировщика: добавление новой ошибки

6. Для тестировщика: посылка команды подтверждения или отклонения исправления активной ошибки
7. Для разработчика: получение списка активных ошибок
8. Для разработчика: посылка команды исправления активной ошибки
9. Разрыв соединения
10. Обработка ситуации отключения клиента сервером

Разработанное клиентское приложение должно предоставлять пользователю настройку IP-адреса или доменного имени удалённого сервера учета ошибок и номера порта, используемого сервером. Разработанное серверное приложение должно хранить списки тестировщиков и разработчиков, список текущих проектов и файлов в каждом проекте.

1.2 Нефункциональные требования

При подключении нового клиента должна производиться процедура аутентификации с помощью логина и пароля. Пользователь не получает доступа к данным пока аутентификация не будет успешной. Один и тот же пользователь может одновременно быть авторизован с нескольких клиентских приложений. Ошибка может иметь два статуса: активна и исправлена. Сразу после добавления ошибка имеет статус "активна". После исправления ошибки, разработчик меняет ее статус на "исправлена". В статусе "исправлена" ошибка ожидает подтверждения статуса "исправлена" от тестировщика. Если тестировщик подтверждает этот статус, ошибка удаляется. Если не подтверждает, ошибка снова имеет статус "активна". Серверное и клиентское приложения должны работать на разных ОС.

1.3 Ограничения

Серверное приложение не поддерживает одновременное подключение более 100 клиентов. Максимальная длина вводимых пользователем строковых данных — 255 символов. Вводимые пользователем данные не должны содержать символа «*», так как он используется в качестве разделяющего символа между аргументами команды.

Глава 2

Реализация для работы по протоколу TSP

2.1 Прикладной протокол

Формат команды, отправляемой клиентом серверу приведем на рисунке 2.1. По рисунку видно, что команда может иметь максимум 3 аргумента, максимальный размер каждого аргумента – 255 символов. В качестве разделяющего символа между аргументами используется символ «*», поэтому пользовательские данные не должны содержать этого символа. Если в команде используется только 1 аргумент, разделяющего символа в конце строки не требуется.

Код команды 3 символа	1 аргумент 255 символов	*	2 аргумент 255 символов	*	3 аргумент 255 символов	*
--------------------------	----------------------------	---	----------------------------	---	----------------------------	---

Рис. 2.1: Формат команды, отправляемой клиентом серверу

Команды, доступные клиентскому приложению приведены в таблице 2.1.

При подключении нового клиента происходит процесс аутентификации. Аутентификация реализована по протоколу CHAP. Sequence-диаграмма процесса аутентификации приведена на рисунке 2.2. Дайджест пароля считается по алгоритму md5.

Описание команд:
1. crt — создание проекта. Команда доступна только разработчикам.

Имя команды	Аргумент 1	Аргумент 2	Аргумент 3
crt	имя проекта	-	-
adf	имя прокета	имя файла	-
ade	имя прокта	имя файла	описание ошибки
prt	имя списка	-	-
fix	номер ошибки	-	-
cnf	номер ошибки	-	-
rej	номер ошибки	-	-

Таблица 2.1: Команды, доступные клиентскому приложению

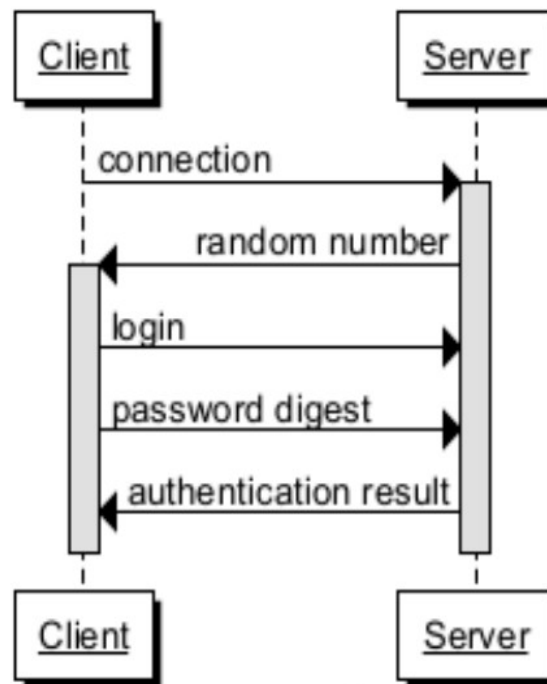


Рис. 2.2: Процесс аутентификации

Возможные ответы сервера:

- 0 – команда успешно выполнена;
- 1 – пользователь не имеет права выполнять данную команду;
- -1 – иная ошибка.

2. `adf` — добавление файла в указанный проект. Команда доступна только разработчикам.

Возможные ответы сервера:

- 0 – команда успешно выполнена;
- 1 – пользователь не имеет права выполнять данную команду;
- -1 – указанный проект не найден;
- -2 – иная ошибка.

3. `fix` — изменение статуса указанной ошибки на "исправлена". Команда доступна только разработчикам.

Возможные ответы сервера:

- 0 – команда успешно выполнена;
- 1 – пользователь не имеет права выполнять данную команду;
- -1 – ошибка с таким номером не найдена;
- -2 – иная ошибка.

4. `ade` — добавление ошибки в файл. Команда доступна только тестировщикам.

Возможные ответы сервера:

- 0 – команда успешно выполнена;
- 1 – пользователь не имеет права выполнять данную команду;
- -1 – указанный проект не найден;
- -2 – указанный файл не найден;
- -3 – иная ошибка.

5. `snf` — изменение статуса указанной ошибки на "исправление подтверждено". Команда доступна только тестирующим.

Возможные ответы сервера:

- 0 – команда успешно выполнена;
- 1 – пользователь не имеет права выполнять данную команду;
- -1 – ошибка с таким номером не найдена;
- -2 – иная ошибка.

6. `snf` — изменение статуса указанной ошибки на "исправление отвергнуто". Команда доступна только тестирующим.

Возможные ответы сервера:

- 0 – команда успешно выполнена;
- 1 – пользователь не имеет права выполнять данную команду;
- -1 – ошибка с таким номером не найдена;
- -2 – иная ошибка.

7. `prt` — получение указанного списка. Команда доступна всем пользователям. В ответ сервер присылает количество элементов в списке, а затем сам список, упакованный в 1 пакет. Различные элементы в списке разделены символом «*».

Пользователю на сервере доступны 2 команды:

1. `print` — вывод списка подключенных клиентов с их номерами;
2. `kill` номер клиента — отключение указанного клиента от сервера;

2.2 Архитектура приложения

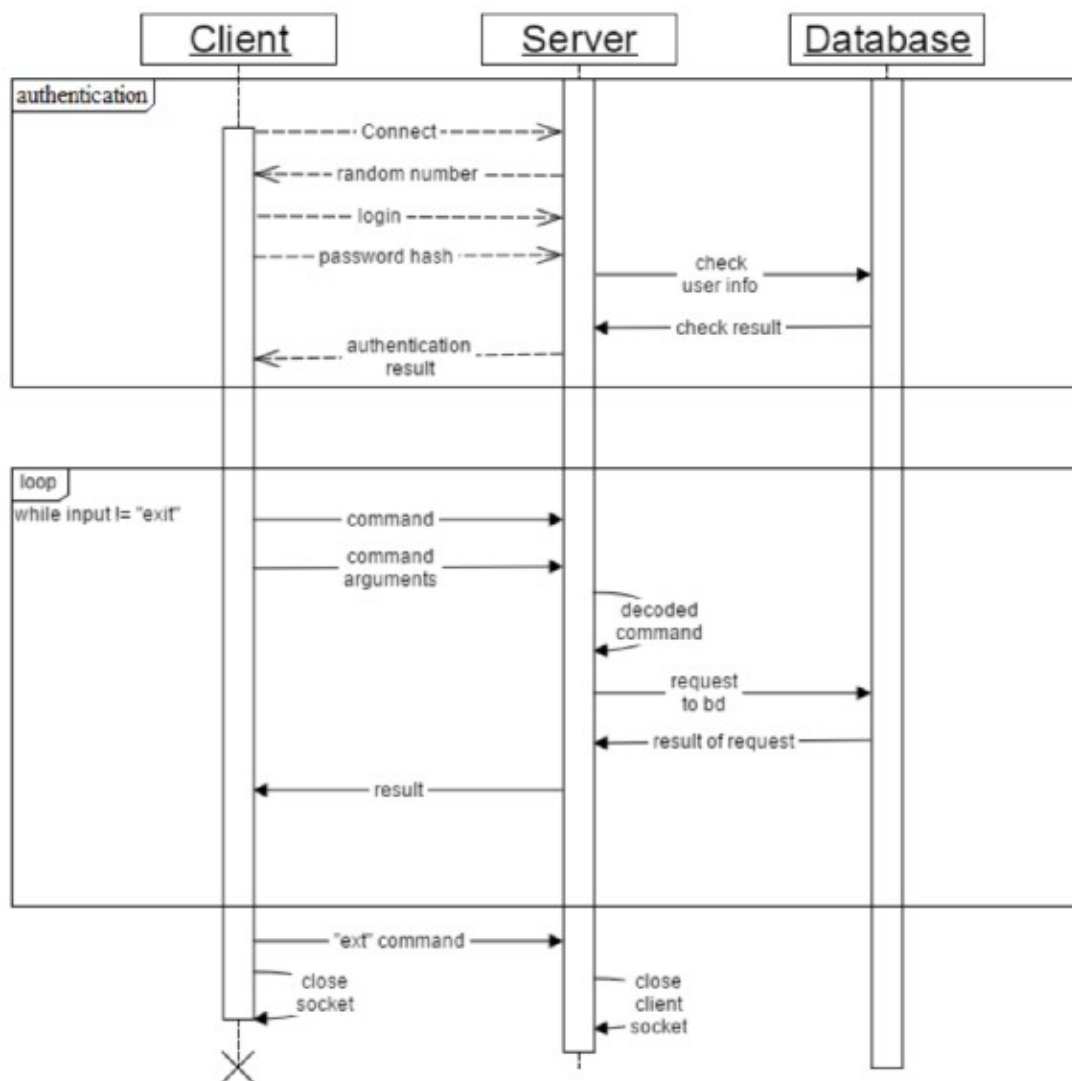


Диаграмма деятельности клиент-серверного приложения приведена на рисунке 2.3.

При подключении нового клиента сервер создает новый поток. Сначала происходит аутентификация подключившегося пользователя. Как говорилось ранее, аутентификация реализована по протоколу SHAP. После аутентификации начинается процесс работы. Клиент отправляет серверу различные команды, сервер выполняет эти команды и отправляет клиенту код результата выполнения операции. Объект "Ошибка" в процессе существования может иметь разные состояния. Диаграмма состояний ошибки приведена на рисунке 2.4.

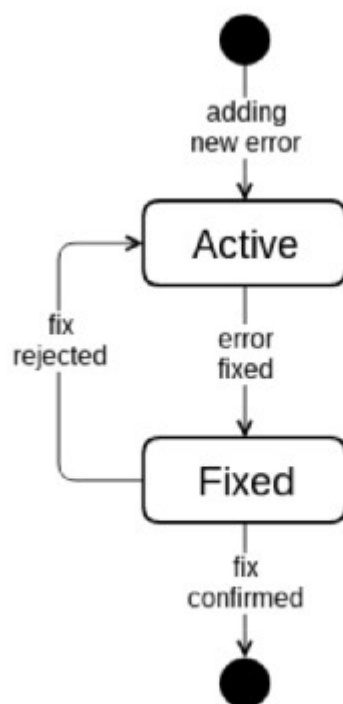


Рис. 2.4: Диаграмма состояний ошибки

Для процесса аутентификации клиент и сервер используют библиотеку OpenSSL версии 1.1.0-dev, в которой присутствует реализация алгоритма хэширования md5. Для хранения данных используется БД MySQL. Для подключения к БД и выполнения запросов к ней сервер использует библиотеку MySQL C Connector версии 6.1.6. ER-диаграмма созданной базы данных приведена на рисунке 2.5.

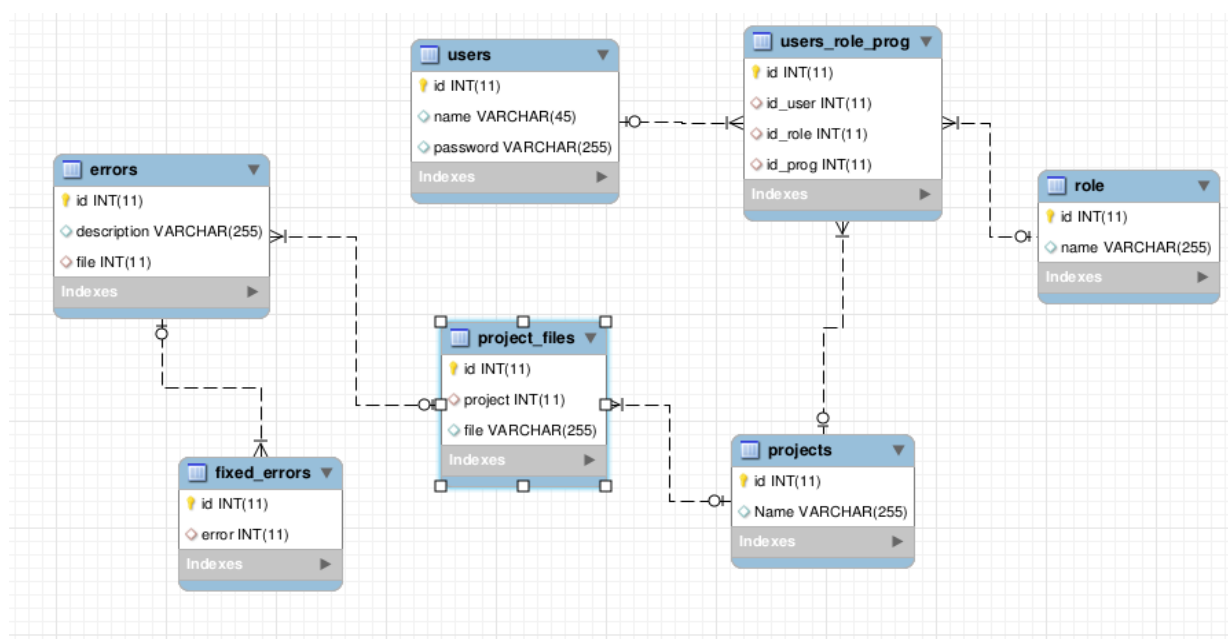


Рис. 2.5: ER-диаграмма базы данных

2.3 Тестирование

2.3.1 Описание тестового стенда и методики тестирования

Серверное приложение реализовано для ОС Linux. Тестирование проводилось на ОС Ubuntu 14.05. Сервер запускался на виртуальной машине с подключением к сети через сетевой мост.

Клиентское приложение реализовано в ОС Windows. Тестирование проводилось на ОС Windows XP.

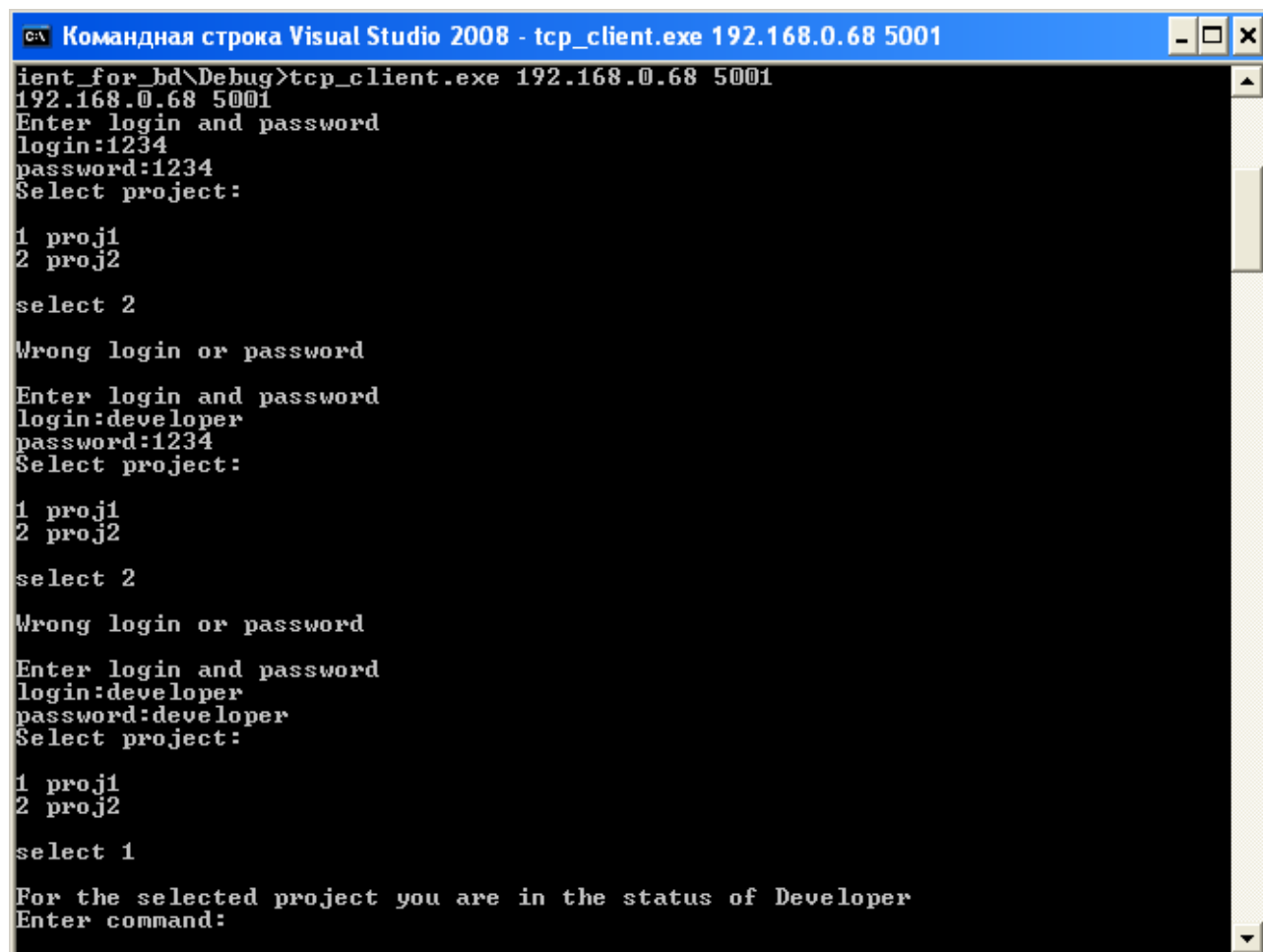
2.3.2 Тестовый план и результаты тестирования

Процесс тестирования:

1. Проверка аутентификации:

- Попытка подключиться с неправильным логином;
- Попытка подключиться с правильным логином и неправильным паролем;
- Попытка подключиться с правильным логином и паролем.

Результаты приведены на рисунке 2.6.



```
C:\> Командная строка Visual Studio 2008 - tcp_client.exe 192.168.0.68 5001
ient_for_bd\Debug>tcp_client.exe 192.168.0.68 5001
192.168.0.68 5001
Enter login and password
login:1234
password:1234
Select project:

1 proj1
2 proj2

select 2

Wrong login or password

Enter login and password
login:developer
password:1234
Select project:

1 proj1
2 proj2

select 2

Wrong login or password

Enter login and password
login:developer
password:developer
Select project:

1 proj1
2 proj2

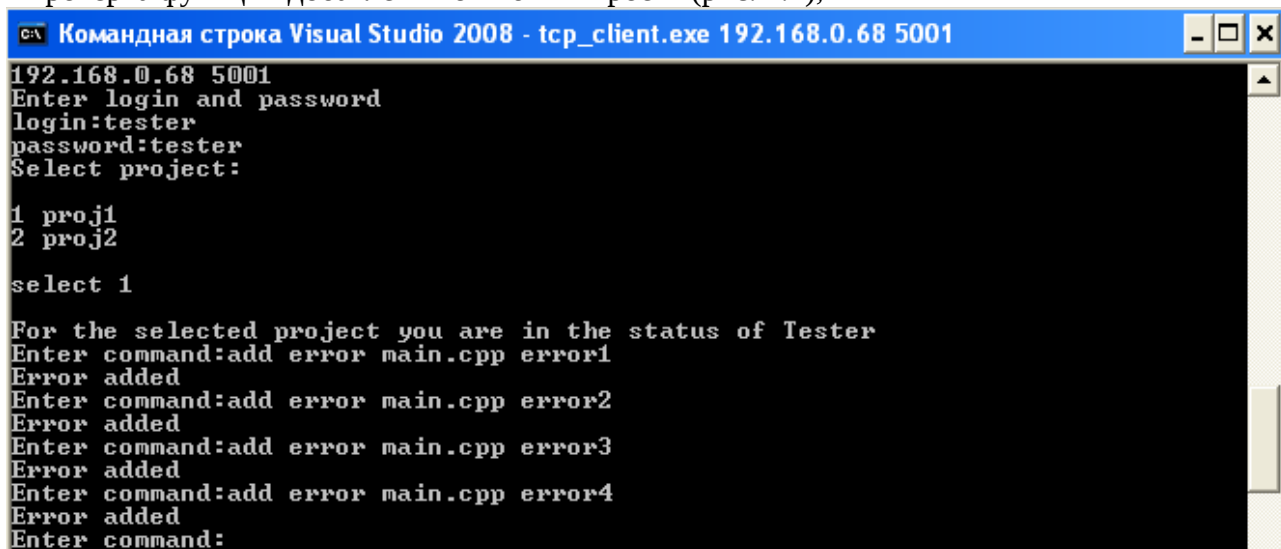
select 1

For the selected project you are in the status of Developer
Enter command:
```

Рис. 2.6: Проверка аутентификации

2. Проверка корректности выполнения всех команд тестировщика:

· Проверка функции добавления ошибки в проект (рис. 2.7);



```
C:\> Командная строка Visual Studio 2008 - tcp_client.exe 192.168.0.68 5001
192.168.0.68 5001
Enter login and password
login:tester
password:tester
Select project:

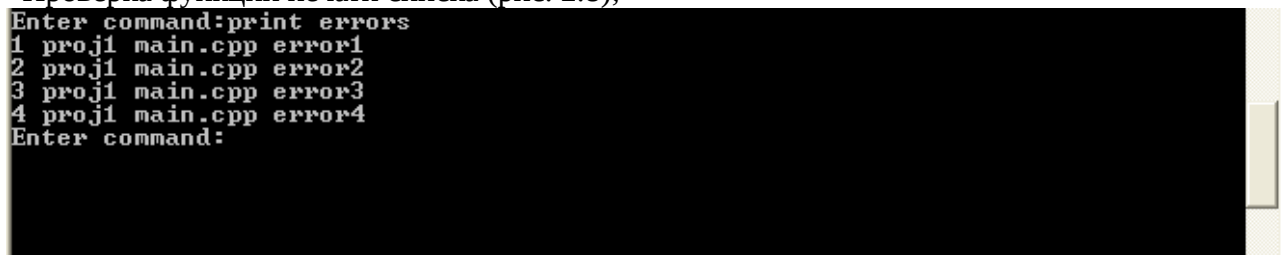
1 proj1
2 proj2

select 1

For the selected project you are in the status of Tester
Enter command:add error main.cpp error1
Error added
Enter command:add error main.cpp error2
Error added
Enter command:add error main.cpp error3
Error added
Enter command:add error main.cpp error4
Error added
Enter command:
```

Рис. 2.7: Проверка добавления ошибки

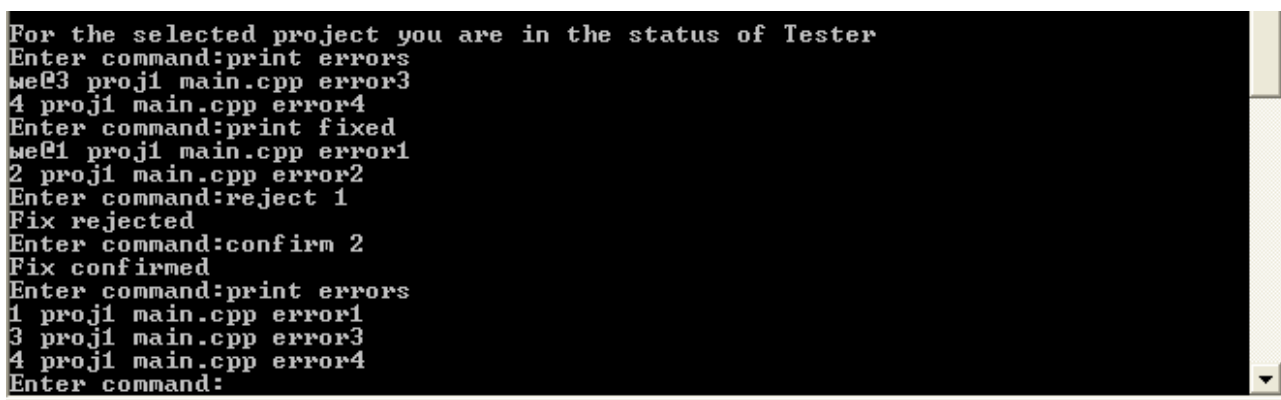
· Проверка функции печати списка (рис. 2.8);



```
Enter command:print errors
1 proj1 main.cpp error1
2 proj1 main.cpp error2
3 proj1 main.cpp error3
4 proj1 main.cpp error4
Enter command:
```

Рис. 2.8: Проверка вывода списка ошибок

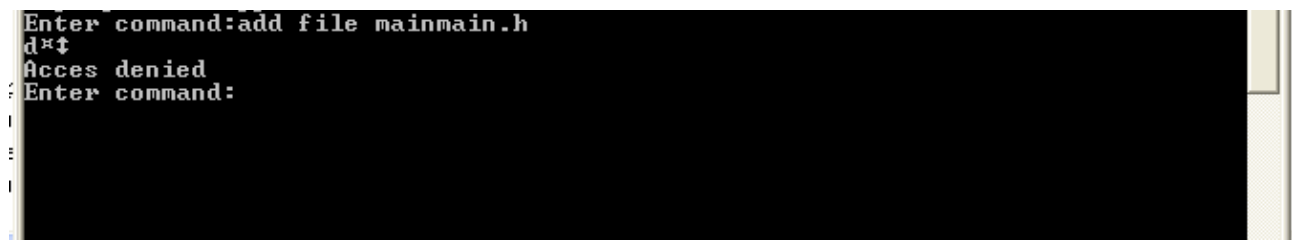
· Проверка функций изменения статуса ошибки (рис. 2.9);



```
For the selected project you are in the status of Tester
Enter command:print errors
3 proj1 main.cpp error3
4 proj1 main.cpp error4
Enter command:print fixed
1 proj1 main.cpp error1
2 proj1 main.cpp error2
Enter command:reject 1
Fix rejected
Enter command:confirm 2
Fix confirmed
Enter command:print errors
1 proj1 main.cpp error1
3 proj1 main.cpp error3
4 proj1 main.cpp error4
Enter command:
```

Рис. 2.9: Проверка изменения статуса ошибок

- Попытка выполнить команды разработчика (рис. 2.13);

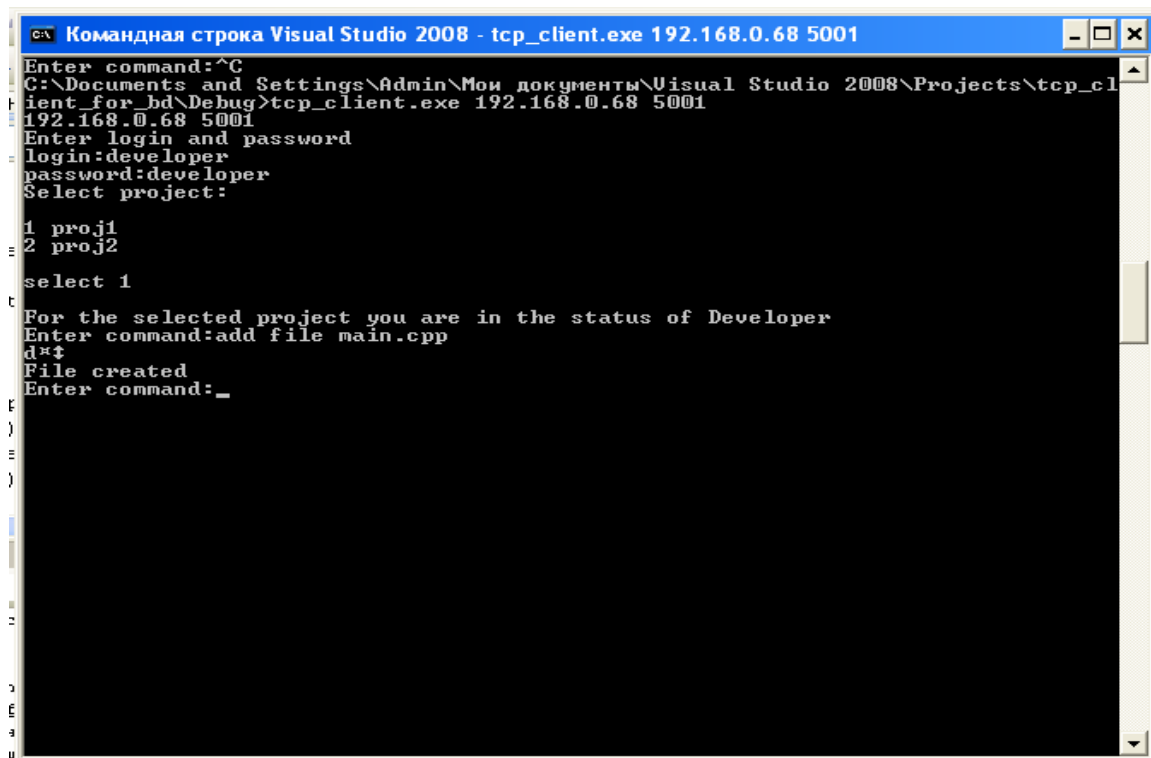


```
Enter command:add file mainmain.h
d*↑
Access denied
Enter command:
```

Рис. 2.10: Попытка выполнить команды разработчика

3. Проверка корректности выполнения команд разработчика:

- Проверка функций добавления файла (рис. 2.11);



```
Командная строка Visual Studio 2008 - tcp_client.exe 192.168.0.68 5001
Enter command:^C
C:\Documents and Settings\Admin\Мои документы\Visual Studio 2008\Projects\tcp_client_for_bd\Debug>tcp_client.exe 192.168.0.68 5001
192.168.0.68 5001
Enter login and password
login:developer
password:developer
Select project:
1 proj1
2 proj2
select 1
For the selected project you are in the status of Developer
Enter command:add file main.cpp
d*↑
File created
Enter command:_
```

Рис. 2.11: Проверка добавления файла

- Проверка функций печати, исправления ошибок (рис. 2.12);

```
192.168.0.68 5001
Enter login and password
login:developer
password:developer
Select project:

1 proj1
2 proj2

select 1

For the selected project you are in the status of Developer
Enter command:print errors
1 proj1 main.cpp error1
2 proj1 main.cpp error2
3 proj1 main.cpp error3
4 proj1 main.cpp error4
Enter command:
```

Рис. 2.12: Проверка печати

- Проверка функции исправления ошибок (рис. 2.12);

```
For the selected project you are in the status of Developer
Enter command:print errors
1 proj1 main.cpp error1
2 proj1 main.cpp error2
3 proj1 main.cpp error3
4 proj1 main.cpp error4
Enter command:add error main.cpp error33
Acces denied
Enter command:fix 1
Error fixed
Enter command:print errors
2 proj1 main.cpp error2
3 proj1 main.cpp error3
4 proj1 main.cpp error4
Enter command:fix 2
Error fixed
Enter command:
```

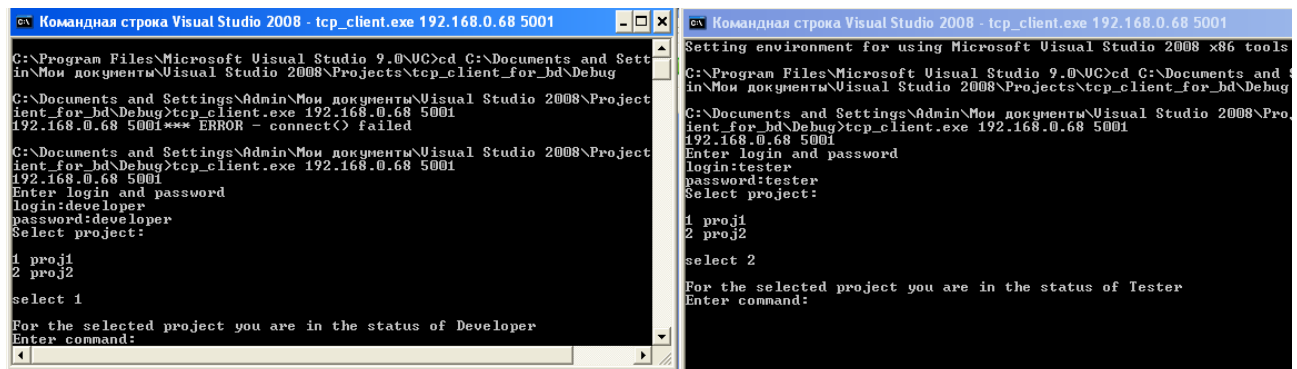
Рис. 2.12: Проверка исправления ошибок

- Попытка выполнить команды тестировщика (рис. 2.13);

```
Enter command:add error main.cpp error33
Acces denied
Enter command:
```

Рис. 2.13: Попытка выполнить команды тестировщика

4. Проверка корректной работы нескольких клиентов (рис. 2.14);

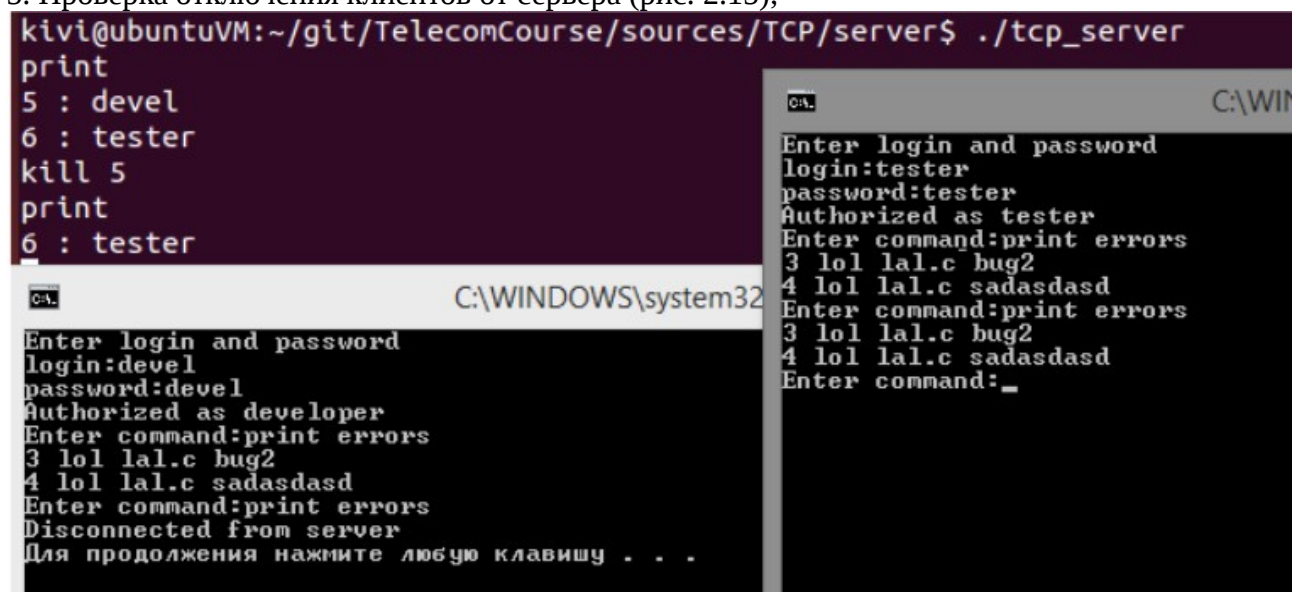


```
Командная строка Visual Studio 2008 - tcp_client.exe 192.168.0.68 5001
C:\Program Files\Microsoft Visual Studio 9.0\VC>cd C:\Documents and Settings\Admin\Мои документы\Visual Studio 2008\Projects\tcp_client_for_bd\Debug
C:\Documents and Settings\Admin\Мои документы\Visual Studio 2008\Projects\tcp_client_for_bd\Debug>tcp_client.exe 192.168.0.68 5001
192.168.0.68 5001*** ERROR - connect() failed
C:\Documents and Settings\Admin\Мои документы\Visual Studio 2008\Projects\tcp_client_for_bd\Debug>tcp_client.exe 192.168.0.68 5001
192.168.0.68 5001
Enter login and password
login:developer
password:developer
Select project:
1 proj1
2 proj2
select 1
For the selected project you are in the status of Developer
Enter command:

Командная строка Visual Studio 2008 - tcp_client.exe 192.168.0.68 5001
Setting environment for using Microsoft Visual Studio 2008 x86 tools
C:\Program Files\Microsoft Visual Studio 9.0\VC>cd C:\Documents and Settings\Admin\Мои документы\Visual Studio 2008\Projects\tcp_client_for_bd\Debug
C:\Documents and Settings\Admin\Мои документы\Visual Studio 2008\Projects\tcp_client_for_bd\Debug>tcp_client.exe 192.168.0.68 5001
192.168.0.68 5001
Enter login and password
login:tester
password:tester
Select project:
1 proj1
2 proj2
select 2
For the selected project you are in the status of Tester
Enter command:
```

Рис. 2.14: Проверка корректной работы нескольких клиентов

5. Проверка отключения клиентов от сервера (рис. 2.15);



```
kivi@ubuntuVM:~/git/TelecomCourse/sources/TCP/server$ ./tcp_server
print
5 : devel
6 : tester
kill 5
print
6 : tester

C:\WINDOWS\system32
Enter login and password
login:devel
password:devel
Authorized as developer
Enter command:print errors
3 lol lal.c bug2
4 lol lal.c sadasdasd
Enter command:print errors
3 lol lal.c bug2
4 lol lal.c sadasdasd
Enter command:_

C:\WINDOWS\system32
Enter login and password
login:tester
password:tester
Authorized as tester
Enter command:print errors
3 lol lal.c bug2
4 lol lal.c sadasdasd
Enter command:print errors
3 lol lal.c bug2
4 lol lal.c sadasdasd
Enter command:_
```

Рис. 2.15: Проверка отключения клиентов от сервера

Тестирование было успешным, все тесты прошли проверку. Тестирование показало, что приложение реализовывает функциональность, описанную в требованиях.

Глава 3

Выводы

В результате работы был создан прикладной протокол взаимодействия клиент-серверного приложения. В соответствии с прикладным протоколом была создана реализация приложения для протокола TCP. Клиентское и серверное приложения были реализованы для двух разных платформ: ОС Windows и Linux. При реализации использовались стандартные сокеты. Реализации сокетов для использованных ОС идентичны, портирование программ с одной платформы на другую выполняется достаточно просто. В результате работы была создана клиент-серверная система регистрации, учета и управления ошибками в программных проектах. Система состоит из сервера, хранящего репозиторий ошибок, и рабочих клиентских мест, позволяющих программистам и тестировщикам управлять ошибками. Для хранения репозитория ошибок использовалась СУБД MySQL. Все поставленные требования были выполнены. Тестирование программ прошло успешно, тесты показали корректность работы приложения.

3.1 Реализация для TCP

Протокол TCP удобен для реализации пользовательских приложений, так как обеспечивает установление соединения и надежную доставку пакетов. Протокол обеспечивает стабильное надежное соединение, поэтому при реализации своего протокола не требуется волноваться об этом. Однако, эти дополнительные средства синхронизации требуют больше времени на доставку, т.е. скорость передачи данных ниже чем в UDP.

Приложения

Описание среды разработки

Серверное приложение реализовывалось в ОС Ubuntu версии 14.05. Среда разработки — CLion версии 1.2.1. Компилятор GCC 4.9.2. Использовалась среда автоматизации сборки ПО CMake версии 3.2.2. Для компиляции серверного приложения так же требуются библиотеки OpenSSL версии 1.1.0-dev и MySQL C Connector версии 6.1.6. Клиентское приложения реализовывалось в ОС Windows XP. Среда разработки — Microsoft Visual Studio 2008.