Министерство образования Республики Беларусь

Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Операционные системы и системное программирование

Лабораторная работа №8
«Сокеты. Взаимодействие процессов.»

Выполнил: студент группы 350501
Маслаков Н. А.

Проверил: старший преподаватель каф. ЭВМ
Поденок Л. П.

Минск 2025

1 УСЛОВИЕ ЛАБОРАТОРНОЙ 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 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 '>'.

Сервер принимает корневой каталог сервера, с которым он будет работать, и номер порта, на котором будет слушать, и выводит протокол работы в `stdout`:

```
$ myserver root_dir port_no
Готов.
```

Формат протокола произвольный, каждое событие занимает одну строку, первое поле - дата и время в формате `YYYY.MM.DD-hh:mm:ss.sss`.

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 при 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.

2 ОПИСАНИЕ АЛГОРИТМОВ И РЕШЕНИЙ

Программа состоит из двух консольных приложений на языке C — многопоточного сервера и клиентской оболочки — взаимодействующих по TCP-сокетам в домене `AF_INET`. Сервер запускается с указанием корневого каталога и порта, создаёт слушающий сокет (`socket()`, `bind()`, `listen()`) и для каждого подключения порождает отдельный поток через `pthreads`. В каждом потоке поддерживается своё состояние: текущий каталог относительно корня сервера, в который клиент может перемещаться командой `CD`, причём проверка выхода за пределы дерева выполняется с помощью `realpath()` и сравнения с корневым путём.

Клиент подключается к серверу через `connect()`, после чего пользователю отображается приветствие. Дальнейшая работа ведётся в интерактивном режиме: пользователь вводит запросы в формате текста, отправляемые на сервер через `send()/recv()` или через обёртку `fdopen()` и `fgets()/fputs()`.

Клиент также умеет читать пакетные команды из файла, указанного префиксом @, и корректно обрабатывать многострочные ответы до специального разделителя (пустой строки). Сборка проекта организована с помощью `Makefile`: исходники лежат в `src/`, объекты и бинарники создаются в `build/debug/` и `build/release/`, флаги включают `-Wall -Wextra -pthread` и стандарты `POSIX`. Такой дизайн позволяет легко расширять функционал (например, добавлять команды `PWD`, `GET` или `PUT`) и надёжно обслуживать несколько клиентов одновременно.

3 ФУНКЦИОНАЛЬНАЯ СТРУКТУРА ПРОЕКТА

Реализация серверной части

Серверная часть реализована вокруг функции `main()`, выполняющей инициализацию сетевого сокета, обработку входящих подключений и управление клиентскими потоками. На этапе запуска осуществляется:

1) Проверка корректности входных параметров (корневой каталог и порт);

2) Чтение файла `info.txt` для формирования приветственного сообщения;

3) Переход в рабочий каталог с использованием `chdir()`;

4) Создание слушающего сокета с привязкой к указанному порту.

Основной цикл сервера использует `accept()` для принятия новых подключений. Каждый клиент обрабатывается в отдельном потоке, что обеспечивает параллельное обслуживание.

Поддерживаемые команды:

1) `ECHO` - возврат клиенту полученного текста;

2) `INFO` - отправка содержимого файла `info.txt`;

3) `CD` - смена текущего каталога с контролем безопасности;

4) `LIST` - вывод содержимого каталога с классификацией файлов;

5) `QUIT` - завершение сеанса.

Структуры данных

1) `PendingFD` - список активных клиентских дескрипторов с мьютексом для синхронизации;

2) `info_buf` - буфер для хранения приветственного сообщения из `info.txt`;

Функции исходных файлов

Серверная часть

```
void handle_sigint(int signo).
```

Функция `handle_sigint` используется для обработки сигнала `SIGINT`.

Принимает аргумент `signo` - номер сигнала.

```
void *handle_client(void *arg).
```

Функция `handle_client` используется для работы с клиентом через отдельный поток.

Принимает аргумент `arg` - указатель на дескриптор сокета клиента.

Клиентская часть

`void handle_sigint(int signo).`

Функция `handle_sigint` используется для обработки сигнала `SIGINT`.

`void interactive_mode(FILE *sock_fp).`

Функция `nteractive_mode` реализует интерактивное взаимодействие с пользователем.

Принимает аргумент `sock_fp` - указатель на `FILE`, связанный с сокетом.

`void process_file_commands(FILE *sock_fp, const char *fn).`

Функция `process_file_commands` используется для обработки команд из файла.

Принимает аргументы:

1) `FILE *sock_fp`. Сокет для обмена с сервером.
2) `const char *fn` . Имя файла с командами.

Логика клиента включает интерактивный режим с поддержкой пакетной обработки команд через файлы (`@filename`). Обработка `SIGINT` на стороне клиента гарантирует отправку `QUIT` перед завершением.

4 ПОРЯДОК СБОРКИ И ЗАПУСКА

1) Перейти в каталог проекта.

```
$ cd 'Маслаков Н.А./lab08'
```

2) Собрать проект с помощью `make`. по умолчанию сборка происходит в режиме отладки.

```
$ make
```

3) Запустить сервер.

```
$ build/debug/server <root_d> <port>
```

4) Запустить клиент.

```
$ build/debug/client <IP:port>
```

5 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Сервер

```
~/lab08$ valgrind --leak-check=full --show-leak-kinds=all --track-
origins=yes ./build/debug/server
/home/hechert/tar_working_dir/Маслаков_Н.А./lab08/src/s_root_dir
12346
==18598== Memcheck, a memory error detector
==18598== Copyright (C) 2002-2024, and GNU GPL'd, by Julian Seward
et al.
==18598== Using Valgrind-3.24.0 and LibVEX; rerun with -h for
copyright info
==18598== Command: ./build/debug/server
/home/hechert/tar_working_dir/____________________.__./lab08/src/s_r
oot_dir 12346
==18598==
Server listening on port 12346, root dir
'/home/hechert/tar_working_dir/Маслаков_Н.А./lab08/src/s_root_dir'
^C
Server terminated gracefully
==18598==
==18598== HEAP SUMMARY:
==18598==     in use at exit: 0 bytes in 0 blocks
==18598==   total heap usage: 17 allocs, 17 frees, 170,404 bytes
allocated
==18598==
==18598== All heap blocks were freed -- no leaks are possible
==18598==
==18598== For lists of detected and suppressed errors, rerun with:
-s
==18598== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0
from 0)
```

Клиент 1

```
~/lab08$ valgrind --leak-check=full --show-leak-kinds=all --
track-origins=yes ./build/debug/client 127.0.0.1:12346
==18633== Memcheck, a memory error detector
==18633== Copyright (C) 2002-2024, and GNU GPL'd, by Julian
Seward et al.
==18633== Using Valgrind-3.24.0 and LibVEX; rerun with -h for
copyright info
==18633== Command: ./build/debug/client 127.0.0.1:12346
==18633==
Welcome to myserver
Версия: 1.0
Автор: Маслаков Н.А.
Дата сборки: 2025.05.26
```

```
> ECHO lll
lll
> ECHOge;ge;
Unknown command
> ECHO ;;;
;;;
> ECHO ppp
ppp
> INFO
Учебный сервер "myserver"
Версия: 1.0
Автор: Маслаков Н.А.
Дата сборки: 2025.05.26
> INFO
Учебный сервер "myserver"
Версия: 1.0
Автор: Маслаков Н.А.
Дата сборки: 2025.05.26
> CD ..

> @cmd.txt
first
second
../
info.txt
t1/
t2/
link_info.txt --> info.txt
cmd.txt

../
link_t2 -->>
/home/hechert/tar_working_dir/Маслаков_Н.А./lab08/src/s_root_dir
/t2
BYE
==18633==
==18633== HEAP SUMMARY:
==18633==     in use at exit: 0 bytes in 0 blocks
==18633==   total heap usage: 6 allocs, 6 frees, 11,184 bytes
allocated
==18633==
==18633== All heap blocks were freed -- no leaks are possible
==18633==
==18633== For lists of detected and suppressed errors, rerun
with: -s
==18633== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0
from 0)
```

Клиент 2

```
~/lab08$ valgrind --leak-check=full --show-leak-kinds=all --
track-origins=yes ./build/debug/client 127.0.0.1:12346
==18638== Memcheck, a memory error detector
```

```
==18638== Copyright (C) 2002-2024, and GNU GPL'd, by Julian
Seward et al.
==18638== Using Valgrind-3.24.0 and LibVEX; rerun with -h for
copyright info
==18638== Command: ./build/debug/client 127.0.0.1:12346
==18638==
Welcome to myserver
Версия: 1.0
Автор: Маслаков Н.А.
Дата сборки: 2025.05.26
> LIST
../
info.txt
t1/
t2/
link_info.txt --> info.txt
cmd.txt
> ECHO gg
gg
> ECHO kkf
kkf
> CD t1

> LIST
../
link_t2 -->>
/home/hechert/tar_working_dir/Маслаков_Н.А./lab08/src/s_root_dir
/t2
> CD ..

> LIST
../
info.txt
t1/
t2/
link_info.txt --> info.txt
cmd.txt
> INFO
Учебный сервер "myserver"
Версия: 1.0
Автор: Маслаков Н.А.
Дата сборки: 2025.05.26
> ECHO g23
g23
> ^C==18638==
==18638== HEAP SUMMARY:
==18638==     in use at exit: 0 bytes in 0 blocks
==18638==   total heap usage: 4 allocs, 4 frees, 6,616 bytes
allocated
==18638==
==18638== All heap blocks were freed -- no leaks are possible
==18638==
```

```
==18638== For lists of detected and suppressed errors, rerun
with: -s
==18638== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0
from 0)
```