БГУИР

Кафедра ЭВМ

Операционные системы и системное программирование
Отчет по лабораторной работе № 8
Тема: «Сокеты. Взаимодействие процессов»

Выполнил:
студент группы 230501 Кочеров Р.С.

Проверил:
Поденок Л.П.

Минск
2024

1 УСЛОВИЕ ЛАБАРАТОРНОЙ РАБОТЫ

Задача разработка многопоточного сервера и клиента, работающих по простому протоколу.

Изучаемые системные вызовы: socket(), bind(), listen(), conect(), accept() и прочих, связанных с адресацией в домене AF_INET.

Протокол должен содержать следующие запросы:

ECHO − эхо-запрос, возвращающий без изменений полученное от клиента;

QUIT − запрос на завершение сеанса;

INFO − запрос на получения общей информации о сервере;

CD − изменить текущий каталог на сервере;

LIST − вернуть список файловых объектов из текущего каталога.

Протокол может содержать дополнительные запросы по выбору студента, не выходящие за пределы корневого каталога сервера и не изменяющих файловую систему в его дереве.

Запросы клиенту отправляются на stdin.

Ответы сервера и ошибки протокола выводятся на stdout.

Ошибки системы выводятся на stderr.

Подсказка клиента для ввода запросов символ− '>'.

Клиент помимо интерактивных запросов принимает запросы из файла. Файл с запросами указывается с использованием префикса '@':

```
$ myclient server.domen
Вас приветсвует учебный сервер 'myserver'
> @file
> ECHO какой-то_текст
какой-то_текст
> LIST
dir1
dir2
file
> CD dir1
dir1> QUIT
BYE
$
```

ECHO − эхо-запрос, возвращающий без изменений полученное от клиента.

```
> ECHO "произвольный текст"
произвольный текст
>
```

QUIT − запрос на завершение сеанса

```
> QUIT
BYE
$
```

INFO − запрос на получения общей информации о сервере.

Сервер отправляет текстовый файл с соответствующей информацией.

```
> INFO
Вас приветсвует учебный сервер 'myserver'
>
```

Этот же файл сервер отправляет клиенту при установлении сеанса.

LIST − вернуть список файловых объектов из текущего каталога.

Текущий каталог каталог в дереве каталогов сервера. Корневой каталог сервера устанавливается из командной строки при старте сервера.

```
> LIST
dir1/
dir2/
file1
file2 --> dir2/file2
file3 -->> dir1/file
>
```

Каталоги выводятся с суффиксом '/' после имени, файлы как есть, симлинки на регулярные файлы разрешаются через '-->', симлинки на симлинки разрешаются через '-->>'. Корневой каталог сервера пр выводе указывается префиксом '/' перед именем.

CD − изменить текущий каталог на сервере

Выход за пределы дерева корневого каталога сервера запрещается, команда безмолвно игнорируется

```
> CD dir2
dir2> LIST
file2
dir2> CD ../dir1
dir1> LIST
file --> /file1
dir1> CD ..
> CD ..
```

>

Соединения функционируют независимо, т.е. текущий каталог у каждого соединения свой.

Примечания:

Раскрашивать вывод не нужно.

Для разработки и отладки лабораторной следует использовать редактор или IDE, поддерживающие несколько запущенных экземпляров, каждый со своей конфигурацией, и поддерживающие отладку прямо в окне с кодом, например, slickedit (лучший выбор).

2 ОПИСАНИЕ АЛГОРИТМОВ И РЕШЕНИЙ

Алгоритмы клиента:

- Подключение к серверу: Клиент создает сокет, соединяет его с сервером по указанному адресу и порту;

- Отправка запросов: Клиент запрашивает команду у пользователя, отправляет ее серверу и получает ответ;

- Получение ответов: Клиент получает ответ от сервера и выводит его на консоль.

Алгоритмы сервера:

1) Обработка запросов клиентов: Сервер использует неблокирующий сокет для приема подключений от клиентов. Когда клиент подключается, сервер создает новый поток для обработки его запросов. Поток клиента читает сообщение от клиента, парсит его и выполняет соответствующую команду;

2) Парсинг сообщений: Сервер парсит сообщения от клиентов, используя префиксы команд;

3) Обработка команд: Сервер поддерживает различные команды, такие как "ECHO", "QUIT", "INFO", "CD", "LIST" и "@". Команда "INFO" возвращает информацию о сервере, команда "CD" изменяет текущий рабочий каталог, команда "LIST" возвращает список файлов и каталогов в указанном каталоге, а команды, начинающиеся с "@", используются для обработки команд, связанных с файлами;

4) Чтение и запись файлов: Сервер использует потоки ввода-вывода для чтения и записи файлов.

3 ОПИСАНИЕ ФУНКЦИОНАЛЬНОЙ СТРУКТУРЫ ПРОЕКТА

Сервер.

Модули:

- Server: Модуль, отвечающий за создание и запуск сервера;
- ClientHandler: Модуль, отвечающий за обработку запросов от клиентов;
- FileHandler: Модуль, отвечающий за обработку команд, связанных с файлами;
- CommandParser: Модуль, отвечающий за парсинг команд от клиентов;
- NetworkManager: Модуль, отвечающий за сетевое взаимодействие с клиентами.

Функциональность:

- Создание и запуск сервера: Сервер создает сокет, связывает его с адресом и портом, а также запускает прослушивание сокета.
- Обработка запросов от клиентов: Сервер принимает подключения от клиентов, создает для каждого клиента поток обработки и передает управление модулю ClientHandler.
- Парсинг команд: Модуль ClientHandler парсит команды от клиентов и вызывает соответствующие функции для их выполнения.
- Обработка команд, связанных с файлами: Модуль FileHandler обрабатывает команды, связанные с чтением, записью и изменением файлов.
- Сетевое взаимодействие: Модуль NetworkManager отправляет и получает данные от клиентов.

Клиент.

Модули:

Client: Модуль, отвечающий за создание и запуск клиента;

CommandSender: Модуль, отвечающий за отправку команд серверу;

ResponseReceiver: Модуль, отвечающий за получение ответов от сервера;

NetworkManager: Модуль, отвечающий за сетевое взаимодействие с сервером.

Функциональность:

- Создание и запуск клиента: Клиент создает сокет, соединяет его с сервером по указанному адресу и порту.
- Отправка команд: Клиент запрашивает команду у пользователя, отправляет ее серверу и получает ответ.
- Получение ответов: Клиент получает ответ от сервера и выводит его на консоль.
- Сетевое взаимодействие: Модуль NetworkManager отправляет и получает данные от сервера.

4 ПОРЯДОК СБОРКИ И ИСПОЛЬЗОВАНИЯ

Порядок сборки и использования:
1) Открываем консоль;
2) Переходим в каталог с makefile;
3) Пишем в командную строку make;
4) Переходим в build;
5) Запускаем программу parent.
Makefile:

```
CXX = g++
CXXFLAGS = -W -Wall -Wno-unused-parameter -Wno-unused-variable -std=c++17
BUILD_DIR = build/debug
RELEASE_DIR = build/release
SRC_DIR = src

all: server client

server: $(BUILD_DIR)/server_main.o $(BUILD_DIR)/Server.o
$(CXX) $(CXXFLAGS) -o $(BUILD_DIR)/server $(BUILD_DIR)/server_main.o $(BUILD_DIR)/Server.o

client: $(BUILD_DIR)/client_main.o $(BUILD_DIR)/Client.o
$(CXX) $(CXXFLAGS) -o $(BUILD_DIR)/client $(BUILD_DIR)/client_main.o $(BUILD_DIR)/Client.o

release: CXXFLAGS += -O2
release: BUILD_DIR = $(RELEASE_DIR)
release: server client

$(BUILD_DIR)/server_main.o: $(SRC_DIR)/server_main.cpp $(SRC_DIR)/Server.h | $(BUILD_DIR)
$(CXX) $(CXXFLAGS) -c $(SRC_DIR)/server_main.cpp -o $(BUILD_DIR)/server_main.o

$(BUILD_DIR)/client_main.o: $(SRC_DIR)/client_main.cpp $(SRC_DIR)/Client.h | $(BUILD_DIR)
```

```
$(CXX) $(CXXFLAGS) -c $(SRC_DIR)/client_main.cpp -o $(BUILD_DIR)/client_main.o

$(BUILD_DIR)/Server.o: $(SRC_DIR)/Server.cpp $(SRC_DIR)/Server.h | $(BUILD_DIR)
$(CXX) $(CXXFLAGS) -c $(SRC_DIR)/Server.cpp -o $(BUILD_DIR)/Server.o

$(BUILD_DIR)/Client.o: $(SRC_DIR)/Client.cpp $(SRC_DIR)/Client.h | $(BUILD_DIR)
$(CXX) $(CXXFLAGS) -c $(SRC_DIR)/Client.cpp -o $(BUILD_DIR)/Client.o

$(BUILD_DIR):
mkdir -p $(BUILD_DIR)

clean:
rm -rf $(BUILD_DIR)/*.o $(BUILD_DIR)/server $(BUILD_DIR)/client $(RELEASE_DIR)/*.o $(RELEASE_DIR)/server $(RELEASE_DIR)/client
```

5 ОПИСАНИЕ МЕТОДА ТЕСТИРОВАНИЯ И РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Тестирование путем запуска программы.

Работа клиента:

```
rlinux@fedora:~/Kocherov/ОСиСП/LAB08$ build/debug/client
127.0.0.1 8080
> INFO
Welcome to our server!
Here are some commands you can use:
ECHO - write message to server
INFO - Get server information
CD <directory> - Change current directory
LIST - List files in the current directory
QUIT - Disconnect from the server

> LIST
Makefile
src/
 Client.h
 Client.cpp
 Server.cpp
 Server.h
 client_main.cpp
 server_main.cpp
 serverInfo.txt
build/
 debug/
   commands.txt
   testdir/
     111.txt
   serverInfo.txt
   server_main.o
   Server.o
   server
   client_main.o
   Client.o
   client
command.txt

> ECHO lala
lala
> ad as
Invalid command
> QUIT
Server closed connection.
```

Работа сервера:

```
rlinux@fedora:~/Kocherov/ОСиСП/LAB08$ build/debug/server 8080
Welcome to the educational server 'myserver'.
Server is ready.
2024-06-03 13:06:50 Client connected
2024-06-03 13:07:50 Client disconnected
```