# СОДЕЖРАНИЕ

[1 Перечень используемых сокращений 3](#_Toc134887365)

[2 ВВЕДЕНИЕ 4](#_Toc134887366)

[3 ОБЗОР МЕТОДОВ И АЛГОРИТМОВ РЕШЕНИЯ ЗАДАЧИ 14](#_Toc134887367)

[4 ОБОСНОВАНИЕ МЕТОДОВ И АЛГОРИТМОВ РЕШЕНИЯ ЗАДАЧИ 15](#_Toc134887368)

[5 ОПИСАНИЕ ПРОГРАММЫ ДЛЯ ПРОГРАММИСТА 16](#_Toc134887369)

[6 ОПИСАНИЕ АЛГОРИТМОВ РЕШЕНИЯ ЗАДАЧИ 17](#_Toc134887370)

[7 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 18](#_Toc134887371)

[8 ЗАКЛЮЧЕНИЕ 19](#_Toc134887372)

[9 СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 20](#_Toc134887373)

# Перечень используемых сокращений

1. ПО – Программное обеспечение.
2. ОС – Операционная система.
3. ЭВМ – Электронно-вычислительная машина.
4. СУБД – Система управления базами данных.
5. ПК – Персональный Компьютер.
6. DNS – Domain Name Service, система доменных имен.
7. FPT - File Transfer Protocol, протокол передачи данных.
8. IP – Internet Protocol, Протокол Интернета.
9. SNMP - Simple Network Management Protocol, простой протокол управления сетями.
10. URL – Universal Resource Locator, универсальный указатель ресурсов.

# ВВЕДЕНИЕ

## Программное обеспечение.

Программное обеспечение (ПО) представляет собой комплекс программ, которые хранятся на запоминающих устройствах компьютера и выполняются процессором. Существует несколько групп ПО, каждая из которых имеет свои задачи и функциональность:

* Системное ПО отвечает за корректное функционирование компьютера. Оно включает в себя операционные системы (ОС), драйверы и другие программы, необходимые для обеспечения работы аппаратных устройств.
* Прикладное ПО предназначено для выполнения определенных задач пользователем. К нему относятся программы и пакеты, которые используются для решения конкретных задач, например, текстовые редакторы, графические редакторы, программы для управления базами данных и другие.
* Инструментальное ПО используется для разработки системного и прикладного ПО. К нему относятся языки программирования, компиляторы, трансляторы и другие средства разработки.

Для данного курсового проекта также важно отметить программы для управления базами данных (СУБД), которые предназначены для хранения, обработки и выдачи данных по запросу пользователя. Кроме того, следует учитывать, что существуют различные версии программного обеспечения, которые могут отличаться по функциональности, стабильности и совместимости с другими программами.

## Операционная система.

Операционная система (ОС) — это неотъемлемая часть любого компьютера, обеспечивающая корректное взаимодействие программного обеспечения и аппаратных компонентов компьютера. Она состоит из нескольких компонентов, включая файловую систему, систему ввода/вывода и командный процессор. Кроме того, важной составляющей ОС являются драйверы, которые управляют работой устройств, подключенных к компьютеру. Операционная система играет роль связующего звена между пользователем и аппаратным обеспечением ЭВМ (рис. 2.1), обеспечивая управление аппаратными средствами и общее управление ЭВМ.

Пользователь

Аппаратное обеспечение

Сервисные и прикладные программы

Операционная Система

Рисунок 2.1 – Роль ОС во взаимодействии пользователя с ЭВМ.

Операционные системы выполняют широкий спектр функций, которые обеспечивают правильную работу компьютера. Среди них можно выделить такие важные функции, как инициация, загрузка программ и их исполнение, обеспечение работы системы управления файлами, планирование и диспетчеризация задач, работа с командами ввода/вывода, управление памятью и организация виртуальной памяти.

Функции операционной системы как правило группируются в соответствии с типами локальных ресурсов, которыми управляет система. Такие группы принято называть подсистемами. Наиболее значимые подсистемы ЭВМ включают подсистему управления процессами, управления памятью, управления файлами, управления внешними устройствами, пользовательского интерфейса и защиты данных и администрирования.

По числу одновременно исполняемых задач операционные системы делятся на однозадачные и многозадачные. Однозадачные системы способны выполнять только одну задачу за раз, тогда как многозадачные могут одновременно выполнять несколько задач. Один из наиболее известных примеров многозадачных ОС является Linux и дистрибутивы, основанные на ядре Linux. Они способны выполнять множество задач одновременно, что делает их очень популярными среди пользователей.

Операционные системы (ОС) с многозадачностью обладают функциями, позволяющими создать виртуальную машину для пользователя и управлять периферийными устройствами, файлами и взаимодействием пользователя с ОС и компьютером. Эти ОС также управляют разделением совместно используемых ресурсов, таких как процессор, оперативная память, файлы и внешние устройства.

Одной из наиболее популярных многозадачных ОС является Linux. Linux – это 32 или 64-разрядная UNIX-подобная ОС, разработанная для работы на широком спектре компьютерного оборудования. Она объединяет функции настольной и серверной ОС. Кроме того, она является POSIX-совместимой ОС, что означает, что она способна работать на различных конфигурациях аппаратных платформ.

Linux поддерживает многозадачность и позволяет большому числу пользователей работать на одной машине, что делает ее идеальным решением как для конфигурации персонального компьютера, так и для серверной ОС. Кроме того, Linux поддерживает разделение пользователей по уровням доступа, позволяя гибко настраивать права для большого числа клиентов сервера.

ОС Linux также поддерживает набор протоколов TCP/IP для сетевой работы и включает драйверы устройств для сетевых карт Ethernet. Это позволяет легко настроить сетевые настройки и обеспечить высокую скорость передачи данных. Кроме того, Linux имеет обширную поддержку сообщества, которое разрабатывает и поддерживает множество программных пакетов и приложений для ОС.

## Вычислительные сети и сетевые технологии.

Сеть – это совокупность устройств и программных средств, которые обеспечивают передачу данных между различными устройствами и пользователей, позволяя им обмениваться информацией и ресурсами. Компьютерные сети играют важную роль в нашей жизни, с их помощью мы можем связываться с друзьями и коллегами, получать и передавать информацию, делать покупки и многое другое.

С развитием компьютерных технологий и телекоммуникационных технологий, компьютерные сети стали более сложными и функциональными. В настоящее время мы имеем широкий спектр различных типов сетей, включая локальные сети, глобальные сети, сети мобильной связи, сети Интернет и многие другие.

Первые компьютерные сети были созданы в 1960-х годах. Они были очень простыми и использовались в основном для обмена файлами и электронной почты. С течением времени сети стали более сложными, и были разработаны новые технологии для улучшения их производительности и безопасности.

Одной из ключевых проблем, которую пришлось решить при создании компьютерных сетей, было разработка протоколов и стандартов, которые обеспечивали бы надежную и эффективную передачу данных. Эти протоколы определяют, как данные передаются между устройствами в сети, как они обрабатываются и как происходит обмен информацией между устройствами.

Сегодня компьютерные сети играют важную роль в бизнесе, образовании, медицине и других отраслях. Они позволяют людям работать вместе, обмениваться информацией и ресурсами, и эффективно использовать ресурсы компьютеров и других устройств.

Исследования в области сетевых технологий показали, что передача данных в виде пакетов более эффективна и быстродейственна, чем передача данных непосредственно в виде потока. Коммутация пакетов является одной из самых популярных и эффективных технологий передачи данных в компьютерных сетях. В коммутируемых сетях каждый пакет данных обрабатывается и передается независимо от других пакетов, что позволяет существенно сократить задержки и повысить пропускную способность сети.

Одним из примеров коммутируемых сетей является сеть Интернет, которая использует технологию коммутации пакетов для передачи данных. Каждый пакет данных, отправляемый через Интернет, имеет заголовок с адресом конечного узла, который позволяет маршрутизаторам на пути следования пакета определить оптимальный путь для доставки данных. Это позволяет минимизировать задержки и увеличить скорость передачи данных в Интернете.

Однако, помимо коммутации пакетов, существуют и другие технологии передачи данных, такие как коммутация каналов и коммутация сообщений, которые находят свое применение в различных областях, в зависимости от требований к скорости, надежности и пропускной способности сети.

Сеть APRANET, появившаяся в 1969 году, стала отправной точкой для создания интернета - наиболее известной в мире сети. APRANET была создана для связи компьютеров разных типов, работающих под управлением различных операционных систем (ОС), использующих общие коммуникационные протоколы. Это позволило компьютерам взаимодействовать друг с другом и обмениваться информацией.

С появлением персональных компьютеров, вычислительные мощности которых росли с каждым годом, стала возникать потребность в развитии сетей и их функций. В результате были созданы локальные вычислительные сети, основанные на различных технологиях, таких как Ethernet, Arcnet, Token Ring, Token Bus и т.д. Эти стандартизированные сетевые технологии позволили упростить процесс подключения компьютеров к сети и снизить затраты на оборудование.

Сетевая технология - это набор программных и аппаратных средств, который позволяет передавать данные по линиям связи и создавать вычислительную сеть. Существует множество различных сетевых технологий, каждая из которых имеет свои особенности и преимущества в зависимости от задач, которые ей предстоит решать.

Вместе с развитием вычислительной техники и телекоммуникационных технологий произошло расширение функциональных возможностей локальных вычислительных сетей. Они стали использоваться не только для обмена данными между компьютерами внутри организации, но и для обеспечения доступа к интернету и удаленным ресурсам. Это привело к развитию новых технологий, таких как Wi-Fi, Bluetooth, и NFC, которые обеспечивают беспроводное подключение к сети и упрощают использование сетевых ресурсов.

Один из ключевых элементов сетевой инфраструктуры - сетевые серверы, выполняющие функции хранения и обработки данных. Серверы могут быть распределенными, т.е. размещаться в разных локациях и взаимодействовать между собой. Для их связи и управления существуют различные протоколы, такие как HTTP, FTP, SMTP, POP3 и другие.

Протокол IP (Internet Protocol) стал доминирующим протоколом в глобальных сетях, в том числе и в интернете. Он обеспечивает маршрутизацию и доставку пакетов данных между различными узлами сети. Для обеспечения безопасности передачи данных в глобальных сетях широко используются различные протоколы шифрования, такие как SSL/TLS, IPSec, SSH и другие.

Ethernet по-прежнему является одной из наиболее распространенных технологий для организации локальных сетей. С появлением новых стандартов, таких как 10 Gigabit Ethernet, 40 Gigabit Ethernet и 100 Gigabit Ethernet, Ethernet стала также применяться для организации высокоскоростных линий связи в глобальных сетях.

В клиент-серверной архитектуре клиентские устройства (как правило, персональные компьютеры) запрашивают данные или услуги у серверов. Это позволяет более эффективно управлять сетевыми ресурсами и упрощает масштабирование сети.

Серверы могут выполнять различные функции, например, обеспечивать доступ к файлам, базам данных, электронной почте, веб-страницам и другим ресурсам. Существуют разные типы серверов, такие как файловые серверы, веб-серверы, почтовые серверы и т.д.

Кроме того, клиент-серверная архитектура позволяет обеспечить безопасность данных и управлять доступом пользователей к различным ресурсам в сети. Сервер может авторизовать пользователей и контролировать их доступ к определенным файлам и другим ресурсам в сети.

Компьютерные сети, в том числе и сети Интернет, используют систему доменных имен (DNS) для преобразования доменных имен в IP-адреса. Когда пользователь вводит URL-адрес в браузере, браузер отправляет запрос на DNS-сервер, чтобы узнать, какой IP-адрес соответствует этому доменному имени.

DNS-серверы работают в иерархической структуре, которая включает в себя корневые серверы, серверы верхнего уровня и серверы нижнего уровня. Корневые серверы хранят информацию о серверах верхнего уровня, которые, в свою очередь, хранят информацию о серверах нижнего уровня. Когда DNS-сервер получает запрос на преобразование доменного имени, он начинает поиск информации, начиная с корневого сервера, который указывает, на какой сервер верхнего уровня необходимо обратиться.

Существует множество различных DNS-серверов, включая общедоступные серверы, управляемые крупными компаниями, такими как Google и OpenDNS, а также серверы, управляемые интернет-провайдерами. В зависимости от местоположения источника запроса DNS-сервер может выбирать наиболее близкий сервер для ускорения процесса преобразования доменного имени в IP-адрес.

## Клиент-серверная архитектура.

Клиент/серверная модель является широко используемой архитектурой в сетевых приложениях. Она позволяет разделить задачи по обработке данных между клиентским и серверным приложениями. При этом клиентская часть приложения отвечает за представление данных пользователю и их обработку, а серверная часть – за хранение данных и их обработку на более высоком уровне (рис. 2.2).

СЕТЬ

Локальная ОС

Серверная часть

Драйвер порта

Клиентская Часть

Драйвер порта

Локальная ОС

Приложение А

Редиректор

Компьютер Б / Сервер

Локальные ресурсы

Компьютер А / Клиент

Локальные ресурсы

Рисунок 2.2 – Взаимодействие программных компонентов при связи двух компьютеров в сети.

Клиент/серверная модель позволяет уменьшить нагрузку на клиентскую сторону, так как большая часть обработки данных выполняется на сервере. Это позволяет увеличить быстродействие и удобство использования приложения. Один сервер может обслуживать множество клиентов, что делает эту модель масштабируемой и гибкой.

Основным механизмом передачи данных между клиентом и сервером является протокол HTTP, который является основой работы большинства веб-приложений. Однако, клиент/серверная модель может использовать и другие протоколы, такие как TCP/IP, UDP и т.д.

Кроме того, клиент/серверная модель может быть расширена с использованием дополнительных серверных компонентов, таких как базы данных, кэширование и балансировка нагрузки. Благодаря этому, приложения на основе клиент/серверной модели могут быть более гибкими и мощными, чем простые веб-приложения.

Клиент/серверная модель является распространенной архитектурой в компьютерных сетях. Для обмена данными между компонентами используются коммуникационные протоколы, такие как TCP/IP. Коммуникационные протоколы обеспечивают надежную доставку пакетов данных и управление потоками данных в сети.

Сетевое программное обеспечение реализуется в виде нескольких уровней протоколов, которые образуют стек протоколов. Каждый уровень отвечает за выполнение определенных функций в процессе передачи данных. Наиболее распространенной моделью уровней протоколов является модель OSI, которая включает семь уровней: физический, канальный, сетевой, транспортный, сеансовый, презентационный и прикладной. Каждый уровень выполняет определенную функцию, такую как управление физическим подключением, маршрутизацией или шифрованием данных.

Клиент/серверная модель является широко используемой архитектурой для разработки приложений, которая позволяет разделять уровень представления данных от их обработки и хранения. Для обмена данными между компонентами используются коммуникационные протоколы, которые определяют процедуру упаковки данных, скорость передачи и преобразование данных в исходный вид.

Сервер Internet предоставляет доступ к сервисам через сокеты. Сервисы представлены портами, имеющими числовые идентификаторы (адреса), через которые обрабатываются запросы. Для использования протокола TCP требуется указание IP-адреса и порта. IP-адрес обеспечивает логический адрес сетевого устройства, а каждому доменному имени компьютера в системе доменных имен DNS соответствует IP-адрес. В случае, если номер порта не указан, то используется номер порта по умолчанию для сервиса. Например, для FTP сервера зарезервирован порт номер 21.

Протокол FTP (File Transfer Protocol) предназначен для передачи файлов между компьютерами. Он позволяет пользователю управлять удаленным файловым хранилищем и осуществлять передачу файлов между клиентом и сервером. FTP работает поверх протокола TCP и использует порты 20 и 21 для передачи данных. Порт 20 используется для передачи данных в активном режиме, а порт 21 - для передачи данных в пассивном режиме.

Клиент/серверная модель приложений использует TCP/IP протокол для обеспечения надежного канала связи между клиентом и сервером. Протокол TCP (Transmission Control Protocol) гарантирует доставку данных в том порядке, в котором они были отправлены, и обеспечивает проверку наличия и правильности полученных данных. Протокол IP (Internet Protocol) обеспечивает передачу данных между устройствами в сети, используя IP-адреса для идентификации устройств и маршрутизации пакетов данных между ними.

Сокеты в клиент/серверной модели представляют программный интерфейс, который используется для обмена информацией между процессами на различных устройствах в сети. Сокеты обеспечивают установление соединения между клиентом и сервером через сеть и управление каналом связи между ними. Каждое TCP соединение может быть однозначно идентифицировано своими двумя конечными точками - IP-адресом и портом. Сокеты используются для обмена данными между клиентом и сервером через установленное соединение.

Один из примеров использования TCP/IP протокола и сокетов - передача электронной почты. Клиентское приложение (например, почтовый клиент) использует сокеты для установления соединения с сервером электронной почты через протокол SMTP (Simple Mail Transfer Protocol). После установления соединения клиент отправляет запрос на отправку электронного письма на сервер через сокеты, а сервер обрабатывает этот запрос и отправляет электронное письмо на указанный адрес через сеть.

## Стек протоколов TCP/IP и FTP-протокол.

Стек TCP/IP (стек Интернет) инкапсулирует данные верхних уровней протокола в пакеты нижних уровней. Он включает 4 уровня: прикладной, транспортный, межсетевой и уровень доступа к среде передачи.

Данные приложения

TCP-Сегмент

TCP-заголовок

Данные TCP-сегмента

IP-дейтаграмма

IP-заголовок

Данные IP-дейтаграммы

Ethernet-заголовок

Данные Ethernet-кадра

Кадр Ethernet

Рисунок 2.3 – Пример инкапсуляции пакетов в стеке TCP/IP.

Уровень доступа к среде передачи (IV) поддерживает все популярные стандарты физического и канального уровней.

Уровень межсетевого взаимодействия (III) отвечает за передачу дейтаграмм через различные локальные сети, линии спецсвязи и т.п. Он использует протокол IP в качестве основного протокола для передачи пакетов в составных сетях.

Уровень II, или транспортный протокол, обеспечивает сквозную доставку данных между двумя прикладными процессами и использует номер порта для идентификации процесса-отправителя и процесса-получателя. Заголовок пакета на транспортном уровне содержит номера портов отправителя и получателя, которые используются для передачи данных между соответствующими прикладными процессами. Формат заголовка зависит от используемого транспортного протокола.

Уровень I – прикладной уровень протокола TCP/IP обеспечивает множество сервисов. Приложение, желающее передать данные другому приложению, использует модуль транспортного уровня.

Протокол SNMP используется для управления сетями, обеспечивая передачу информации и регламентацию данных. Протокол FTP обеспечивает удаленный доступ к файлам и предоставляет интерактивную работу с удаленной машиной. Он использует протокол TCP для обеспечения надежной передачи данных и обеспечивает аутентификацию пользователей. Служба FTP предназначена для обмена файлами между машинами с разной архитектурой, использует архитектуру клиент/сервер и основана на протоколе FTP. Клиент отправляет запросы серверу, а сервер обрабатывает запросы на получение файла.

FTP - это служба передачи файлов, основанная на двух стандартах: URL - универсальном способе передачи ресурсов в сети, и протоколе FTP. Протокол FTP работает на прикладном уровне и использует TCP/IP для передачи данных, где TCP обеспечивает надежную передачу данных, а IP - доставку пакетов получателю. TCP разбивает информацию на пакеты, нумерует их и проверяет факт получения всех пакетов, после чего восстанавливает данные в нужном порядке.

Протокол FTP работает через два TCP соединения: управляющее для отправки команд серверу и получения ответов, и соединение данных для передачи файлов.

FTP-соединение инициирует интерпретатор протокола пользователя по каналу управления Telnet. Команды FTP передаются на сервер, а ответы сервера отправляются пользователю через канал управления. Команды FTP определяют параметры передачи данных и работу с файловыми системами. Для передачи данных сервер инициирует обмен данными в соответствии с согласованными параметрами сессии управления.

Для установления канала данных используется тот же хост, что и для канала управления. Он может использоваться для передачи и приема данных. При передаче данных между машинами канал управления должен быть открыт, иначе передача будет прервана.

Протокол FTP имеет несколько этапов взаимодействия клиента и сервера:

1. Аутентификация (ввод имени и пароля)
2. Выбор каталога
3. Определение режима обмена (блочный, поточный, двоичный, ASCII)
4. Выполнение команд обмена
5. Завершение процедуры.

В современных версиях FTP серверов порты от N>1024 могут быть назначены для канала данных сервером.

# ОБЗОР МЕТОДОВ И АЛГОРИТМОВ РЕШЕНИЯ ЗАДАЧИ

Проведем декомпозицию проекта на модули, необходимые для реализации поставленной задачи. Цель работы – создание FTP-клиента, способного работать с файловой системой, архивами. Составим модель многоуровневого представления разделив решение на иерархические уровни с набором запросов функций, с которым к одному уровню могут обращаться модули вышележащего уровня. Таким образом будут реализованы интерфейсы, выполняющиеся уровнем для вышележащего уровня.

Так же будут применен стек протоколов разных уровней, для межсетевого взаимодействия. В качестве стеков протоколов необходимым выбором является стек TCP/IP лежащий в основе работы сети Интернет.

Аутентификация клиента на сервере по схеме имя/пароль. Для отправки имени и пароля серверу применим команды USER и PASS соответственно.

Необходимые для реализации функций клиента модули:

1. Модуль подключения к FTP-серверу с выводом корневого каталога;
2. Модуль отключения от FTP-сервера;
3. Модуль загрузки файла на FTP-сервер;
4. Модуль скачивания файла с FTP-сервера;
5. Модуль удаления файлам с FTP-сервера;
6. Модуль архивации файла;
7. Модуль разархивации файла;
8. Модуль help описывающий команды для пользователя.

# ОБОСНОВАНИЕ МЕТОДОВ И АЛГОРИТМОВ РЕШЕНИЯ ЗАДАЧИ

# ОПИСАНИЕ ПРОГРАММЫ ДЛЯ ПРОГРАММИСТА

# ОПИСАНИЕ АЛГОРИТМОВ РЕШЕНИЯ ЗАДАЧИ

# РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

# ЗАКЛЮЧЕНИЕ

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Т. П. Куль, Операционные системы: учебное пособие / Т. П. Куль. – Минск: РИПО, 2019. – 312 с.,
2. В. Олифер, Компьютерные сети. Принципы, технологии, протоколы: Юбилейное издание / Олифер В. - Олифер Н. СПб.: Питер, 2020. — 1008 с.
3. Дубаков А.А. Сетевое программирование: учебное пособие / А.А. Дуба-ков – СПб: НИУ ИТМО, 2013. – 248 с.
4. Гайсина Л.Ф. FTP-клиент: Методические указания к лабораторному практикуму / Гайсина Л.Ф. – Оренбург: ГОУ УГУ, 2005 – 35с.
5. Герберт Шилдт, C++ Руководство для начинающих. Второе издание. / Герберт Шилдт, С.Н. Тригуб, Н.М. Ручко – М.: Издательский Дом Вильямс. 2005.