

ГУАП
КАФЕДРА №51

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ
ПРЕПОДАВАТЕЛЬ

должность, уч. степень, звание

подпись, дата

инициалы, фамилия

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №11

ЧАТ ДЛЯ ДВУХ ПОЛЬЗОВАТЕЛЕЙ

по курсу: ОСНОВЫ ПРОГРАММИРОВАНИЯ II

РАБОТУ ВЫПОЛНИЛ
СТУДЕНТ ГР. № 5511

подпись, дата

инициалы, фамилия

Санкт-Петербург, 2017

1 Задание

Написать текстовый чат для двух пользователей на сокетах. Чат должен быть реализован по принципу клиент-сервер. Один пользователь находится на сервере, второй — на клиенте. Адреса и порты задаются через командную строку: клиенту — куда соединяться, серверу — на каком порту слушать. При старте программы выводится текстовое приглашение, в котором можно ввести одну из следующих команд:

- задать имя пользователя (@name Vasya)
- послать текстовое сообщение (Hello)
- выход (@quit)

Принятые сообщения автоматически выводятся на экран. Программа работает по протоколу UDP.

2 Дополнительное задание

Добавить команды @cd, @pwd, @ls для просмотра файловой системы собеседника. В ответ на эти команды необходимо выполнить действие с файловой системой и вернуть назад результат.

3 Реализация

Программа реализована двумя частями — клиент и сервер.

3.1 Клиент

За работу с клиентом отвечает класс `FileSystemClient`. В конструкторе получает адрес и порт для создания подключения. Затем в двух потоках запускаются методы `client::sendMessage` и `client::recieveMessage`, которые представляют собой бесконечные циклы, которые ожидают ввода пользователя или сообщения с сервера.

- `client::sendMessage` — отправляет сообщение, полученное из `System.in`, если оно не было командой @quit и @name, на сервер через `DatagramSocket`.
- `client::recieveMessage` — принимает сообщения от сервера, а затем выводит их в терминал.

3.2 Сервер

Сервер имплементирован в классе `Server`. У него, так же как у клиента, есть два метода `server::listener` и `server::respond`, которые запущены в отдельных потоках.

- `server::respond` — отправляет сообщение, полученное из `System.in`, так как сервер одновременно является и клиентом.
- `client::recieveMessage` — принимает сообщения от клиента, затем проверяет, являются ли они командами, если нет то выводит само сообщение в терминал.

Команды @cd, @ls и @pwd проверяются в соответствующих методах `server::checkCd`, `server::checkLs` и `server::checkPwd`. В них сообщение сравнивается с необходимым с помощью метода `String::compareTo("command")`, если они равны, то создаётся новый процесс (`Process`), равный выполнению данной команды в среде выполнения программы, с помощью `Runtime::exec("command")`. Результат выполнения данной команды берется из `InputStream` данного `Process`. А затем сервер отвечает клиенту с помощью метода `server::respondWithMessage("message")`, который отсылает необходимое сообщение через `DatagramSocket`.

4 Инструкция

4.1 Запуск сервера

Сервер запускается на порту указанном в 0 аргументе командной строки. Пример запуска — `java Server.java 5775`.

4.2 Запуск клиента

При запуске клиента необходимо ввести данные адреса и порта сервера после того, как вас об этом попросят. Если все произошло успешно вы можете использовать все доступные команды.

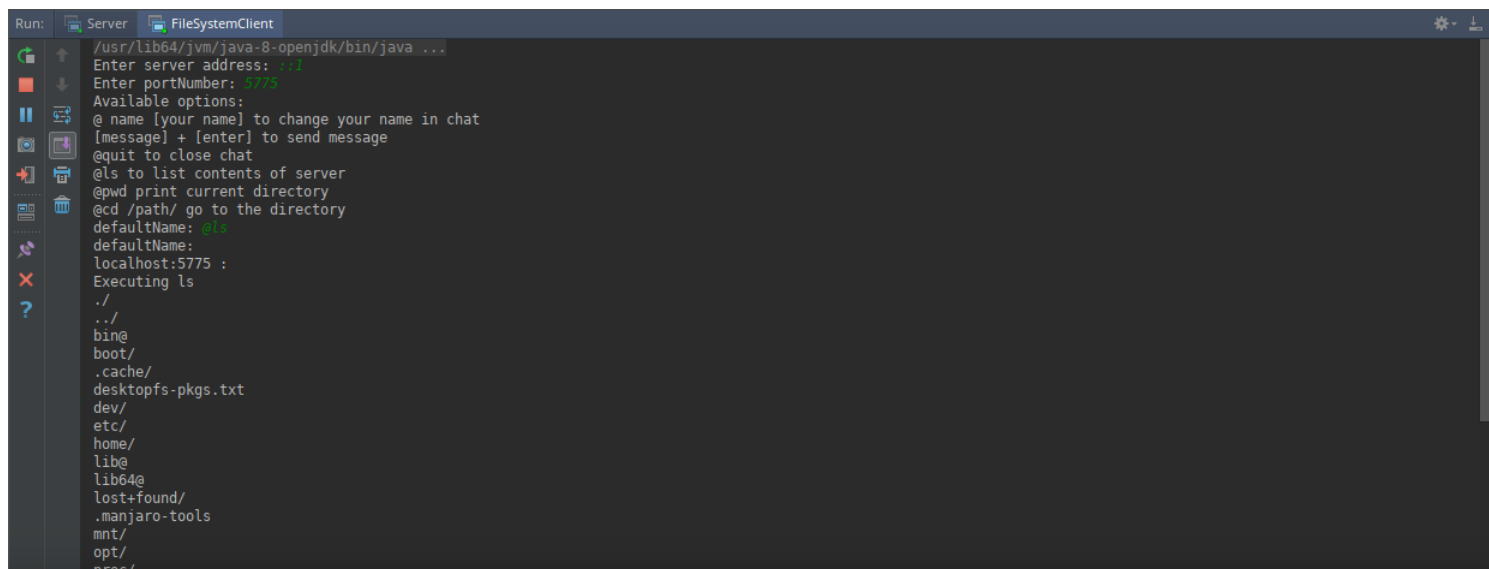
Можно настроить имя, которое будет отображаться у собеседника, используя команду @name. Пример использования — @name User2.

Для выхода из программы нужно ввести команду @end.

Для просмотра файловой системы собеседника необходимо ввести команды @ls, @cd, @pwd.

5 Тестирование

5.1 Пример работы программы



```
Run: Server FileSystemClient
/usr/lib64/jvm/java-8-openjdk/bin/java ...
Enter server address: localhost
Enter portNumber: 5775
Available options:
@ name [your name] to change your name in chat
[message] + [enter] to send message
@quit to close chat
@ls to list contents of server
@pwd print current directory
@cd /path/ go to the directory
defaultName: lib64@
localhost:5775 :
Executing ls
./
../
bin@
boot/
.cache/
desktopfs-pkgs.txt
dev/
etc/
home/
lib@
lib64@
lost+found/
.manjaro-tools
mnt/
opt/
proc/
```

Рис. 1: Пример подключения к серверу и просмотра его файловой через команду @ls