МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

## Факультет информационных технологий и робототехники

Кафедра программного обеспечения информационных систем

и технологий

**Отчет**

**по лабораторной работе № 11**

по дисциплине: ”Системное программирование”

на тему: ***”*** Сокеты и удаленное межпроцессное взаимодействие***”***

Выполнил**:** студент группы *10702121* Агеенко А. К.

Высоцкий М. Л.

Градусов А. Р.

Бродко Г. Г.

Принял**:** пр. Мисякова В. А.

Минск 2023

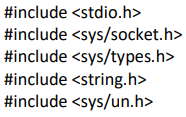
# Лабораторная работа №11.

**Цель работы:** Изучить механизм сокетного сетевого взаимодействия в LINUX

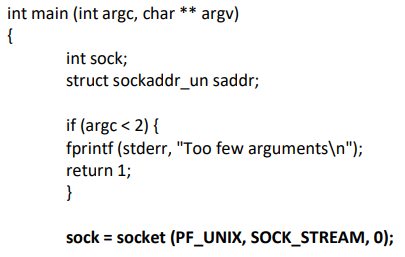
**Задание 1** Создать локальный сокет и назначить ему локальный адрес.

Решение

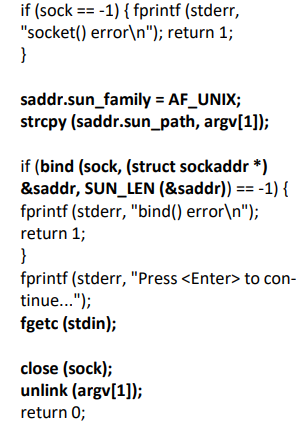
**Подключаем библиотеки**



**Создаем сокет**



**Назначаем ему локальный адрес**



**Проверяем:**



**Откроем другое терминальное окно и посмотрим на наш сокет:**



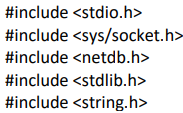
После нажатия клавиши в первом терминале сокет исчезнет.

**Какой консольной командой и каким системным вызовом в программах удаляется сокет из системы?**

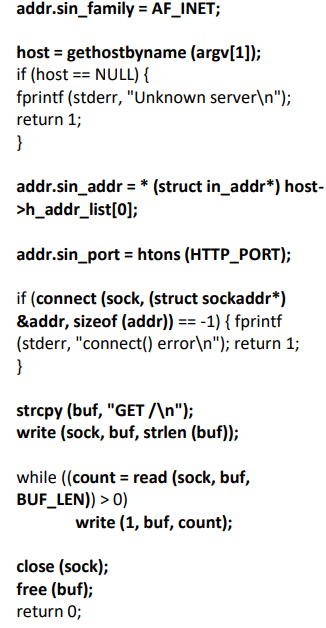
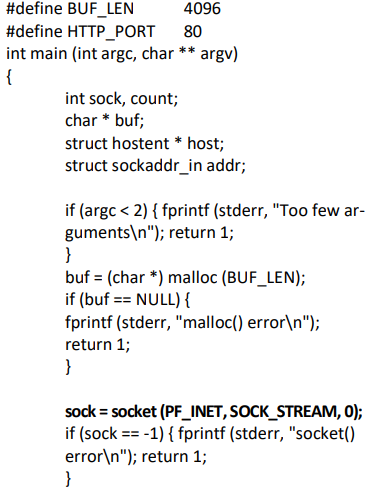
В ОС Linux консольной командой для удаления сокета является rm (remove). Однако, для удаления сокета из программы в ОС Linux часто используется системный вызов unlink, который удаляет файл из файловой системы по его имени. Сокеты в ОС Linux представлены в файловой системе как специальные файлы, поэтому удаление сокета в программе производится с помощью вызова unlink.

**Задание 2: Реализуйте чтение главное страницы из указанного web-сервиса**

**Подключаем библиотеки**



**Реализуем чтение**



**Подключаемся к интернету**



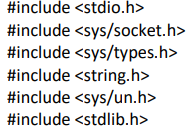
**Заносим инфу в файл**



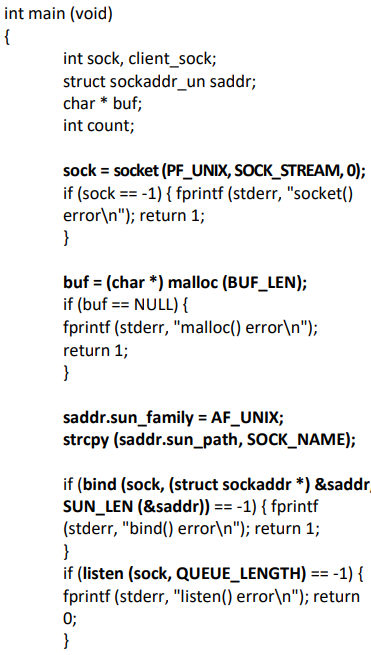
**Задание 3 Реализовать межпроцессное взаимодействие с использованием локальных потоковых сокетов.**

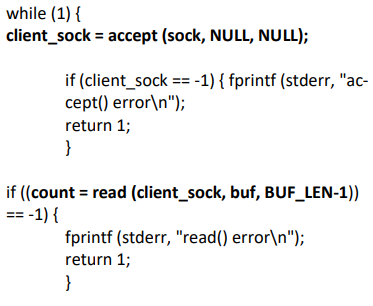
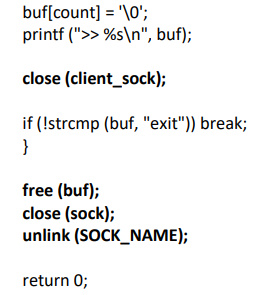
Создадим локальное клиент-серверное приложение.

Подключим библиотеки

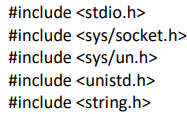


**Создадим сервер**

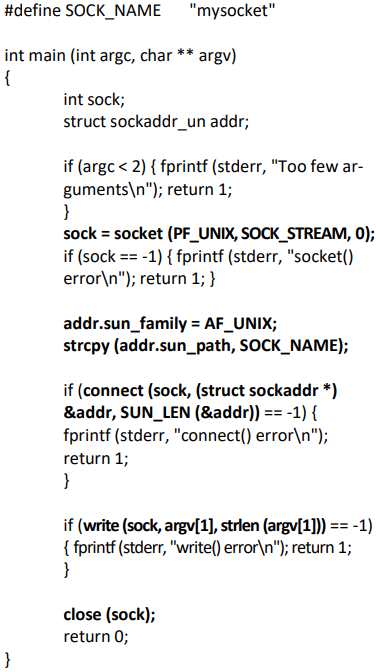


**Подключим библиотеки**



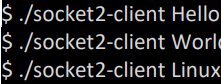
**Создадим клиента**



**После создания сервера переходим в режим прослушивания сокета**

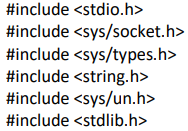
**Откроем другое терминальное окно и передадим серверу запросы:**

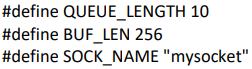
**Задание 4 Реализовать прием и передачу данных через дейтаграммные сокеты**

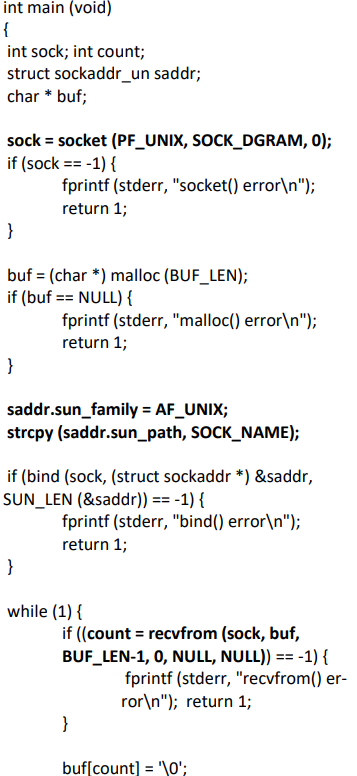
**Создаем сервер**

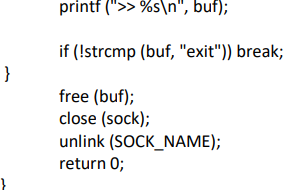
**Подключаем библиотеки**



**Реализуем**

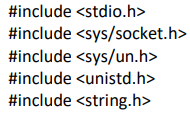






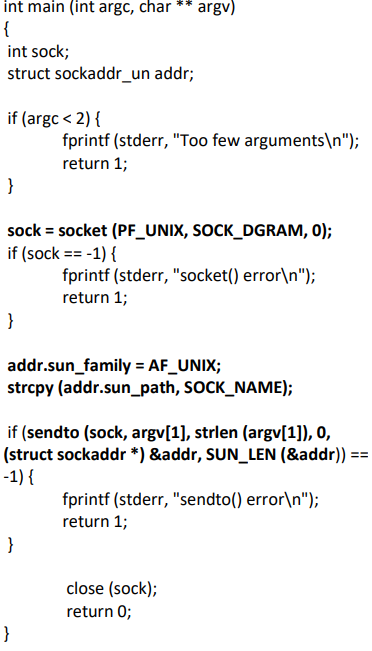
**Создаем клиента**

**Подключаем библиотеки**



**Реализуем**





**Контрольные**

1) **Что такое удаленное межпроцессное взаимодействие?**

Библиотеки позволяют разным программам использовать один и тот же объектный код. Библиотека (library) – это набор соединенных определенным образом объектных файлов. Библиотеки подразделяются на две категории: − статические (архивы); − динамические (совместно используемые)

2) **Как переводится слово «socket» с английского на русском язык?**

Слово "socket" переводится на русский язык как "сокет".

3) **Что такое сокет? Как он представлен в операционной системе?**

Сокет - это абстрактное понятие для реализации механизма межпроцессного взаимодействия (IPC) в операционной системе. Он представляет собой интерфейс для обмена данными между процессами, работающими на одной машине или на разных машинах в сети.

В операционной системе сокет представлен в виде файлового дескриптора, что позволяет программам использовать стандартные файловые операции для взаимодействия с сокетом. Когда сокет создается, операционная система выделяет соответствующий файловый дескриптор и связывает его с сетевыми ресурсами, такими как IP-адрес и порт. Это позволяет процессам читать и записывать данные через сокет, а также выполнять другие операции, такие как установка соединения и прослушивание запросов от клиентов.

Сокеты позволяют процессам различных программ обмениваться данными через сеть или между различными частями одной программы, что делает их мощным инструментом для разработки сетевых приложений и клиент-серверных систем.

4) **Какие операции можно производить над сокетом?**

Сокет предоставляет различные операции для взаимодействия с протоколами в сети. Вот некоторые основные операции, которые можно производить над сокетом:

1. Создание сокета: Сокет можно создать с помощью функции socket(), указав тип сокета (например, SOCKSTREAM для TCP или SOCKDGRAM для UDP) и домен (например, AFINET для IPv4 или AFINET6 для IPv6).

2. Связывание сокета: С помощью функции bind() сокет можно связать с конкретным сетевым адресом и портом. Это позволяет сокету прослушивать входящие соединения на определенном адресе и порту.

3. Установка в режим прослушивания: Для сокета TCP можно установить режим прослушивания (listening) с помощью функции listen(). Это позволяет сокету ожидать входящих подключений от клиентов.

4. Принятие соединения: Для сокета TCP функция accept() позволяет принять входящее подключение от клиента. Она создает новый сокет для этого соединения и возвращает файловый дескриптор этого сокета.

5. Чтение и запись данных: Сокеты позволяют производить операции чтения и записи данных. Для сокета TCP функции read() и write() используются для чтения данных из сокета и записи данных в сокет соответственно. Для сокета UDP используется функция sendto() для отправки данных и recvfrom() для приема данных.

6. Закрытие сокета: Когда работа с сокетом завершена, его следует закрыть с помощью функции close(). Это освободит ресурсы, связанные с сокетом, и закроет соединение, если оно было установлено.

Кроме этих основных операций, также существуют дополнительные операции, такие как настройка параметров сокета, установка и разрыв соединений, установка таймаутов и т.д., которые зависят от конкретного протокола и типа сокета.

5) **Какой способ взаимодействия процессов обеспечивает сокет?**

Сокет обеспечивает взаимодействие между процессами при помощи сетевого протокола. Сокеты позволяют процессам обмениваться данными через сеть, как если бы они были локальными процессами. С использованием сокетов программисты могут создавать клиент-серверные приложения, где клиентские программы могут отправлять запросы серверу, а серверные программы могут отвечать на эти запросы и передавать данные обратно клиентам. Таким образом, сокеты обеспечивают удаленное взаимодействие процессов в распределенной сетевой среде.

6) **Что такое архитектура приложений клиент/сервер, и в чем ее отличие от архитектуры взаимодействия процессов «источник» - «приёмник»**.

Архитектура приложений клиент/сервер - это модель взаимодействия между различными компьютерными процессами (клиентом и сервером), где клиентские процессы инициируют запросы, а серверные процессы отвечают на эти запросы, предоставляя необходимые данные или услуги.

В архитектуре клиент/сервер обычно существует централизованный сервер, который предоставляет определенные функции или ресурсы, а клиенты обращаются к серверу для доступа к этим функциям или ресурсам. Клиенты и серверы могут работать на разных компьютерах в сети или на одной машине.

Отличие архитектуры клиент/сервер от архитектуры взаимодействия процессов «источник» - «приёмник» состоит в следующем:

1. Централизованность: В архитектуре клиент/сервер существует центральный сервер, который управляет всеми клиентскими запросами и отвечает на них. В то время как в архитектуре "источник" - "приёмник" каждый процесс является источником и получателем данных, и обмен данными происходит напрямую между ними без централизации.

2. Распределенность: В архитектуре клиент/сервер клиенты и серверы могут работать на разных компьютерах в сети, обмениваясь данными удаленно. В архитектуре "источник" - "приёмник" процессы обычно работают на одной машине и обмениваются данными локально.

3. Роли и ответственности: В архитектуре клиент/сервер сервер обычно отвечает за предоставление определенных услуг или данных, а клиенты запрашивают эти услуги или данные. В архитектуре "источник" - "приёмник" обмен данными может происходить

7) **Что такое потоковые и дейтаграммные сокеты, для чего они используются?**

Потоковые сокеты и дейтаграммные сокеты являются двумя основными типами сокетов в сетевом программировании.

1. Потоковые сокеты (stream sockets) - предоставляют надежное и упорядоченное двустороннее соединение между двумя сетевыми узлами. Этот тип сокетов основан на протоколе TCP (Transmission Control Protocol) и гарантирует доставку данных без потерь, дублирования или изменения порядка. Поскольку потоковые сокеты поддерживают надежную передачу данных, они широко используются для передачи файлов, передачи потоков видео или аудиоданных, а также для выполнения удаленных вызовов процедур (RPC).

2. Дейтаграммные сокеты (datagram sockets) - предоставляют ненадежное и негарантированное соединение между двумя сетевыми узлами. Этот тип сокетов основан на протоколе UDP (User Datagram Protocol) и не гарантирует доставку данных или упорядоченность их получения. Дейтаграммные сокеты широко используются для приложений, где потеря данных или небольшая задержка являются приемлемыми, например, в многопользовательских играх, видеоконференциях, трансляциях в реальном времени и т. д.

Каждый тип сокетов имеет свои преимущества и недостатки, и выбор между ними зависит от требований конкретного приложения и предпочтений программиста.

8) **Перечислите основные операции процесса-сервера**

Основные операции процесса-сервера могут включать:

1. Привязка к сетевому порту: Процесс-сервер должен привязаться к определенному сетевому порту, чтобы начать прослушивание входящих соединений или прием дейтаграмм.

2. Прослушивание на входящие соединения: Процесс-сервер может выполнять операцию прослушивания входящих соединений на связанном порту, ожидая запросов от клиентских приложений.

3. Принятие соединения: Когда процесс-сервер получает запрос на соединение от клиентского приложения, он может произвести операцию принятия соединения, чтобы установить сетевой канал связи с клиентом.

4. Обработка запроса: Процесс-сервер выполняет операцию обработки запроса, которая может варьироваться в зависимости от типа приложения. Обработка может включать чтение и запись данных с клиентом, выполнение запрошенных операций или обработку данных пользователя.

5. Отправка ответа: После обработки запроса процесс-сервер может выполнить операцию отправки ответа, чтобы вернуть результаты клиенту. Это может включать отправку данных, статуса или любых других необходимых информационных сообщений.

6. Закрытие соединения: При завершении обработки запроса или на основании определенных условий процесс-сервер может выполнить операцию закрытия соединения. Это освобождает занятые ресурсы и устанавливает конечную связь с клиентом.

7. Управление ошибками: В процесс-сервере также должны быть встроены операции управления ошибками для обнаружения и обработки ошибок в процессе обмена данными с клиентом или выполнения операций.

Это лишь некоторые из основных операций, которые процесс-сервер может выполнять. Фактический список операций может варьироваться в зависимости от типа приложения и требований.

9) **Перечислите основные операции процесса-клиента.**

Основные операции процесса-клиента включают:

1. Создание сокета: Процесс-клиент создает сокет, который будет использоваться для установления соединения с сервером.

2. Установка соединения: Процесс-клиент устанавливает соединение с сервером, отправляя запрос на соединение и ожидая подтверждения со стороны сервера.

3. Отправка запроса: После установки соединения процесс-клиент может отправить запрос на сервер, содержащий данные, команды или другую информацию, необходимую для выполнения операции на сервере.

4. Получение ответа: Когда сервер обрабатывает запрос, процесс-клиент ожидает получения ответа от сервера. Ответ может содержать необходимые данные, результат операции или статус выполнения запроса.

5. Обработка ответа: После получения ответа от сервера процесс-клиент обрабатывает полученные данные, результаты операции или выполняет необходимые действия на основании статуса ответа.

6. Отправка дополнительных запросов: При необходимости процесс-клиент может отправить дополнительные запросы на сервер для получения дополнительных данных или выполнения дополнительных операций.

7. Закрытие соединения: При завершении коммуникации с сервером процесс-клиент выполняет операцию закрытия соединения, освобождая ресурсы и устанавливая конечную связь с сервером.

8. Обработка ошибок: Процесс-клиент также должен обрабатывать возможные ошибки, которые могут возникнуть в процессе обмена данными с сервером или выполнения операций.

Это лишь некоторые из основных операций, которые процесс-клиент может выполнять. Реальные операции будут зависеть от конкретного приложения и требований.

10) **Какие системные вызовы обеспечивают основные операции с сокетами?**

Основные операции с сокетами в сетевом программировании обеспечиваются с помощью следующих системных вызовов:

1. socket(): Создание сокета. Этот системный вызов создает новый сокет и возвращает его дескриптор, который используется для дальнейшей работы с сокетом.

2. bind(): Привязка сокета к определенному адресу (IP-адресу и порту).

3. listen(): Прослушивание входящих соединений. Для потоковых сокетов этот вызов указывает, что сокет должен начать прослушивать входящие соединения.

4. accept(): Принятие входящего соединения. Этот вызов используется на стороне сервера для принятия новых соединений от клиентских приложений. Он создает новый сокет для этого соединения и возвращает дескриптор нового сокета.

5. connect(): Установка соединения с удаленным сервером. Этот вызов используется на стороне клиента для установки соединения с серверным сокетом.

6. send()/write(): Отправка данных через сокет. Эти вызовы используются для отправки данных на удаленный сокет.

7. recv()/read(): Прием данных через сокет. Эти вызовы используются для чтения данных из сокета, полученных от удаленного сокета.

8. close(): Закрытие сокета. Этот вызов используется для закрытия сокета после завершения обмена данными или прекращения соединения.

Это лишь некоторые из основных системных вызовов, связанных с сокетами. Фактический набор системных вызовов может различаться в зависимости от операционной системы и используемого протокола передачи данных (TCP, UDP и т. д.).

11) **Какой консольной командой и каким системным вызовом в программах удаляется сокет из системы?**

В сети Unix-подобных систем, чтобы удалить сокет из системы, можно воспользоваться следующей консольной командой:

rm <путь к файлу сокета>

Эта команда удаляет файл сокета из файловой системы.

Также, для удаления сокета с помощью системного вызова в программе на языке программирования C или других языках, используется функция unlink():

#include <unistd.h>

int unlink(const char \*pathname);

Функция unlink() удаляет файл по указанному пути. В случае с сокетами, передавайте в качестве аргумента функции unlink() путь до файла сокета, который вы хотите удалить.

Обратите внимание, что удаление сокета связано с закрытием всех открытых соединений и освобождением связанных ресурсов.

12) **Перечислите основные типы сокетов.**

Основные типы сокетов, используемые в сетевом программировании, включают:

1. Потоковые сокеты (Stream Sockets): Эти сокеты используют протокол TCP (Transmission Control Protocol), который обеспечивает надежную и упорядоченную передачу данных. Потоковые сокеты обеспечивают двунаправленное соединение, гарантирующее, что данные будут доставлены без потерь, дублирования или изменения порядка. Они обеспечивают надежный поток данных, похожий на конвейер или трубу.

2. Дейтаграммные сокеты (Datagram Sockets): Эти сокеты используют протокол UDP (User Datagram Protocol), который обеспечивает ненадежную, негарантированную передачу данных. Дейтаграммные сокеты предоставляют возможность отправки и получения дискретных сообщений, известных как дейтаграммы, без гарантии их доставки, упорядоченности или ненарушаемости.

3. RAW сокеты: RAW сокеты позволяют сетевым приложениям взаимодействовать напрямую с сетевым стеком и отправлять и получать пакеты на низком уровне, обходя традиционные протоколы транспортного уровня, такие как TCP и UDP. RAW сокеты могут быть полезны для разработки сетевых утилит, анализа сетевого трафика и создания собственных сетевых протоколов.

4. Последовательные сокеты (Sequential Sockets): Это специальный тип сокета, который позволяет установить последовательное соединение с последовательным устройством, таким как последовательный порт.

Это лишь некоторые из основных типов сокетов, используемых в сетевом программировании. Выбор типа сокета зависит от требований конкретного приложения и используемого протокола.

13) **Что такое коммуникационный домен и семейство протоколов?**

Коммуникационный домен - это группа устройств, которые могут взаимодействовать друг с другом в рамках локальной сети или сети обмена данными. Коммуникационный домен обычно определяется физическими или логическими ограничениями, такими как кабели, коммутаторы, маршрутизаторы или протоколы сети. Устройства внутри коммуникационного домена могут обмениваться информацией и ресурсами без необходимости направления этой информации за пределы домена.

Семейство протоколов - это набор связанных протоколов, которые определяют правила и формат обмена информацией между устройствами в сети. Каждый протокол внутри семейства выполняет определенные задачи, такие как управление передачей данных, адресацию, маршрутизацию и т. д. Примером семейства протоколов является семейство TCP/IP, которое используется в большинстве сетей Интернет. Это семейство включает протоколы, такие как IP (протокол интернета) для маршрутизации пакетов данных и TCP (протокол управления передачей) для обеспечения надежной доставки данных.