Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей
Кафедра электронных вычислительных машин

Лабораторная работа №8
по курсу
«Операционные системы и системное программирование»
на тему
«Сокеты. Взаимодействие процессов»

Выполнил: студент группы 350501
Слепица О.Н.
Проверил: старший преподаватель каф. ЭВМ
Поденок Л.П.

Минск 2025

1 ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

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`.

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

2.1 Общая структура

Программа представляет собой многопоточный сервер для работы с файловой системой, реализующий следующие функции:
- обработка клиентских подключений через TCP-сокеты;
- поддержка базовых команд: ECHO, INFO, CD, LIST, QUIT;
- безопасный доступ к файлам с проверкой прав доступа;
- логирование событий с временными метками;
- ограничение числа одновременных клиентов (MAX_CLIENTS).

Сервер использует:
- потоки (pthread) для обработки клиентов;
- механизм select() для мониторинга входящих подключений;
- реализацию безопасных путей через realpath().

2.2 Алгоритм работы основного процесса

Алгоритм работы основного процесса включает в себя:
1) инициализация:
- парсинг аргументов (корневой каталог, порт);
- валидация корневого каталога (stat, realpath);
- создание TCP-сокета с настройкой SO_REUSEADDR.
2) главный цикл сервера:
- мониторинг входящих подключений через select();
- принятие новых соединений (accept);
- создание потока для каждого клиента (pthread_create).
3) обработка сигналов:
- корректное завершение по SIGINT/SIGTERM;
- освобождение ресурсов (закрытие сокетов).
4) завершение работы:
- логирование события остановки.

2.3 Работа с клиентами

Поток обработки (handle_connection) включает в себя:
1) установка соединения:
- логирование нового клиента (IP, порт);
- отправка приветственного сообщения (INFO).
2) цикл обработки команд:
- чтение запроса (recv);
- парсинг команды и аргументов (strtok);
- вызов соответствующего обработчика.

3) поддерживаемые команды:
`ECHO` - возврат введенного текста;
`INFO` - информация о сервере;
`CD` - смена текущего каталога;
`LIST` - список файлов;
`QUIT` - разрыв соединения.

2.4 Механизмы безопасности

Механизм безопасности состоит из:
1) проверка путей:
- контроль выхода за корневой каталог (`is_subpath`);
- нормализация путей через `realpath()`.
2) обработка файлов:
- валидация существования файлов (`stat`);
- проверка типов (файл/каталог/ссылка).
3) ограничения:
- максимальное число клиентов;
- буферизация вывода (предотвращение переполнения).

2.5 Особенности реализации

1) многопоточность:
- каждый клиент обслуживается в отдельном потоке;
- потоки создаются в отсоединенном состоянии (`detach`).
2) работа с файловой системой:
- однократная проверка симлинков (`lstat + readlink`);
- безопасное построение путей через `make_safe_path()`.
3) логирование:
- форматированный вывод с миллисекундной точностью;
- фиксация всех клиентских запросов.
4) сетевое взаимодействие:
- таймауты в `select()` для плавного завершения;
- обработка ошибок ввода-вывода.

3 ФУНКЦИОНАЛЬНАЯ СТРУКТУРА ПРОЕКТА

3.1 Модули проекта

Система состоит из двух основных компонентов - сервера и клиента, взаимодействующих по сети через `TCP`-соединение. Каждый компонент имеет четко определенные модули с конкретными функциями.

Модуль сервера (`server.c`) выполняет следующие функции:

1) Инициализация сервера. Создание и настройка серверного сокета;
2) Обработка подключений;
3) Обработка команд клиента (`handle_connection()`);
4) Контроль доступа к файловой системе. Корректная обработка симлинков. Защита от переполнения буферов;
5) Логирование.

Модуль клиента (`client.c`) выполняет следующие функции:

1) Установка соединения;
2) Интерфейс пользователя;
3) Обработка команд;
4) Обработка ошибок.

3.2 Особенности реализации

1) Многопоточная архитектура сервера:
- сервер обрабатывает каждого клиента в отдельном потоке;
- безопасный доступ к общим ресурсам;
- контроль числа одновременных подключений.

2) Безопасная работа с файловой системой:
- нормализация путей через `realpath()`;
- проверка `is_subpath()` для всех операций с файлами;
- однократное чтение симлинков;
- защита от переполнения буферов.

3) Подробное логирование всех операций:
- подробное логирование;
- логирование всех клиентских действий;
- `graceful shutdown`.

4) Кроссплатформенность:
- использование `POSIX`-совместимых вызовов;
- стандартные библиотеки;
- переносимая обработка сигналов.

4 ПОРЯДОК СБОРКИ И ЗАПУСКА ПРОЕКТА

Порядок сборки и запуска состоит в следующем:
1) Разархивировать каталог с проектом;
2) Перейти в каталог с проектом
`$cd "Слепица О.Н./lab08";`
3) Собрать проект используя `make`;
4) После сборки проекта можно использовать, прописав
`$./build/myserver test_server_root 8080`
`$./build/myclient localhost 8080.`

5 РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ

```
helga@fedora:~/tar_working_dir/Слепица О.Н./lab08$
./build/myserver test_server_root 8080
2025.05.26-15:36:06.996 Server started on port 8080, root
directory: /home/helga/tar_working_dir/Слепица О.Н./lab08/
test_server_root
Готов.
2025.05.26-15:36:10.110 Client connected: 127.0.0.1:50302
2025.05.26-15:36:10.118 Client 127.0.0.1:50302: 'INFO'
2025.05.26-15:36:10.165 Client 127.0.0.1:50302: 'ECHO Hello,
World!'
2025.05.26-15:36:10.167 Client 127.0.0.1:50302: ''
2025.05.26-15:36:10.208   Client   127.0.0.1:50302:   'ECHO
Тестовая строка с русскими символами'
2025.05.26-15:36:10.208 Client 127.0.0.1:50302: ''
2025.05.26-15:36:10.251 Client 127.0.0.1:50302: 'LIST'
2025.05.26-15:36:10.269 Client 127.0.0.1:50302: ''
2025.05.26-15:36:10.269 Client 127.0.0.1:50302: 'CD subdir1'
2025.05.26-15:36:10.282 Client 127.0.0.1:50302: ''
2025.05.26-15:36:10.322 Client 127.0.0.1:50302: 'LIST'
2025.05.26-15:36:10.364 Client 127.0.0.1:50302: 'ECHO Сейчас
мы в subdir1'
2025.05.26-15:36:10.365 Client 127.0.0.1:50302: ''
2025.05.26-15:36:10.405 Client 127.0.0.1:50302: 'CD subdir2'
2025.05.26-15:36:10.406 Client 127.0.0.1:50302: ''
2025.05.26-15:36:10.447 Client 127.0.0.1:50302: 'LIST'
2025.05.26-15:36:10.448 Client 127.0.0.1:50302: 'CD ../..'
2025.05.26-15:36:10.452 Client 127.0.0.1:50302: ''
2025.05.26-15:36:10.492 Client 127.0.0.1:50302: 'LIST'
2025.05.26-15:36:10.535   Client   127.0.0.1:50302:   'ECHO
Вернулись в корень'
2025.05.26-15:36:10.535 Client 127.0.0.1:50302: ''
2025.05.26-15:36:10.576    Client    127.0.0.1:50302:    'CD
nonexistent'
2025.05.26-15:36:10.577 Client 127.0.0.1:50302: ''
2025.05.26-15:36:10.617 Client 127.0.0.1:50302: 'LIST'
2025.05.26-15:36:10.660 Client 127.0.0.1:50302: 'QUIT'
2025.05.26-15:36:10.661 Client disconnected: 127.0.0.1:50302

helga@fedora:~/tar_working_dir/Слепица О.Н./lab08$
./build/myclient localhost 8080 < test_commands.txt
Connecting to 127.0.0.1:8080...
Connected to server.
Вас приветствует учебный сервер 'myserver'
Доступные команды:
  INFO
  ECHO <текст>
  LIST
```

```
  CD <путь>
  QUIT
> Вас приветствует учебный сервер 'myserver'
Доступные команды:
  INFO
  ECHO <текст>
  LIST
  CD <путь>
  QUIT
> Hello, World!
> Тестовая строка с русскими символами
> subdir1/
subdir3/
file1.txt
file2.txt
link_to_file1 --> file1.txt
link_to_subdir1 --> subdir1
broken_link --> nonexistent
link_to_link -->> link_to_file1
> subdir1> subdir2/
subfile.txt
subdir1> Сейчас мы в subdir1
subdir1> subdir1/subdir2> subdir1/subdir2> > subdir1/
subdir3/
file1.txt
file2.txt
link_to_file1 --> file1.txt
link_to_subdir1 --> subdir1
broken_link --> nonexistent
link_to_link -->> link_to_file1
> Вернулись в корень
> > subdir1/
subdir3/
file1.txt
file2.txt
link_to_file1 --> file1.txt
link_to_subdir1 --> subdir1
broken_link --> nonexistent
link_to_link -->> link_to_file1
> BYE
```