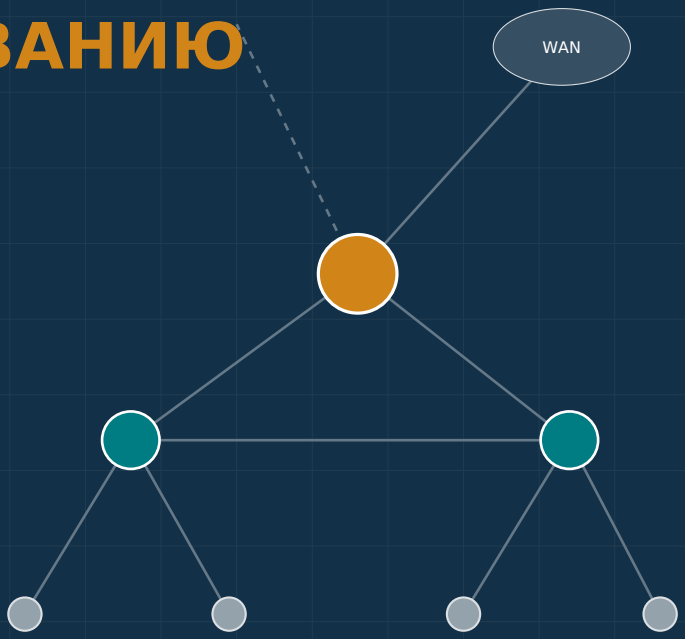
ПРАКТИЧЕСКОЕ ПОСОБИЕ

ПОСОБИЕ ПО СИСТЕМНОМУ И СЕТЕВОМУ АДМИНИСТРИРОВАНИЮ



Linux · Windows · SQL (SQLite)
Cisco Packet Tracer

```
$ ls -l ~/linux-lab
sqlite> SELECT name FROM products;
Router(config)# ip route 0.0.0.0 0.0.0.0
PS> Get-Service | Where Status -eq Running
```

КНИГА I

СИСТЕМНОЕ АДМИНИСТРИРОВАНИЕ

Базовые команды Linux
Linux и Windows: учётные записи, права, службы, журналы, копии
Базы данных и язык SQL

КНИГА II

СЕТЕВОЕ АДМИНИСТРИРОВАНИЕ

Адреса, VLAN, маршрутизация, службы
Безопасность, IPv6, беспроводные сети
Лабораторные работы и проекты

Русскоязычный курс для самостоятельного обучения

Редакция 4.0 · 12 октября 2026 г. · исходник XeLaTeX, векторные схемы, справочники

Как пользоваться этой книгой

Это учебное пособие для человека, который хочет стать администратором: научиться устанавливать, настраивать, проверять и чинить компьютеры, службы, данные и сети. Знаний и опыта от читателя не требуется: сначала объясняется, зачем нужно новое умение, затем показывается пример, и только потом даётся самостоятельное задание. Каждое задание можно проверить, а каждая работа начинается с вопроса «какую проблему мы решаем».

Пособие состоит из **двух книг**, которые можно изучать подряд или по очереди.



Устройство двух книг

Книга	Часть	Объём практики	Где выполняется
I. Системное администрирование	Базовые команды Linux	18 разделов, 170 заданий и итоговое задание	терминал Linux, полигон `linux-lab`
	Практикум системного администрирования	21 модуль (Linux и Windows)	полигоны для Linux и Windows PowerShell
	Базы данных и язык SQL (SQLite)	18 модулей, 590 заданий	консоль `sqlite3`, полигон на Python
II. Сетевое администрирование	Первые шаги, маршруты и коммутация, сетевые службы, динамические маршруты, IPv6 и беспроводные сети, обслуживание и проекты	40 лабораторных работ	Cisco Packet Tracer 8.2.1
	Системы в сети: проекты на стыке двух книг	5 проектов (I01–I05)	учебные виртуальные машины

Как устроена практика

В книге I практика — это *рабочая стажировка*. Ученик получает от руководителя вымышленной компании «Северный ветер» **заявки**: у каждой есть повод, требование и критерий приёмки. Заявки выполняются на учебном сервере (**полигоне**), который создаётся одной командой; результат принимает программа `proverka`, а если задание не принято, она называет причину, но не показывает решения. У каждого ученика свои данные, поэтому списать ответ нельзя.

В книге II практика — *лабораторная работа* в Cisco Packet Tracer. Каждая работа построена одинаково: *какую проблему решает работа* (зачем она нужна в реальной сети) → *что вы научитесь делать* → *стенд и адреса* → *порядок работы* по шагам → *проверка результата* → *управляемая неисправность* (одна поломка, чтобы научиться диагностике) → *что сдать и объяснить* → *самостоятельное усложнение*. Пока вы не поняли, зачем нужна работа, не переходите к шагам: именно эта проблема определяет, что вы проверяете в конце.

Маршруты обучения

Маршрут	Что изучать	Результат
Системный администратор	Книга I целиком: базовые команды Linux, модули 1–14, темы SQL 1–17	Администрирование Linux и Windows, работа с данными и базами
Сетевой администратор	Книга II целиком; из книги I — базовые команды Linux и модули 1–5	Проектирование, настройка и диагностика сети организации
Полный курс	Книга I, затем книга II, затем проекты I01–I05	Администратор небольшой организации: системы, данные и сеть
Быстрый старт	Базовые команды Linux; модули 1–5, 8, 10; темы SQL 1–8; лабораторные 01–09	Терминал, учётные записи, доступ, первые запросы и первая работающая сеть
Повторение	Модули 3–4, 7, 10–14; темы SQL 9–17; лабораторные 37–40; проекты I02–I05	Практика доступа, служб, баз данных, диагностики и восстановления

С чтением теории, повторением и отчётами полный курс рассчитан ориентировочно на 250–330 часов; на раздел, модуль или обычную лабораторную отводится 1–4 часа, на проект — несколько занятий. Это оценка, а не обещание пройти тему за фиксированное время.

Правило одного изменения

Стендом называется небольшая учебная система, на которой проводится опыт. После сборки и проверки её полезно сохранить. Для опыта с неисправностью делают отдельную копию и меняют только одно условие: например, отсоединяют один кабель или забирают одно право. Так можно связать наблюдаемое изменение с конкретной причиной. Если изменить сразу несколько условий, причина результата станет неясной. В книге I роль копии играет полигон: модуль можно пересоздать. В книге II имена файлов `L01-working.pkt` и `L01-fault.pkt` означают «лабораторная 01, исправная сеть» и «лабораторная 01, опыт с неисправностью».

Условные обозначения

Команды Linux, PowerShell, CMD, SQL и Cisco IOS подписаны отдельно: одинаковое окно терминала не означает одинаковый язык команд. Команды набраны моноширинным шрифтом. Английские слова в тексте помогают найти нужный элемент программы: рядом приводятся перевод и назначение; например, *File / Save As* означает пункт «Сохранить как» в меню «Файл». Короткие имена устройств на сетевых схемах (PC-A, S1, R1) — подписи к рисунку: их назначение объясняется в начале книги II. В системных заданиях anna, boris и другие имена обозначают сотрудников компании «Северный ветер»; в объединяющих проектах встречаются ещё student и labanna.

Что понадобится

Книга I: компьютер с Linux (Ubuntu 24.04) или Windows 11 с Windows PowerShell 5.1; Python 3; для SQL — консоль `sqlite3`. Вся практика выполняется от имени обычного пользователя. **Книга II:** Cisco Packet Tracer 8.2.1 (бесплатная учебная программа Cisco Networking Academy). **Проекты I01–I05:** две учебные виртуальные машины (Linux и Windows) и гипервизор.

Что именно входит в комплект

Пособие содержит объяснения, команды, сценарии, адресные планы, критерии проверки и проекты. Рядом с книгой поставляются полигоны (`setup.sh`, `setup.ps1`, `setup.py`) и самопроверки (`selftest`). Готовые образы виртуальных машин, файлы Packet Tracer `.pkt` и задания `.pka` в комплект не входят. Сборка LaTeX проверяет документ; выполнение команд на конкретных версиях ОС и моделях проверяется отдельно, границы проверки указаны в приложениях к каждой книге.

Различия версий

Рисунки и «эталонные выводы» в книге II сняты в Packet Tracer 8.2.1 (Windows, английский интерфейс). В других выпусках положение кнопок может отличаться; адреса, имена команд и смысл вывода при этом сохраняются. Пользуйтесь рисунками как ориентиром, а не как эталоном пикселей. В группе удобно использовать одну версию Packet Tracer: файл из нового выпуска может не открыться в старом.

Оглавление

Как пользоваться этой книгой i

КНИГА I. Системное администрирование 1

О книге I: системное администрирование 2

I Базовые команды Linux 4

О первой части: практикум «Базовые команды Linux» 5

Часть 1. Навигация по файловой системе и создание объектов 6

Раздел 1. Команда pwd 9
1.1. Теоретические сведения . . . 9
1.2. Задания . . . 9

Раздел 2. Команда cd 11
2.1. Теоретические сведения . . . 11
2.2. Задания . . . 11

Раздел 3. Команда ls 13
3.1. Теоретические сведения . . . 13
3.2. Задания . . . 14

Раздел 4. Команда mkdir 15
4.1. Теоретические сведения . . . 15
4.2. Задания . . . 15

Раздел 5. Команда touch 17
5.1. Теоретические сведения . . . 17
5.2. Задания . . . 17

Часть 2. Копирование, перемещение, удаление и просмотр файлов 19

Раздел 6. Команда cp 20
6.1. Теоретические сведения . . . 20
6.2. Задания . . . 20

Раздел 7. Команда mv 22
7.1. Теоретические сведения . . . 22
7.2. Задания . . . 22

Раздел 8. Команда rm 24
8.1. Теоретические сведения . . . 24
8.2. Задания . . . 24

Раздел 9. Команда cat 26
9.1. Теоретические сведения . . . 26

9.2. Задания . . . 26

Раздел 10. Команда less **28**
10.1. Теоретические сведения . . . 28
10.2. Задания . . . 28

Часть 3. Обработка текста, поиск, права доступа и справочная система **30**

Раздел 11. Команда head **31**
11.1. Теоретические сведения . . . 31
11.2. Задания . . . 31

Раздел 12. Команда tail **32**
12.1. Теоретические сведения . . . 32
12.2. Задания . . . 32

Раздел 13. Команда echo **33**
13.1. Теоретические сведения . . . 33
13.2. Задания . . . 33

Раздел 14. Команда grep **35**
14.1. Теоретические сведения . . . 35
14.2. Задания . . . 35

Раздел 15. Команда find **37**
15.1. Теоретические сведения . . . 37
15.2. Задания . . . 37

Раздел 16. Команда chmod **39**
16.1. Теоретические сведения . . . 39
16.2. Задания . . . 40

Раздел 17. Команда man **41**
17.1. Теоретические сведения . . . 41
17.2. Задания . . . 41

Раздел 18. Итоговое задание **43**

II Практикум системного администрирования: Linux и Windows **44**

Как устроен практикум **45**

Подготовка: полигон, терминал и первая заявка **49**

1 Модуль 1. Сервер и его паспорт **52**

2 Модуль 2. Учётные записи и группы **56**

3 Модуль 3. Права доступа: chmod, umask, setgid, ACL **60**

4 Модуль 4. sudo и минимальные полномочия **64**

5 Модуль 5. Процессы и ресурсы **68**

6 Модуль 6. Пакеты и репозитории 72

7 Модуль 7. Службы и автозапуск (systemd) 76

8 Модуль 8. Журналы и расследование 81

9 Модуль 9. Диски, место и файловые системы 86

10 Модуль 10. Резервное копирование и восстановление 90

11 Модуль 11. Автоматизация: сценарии и cron 95

12 Модуль 12. Сеть, SSH и межсетевой экран 100

13 Модуль 13. Аудит безопасности 105

14 Модуль 14. Итоговый инцидент: «сервер тормозит» 110

15 Модуль 15. Сетевые настройки хоста 115

16 Модуль 16. Хранилище: разделы, LVM, монтирование, swap 119

17 Модуль 17. Мониторинг и производительность 123

18 Модуль 18. Хеши, шифрование и сертификаты 127

19 Модуль 19. Веб-сервер: конфигурация и журнал доступа 130

20 Модуль 20. Конфигурации под контролем версий 133

21 Модуль 21. Копии: ротация, инкременты и восстановление 136

III Базы данных и язык SQL (SQLite) 140

О части: базы данных и язык SQL 141

Тема 1. Первое знакомство: база данных в одном файле 144

Раздел 1. Команда sqlite3 — запуск консоли 146
1.1. Теоретические сведения 146
1.2. Задания 146

Раздел 2. Команда .tables, .schema — устройство базы 148
2.1. Теоретические сведения 148
2.2. Задания 149

Раздел 3. Команда SELECT … FROM — первый запрос 150
3.1. Теоретические сведения 150
3.2. Задания 150

Раздел 4. Команда .headers, .mode — вид вывода 152
4.1. Теоретические сведения 152
4.2. Задания 152

Раздел 5. Команда .output — вывод в файл 154
5.1. Теоретические сведения 154

5.2. Задания . . . 154

Тема 2. Выбор столбцов, выражения и сортировка 156

Раздел 6. Команда SELECT … FROM — выбор столбцов 158
6.1. Теоретические сведения . . . 158
6.2. Задания . . . 158

Раздел 7. Команда AS — выражения и псевдонимы 160
7.1. Теоретические сведения . . . 160
7.2. Задания . . . 160

Раздел 8. Команда ORDER BY — сортировка 162
8.1. Теоретические сведения . . . 162
8.2. Задания . . . 162

Раздел 9. Команда DISTINCT — без повторов 164
9.1. Теоретические сведения . . . 164
9.2. Задания . . . 164

Раздел 10. Команда LIMIT, OFFSET — часть результата 166
10.1. Теоретические сведения . . . 166
10.2. Задания . . . 166

Тема 3. Условия отбора: WHERE 168

Раздел 11. Команда WHERE — отбор строк и сравнения 170
11.1. Теоретические сведения . . . 170
11.2. Задания . . . 171

Раздел 12. Команда AND, OR, NOT — логические операции 172
12.1. Теоретические сведения . . . 172
12.2. Задания . . . 173

Раздел 13. Команда IN — проверка по списку 174
13.1. Теоретические сведения . . . 174
13.2. Задания . . . 174

Раздел 14. Команда BETWEEN — диапазон 176
14.1. Теоретические сведения . . . 176
14.2. Задания . . . 176

Раздел 15. Команда LIKE — поиск по шаблону 178
15.1. Теоретические сведения . . . 178
15.2. Задания . . . 179

Раздел 16. Команда IS NULL — пустое значение 180
16.1. Теоретические сведения . . . 180
16.2. Задания . . . 180

Тема 4. Функции, выражения CASE и работа с NULL 182

Раздел 17. Команда LENGTH, SUBSTR, INSTR — разбор строки 184
17.1. Теоретические сведения . . . 184
17.2. Задания . . . 185

Раздел 18. Команда REPLACE, TRIM, UPPER, LOWER — преобразование строки **186**
18.1. Теоретические сведения . . . 186
18.2. Задания . . . 186

Раздел 19. Числовые функции **188**
19.1. Теоретические сведения . . . 188
19.2. Задания . . . 189

Раздел 20. Функции даты и времени **190**
20.1. Теоретические сведения . . . 190
20.2. Задания . . . 191

Раздел 21. Команда CASE — условное выражение **192**
21.1. Теоретические сведения . . . 192
21.2. Задания . . . 193

Раздел 22. Команда COALESCE, NULLIF, IFNULL — работа с NULL **194**
22.1. Теоретические сведения . . . 194
22.2. Задания . . . 194

Раздел 23. Команда CAST и TYPEOF — типы значений **196**
23.1. Теоретические сведения . . . 196
23.2. Задания . . . 196

Тема 5. Агрегатные функции и группировка **198**

Раздел 24. Команда COUNT — подсчёт строк **200**
24.1. Теоретические сведения . . . 200
24.2. Задания . . . 200

Раздел 25. Команда SUM и AVG — сумма и среднее **202**
25.1. Теоретические сведения . . . 202
25.2. Задания . . . 202

Раздел 26. MIN, MAX и GROUP_CONCAT **204**
26.1. Теоретические сведения . . . 204
26.2. Задания . . . 204

Раздел 27. Команда GROUP BY — группировка **206**
27.1. Теоретические сведения . . . 206
27.2. Задания . . . 206

Раздел 28. Команда HAVING — условие на группы **208**
28.1. Теоретические сведения . . . 208
28.2. Задания . . . 208

Тема 6. Соединение таблиц: JOIN **210**

Раздел 29. Ключи и связи таблиц **212**
29.1. Теоретические сведения . . . 212
29.2. Задания . . . 213

Раздел 30. Команда JOIN … ON — внутреннее соединение **214**
30.1. Теоретические сведения . . . 214

30.2. Задания . . . 215

Раздел 31. Цепочка из трёх и более таблиц 216
31.1. Теоретические сведения . . . 216
31.2. Задания . . . 217

Раздел 32. Команда LEFT JOIN — сохранение строк левой таблицы 218
32.1. Теоретические сведения . . . 218
32.2. Задания . . . 218

Раздел 33. Поиск строк без пары: LEFT JOIN … IS NULL 220
33.1. Теоретические сведения . . . 220
33.2. Задания . . . 220

Раздел 34. Самосоединение таблицы 222
34.1. Теоретические сведения . . . 222
34.2. Задания . . . 223

Раздел 35. Агрегаты по соединению 224
35.1. Теоретические сведения . . . 224
35.2. Задания . . . 224

Тема 7. Подзапросы и операции над множествами 226

Раздел 36. Подзапрос, возвращающий одно значение 228
36.1. Теоретические сведения . . . 228
36.2. Задания . . . 229

Раздел 37. Команда IN, NOT IN — значение в списке из подзапроса 230
37.1. Теоретические сведения . . . 230
37.2. Задания . . . 231

Раздел 38. Команда EXISTS — есть ли связанная строка 232
38.1. Теоретические сведения . . . 232
38.2. Задания . . . 233

Раздел 39. Подзапрос в списке столбцов 234
39.1. Теоретические сведения . . . 234
39.2. Задания . . . 235

Раздел 40. Подзапрос в FROM 236
40.1. Теоретические сведения . . . 236
40.2. Задания . . . 237

Раздел 41. Команда UNION, UNION ALL — объединение результатов 238
41.1. Теоретические сведения . . . 238
41.2. Задания . . . 239

Раздел 42. Команда INTERSECT, EXCEPT — пересечение и разность 240
42.1. Теоретические сведения . . . 240
42.2. Задания . . . 241

Тема 8. CTE, рекурсия и оконные функции 243

Раздел 43. Команда WITH — общее табличное выражение (CTE) 245
43.1. Теоретические сведения . . . 245
43.2. Задания . . . 246

Раздел 44. Команда WITH RECURSIVE — рекурсивные запросы 247
44.1. Теоретические сведения . . . 247
44.2. Задания . . . 248

Раздел 45. Команда OVER, PARTITION BY — оконные агрегаты 250
45.1. Теоретические сведения . . . 250
45.2. Задания . . . 251

Раздел 46. Команда ROW_NUMBER, RANK, DENSE_RANK — нумерация и ранги 252
46.1. Теоретические сведения . . . 252
46.2. Задания . . . 253

Раздел 47. Нарастающий итог и рамка окна 254
47.1. Теоретические сведения . . . 254
47.2. Задания . . . 255

Раздел 48. Команда LAG, LEAD — соседние строки 256
48.1. Теоретические сведения . . . 256
48.2. Задания . . . 257

Раздел 49. Отбор по результату оконной функции 258
49.1. Теоретические сведения . . . 258
49.2. Задания . . . 259

Тема 9. Изменение данных: INSERT, UPDATE, DELETE 261

Раздел 50. Изменение данных: INSERT, UPDATE, DELETE 263
50.1. Теоретические сведения . . . 263
50.2. Задания . . . 267

Тема 10. Создание таблиц и ограничения 268

Раздел 51. Создание таблиц и ограничения 270
51.1. Теоретические сведения . . . 270
51.2. Задания . . . 277

Тема 11. Проектирование схемы и нормализация 278

Раздел 52. Проектирование схемы и нормализация 280
52.1. Теоретические сведения . . . 280
52.2. Задания . . . 285

Тема 12. Индексы и планы запросов 287

Раздел 53. Индексы и планы запросов 289
53.1. Теоретические сведения . . . 289
53.2. Задания . . . 293

Тема 13. Представления и триггеры 295

Раздел 54. Представления и триггеры **297**
54.1. Теоретические сведения . . . 297
54.2. Задания . . . 301

Тема 14. Транзакции, целостность и PRAGMA **303**

Раздел 55. Транзакции, целостность и PRAGMA **305**
55.1. Теоретические сведения . . . 305
55.2. Задания . . . 310

Тема 15. Резервные копии, обмен данными и обслуживание базы **312**

Раздел 56. Резервные копии, обмен данными и обслуживание базы **314**
56.1. Теоретические сведения . . . 314
56.2. Задания . . . 320

Тема 16. SQL в сценариях, безопасность и анализ журналов **322**

Раздел 57. SQL в сценариях, безопасность и анализ журналов **324**
57.1. Теоретические сведения . . . 324
57.2. Задания . . . 329

Тема 17. Итоговый проект: заявки службы поддержки **332**

Раздел 58. Итоговый проект: заявки службы поддержки **334**
58.1. Теоретические сведения . . . 334
58.2. Задания . . . 338

Тема 18. JSON, полнотекстовый поиск и подключение баз **340**

Раздел 59. JSON, полнотекстовый поиск и подключение баз **342**
59.1. Теоретические сведения . . . 342
59.2. Задания . . . 347

КНИГА II. Сетевое администрирование 349

О книге II: сетевое администрирование **350**

I Первые шаги и устройство сети 352

22 Что такое сеть и зачем нужен Packet Tracer **353**
22.1 Начнём с обычной ситуации . . . 353
22.2 Какие устройства участвуют в сети . . . 353
22.3 Что именно делает Packet Tracer . . . 354
22.4 Первый взгляд на окно программы . . . 355
22.5 PC-PT, 2960 и PC-A: три разных вида названий . . . 356
22.6 Порт и кабель: как появляются соединения . . . 357
22.7 Разобранный пример: одно рабочее место . . . 357
01. Первая схема: повторяем пример и добавляем второй компьютер . . . 361

23 Пакеты, адреса и путь сообщения **364**
23.1 Зачем компьютеру адрес . . . 364
23.2 Маска: как отличить соседей от другой сети . . . 364

23.3 Что означают остальные поля настройки . . . 365
23.4 Проверочное сообщение: что делает ping . . . 366
02. Первая локальная сеть и ping . . . 367
23.5 Что находится внутри обмена . . . 369
23.6 Как читать пошаговую симуляцию . . . 370
03. Пошаговое наблюдение ARP и ICMP . . . 371

24 IPv4: от маски до адресного плана 376
24.1 Бит, октет и двоичная запись . . . 376
24.2 Префикс и диапазон . . . 376
24.3 Какие адреса использовать в учебной сети . . . 377
24.4 VLSM и резерв роста . . . 377
04. Расчёт подсетей и проверка маски . . . 378
05. VLSM для небольшой организации . . . 379

25 CLI и базовая конфигурация IOS 381
25.1 Режимы и помощь . . . 383
25.2 Состояние и сохранение . . . 383
06. Первичная настройка и сохранение . . . 384
07. Управление коммутатором через SSH . . . 386
08. Маршрутизатор между двумя LAN . . . 388

II Маршруты и коммутация 390

26 Статическая маршрутизация 391
26.1 Таблица маршрутов и обратный путь . . . 391
09. Три маршрутизатора и статические маршруты . . . 391
10. Резервный путь и floating static route . . . 394

27 VLAN, trunk и маршрутизация между отделами 397
27.1 Границы широковещания . . . 397
11. Два отдела на одном коммутаторе . . . 397
12. Trunk между коммутаторами . . . 399
13. Router-on-a-stick . . . 401
14. Маршрутизация на L3-коммутаторе . . . 404

28 Петли, резервирование и защита доступа 407
28.1 Почему два кабеля не всегда лучше одного . . . 407
15. STP: дерево и переключение после отказа . . . 407
16. EtherChannel с LACP . . . 410
17. Port Security, PortFast и BPDU Guard . . . 412
18. Инвентаризация через CDP и описание портов . . . 414

III Сетевые службы и политики доступа 416

29 Автоматическая настройка и приложения 417
29.1 DHCP и relay . . . 417
29.2 DNS, HTTP и транспорт . . . 417
19. DHCP на маршрутизаторе . . . 417
20. DHCP-сервер в другой подсети . . . 419
21. DNS, веб-сервер и TCP в Simulation . . . 422

22. FTP и учебная электронная почта . . . 425

30 ACL: формулируем и проверяем политику 430
30.1 Правило первого совпадения . . . 430
23. Standard ACL и место применения . . . 430
24. Extended ACL для гостевого доступа . . . 432

31 NAT и учебный провайдер 434
31.1 Адреса до и после трансляции . . . 434
25. Static NAT и публикация сервера . . . 434
26. PAT: несколько клиентов за одним адресом . . . 436

IV Динамические маршруты, IPv6 и беспроводные сети 438

32 OSPF и альтернативные протоколы 439
32.1 Соседство, база топологии и маршрут . . . 439
32.2 Почему маршрутизатор не стал соседом . . . 439
27. Single-area OSPF . . . 439
28. Стоимость OSPF и отказ удалённой линии . . . 441
29. Две области OSPF . . . 442
30. Сравнение EIGRP и RIPv2 . . . 444

33 IPv6 и параллельная работа стеков 446
33.1 Адреса и Neighbor Discovery . . . 446
31. IPv6: две LAN и SLAAC . . . 446
32. Статические маршруты IPv6 . . . 449

34 Wi-Fi и Интернет вещей 451
34.1 Ассоциация, адрес и приложение . . . 451
33. Домашняя WLAN с WPA2 Personal . . . 451
34. Централизованная WLAN через WLC . . . 456
35. IoT: регистрация и простое правило . . . 459

V Обслуживание, диагностика и проекты 463

35 Наблюдаемость и восстановление 464
35.1 Что сеть должна сообщать администратору . . . 464
36. NTP, Syslog и резервная копия TFTP . . . 464
37. Практикум поиска неисправностей . . . 466

36 Комплексные проекты 469
38. Кампус: три отдела и серверная . . . 470
39. Головной офис, филиал и учебный ISP . . . 473
40. Итоговый проект: сеть учебного центра . . . 475

VI Отказоустойчивость, защита и мониторинг 477

37 Расширенные работы 41–48 478
37.1 Как читать эти работы . . . 478
41. Резервный шлюз: HSRP . . . 479
42. Защита от чужого DHCP-сервера . . . 482

43. Быстрое остовное дерево и корневой мост . . . 485
44. Мониторинг по SNMP . . . 487
45. Туннель GRE и защита IPsec . . . 489
46. Централизованная проверка прав: AAA . . . 492
47. IP-телефония: голосовая VLAN и приоритет . . . 494
48. Обмен маршрутами между организациями: основы BGP . . . 497

VII Системы в сети: проекты на стыке двух книг 499

38 От модели сети к настоящей службе 500
38.1 Два стенда и одна проверяемая задача . . . 500
I01. Две ОС в одной учебной сети . . . 500
38.2 Samba: сетевой ресурс и права на данные . . . 502
I02. Linux-файловый сервер и Windows-клиент . . . 502
38.3 Веб-служба: процесс, порт, конфигурация и файл . . . 503
I03. Linux: сайт, доступ и диагностика отказа . . . 504

39 Восстановление и передача инфраструктуры 506
39.1 Сохранить данные вместе с политикой доступа . . . 506
I04. Восстановление общего ресурса и прав . . . 506
I05. Итог: инфраструктура учебного отдела . . . 507

ПРИЛОЖЕНИЯ. Книга I. Системное администрирование 509

A Книга преподавателя, модель и проверка полигонов 510
A.1 Книга преподавателя: практикум «Базовые команды Linux» . . . 510
A.2 Книга преподавателя: эталонные решения полигонов . . . 522
A.3 Учебные команды полигона и расхождения с настоящими . . . 575
A.4 Как проверен полигон . . . 577
A.5 Книга преподавателя: практикум по SQL . . . 578

B Справочник команд системного администратора 663
B.1 Один вопрос – несколько средств проверки . . . 663
B.2 Минимальный отчёт и оценивание . . . 664
B.3 Что изучать после базового курса . . . 664

ПРИЛОЖЕНИЯ. Книга II. Сетевое администрирование 665

C Карманный справочник и безопасный порядок действий 666
C.1 Среда передачи и кабели . . . 666
C.2 Состояния интерфейса . . . 666
C.3 Команды по задачам . . . 667
C.4 Часто используемые службы . . . 667
C.5 Дополнительный WAN-эксперимент: Serial и PPP . . . 668

D Шаблоны отчётов, план занятий и оценивание 669
D.1 Отчёт по лабораторной . . . 669
D.2 Журнал поиска ошибки . . . 669
D.3 Пример учебного календаря . . . 670
D.4 Рубрика оценивания . . . 670

E Контрольные расчёты и разборы **672**
E.1 IPv4 без калькулятора 672
E.2 Ключ к вопросам первых работ 672
E.3 Ключ к коммутации 673
E.4 Ключ к службам и безопасности 673
E.5 Ключ к расширенным работам 673
E.6 Ориентир конфигурации MLS1 для проекта 38 674
E.7 Один адресный вариант для проекта 40 675

F Словарь сетевых понятий **676**

G Источники и границы применимости (обе книги) **679**
G.1 Как читать документацию вместе с учебным стендом 679
G.2 Что проверено в Packet Tracer 8.2.1 679

Следующий шаг **684**

КНИГА I

Системное администрирование

Как устроены, настраиваются, защищаются и восстанавливаются компьютеры, службы и данные. Базовые команды Linux, администрирование Linux и Windows, базы данных и язык SQL.

О книге I: системное администрирование

Что делает системный администратор

Системный администратор отвечает за то, чтобы компьютеры и программы организации работали, данные не терялись, а доступ имели только те, кому он нужен. На практике это пять повторяющихся задач: **подготовить** систему (установить, настроить, создать пользователей), **защитить** её (права, обновления, минимальные полномочия), **наблюдать** за ней (процессы, журналы, место на диске), **чинить** (найти причину сбоя и устранить её) и **восстанавливать** (копии, проверка восстановления). Все четыре части этой книги отвечают на один вопрос: «что делать, когда нужно …и как убедиться, что получилось».

Три части книги

Часть	Что вы научитесь делать
Базовые команды Linux	Работать в терминале: перемещаться по файловой системе, создавать, копировать, перемещать и удалять файлы, искать, менять права, пользоваться справкой. Это основа для всего остального
Практикум системного администрирования	Проверять и «паспортизировать» сервер, управлять учётными записями и правами, настраивать `sudo`, находить и останавливать процессы, ставить программы, управлять службами, разбирать журналы, чистить диск, делать и восстанавливать копии, автоматизировать работу, защищать сеть хоста, проводить аудит и расследовать инцидент — и в Linux, и в Windows
Базы данных и язык SQL	Читать и изменять данные запросами, проектировать таблицы, ускорять запросы индексами, защищать данные ограничениями, триггерами и транзакциями, делать и восстанавливать копии, не допускать SQL-инъекций

Как работать с книгой I

Все три части построены одинаково: теория и примеры, затем заявки с самопроверкой. Порядок такой: **(1)** прочитайте теорию и разберите примеры; **(2)** создайте полигон одной командой (`bash setup.sh`, `setup.ps1` или `python3 setup.py`); **(3)** выполните заявки по порядку; **(4)** после каждой проверьте результат своим способом и только потом запустите `proverka`; **(5)** при ошибке прочитайте причину, при необходимости возьмите подсказку (`podskazka`); **(6)** ответьте на вопросы для размышления. Если полигон испорчен, раздел или модуль пересоздаётся одной командой, результаты других модулей не страдают.

Рекомендуемый порядок: базовые команды Linux → модули системного администрирования 1–14 → SQL. Базовые команды нужны всем; SQL можно начинать после модуля 5. Книгу II (сети) удобно читать после модулей 1–5.

Правило безопасности

Ломать можно только каталог полигона. Практика выполняется от имени обычного пользователя, не root и не администратора. Не вставляйте в настоящую систему команды, смысл которых вам неясен: в книге всё объясняется, а модель учебных команд описана в приложениях к книге I.

Часть I

Базовые команды Linux

О первой части: практикум «Базовые команды Linux»

Книга начинается с практикума по командной строке Linux. Он учит работать с файловой системой: перемещаться по каталогам, создавать, копировать, перемещать и удалять файлы, просматривать и искать данные, менять права доступа и пользоваться справочной системой. Эти приёмы нужны во всех следующих частях: без них нельзя ни разбирать журналы, ни править настройки служб, ни составлять сценарии.

Практикум состоит из 18 разделов и содержит 170 заданий и итоговое задание из семи шагов. Его материалы приводятся здесь без изменений: теоретические сведения и задания разделов 1–18 (три части), а в приложении — методические указания и эталонные решения для преподавателя. Раздел практикума ничем не связан с «модулями» следующей части книги: у них разные полигоны, разные каталоги и разная нумерация.

Полигон и проверка

Рабочие каталоги и учебные файлы создаёт сценарий `setup.sh` (в комплекте книги: каталог `sysadmin-lab/linux-basics`). Его запускают один раз: `bash setup.sh`. Он создаёт каталог `~/linux-lab` и программу проверки `proverka`. Задания выполняются от имени обычного пользователя; результат каждого задания — файл, каталог или файл ответа в `otvety`, который проверяет `~/linux-lab/proverka`. Подробности — в общих указаниях ниже.

После этого практикума переходите к практикуму системного администрирования (модули 1–14), а затем — к работе с базами данных и к сетям.

Часть 1. Навигация по файловой системе и создание объектов

Команды: `pwd`, `cd`, `ls`, `mkdir`, `touch`.

Общие указания

Цель практикума

Освоить базовые команды командной оболочки Linux для навигации по файловой системе, работы с файлами и каталогами, просмотра и поиска данных, а также управления правами доступа.

Структура практикума

Часть	Разделы	Команды
1. Навигация по файловой системе и создание объектов	1–5	`pwd`, `cd`, `ls`, `mkdir`, `touch`
2. Копирование, перемещение, удаление и просмотр файлов	6–10	`cp`, `mv`, `rm`, `cat`, `less`
3. Обработка текста, поиск, права доступа и справочная система	11–18	`head`, `tail`, `echo`, `grep`, `find`, `chmod`, `man`, итоговое задание

Каждый раздел состоит из теоретических сведений и блока из десяти заданий.

Требования к рабочей среде

1. Операционная система Linux (или WSL в Windows) с командной оболочкой bash.
2. Работа выполняется от имени обычного пользователя, а не суперпользователя (root).
3. Терминал использует кодировку UTF-8.

Подготовка учебного полигона

Учебные файлы создаются скриптом `setup.sh`, который выдаёт преподаватель. Скрипт запускается **один раз** перед началом практикума:

```
bash setup.sh
```

Скрипт создаёт каталог `~/linux-lab`, а в нём — отдельный рабочий каталог для каждого раздела:

```
~/linux-lab/
├── 01_pwd/      ├── 07_mv/       ├── 13_echo/
├── 02_cd/       ├── 08_rm/       ├── 14_grep/
├── 03_ls/       ├── 09_cat/      ├── 15_find/
├── 04_mkdir/    ├── 10_less/     ├── 16_chmod/
├── 05_touch/    ├── 11_head/     ├── 17_man/
```

```
├── 06_cp/      ├── 12_tail/    ├── 18_final/
                                └── proverka    программа проверки заданий
```

Все задания раздела выполняются в его рабочем каталоге, поэтому изменения, сделанные в одном разделе, не влияют на другие.

Если содержимое рабочего каталога было повреждено, его можно восстановить в исходном виде, указав номер раздела:

```
bash setup.sh 7        # пересоздать только каталог 07_mv
```

Если в момент восстановления вы находились внутри этого каталога, после восстановления выполните `cd` в него повторно.

Состав учебного набора

Каждый рабочий каталог содержит одинаковый набор файлов:

```
NN_имя/
├── .hidden_note          скрытый файл
├── .secret/hint.txt      скрытый каталог с файлом
├── archive/
│   ├── 2024/             report.bak, notes.bak, photos.bak, budget.bak
│   └── 2025/             report.txt, notes.txt, photos.txt, budget.txt
├── cities.txt            список из 12 городов
├── config.conf           конфигурационный файл с комментариями
├── hello.sh              сценарий оболочки
├── kitchen/
│   ├── fridge.txt        пустой файл
│   └── recipes/          borsch.txt, olivie.txt, pancakes.txt
├── logs/
│   ├── app.log           журнал событий (40 записей)
│   └── old.log
├── numbers.txt           числа от 1 до 100
├── projects/site/        index.html, css/style.css, img/ (logo.png, cat.jpg,
Cat2.JPG)
├── secret.txt
├── space/                planets/ (пустой каталог), planets.txt, telescope.raw,
photo_moon.jpg
├── otvety/               каталог для сохранения ответов
├── story.txt             короткий текст
└── zoo/
    ├── animals.txt       записи вида «животное;класс;континент»
    └── mammals/, birds/, reptiles/   (пустые каталоги)
```

Порядок выполнения

1. Перед началом раздела перейдите в его рабочий каталог командой `cd`.
2. Выполняйте задания по порядку: некоторые задания опираются на результат предыдущих.
3. Результатом каждого задания является изменение файловой системы: созданный или перемещённый файл, каталог, изменённые права доступа либо файл с сохранённым выводом команды.

Сохранение результата в файл

Если в задании требуется сохранить результат, вывод команды записывается в файл с помощью символа `>`:

```
pwd > otvety/1          # вывод команды pwd записывается в файл otvety/1
ls -a > otvety/3        # список файлов записывается в файл otvety/3
echo 5 > otvety/7       # в файл otvety/7 записывается число 5
```

Файлы ответов хранятся в каталоге `otvety` рабочего каталога раздела и называются по номеру задания. Если текущий каталог был изменён командой `cd`, путь к файлу ответа указывается относительно нового текущего каталога (`pwd > ../../otvety/2`) или абсолютно (`pwd > ~/linux-lab/01_pwd/otvety/2`). Повторное выполнение команды перезаписывает файл. Содержимое файла ответа можно просмотреть командой `cat otvety/1`.

Подробно перенаправление вывода рассматривается в разделах 9 и 13.

Проверка выполнения

Выполнение заданий проверяется программой `proverka`:

```
~/linux-lab/proverka 1      # подробная проверка раздела 1
~/linux-lab/proverka        # сводка по всем разделам
```

Для каждого задания программа сообщает «выполнено» или «не выполнено». Если задание не выполнено, программа указывает причину: отсутствует файл с ответом либо результат не соответствует условию. Ошибку можно исправить и повторить проверку.

Раздел 1. Команда pwd

1.1. Теоретические сведения

Назначение. Команда pwd (print working directory) выводит абсолютный путь к текущему рабочему каталогу. В каждый момент времени командная оболочка связана с одним каталогом файловой системы, и относительные пути отсчитываются от него.

Синтаксис:

```
pwd [-L | -P]
```

Параметры:

Параметр	Описание
-L	вывести логический путь, в том числе содержащий символические ссылки (по умолчанию)
-P	вывести физический путь, в котором символические ссылки заменены реальными каталогами

Связанные понятия:

- **Абсолютный путь** начинается от корня файловой системы /, например /home/anna/docs.
- **Относительный путь** отсчитывается от текущего каталога, например docs/plan.txt.
- Символ ~ обозначает домашний каталог пользователя.
- Переменная окружения $PWD содержит тот же путь, который выводит pwd.

Приглашение командной строки (например, anna@pc:~/docs$) также отображает текущий каталог, но в сокращённом виде.

Пример:

```
$ pwd
/home/anna
```

1.2. Задания

Рабочий каталог: ~/linux-lab/01_pwd. Ответы сохраняются в каталог ~/linux-lab/01_pwd/otvety.

1. Перейдите в рабочий каталог раздела. Сохраните путь к нему в файл otvety/1.
2. Перейдите в каталог zoo/birds. Сохраните его путь в файл ~/linux-lab/01_pwd/otvety/2.
3. Перейдите в корневой каталог файловой системы. Сохраните его путь в файл otvety/3 рабочего каталога раздела (используйте абсолютный путь к файлу).

4. Перейдите в домашний каталог командой `cd` без аргументов. Сохраните его путь в файл `otvety/4` рабочего каталога раздела.
5. Выполните последовательно `cd /usr/share` и `cd ../lib`. Сохраните путь текущего каталога в файл `otvety/5` рабочего каталога раздела.
6. Перейдите в каталог `space/planets` рабочего каталога. Сохраните в файл `otvety/6` строку вида «Я нахожусь в <путь>», сформированную командой `echo "Я нахожусь в $(pwd)"`.
7. В рабочем каталоге создайте символическую ссылку `sky`, указывающую на каталог `space`: `ln -s space sky`.
8. Перейдите по ссылке в каталог `sky`. Сохраните логический путь (`pwd`) в файл `otvety/8a`, физический путь (`pwd -P`) — в файл `otvety/8b`. Сравните содержимое файлов.
9. Перейдите в каталог `kitchen`. Сохраните в файл `otvety/9` значение переменной окружения `PWD`, выведенное командой `echo $PWD`.
10. Перейдите в каталог `logs`. Сохраните в файл `otvety/10` путь текущего каталога, полученный без команды `pwd`, — командой `realpath ..`.

Раздел 2. Команда cd

2.1. Теоретические сведения

Назначение. Команда cd (change directory) изменяет текущий рабочий каталог. При успешном выполнении команда ничего не выводит.

Синтаксис:

```
cd [каталог]
```

Основные формы:

Команда	Результат
cd каталог	переход в каталог по относительному пути
cd /путь/к/каталогу	переход по абсолютному пути
cd ..	переход в родительский каталог
cd ../..	переход на два уровня вверх
cd или cd ~	переход в домашний каталог
cd -	переход в предыдущий рабочий каталог (с выводом его пути)
cd /	переход в корневой каталог

Специальные обозначения:

Обозначение	Значение
/	корневой каталог
~	домашний каталог пользователя
.	текущий каталог
..	родительский каталог

Автодополнение. Клавиша Tab дополняет имя файла или каталога по введённому началу. Двойное нажатие Tab выводит список возможных вариантов, если однозначное дополнение невозможно.

Замечания:

- Команда, её параметры и аргументы разделяются пробелами. Запись cd.. без пробела в bash не распознаётся.
- Имя, содержащее пробел, необходимо заключать в кавычки: cd "my folder".
- Аргументом cd может быть только каталог.

2.2. Задания

Рабочий каталог: ~/linux-lab/02_cd. В каждом задании после перехода сохраните результат команды pwd в файл с номером задания в каталоге ~/linux-lab/02_cd/otvety. Путь к файлу ответа указывайте относительно текущего каталога (например, pwd > ../../otvety/2) или абсолютно (pwd > ~/linux-lab/02_cd/otvety/2).

1. Из рабочего каталога перейдите в каталог `zoo/mammals`, затем дважды выполните `cd ..`. Сохраните путь в `otvety/1`.
2. Находясь в рабочем каталоге, последовательно перейдите в каталоги `zoo/birds`, `space/planets`, `kitchen/recipes`, используя для каждого перехода одну команду с относительным путём. В последнем каталоге сохраните путь командой `pwd > ../../otvety/2`.
3. Перейдите в каталог `projects/site/css`, затем выполните `cd ../../..`. Сохраните путь в `otvety/3`.
4. Перейдите в каталог `~/linux-lab/02_cd/zoo`, затем в `~/linux-lab/02_cd/space`. Выполните команду `cd -` три раза. Сохраните путь в `otvety/4`.
5. Команда `cd projects site` завершается ошибкой. Выполните переход в каталог `projects/site` правильной командой и сохраните путь в `otvety/5`.
6. В рабочем каталоге находится каталог `my photos`, имя которого содержит пробел. Перейдите в него и сохраните путь в `otvety/6`.
7. Из рабочего каталога перейдите в каталог `projects/site/img`, набирая имена с помощью автодополнения (клавиша Tab). Сохраните путь в `otvety/7`.
8. Перейдите в скрытый каталог `.secret`. Сохраните путь в `otvety/8`.
9. Перейдите в каталог `projects/site/img`, а из него одной командой с относительным путём — в каталог `zoo/reptiles`. Сохраните путь в `otvety/9`.
10. Выполните приведённую последовательность команд и сохраните путь итогового каталога в `otvety/10`:

```
cd ~/linux-lab/02_cd
cd zoo/./birds/../../space
cd planets/..
cd ./../kitchen
```

Раздел 3. Команда `ls`

3.1. Теоретические сведения

Назначение. Команда `ls` (list) выводит список файлов и подкаталогов. Без аргументов выводится содержимое текущего каталога, при указании аргумента — содержимое указанного каталога.

Синтаксис:

```
ls [параметры] [файл или каталог ...]
```

Основные параметры:

Параметр	Описание
`-l`	подробный (длинный) формат: тип, права доступа, владелец, группа, размер, дата изменения
`-a`	включить скрытые файлы (имена которых начинаются с точки)
`-h`	размеры в единицах K, M, G (совместно с `-l`)
`-t`	сортировка по времени изменения, новые файлы первыми
`-r`	обратный порядок сортировки
`-S`	сортировка по размеру
`-R`	рекурсивный вывод содержимого подкаталогов
`-d`	вывести сведения о самом каталоге, а не о его содержимом
`-1`	выводить по одному имени в строке

Однобуквенные параметры допускается объединять: `ls -lah` равнозначно `ls -l -a -h`.

Шаблоны имён (обрабатываются командной оболочкой и применимы в любой команде):

Шаблон	Значение
`*`	любая последовательность символов, в том числе пустая
`?`	ровно один произвольный символ
`[abc]`	один символ из перечисленных
`[a-z]`	один символ из диапазона

Формат строки `ls -l`:

```
-rw-r--r-- 1 anna anna 1.2K Sep 21 14:02 notes.txt
│└───┬───┘  │  └─┬─┘ └─┬─┘ └──┬──┘ └──────┬──────┘ └────┬────┘
│    │      │    │     │      │           │             │
│  права   владелец группа размер  дата       имя
└ тип объекта: - обычный файл, d каталог, l символическая ссылка
```

3.2. Задания

Рабочий каталог: `~/linux-lab/03_ls`. Вывод каждой команды сохраняется в файл с номером задания в каталоге `otvety`, например `ls -a > otvety/1`. Сохранённый результат можно просмотреть командой `cat otvety/1`.

1. Сохраните в `otvety/1` список файлов рабочего каталога, включая скрытые.
2. Найдите скрытый каталог. Сохраните в `otvety/2` список его содержимого, включая скрытые файлы.
3. Сохраните в `otvety/3` подробный список каталога `space` с размерами файлов в единицах K, M, G.
4. Сохраните в `otvety/4` список файлов каталога `space`, отсортированный по размеру.
5. Сохраните в `otvety/5` рекурсивный список каталога `zoo`.
6. Используя шаблон `?`, сохраните в `otvety/6` список файлов рабочего каталога, имя которых состоит ровно из пяти символов и расширения `.txt`.
7. Сохраните в `otvety/7` список только каталогов рабочего каталога (`ls -d */`).
8. Используя шаблон `*`, сохраните в `otvety/8` список всех файлов с расширением `.bak` из всех подкаталогов каталога `archive` одной командой.
9. Сохраните в `otvety/9` подробные сведения о файле `hello.sh`. Выведите ответ командой `cat otvety/9`: в первой колонке видны права доступа, в третьей — владелец файла.
10. Сохраните в `otvety/10` количество файлов в каталоге `archive/2024`, полученное конвейером `ls -1 archive/2024 | wc -l`.

Раздел 4. Команда `mkdir`

4.1. Теоретические сведения

Назначение. Команда `mkdir` (make directory) создаёт каталоги.
Синтаксис:

```
mkdir [параметры] каталог ...
```

Основные параметры:

Параметр	Описание
`-p`	создать все недостающие промежуточные каталоги; не сообщать об ошибке, если каталог уже существует
`-v`	выводить сообщение о каждом созданном каталоге
`-m права`	задать права доступа создаваемого каталога

Замечания:

- Одной командой можно создать несколько каталогов: `mkdir a b c`.
- Имя, содержащее пробел, заключается в кавычки: `mkdir "my folder"`.
- Фигурные скобки формируют набор имён: `mkdir {jan,feb,mar}` создаёт три каталога, `mkdir month{01..12}` — двенадцать.

Пример:

```
$ mkdir -p trips/2026/summer
$ ls trips/2026
summer
```

4.2. Задания

Рабочий каталог: `~/linux-lab/04_mkdir`.

1. Создайте в рабочем каталоге каталог `sandbox` и перейдите в него. Задания 2–10 выполняются в каталоге `sandbox`.
2. Создайте одной командой каталоги `red`, `green`, `blue`.
3. Создайте цепочку вложенных каталогов `one/two/three`. Команда `mkdir one/two/three` без параметров завершается ошибкой; подберите параметр, при котором цепочка создаётся одной командой.
4. Создайте один каталог с именем `my folder`. Если ранее вы выполнили команду без кавычек и получили каталоги `my` и `folder`, удалите их командой `rmdir my folder`.
5. Создайте одной командой каталоги `month01`, `month02`, …, `month12`.
6. Создайте одной командой следующую структуру каталогов:

```
school/
├── class1/
│   ├── math/
```

```
│   └── art/
└── class2/
    ├── math/
    └── art/
```

7. Создайте скрытый каталог `.invisible` и выведите его в списке файлов.
8. Создайте одной командой с параметрами `-p` и `-v` структуру каталогов музыкального коллектива: `band/albums/2024`, `band/albums/2025`, `band/concerts`, `band/photos`.
9. Создайте одной командой цепочку из десяти вложенных каталогов `1/2/3/4/5/6/7/8/9/10`. Перейдите в самый глубокий каталог одной командой `cd`, проверьте путь командой `pwd` и вернитесь в `sandbox` командой `cd -`.
10. Создайте каталог `private` с правами доступа `700`, указав права при создании (параметр `-m`). Проверьте права командой `ls -ld private`.

Раздел 5. Команда touch

5.1. Теоретические сведения

Назначение. Команда `touch` изменяет время доступа и модификации файла. Если указанный файл не существует, создаётся пустой файл. Содержимое существующего файла не изменяется.

Синтаксис:

```
touch [параметры] файл ...
```

Основные параметры:

Параметр	Описание
`-c`	не создавать файл, если он не существует
`-d дата`	установить указанную дату вместо текущей, например `-d "2000-01-01 12:00"`
`-r образец`	установить время как у файла-образца

Замечания:

- Команда не создаёт промежуточные каталоги.
- Фигурные скобки позволяют создать серию файлов: `touch day{1..7}.txt`.
- Файл, имя которого начинается с точки, является скрытым.

Пример:

```
$ touch todo.txt
$ ls -l todo.txt
-rw-r--r-- 1 anna anna 0 Sep 21 14:02 todo.txt
```

5.2. Задания

Рабочий каталог: `~/linux-lab/05_touch/sandbox`. Все пути в заданиях указаны относительно этого каталога.

1. Создайте пустой файл `note.txt`. Проверьте его размер командой `ls -l`.
2. Создайте одной командой файлы `day1.txt`, `day2.txt`, ..., `day7.txt`.
3. Создайте файл `old.txt` с датой изменения 1 января 2000 года, 12:00. Проверьте дату командой `ls -l`.
4. Создайте файл `future.txt` с датой изменения 1 января 2030 года.
5. Создайте файл `newdir/file.txt`. Команда `touch newdir/file.txt` завершится ошибкой, пока не существует каталог `newdir`.
6. Создайте один файл с именем `my notes.txt`.
7. Создайте каталог `../kitchen/fridge_items` и одной командой создайте в нём пять файлов с названиями продуктов.
8. Создайте скрытый файл `.todo` и выведите его в списке файлов.
9. Создайте одной командой 100 файлов с именами `file001.txt`, `file002.txt`, ..., `file100.txt`. Проверьте их количество командой `ls file*.txt | wc -l`.

10. Создайте файл `same_time.txt` с такой же датой изменения, как у файла `old.txt` (параметр `-r`). Сравните даты командой `ls -l old.txt same_time.txt`.

Часть 2. Копирование, перемещение, удаление и просмотр файлов

Команды: `cp`, `mv`, `rm`, `cat`, `less`.

Общие указания

1. Учебный полигон `~/linux-lab` создан скриптом `setup.sh` перед началом первой части. Повторно запускать скрипт не требуется.
2. Каждый раздел выполняется в собственном рабочем каталоге, указанном перед блоком заданий.
3. Результат каждого задания — созданный или изменённый файл, каталог либо файл с ответом в каталоге `otvety` (запись вывода команды: `команда > otvety/N`; запись значения: `echo значение > otvety/N`).
4. Выполнение проверяется программой `proverka`: `~/linux-lab/proverka 8` — подробная проверка раздела 8, `~/linux-lab/proverka` — сводка по всем разделам.
5. Если содержимое рабочего каталога было повреждено, восстановите его командой `bash setup.sh <номер раздела>`, например `bash setup.sh 8`. Результаты выполнения заданий этого раздела при этом удаляются.

Раздел 6. Команда cp

6.1. Теоретические сведения

Назначение. Команда `cp` (copy) копирует файлы и каталоги.
Синтаксис:

```
cp [параметры] источник назначение
cp [параметры] источник ... каталог
```

Правила определения назначения:

- если назначение — существующий каталог, копия создаётся внутри него с исходным именем;
- если назначение — несуществующее имя, создаётся копия с этим именем;
- если файл с именем назначения уже существует, он **перезаписывается без предупреждения**;
- при копировании нескольких источников последним аргументом должен быть каталог.

Основные параметры:

Параметр	Описание
`-r`	рекурсивное копирование каталога со всем содержимым
`-i`	запрашивать подтверждение перед перезаписью
`-v`	выводить сведения о каждой операции копирования
`-p`	сохранять время изменения и права доступа

Пример:

```
$ cp notes.txt docs/
$ cp -r docs docs_copy
```

6.2. Задания

Рабочий каталог: `~/linux-lab/06_cp`.

1. Создайте резервную копию файла `story.txt` с именем `story_backup.txt`.
2. Скопируйте одной командой все файлы из каталога `kitchen/recipes` в каталог `sandbox`.
3. Создайте копию каталога `zoo` с именем `zoo_copy`. Команда без параметров завершится сообщением `omitting directory`; подберите нужный параметр.
4. Скопируйте файл `cities.txt` в каталог `space`, указав каталог в качестве назначения. Проверьте результат командой `ls space`.
5. Скопируйте каталог `projects/site` в `projects/site_v2`. Допишите строку в файл копии командой `echo "<p>v2</p>" >> projects/site_v2/index.html`. Убедитесь командой `cat`, что исходный файл `projects/site/index.html` не изменился.

6. Скопируйте файл из скрытого каталога `.secret` в рабочий каталог под именем `clue.txt`.
7. Создайте каталог `sandbox/pictures` и скопируйте в него одной командой все изображения из `projects/site/img` с расширением `.jpg` в любом регистре (`cat.jpg` и `Cat2.JPG`). Шаблон `*.jpg` выберет только один файл; составьте шаблон с классами символов `[ ]`.
8. Скопируйте файлы `cities.txt`, `story.txt` и `config.conf` в каталог `sandbox` с параметром `-v` и сохраните вывод команды в файл `otvety/8`.
9. Скопируйте файл `logs/old.log` в каталог `sandbox` с сохранением времени изменения (параметр `-p`). Сравните время исходного файла и копии командой `ls -l`.
10. Создайте резервную копию всего рабочего каталога раздела: `cp -r ~/linux-lab/06_cp ~/linux-lab/06_cp_backup`. Убедитесь командой `ls -a`, что скрытые файлы и каталоги скопированы.

Раздел 7. Команда mv

7.1. Теоретические сведения

Назначение. Команда mv (move) перемещает и переименовывает файлы и каталоги.

Синтаксис:

```
mv [параметры] источник назначение
mv [параметры] источник ... каталог
```

Правила определения назначения:

- если назначение — существующий каталог, объект перемещается в него;
- в противном случае объект переименовывается;
- файл с именем назначения **заменяется без предупреждения**;
- при перемещении нескольких объектов последним аргументом должен быть каталог.

В отличие от cp, для перемещения каталогов параметр -r не требуется.

Основные параметры:

Параметр	Описание
-i	запрашивать подтверждение перед заменой
-v	выводить сведения о каждой операции

Пример:

```
$ mv draft.txt final.txt
$ mv final.txt docs/
```

7.2. Задания

Рабочий каталог: ~/linux-lab/07_mv/sandbox. Все пути в заданиях указаны относительно этого каталога.

1. Переименуйте файл note.txt в notes.txt.
2. Создайте каталог week и переместите в него все файлы day*.txt одной командой.
3. Переименуйте каталог ../zoo/reptiles в ../zoo/reptilia.
4. Содержимое файлов left.txt и right.txt перепутано: в left.txt записано «право», в right.txt — «лево». Исправьте ошибку, поменяв файлы местами с помощью команды mv (потребуется временное имя). Проверьте результат командой cat.
5. Переместите одной командой все файлы .txt из каталога ../archive/2025 в каталог ../archive/2024.
6. Переименуйте четыре файла .bak в каталоге ../archive/2024, добавив к имени год: report.bak → report_2024.bak, notes.bak → notes_2024.bak и т. д.

7. Каталог `mess` содержит файлы разных типов. Создайте в нём каталоги `images`, `music`, `docs`, `video` и распределите по ним файлы (`.jpg`, `.mp3`, `.txt`, `.mp4`), используя одну команду `mv` на каждый тип.
8. Создайте каталог `letters` и переместите в него файлы `a.txt` и `b.txt` одной командой.
9. Требуется переименовать `plan.txt` в `final.txt`, но файл `final.txt` уже существует и будет перезаписан. Сохраните существующий файл под именем `final_old.txt`, затем выполните переименование.
10. Переместите каталог `../projects/site/img` в каталог `../space` с новым именем `images` одной командой.

Раздел 8. Команда rm

8.1. Теоретические сведения

Назначение. Команда rm (remove) удаляет файлы и каталоги. Удаление выполняется безвозвратно: удалённые объекты не помещаются в корзину.

Синтаксис:

```
rm [параметры] файл ...
rmdir каталог ...
```

Основные параметры:

Параметр	Описание
-i	запрашивать подтверждение перед удалением каждого объекта
-r	рекурсивное удаление каталога со всем содержимым
-f	не запрашивать подтверждение и не сообщать об отсутствующих файлах
-v	выводить сведения о каждом удалённом объекте

Команда rmdir удаляет только пустые каталоги.

Правила безопасной работы:

1. Перед удалением по шаблону выведите список файлов командой ls с тем же шаблоном и убедитесь, что он соответствует ожиданиям.
2. Используйте параметр -i при удалении важных данных.
3. Не выполняйте команды вида rm -rf с непроверенными путями.

Особые случаи:

- Имя, содержащее пробел, заключается в кавычки: rm "my notes.txt".
- Имя, начинающееся с дефиса, указывается после разделителя -- (rm -- -file) или с явным путём (rm ./-file).

8.2. Задания

Рабочий каталог: ~/linux-lab/08_rm/sandbox. Все пути в заданиях указаны относительно этого каталога.

Каталог sandbox заранее содержит: файлы file001.txt…file100.txt, day1.txt…day7.txt, my notes.txt, story_backup.txt, -oops; пустой каталог red; каталог box с тремя файлами; каталог week с файлами day1.txt…day7.txt; цепочку вложенных каталогов 1/2/…/10. Перед каждым удалением просматривайте содержимое каталога командой ls.

1. Удалите файл story_backup.txt.
2. Выполните команду rm -i week/*. На запрос об удалении day7.txt ответьте n, на остальные — y.
3. Удалите пустой каталог red командой rmdir.
4. Удалите каталог box вместе с содержимым. Команда rmdir box завершится ошибкой, так как каталог не пуст.

5. Удалите одной командой файлы `file001.txt`…`file099.txt`, оставив `file100.txt`. Используйте шаблон и предварительно проверьте его командой `ls`.
6. Удалите все файлы `.bak` из каталога `../archive/2024`. Сначала выведите удаляемые файлы командой `ls` с тем же шаблоном.
7. Удалите файл `my notes.txt`.
8. Удалите файл `-oops`. Команда `rm -oops` завершится ошибкой, так как имя воспринимается как параметр.
9. Удалите цепочку каталогов `1/2/…/10` одной командой.
10. Удалите одной командой, без параметра `-i`, файлы `day1.txt`…`day6.txt` в каталоге `sandbox`, оставив `day7.txt`. Используйте шаблон с диапазоном `[ ]`.

Раздел 9. Команда cat

9.1. Теоретические сведения

Назначение. Команда `cat` (от concatenate — сцеплять) выводит содержимое одного или нескольких файлов в стандартный поток вывода. Применяется для просмотра небольших файлов и объединения файлов.

Синтаксис:

```
cat [параметры] [файл ...]
```

Основные параметры:

Параметр	Описание
`-n`	нумеровать все строки
`-b`	нумеровать только непустые строки
`-E`	отмечать конец каждой строки символом `$`

Перенаправление вывода:

Запись	Действие
`команда > файл`	записать вывод в файл; прежнее содержимое файла удаляется
`команда >> файл`	дописать вывод в конец файла

Примеры:

```
$ cat a.txt b.txt > ab.txt     # объединить два файла
$ cat > note.txt               # записать ввод с клавиатуры; завершение — Ctrl+D
```

Родственные команды: `tac` выводит строки файла в обратном порядке; `rev` выводит каждую строку с символами в обратном порядке.

9.2. Задания

Рабочий каталог: `~/linux-lab/09_cat`. Результаты сохраняются в каталог `sandbox`.

1. Объедините все файлы каталога `kitchen/recipes` в файл `sandbox/cookbook.txt`. Проверьте количество строк командой `wc -l`.
2. Сохраните содержимое файла `cities.txt` с нумерацией строк в файл `sandbox/cities_numbered.txt`.
3. Сохраните содержимое файла из скрытого каталога `.secret` в файл `sandbox/secret_copy.txt` с помощью `cat` и `>`.
4. Создайте файл `sandbox/two.txt`, содержащий сначала файл `.hidden_note`, затем файл `story.txt`, одной командой.

5. Выполните команду `cat > sandbox/diary.txt`, введите не менее трёх строк текста и завершите ввод сочетанием Ctrl+D. Выведите файл на экран.
6. Создайте файл `sandbox/all_logs.txt`: запишите в него `logs/old.log` с помощью `>`, затем допишите `logs/app.log` с помощью `>>`. Проверьте, что файл содержит 41 строку.
7. Сохраните содержимое файла `config.conf` с отметками конца строк (`-E`) в файл `sandbox/config_marked.txt`. Найдите в нём пустую строку.
8. Файл `sandbox/story_copy.txt` содержит неполную копию сказки. Выполните команду `cat sandbox/story_copy.txt > sandbox/story_copy.txt` и проверьте размер файла командой `ls -l`: файл станет пустым, так как `>` очищает его до начала чтения. Восстановите полное содержимое из `story.txt` с помощью `cat` и `>`.
9. Сохраните список городов в обратном порядке строк в файл `sandbox/cities_reverse.txt` (команда `tac`).
10. Сохраните строки файла `story.txt` с обратным порядком символов в файл `sandbox/story_reversed.txt` (команда `rev`).

Раздел 10. Команда less

10.1. Теоретические сведения

Назначение. Программа less предназначена для постраничного просмотра текстовых файлов и вывода других команд. Файл не загружается в память целиком, что позволяет просматривать файлы большого объёма.

Синтаксис:

```
less [параметры] файл ...
команда | less
```

Основные параметры:

Параметр	Описание
-N	отображать номера строк
-S	не переносить длинные строки
-i	поиск без учёта регистра
+F	открыть файл в режиме отслеживания новых строк

Управление в режиме просмотра:

Клавиша	Действие
Пробел, f	на страницу вперёд
b	на страницу назад
↓ / ↑, j / k	на строку вперёд / назад
g / G	в начало / в конец файла
Ng	к строке с номером N
/шаблон	поиск вперёд
?шаблон	поиск назад
n / N	следующее / предыдущее совпадение
=	сведения о текущей позиции в файле
F	режим отслеживания новых строк (выход — Ctrl+C)
:n / :p	следующий / предыдущий файл
h	справка
q	выход

Программа man использует less для отображения справки, поэтому перечисленные клавиши действуют и при чтении справочных страниц.

10.2. Задания

Рабочий каталог: ~/linux-lab/10_less. Сведения, найденные при просмотре, записываются в каталог otvety командой echo, например echo 7 > otvety/1.

1. Откройте файл logs/app.log в программе less. Выполните поиск /ERROR и, переходя по совпадениям клавишей n, подсчитайте записи со словом ERROR в верхнем регистре. Запишите число в otvety/1.

2. Откройте `logs/app.log`, перейдите в конец файла клавишей `G`. Запишите время последней записи (в формате `ЧЧ:ММ`) в `otvety/2`.
3. Откройте файл `story.txt` с нумерацией строк (`less -N`). Найдите строку со словом «ракету» и запишите её номер в `otvety/3`.
4. Используя поиск в программе `less`, найдите в файле `zoo/animals.txt` всех животных из Азии. Запишите их названия в `otvety/4`, по одному в строке: `echo -e "первое\nвторое\nтретье" > otvety/4`.
5. Откройте два файла одновременно: `less -N cities.txt space/planets.txt`. Перейдите ко второму файлу командой `:n`, найдите строку «Марс» и запишите её номер в `otvety/5`.
6. Выполните команду `grep -i error logs/app.log | less`. Запишите время первой записи с ошибкой в `otvety/6`.
7. Откройте встроенную справку программы `less` (клавиша `h`). Найдите клавишу, отображающую текущую позицию в файле, и запишите её в `otvety/7`.
8. Откройте `logs/app.log` и введите `&INFO`, чтобы отобразить только записи уровня `INFO`. Запишите время последней такой записи в `otvety/8`.
9. Откройте `logs/app.log`, перейдите в конец файла и выполните поиск назад `?WARNING`. Запишите время последней записи уровня `WARNING` в `otvety/9`.
10. Откройте два терминала. В первом выполните `less +F ~/linux-lab/10_less/logs/app.log`. Во втором добавьте запись: `echo "2026-09-20 17:00 ERROR test" >> ~/linux-lab/10_less/logs/app.log`. Убедитесь, что запись появилась в первом терминале. Выйдите из режима отслеживания (Ctrl+C) и завершите программу (`q`).

Часть 3. Обработка текста, поиск, права доступа и справочная система

Команды: `head`, `tail`, `echo`, `grep`, `find`, `chmod`, `man`. Итоговое задание.

Общие указания

1. Учебный полигон `~/linux-lab` создан скриптом `setup.sh` перед началом первой части. Повторно запускать скрипт не требуется.
2. Каждый раздел выполняется в собственном рабочем каталоге, указанном перед блоком заданий.
3. Результат каждого задания — созданный или изменённый файл, каталог либо файл с ответом в каталоге `otvety` (запись вывода команды: `команда > otvety/N`; запись значения: `echo значение > otvety/N`).
4. Выполнение проверяется программой `proverka`: `~/linux-lab/proverka 16` — подробная проверка раздела 16, `~/linux-lab/proverka` — сводка по всем разделам.
5. Если содержимое рабочего каталога было повреждено, восстановите его командой `bash setup.sh <номер раздела>`, например `bash setup.sh 16`. Результаты выполнения заданий этого раздела при этом удаляются.
6. Задания раздела 16 выполняются от имени обычного пользователя: на суперпользователя (root) ограничения прав доступа не распространяются.

Раздел 11. Команда head

11.1. Теоретические сведения

Назначение. Команда `head` выводит начальную часть файла, по умолчанию — первые 10 строк.

Синтаксис:

```
head [параметры] [файл ...]
```

Основные параметры:

Параметр	Описание
`-n N`	вывести первые N строк
`-n -N`	вывести все строки, кроме последних N
`-c N`	вывести первые N байт
`-q`	не выводить заголовки с именами файлов при обработке нескольких файлов

При обработке нескольких файлов перед содержимым каждого выводится заголовок вида `==> имя <==`.

Конвейер. Символ `|` передаёт вывод одной команды на вход другой. Например, `ls -t | head -n 3` выводит три последних изменённых файла.

Замечание. В кодировке UTF-8 символы латинского алфавита занимают 1 байт, символы кириллицы — 2 байта.

11.2. Задания

Рабочий каталог: `~/linux-lab/11_head`. Вывод каждой команды сохраняется в файл с номером задания в каталоге `otvety`.

1. Сохраните в `otvety/1` результат команды `head cities.txt`. Сравните количество строк в ответе (`wc -l otvety/1`) и в исходном файле.
2. Сохраните в `otvety/2` первые три строки файла `space/planets.txt`.
3. Сохраните в `otvety/3` первые 20 байт файла `story.txt`. Выведите ответ командой `cat`: символов в нём меньше 20, так как буква кириллицы в UTF-8 занимает 2 байта.
4. Сохраните в `otvety/4` все строки файла `cities.txt`, кроме последних пяти.
5. Сохраните в `otvety/5` первые строки всех файлов каталога `kitchen/recipes` без заголовков с именами файлов.
6. Сохраните в `otvety/6` только пятую строку файла `cities.txt`, используя конвейер из команд `head` и `tail`.
7. Сохраните в `otvety/7` строки файла `numbers.txt` с 41-й по 45-ю.
8. Сохраните в `otvety/8` строки файла `numbers.txt` с 90-й по 95-ю.
9. Обновите время изменения файла `config.conf` командой `touch`. Сохраните в `otvety/9` имена трёх последних изменённых объектов рабочего каталога (`ls -t` и `head`).
10. Сохраните в `otvety/10` первую запись файла `zoo/animals.txt`.

Раздел 12. Команда tail

12.1. Теоретические сведения

Назначение. Команда `tail` выводит конечную часть файла, по умолчанию — последние 10 строк. Широко применяется для анализа журналов событий.

Синтаксис:

```
tail [параметры] [файл ...]
```

Основные параметры:

Параметр	Описание
`-n N`	вывести последние N строк
`-n +N`	вывести строки начиная с N-й
`-c N`	вывести последние N байт
`-f`	отслеживать дополнение файла и выводить новые строки по мере их появления
`-F`	то же, что `-f`, с повторным открытием файла при его пересоздании

Режим отслеживания (`-f`) не завершается самостоятельно; для выхода используется сочетание клавиш Ctrl+C.

12.2. Задания

Рабочий каталог: `~/linux-lab/12_tail`. Вывод каждой команды сохраняется в файл с номером задания в каталоге `otvety`.

1. Сохраните в `otvety/1` результат команды `tail cities.txt`.
2. Сохраните в `otvety/2` последнюю строку файла `space/planets.txt`.
3. Сохраните в `otvety/3` строки файла `cities.txt`, начиная с 10-й.
4. Откройте два терминала. В первом выполните `tail -f ~/linux-lab/12_tail/logs/app.log`. Во втором трижды выполните `echo "TEST" >> ~/linux-lab/12_tail/logs/app.log` и наблюдайте появление строк в первом терминале. Завершите отслеживание сочетанием Ctrl+C.
5. Сохраните в `otvety/5` результат конвейера `tail -n 3 numbers.txt | head -n 1`.
6. Сохраните в `otvety/6` последнюю запись файла `logs/app.log`, содержащую слово `error` в любом регистре (конвейер `grep` и `tail`).
7. Сохраните в `otvety/7` по две последние строки файлов `logs/app.log` и `logs/old.log`, полученные одной командой.
8. Сохраните в `otvety/8` последнюю строку файла `story.txt`.
9. Сохраните в `otvety/9` три последние записи уровня `INFO` из файла `logs/app.log`.
10. Сохраните в `otvety/10` первую запись уровня `WARNING` из файла `logs/app.log`.

Раздел 13. Команда echo

13.1. Теоретические сведения

Назначение. Команда `echo` выводит свои аргументы в стандартный поток вывода. В сочетании с перенаправлением применяется для записи текста в файлы.

Синтаксис:

```
echo [параметры] [строка ...]
```

Основные параметры:

Параметр	Описание
`-n`	не выводить перевод строки в конце
`-e`	интерпретировать управляющие последовательности: `\n` — перевод строки, `\t` — табуляция

Подстановки, выполняемые командной оболочкой:

Запись	Результат
`$ИМЯ`	значение переменной, например `$HOME`, `$USER`, `$SHELL`
`$(команда)`	вывод команды
`*`	список имён файлов текущего каталога
`{a,b,c}`, `{1..5}`	последовательность значений

Кавычки:

- в двойных кавычках подстановка переменных выполняется: `echo "Дом: $HOME"` → `Дом: /home/anna`;
- в одинарных кавычках текст выводится без изменений: `echo 'Дом: $HOME'` → `Дом: $HOME`;
- внутри кавычек любого вида шаблон `*` не раскрывается.

Перенаправление: `>` записывает вывод в файл с удалением прежнего содержимого, `>>` дописывает вывод в конец файла.

13.2. Задания

Рабочий каталог: `~/linux-lab/13_echo`.

1. Запишите в `otvety/1` строку «Привет, Linux!».
2. Запишите в `otvety/2` имя пользователя и путь домашнего каталога через пробел, используя переменные `USER` и `HOME`.
3. Создайте файл `sandbox/diary.txt` из трёх строк: первую строку запишите с помощью `>`, две следующие — с помощью `>>`.
4. Создайте файл `sandbox/list.txt`, содержащий ровно две строки: `B` и `C`. Если файл получился неверным, перезапишите его командой с `>`.
5. Запишите в `otvety/5` результат команды `echo -e "раз\nдва\tтри"`. Выведите файл командой `cat` и сравните с результатом команды без параметра `-e`.

6. Запишите в `otvety/6` строку `Мой дом: $HOME` так, чтобы переменная не была заменена её значением.
7. Запишите в `otvety/7` результат команды `echo *`. Выведите файл командой `cat`: вместо символа `*` в нём записан список имён файлов рабочего каталога.
8. Допишите слово «молоко» в конец файла `kitchen/fridge.txt`.
9. Создайте файл `kitchen/shopping.txt`: первая строка — заголовок «Список покупок», далее пять пунктов вида `1. хлеб`, `2. …`. Используйте только команду `echo`.
10. Запишите в `otvety/10` все строчные буквы латинского алфавита через пробел, используя одну команду `echo` с фигурными скобками.

Раздел 14. Команда grep

14.1. Теоретические сведения

Назначение. Команда `grep` выполняет поиск строк, соответствующих шаблону, в файлах или в стандартном потоке ввода и выводит найденные строки.

Синтаксис:

```
grep [параметры] шаблон [файл ...]
команда | grep [параметры] шаблон
```

Основные параметры:

Параметр	Описание
`-i`	не различать строчные и прописные буквы
`-n`	выводить номера строк
`-c`	вывести только количество найденных строк
`-v`	вывести строки, не соответствующие шаблону
`-w`	искать шаблон только как отдельное слово
`-r`	рекурсивный поиск во всех файлах каталога
`-l`	вывести только имена файлов, содержащих совпадения
`-E`	использовать расширенные регулярные выражения (например, `"a|b"` — a или b)

Элементы регулярных выражений:

Элемент	Значение
`^`	начало строки
`$`	конец строки
`.`	любой одиночный символ

Шаблон, содержащий пробелы или специальные символы, заключается в кавычки. При отсутствии имени файла `grep` читает данные со стандартного ввода (с клавиатуры).

Замечание. Параметр `-c` подсчитывает строки, а не отдельные вхождения шаблона.

14.2. Задания

Рабочий каталог: `~/linux-lab/14_grep`. Вывод каждой команды сохраняется в файл с номером задания в каталоге `otvety`.

1. Сохраните в `otvety/1` записи файла `zoo/animals.txt` о животных, обитающих в Африке.
2. Сохраните в `otvety/2` количество птиц в файле `zoo/animals.txt`, полученное одной командой `grep`.

3. Сохраните количество строк файла `story.txt`, содержащих «кот»: с учётом регистра — в `otvety/3a`, без учёта регистра — в `otvety/3b`, без учёта регистра и только как отдельное слово — в `otvety/3c`. Сравните три числа.
4. Сохраните количество строк файла `logs/app.log`, содержащих `ERROR` в верхнем регистре, в `otvety/4a`, а количество строк со словом `error` в любом регистре — в `otvety/4b`.
5. Сохраните в `otvety/5` только действующие параметры файла `config.conf`, исключив строки комментариев (начинающиеся с `#`) и пустые строки.
6. Сохраните в `otvety/6` имена файлов каталога `kitchen`, содержащих слово «картофель».
7. Сохраните в `otvety/7` номер и текст строки файла `space/planets.txt`, содержащей «Марс».
8. Сохраните в `otvety/8` все записи файла `logs/app.log`, содержащие фразу `disk usage`. Команда `grep disk usage logs/app.log` без кавычек выдаёт ошибку.
9. Сохраните в `otvety/9` города из файла `cities.txt`, названия которых начинаются на «К».
10. Сохраните в `otvety/10` количество млекопитающих, обитающих в Африке, используя конвейер из двух команд `grep`.

Раздел 15. Команда `find`

15.1. Теоретические сведения

Назначение. Команда `find` выполняет поиск файлов и каталогов в иерархии каталогов по заданным условиям. В отличие от `grep`, выполняющей поиск в содержимом файлов, `find` выполняет поиск по атрибутам объектов: имени, типу, размеру, времени изменения.

Синтаксис:

```
find [каталог ...] [условия] [действия]
```

Основные условия:

Условие	Описание
`-name "шаблон"`	имя соответствует шаблону (с учётом регистра)
`-iname "шаблон"`	имя соответствует шаблону без учёта регистра
`-type f` / `-type d`	обычный файл / каталог
`-size +N` / `-size -N`	размер больше / меньше N (суффиксы `k`, `M`, `G`)
`-mtime -N`	изменён менее N суток назад
`-mmin -N`	изменён менее N минут назад
`-empty`	пустой файл или каталог
`-not`, `-path`	отрицание условия; соответствие полного пути шаблону

Действия:

Действие	Описание
`-print`	вывести путь (по умолчанию)
`-delete`	удалить найденный объект
`-exec команда {} \;`	выполнить команду для каждого объекта; `{}` заменяется путём объекта

Замечание. Шаблон имени необходимо заключать в кавычки, чтобы командная оболочка не раскрыла его до запуска `find`.

15.2. Задания

Рабочий каталог: `~/linux-lab/15_find`. Вывод каждой команды сохраняется в файл с номером задания в каталоге `otvety`.

1. Сохраните в `otvety/1` список всех файлов с расширением `.txt` в рабочем каталоге и подкаталогах.
2. Сохраните в `otvety/2` список всех каталогов внутри `zoo`, включая сам `zoo`.
3. Сохраните в `otvety/3` все файлы, имя которых начинается с `cat` в любом регистре.
4. Сохраните в `otvety/4` все файлы размером более 1 МБ.

5. Сохраните в `otvety/5` все пустые файлы в каталогах `archive`, `projects` и `kitchen` (все три каталога указываются в одной команде).
6. Сохраните в `otvety/6` количество файлов с расширением `.bak` (конвейер `find` и `wc -l`).
7. Обновите время изменения файла `cities.txt` командой `touch`. Сохраните в `otvety/7` список файлов `.txt`, изменённых в течение последней минуты.
8. Сохраните в `otvety/8` все файлы с расширением `.log`. Шаблон имени заключите в кавычки.
9. Удалите все файлы `.bak` одной командой `find` с действием `-delete`. Перед удалением выведите список удаляемых файлов той же командой без `-delete`.
10. В рабочем каталоге спрятан файл `treasure.txt`, расположение и содержимое которого у каждого обучающегося своё. Найдите его командой `find` и сохраните его содержимое в `otvety/10`.

Раздел 16. Команда chmod

16.1. Теоретические сведения

Назначение. Команда chmod (change mode) изменяет права доступа к файлам и каталогам.

Модель прав доступа. Для каждого объекта задаются права трёх категорий пользователей:

```
-rwxr-xr--
 └┘└┘└┘└┘
  │  │  └── остальные пользователи (o)
  │  └───── группа (g)
  └──────── владелец (u)
```

Право	Для файла	Для каталога
r (read)	чтение содержимого	получение списка имён
w (write)	изменение содержимого	создание, удаление и переименование объектов внутри
x (execute)	выполнение как программы	переход в каталог и доступ к объектам внутри

Синтаксис:

```
chmod [параметры] права файл ...
```

Символьная запись прав состоит из трёх частей:

- категория: u (владелец), g (группа), o (остальные), a (все);
- операция: + (добавить), - (убрать), = (установить точно);
- права: r, w, x.

Примеры: chmod +x script.sh, chmod go-w file, chmod u=rw,go= file.

Числовая (восьмеричная) запись. Каждое право имеет вес: r = 4, w = 2, x = 1. Для каждой категории веса суммируются.

Число	Права
7	rwx
6	rw-
5	r-x
4	r--
0	---

Примеры: chmod 755 file → rwxr-xr-x, chmod 644 file → rw-r--r--, chmod 600 file → rw-------.

Основные параметры:

Параметр	Описание
`-R`	рекурсивное изменение прав для каталога и всего его содержимого
`-v`	выводить сведения о каждом изменении

Замечания:

- Изменять права объекта может его владелец (и суперпользователь), даже если сам владелец лишён прав на чтение или запись.
- На суперпользователя (root) ограничения прав на чтение и запись не распространяются.

16.2. Задания

Рабочий каталог: `~/linux-lab/16_chmod`. После каждого изменения проверяйте права командой `ls -l` (для каталогов — `ls -ld`).

1. Попытайтесь выполнить сценарий командой `./hello.sh` и получите сообщение `Permission denied`. Разрешите выполнение файла `hello.sh`.
2. Выполните сценарий `hello.sh` и сохраните его вывод в `otvety/2`.
3. Установите права в числовой записи: `p1` — `rwxr-x---`, `p2` — `rw-r--r--`, `p3` — `rw-rw-r--`, `p4` — `rwx--x--x`.
4. Установите для файла `secret.txt` права в символьной записи: владелец — чтение и запись, группа — чтение, остальные — нет прав.
5. Запретите запись в файл `readonly.txt` для всех категорий. Убедитесь, что команда `echo "текст" >> readonly.txt` завершается ошибкой.
6. Запретите вход в каталог `closed` для всех категорий. Убедитесь, что команда `cd closed` завершается ошибкой.
7. Лишите группу и остальных пользователей всех прав на файл `diary.txt`, не изменяя права владельца.
8. Запретите одной командой запись группе и остальным пользователям для каталога `projects/site` и всего его содержимого.
9. Создайте каталог `safe`, доступ к которому имеет только владелец.
10. Создайте в каталоге `safe` непустой файл `gold.txt` и сделайте его доступным только для чтения и только владельцу. Убедитесь, что запись в файл невозможна.

Раздел 17. Команда man

17.1. Теоретические сведения

Назначение. Команда man (manual) отображает справочное руководство по командам, системным вызовам и форматам файлов. Страница руководства просматривается средствами программы less.

Синтаксис:

```
man [раздел] имя
man -k ключевое_слово
```

Основные разделы руководства:

Раздел	Содержание
1	пользовательские команды
5	форматы файлов
8	команды системного администрирования

Структура справочной страницы: NAME (название и краткое описание), SYNOPSIS (синтаксис), DESCRIPTION (описание и параметры), EXAMPLES (примеры, если есть), SEE ALSO (связанные команды).

Связанные средства:

Средство	Назначение
команда --help	краткая справка, выводимая самой командой
man -k слово, apropos слово	поиск команд по ключевому слову в кратких описаниях
whatis команда	однострочное описание команды
type команда	определение вида команды (внешняя программа, встроенная команда оболочки, псевдоним)
help команда	справка по встроенным командам оболочки bash

17.2. Задания

Рабочий каталог: ~/linux-lab/17_man. Найденные параметры записываются в каталог otvety командой echo, например echo "-x" > otvety/1. Вывод команд сохраняется с помощью >.

1. Найдите в справочной странице команды ls параметр сортировки по размеру файла и запишите его в otvety/1.
2. Найдите в справочной странице команды grep (поиск /recursive) параметр рекурсивного поиска и запишите его в otvety/2.
3. Найдите в справочной странице команды head параметр, отключающий вывод заголовков с именами файлов, и запишите его в otvety/3.
4. Найдите в справочной странице команды ls параметр, добавляющий символ / после имён каталогов, и запишите его в otvety/4.

5. Сохраните в `otvety/5` краткое описание команды `ls`, полученное командой `whatis`.
6. Команда `man cd` не находит справочную страницу. Сохраните в `otvety/6` результат команды `type cd` и прочитайте справку командой `help cd`.
7. Сохраните в `otvety/7` список команд, связанных с копированием, полученный командой `man -k copy`.
8. Найдите с помощью `man -k sort` команду сортировки строк. Сохраните в `otvety/8` содержимое файла `cities.txt`, отсортированное по алфавиту.
9. Найдите с помощью справочной системы команду подсчёта слов. Сохраните в `otvety/9` количество слов в файле `story.txt`.
10. Найдите в справочной странице команды `ls` параметр, выводящий размеры в единицах K, M, G, и запишите его в `otvety/10`.

Раздел 18. Итоговое задание

Рабочий каталог: `~/linux-lab/18_final`.

Выполните последовательно, используя изученные команды:

1. Создайте каталог `report`.
2. Запишите в файл `report/errors.txt` все строки файла `logs/app.log`, содержащие слово `error` в любом регистре.
3. Допишите в конец файла `report/errors.txt` строку «Всего ошибок: N», где N — количество найденных строк, вычисленное командой (используйте конструкцию `$(...)`).
4. Скопируйте в каталог `report` все файлы рабочего каталога размером более 100 КБ одной командой `find` с действием `-exec`.
5. Установите для файла `report/errors.txt` права, разрешающие только чтение и только владельцу.
6. Сохраните последние три строки файла `report/errors.txt` в `otvety/6`.
7. Удалите из каталога `report` файл `telescope.raw`, предварительно проверив содержимое каталога командой `ls`.

Часть II

Практикум системного администрирования: Linux и Windows

Как устроен практикум

Чтобы научиться администрировать сервер, мало прочитать, что делают команды: нужен сервер, на котором можно ошибаться. Настоящую рабочую машину ломать нельзя, виртуальная машина для каждого ученика неудобна, а «выполнить и сравнить с ответом в конце книги» не учит ничему, кроме внимательности при переписывании. Поэтому эта часть книги построена вокруг **полигонов**: небольших заранее подготовленных учебных серверов, которые ученик создаёт на своём компьютере одной командой, ломает, чинит, а затем проверяет *программой*, а не глазами преподавателя.

Роль ученика и ход работы

Ученик — стажёр системного администратора в вымышленной компании «Северный ветер». В компании пятнадцать сотрудников, один сервер `srv01` (в Linux-варианте) или `SRV01` (в Windows-варианте) и руководитель ИТ, который присылает **заявки**. Заявка похожа на настоящую рабочую: у неё есть повод, требование и критерий приёмки.

Устройство заявки

Повод. Кто и зачем просит (например, «уволен дизайнер Глеб»).
Требование. Что нужно получить в итоге, но не пошаговая инструкция.
Критерий приёмки. Как будет видно, что работа выполнена: какой файл, какое значение, какое состояние системы проверит программа `proverka`.

Работа над заявкой идёт по одному и тому же циклу: прочитать заявку, выяснить текущее состояние (команды только для чтения), внести изменение, проверить результат *тем же способом, каким проверит программа*, и только затем запустить `proverka`. Если задание не принято, программа называет причину: «нет файла с ответом `otvety/3`» или «результат не соответствует условию заявки». Решение она не показывает. Для каждого задания есть одна короткая **подсказка-направление** (`podskazka 4.3`); обращаться к ней не стыдно, но сначала стоит перечитать заявку.

Что входит в каждый модуль

Практикум состоит из 21 модуля. Каждый модуль содержит:

- **учебный сервер**: каталог `srv` с файлами, которые бывают на настоящем сервере (`etc`, `var`, `home` в Linux; `etc`, `Users`, `var` в Windows). Модуль можно пересоздать командой `setup` с номером модуля, и он вернётся в исходное состояние;
- **заявки** (6–10 заданий): чтение состояния, изменение, проверка, итоговое задание-«контрольная»;
- **каталог ответов** `otvety`: туда ученик записывает найденные значения, когда задание состоит в расследовании (имя процесса, адрес атакующего, число файлов);
- **два вопроса для размышления**: они не проверяются, но формулируют, что важно запомнить.

Модули идут по нарастающей трудности. В первых модулях в заявке почти на-

звана команда, дальше остаётся только цель, а в последнем модуле ученик расследует инцидент без подсказок о порядке действий. Контрольные задания в конце модулей повторяют приёмы прошлых модулей в новом контексте.

№	Модуль	О чём заявки
1	Сервер и его паспорт	версия ОС, процессоры, память, время работы, свободное место, паспорт сервера
2	Учётные записи и группы	аудит записей, создание, блокировка, срок действия, увольнение
3	Права доступа	Linux: chmod, umask, setgid, sticky, ACL; Windows: списки NTFS, наследование, запрет
4	Минимальные полномочия	Linux: sudo и sudoers; Windows: состав группы администраторов и делегирование
5	Процессы и ресурсы	найти зависший и посторонний процесс, завершить, изменить приоритет, фон
6	Пакеты и программы	apt и dpkg; winget: поиск, установка, удаление, обновление, закрепление версии
7	Службы и автозапуск	systemd, unit-файлы; службы Windows, журнал событий
8	Журналы и расследование	разбор журнала веб-сервера и журнала входов, ротация и архивирование
9	Диски и место	кто занимает диск, очистка, fstab и inode; тома Windows
10	Резервное копирование	tar и Compress-Archive, контрольные суммы, восстановление
11	Автоматизация	сценарии Bash и PowerShell, коды выхода, cron и планировщик заданий
12	Сеть, SSH, брандмауэр	порты, ключи SSH, правила ufw и брандмауэра Windows
13	Аудит безопасности	пароли в файлах, чужие задания, ключи уволенных, лишние службы
14	Итоговый инцидент	«сервер тормозит»: процесс, диск, служба, атака, восстановление, отчёт

Индивидуальные данные

При создании полигона запоминается случайное число, из которого строятся индивидуальные данные: адреса атакующих, имена «забытых» процессов, число неудачных входов, какие файлы остались открытыми всем. Поэтому ответы соседа не подойдут, а повторное пересоздание модуля даёт те же данные: работу можно продолжать после перерыва. Проверка сравнивает ответ не с числом из книги, а с тем, что получается из *вашего* полигона.

Оценка и уровни

Сводка `proverka` показывает число выполненных заданий в каждом модуле, полоску прогресса и общий процент. По проценту присваивается уровень:

Выполнено	Уровень
0–24 %	Стажёр
25–49 %	Помощник администратора
50–79 %	Администратор
80–100 %	Старший администратор

Уровень — ориентир для самого ученика, а не оценка. Преподавателю удобнее отчёт, который создаёт `proverka -o`: в нём перечислены выполненные и невыполненные задания.

Что настоящее, а что учебное

Полигон должен учить настоящим командам, поэтому там, где это безопасно, всё работает по-настоящему. Привилегированные действия (создание учётных записей, установка пакетов, запуск служб, правила межсетевого экрана) на чужом компьютере выполнять нельзя и не нужно, поэтому для них в полигоне есть учебные команды с *тем же синтаксисом и теми же сообщениями об ошибках*. Они работают только с файлами в каталоге `srv` и ничего не меняют в вашей системе.

	Linux	Windows
По-настоящему	chmod, umask, setfacl, ps, kill, nice, find, grep, awk, sed, tar, sha256sum, ssh-keygen, ssh -G, bash-сценарии	icacls, Get-Acl, Get-Process, Stop-Process, Start-Process, Compress-Archive, Expand-Archive, robocopy, Get-FileHash, ssh-keygen, ssh -G, сценарии PowerShell
Учебная модель	useradd, usermod, passwd, chage, apt, dpkg, systemctl, journalctl, crontab, ufw, sudo -l, visudo -c, df, ss, lsblk, blkid	Get-LocalUser, New-LocalUser, Get-Service, Start-Service, sc.exe, Get-WinEvent, winget, Register-ScheduledTask, New-NetFirewallRule, Get-NetTCPConnection, Get-Volume, Get-ComputerInfo

Модель не заменяет настоящую систему. Она упрощена: например, учебный `sudo` умеет только разбирать правила (`sudo -l`), а настоящий планировщик заданий умеет намного больше. Расхождения перечислены в приложении. После прохождения курса те же команды нужно повторить на настоящей виртуальной машине.

Правила безопасности

Ломать можно только каталог полигона (`~/sysadmin-lab` в Linux, `$HOME\sysadmin-lab` в Windows). Не выполняйте задания от имени root или администратора: часть заданий о правах доступа в этом случае не даст ожидаемых ошибок. Если модуль сломан безвозвратно, его можно пересоздать. Команды, которых вы не понимаете, не вставляйте в настоящую систему.

Как читать задания в этой книге

Каждая глава содержит краткую теорию, параллельные примеры для Linux и Windows и два блока заявок: «Практикум Linux» и «Практикум Windows». Заявки в книге и в файле `ZAYAVKI.md` внутри полигона одинаковы. В тексте заявок команды и пути набраны моноширинным шрифтом; пути Linux записаны через `/`, пути Windows — через `\`. Выполнять можно любую дорожку: если вы изучаете одну ОС, вторую пропустите; если обе, сравнивайте решения. Эталонные решения для преподавателя собраны в приложении и в учебный текст не входят.

Подготовка: полигон, терминал и первая заявка

Что нужно уметь до начала

Практикум предполагает, что вы уже умеете в терминале: переходить по каталогам, просматривать и создавать файлы, пользоваться перенаправлением и конвейером, искать текст (`grep` в Linux, `Select-String` в PowerShell) и редактировать текстовый файл. Это тема первой части книги, практикума «Базовые команды Linux»: если команды `cd`, `ls`, `cat`, `cp`, `mv`, `rm`, `find`, `grep`, `head`, `tail`, `chmod` вызывают сомнения, вернитесь к ней. Утилиты `sort`, `uniq`, `wc`, `awk` и `sed`, которые встречаются в заданиях, объясняются по мере появления.

Что нужно	Linux	Windows
Система	Ubuntu 24.04 или другая с bash 4+	Windows 10/11, Windows PowerShell 5.1
Права	обычный пользователь, **не** root	обычный пользователь, **не** запуск от администратора
Программы	python3, awk, sed, grep, find, tar, sha256sum, ssh-keygen	клиент OpenSSH (`Get-Command ssh, ssh-keygen`), `icacls`, `robocopy`
Место	около 20 МБ	около 30 МБ

Установка полигона

Полигон создаёт один сценарий. Он изменяет только каталог `sysadmin-lab` в вашем домашнем каталоге и ничего не требует от администратора.

Linux

```
bash setup.sh            # создать полигон для всех модулей (один раз)
source ~/sysadmin-lab/env.sh
lab 1                    # перейти в модуль 1
zayavki                  # прочитать заявки модуля
proverka                 # сводка по всем модулям
```

Команду `source ~/sysadmin-lab/env.sh` нужно выполнять в каждом новом терминале: она добавляет учебные команды в `PATH` и определяет `lab`, `zayavki`, `podskazka`, `proverka`. Чтобы восстановить один модуль, используйте `bash setup.sh 5`; чтобы пересоздать всё, `bash setup.sh --force`.

Windows

```
Set-ExecutionPolicy -Scope Process Bypass     # только для этого окна
.\setup.ps1                                    # создать полигон
. $HOME\sysadmin-lab\env.ps1                  # подключить в каждом окне
lab 1
zayavki
```

```
proverka
```

Точка и пробел перед путём к `env.ps1` обязательны: так сценарий выполняется в текущем окне и оставляет в нём команды `lab`, `zayavki`, `podskazka`, `proverka`. Учебные версии командлетов (`Get-Service`, `New-LocalUser`, `winget`) подключаются тем же сценарием и действуют только в этом окне; настоящие командлеты остаются доступны по полному имени, например `Microsoft.PowerShell.Management\Get-Service`.

Рабочие команды

Команда	Что делает
`lab N`	переходит в рабочий каталог модуля `N` и выбирает его учебный сервер
`zayavki`	выводит заявки текущего модуля
`proverka`	сводка по всем модулям и уровень
`proverka N`	подробная проверка модуля `N`: каждое задание выполнено или нет, и причина
`podskazka N.M`	подсказка к заданию `M` модуля `N`
`proverka -o`	отчёт `otchet.txt` для преподавателя

Первая заявка целиком

Разберём, как выглядит работа на примере заявки 2.5 (учётные записи без пароля). Допустим, в заявке 2.4 вы нашли записи `temp` и `guest` и записали их имена в `otvety/4`. Заявка 2.5 требует заблокировать каждую из них.

```
$ lab 2
$ cat otvety/4                       # какие имена мы нашли
guest
temp
$ for n in $(cat otvety/4); do passwd -l "$n"; done
$ getent shadow temp                 # проверка: второе поле начинается с !
$ proverka 2
  [выполнено]    2.4    Учётные записи без пароля
  [выполнено]    2.5    Закрыть дыру
```

Обратите внимание на порядок: перед изменением выясняется состояние, после изменения состояние проверяется самим учеником тем же способом, каким проверит программа, и лишь затем запускается `proverka`. Так вырабатывается привычка, ради которой всё и затевалось: не считать работу выполненной, пока результат не подтверждён наблюдением.

Если что-то не получается

Симптом	Что делать
«не найден учебный сервер»	Вы не в каталоге модуля. Выполните `lab N`.

«нет файла с ответом otvety/3»	Задание требует записать найденное значение в файл. Проверьте имя и каталог файла.
«результат не соответствует условию»	Перечитайте критерий заявки; выполните проверку вручную (например, `stat -c %a файл`); затем `podskazka`.
Модуль испорчен	Linux: `bash setup.sh N`; Windows: `.\setup.ps1 N`. Результаты других модулей не затрагиваются.
Кракозябры вместо русских букв	Linux: `export LANG=C.UTF-8`; Windows: сохраняйте файлы в UTF-8 (`Set-Content -Encoding UTF8`).
Windows не запускает сценарии	`Set-ExecutionPolicy -Scope Process Bypass` в текущем окне.

Глава 1

Модуль 1. Сервер и его паспорт

Идея и словарь

Администратор начинает не с изменений, а с **паспорта**: короткого документа, который отвечает на вопросы «что это за сервер», «сколько у него ресурсов» и «сколько он уже работает». Паспорт нужен по трём причинам. Во-первых, команда, безопасная на одном сервере, может быть вредной на другом (поэтому надо знать версию системы). Во-вторых, он даёт точку отсчёта: после неудачного изменения можно сравнить состояния «до» и «после». В-третьих, он защищает от ошибки «я думал, что это другой сервер».

Из чего состоит паспорт. **Операционная система и версия** определяют набор команд и сроки поддержки. **Число логических процессоров и объём памяти** показывают, какую нагрузку сервер выдержит. **Время работы** (*uptime*) — сколько времени прошло с последней загрузки: очень большое значение означает, что обновления ядра давно не применялись; очень малое — что сервер недавно перезагружался и стоит узнать почему. **Свободное место** на дисках — главный запас, который кончается первым. Ещё одно полезное сведение — **нагрузка** (*load average*): среднее число процессов, готовых к выполнению, за 1, 5 и 15 минут; значение, сравнимое с числом процессоров, означает полную загрузку.

Все эти сведения система хранит в файлах или выдаёт командами. В Linux многие из них доступны как файлы в каталоге `/proc` (информация о ядре и оборудовании в виде текста) и `/etc` (настройки). В Windows те же данные дают командлеты `Get-ComputerInfo`, `Get-Volume`, `Get-Process`. Принцип один: прочитать, а не гадать.

Linux: инструменты

```
cat /etc/os-release           # название и версия дистрибутива
hostname                      # имя сервера
uname -r                      # версия ядра
nproc                         # число логических процессоров
grep MemTotal /proc/meminfo   # память в килобайтах
cat /proc/uptime              # секунды с загрузки (первое число)
cat /proc/loadavg             # нагрузка за 1, 5 и 15 минут
df -h                         # свободное место
```

В полигоне эти файлы лежат в `srv/proc` и `srv/etc`, поэтому команды применяются к ним: `grep MemTotal srv/proc/meminfo`. Память в `/proc/meminfo` указана в килобайтах: чтобы получить мегабайты, разделите значение на 1024. Время работы удобно переводить в сутки целочисленным делением (`awk '{print int($1/86400)}' файл`).

Windows: инструменты

```
Get-ComputerInfo -Property OsName, OsVersion, CsName
(Get-ComputerInfo).CsNumberOfLogicalProcessors
```

```
[math]::Round((Get-ComputerInfo).CsTotalPhysicalMemory / 1GB)
((Get-Date) - (Get-ComputerInfo).OsLastBootUpTime).Days
Get-Volume -DriveLetter C
```

Результат `Get-ComputerInfo` — объект, а не текст: у него есть свойства, которые можно взять через точку и использовать в вычислениях. Разность двух дат (`Get-Date` минус дата загрузки) — объект `TimeSpan`, у него есть свойства `Days` и `TotalDays`. Константы `1KB`, `1MB`, `1GB` встроены в язык и удобны для пересчёта байтов.

Что чаще всего идёт не так

Память и диски в разных утилитах выводятся в разных единицах (КБ, МБ, ГБ; десятичные и двоичные). Всегда выясняйте единицу измерения, прежде чем записывать число в паспорт. Паспорт без единицы измерения бесполезен.

Практикум Linux: заявки модуля 1

Каталог полигона: `01_pasport`. Запуск: `lab 1`, затем `zayavki`.

Первый рабочий день стажёра. Дмитрий (старший администратор ООО «Северный ветер») выдал доступ к серверу `srv01` и попросил составить его паспорт: что за система, сколько ресурсов, сколько работает, где что лежит. Сервер учебный: его «железо» описано файлами в каталоге `srv/proc`, настройки — в `srv/etc`. Настоящую систему вы не затрагиваете.

Заявка Linux 1.1. Версия операционной системы

Запишите в `otvety/1` название операционной системы: значение параметра `PRETTY_NAME` из файла `srv/etc/os-release` без кавычек.

Заявка Linux 1.2. Имя сервера

Запишите в `otvety/2` имя сервера из файла `srv/etc/hostname`.

Заявка Linux 1.3. Версия ядра

В файле `srv/proc/version` третье слово первой строки — номер версии ядра. Запишите его в `otvety/3`.

Заявка Linux 1.4. Число процессорных ядер

Каждое ядро описано в `srv/proc/cpuinfo` блоком со строкой `processor`. Посчитайте ядра и запишите число в `otvety/4`.

Заявка Linux 1.5. Оперативная память

В `srv/proc/meminfo` значение `MemTotal` задано в килобайтах. Запишите в `otvety/5` объём памяти в мегабайтах — целое число (1 МБ = 1024 КБ).

Заявка Linux 1.6. Сколько суток работает сервер

Первое число в `srv/proc/uptime` — время работы в секундах. Запишите в `otvety/6` число полных суток.

Заявка Linux 1.7. Загрузка сервера

Третье число в `srv/proc/loadavg` — средняя загрузка за 15 минут. Запишите его в `otvety/7a`. В `otvety/7b` запишите `да`, если загрузка больше числа ядер (сервер перегружен), иначе `нет`.

Заявка Linux 1.8. Файловые системы

В `srv/proc/mounts` есть настоящие дисковые разделы (устройство начинается с `/dev/`) и служебные псевдо-системы. Запишите в `otvety/8` точки монтирования (второй столбец) только настоящих разделов, по одной в строке, отсортировав по алфавиту.

Заявка Linux 1.9. Где что лежит

Командой `find` определите, в каком каталоге внутри `srv` находятся файлы `sshd_config`, `meminfo` и `start.sh`. Запишите пути каталогов **относительно `srv`** (например, `etc/x`) в `otvety/9a`, `otvety/9b`, `otvety/9c` соответственно.

Заявка Linux 1.10. Паспорт сервера

Создайте в каталоге модуля файл `pasport.txt` из шести строк вида `ключ: значение` в таком порядке: `host`, `os`, `kernel`, `cpu`, `ram_mb`, `uptime_days`. Значения — из заданий 1–6.

Практикум Windows: заявки модуля 1

Каталог полигона: `01_pasport`. Запуск: `lab 1`, затем `zayavki`.

Вы приняли в управление сервер `SRV01` (Windows Server). Прежде чем что-то менять, нужно составить «паспорт» сервера: версия системы, число процессоров, память, время работы, свободное место. На настоящем сервере эти сведения даёт `Get-ComputerInfo` и `Get-Volume`; в полигоне те же команды читают учебные данные из `srv\var\lib\lab`, а значения у каждого ученика свои (из `.seed`). Работайте в Windows PowerShell 5.1: `. $HOME\sysadmin-lab\env.ps1`, затем `lab 1`.

Заявка Windows 1.1. Версия системы

Выясните название операционной системы сервера (свойство `OsName` команды `Get-ComputerInfo`) и запишите его полностью в `otvety\1`.

Заявка Windows 1.2. Сколько процессоров

Сколько логических процессоров видит система (`CsNumberOfLogicalProcessors`)? Запишите число в `otvety\2`.

Заявка Windows 1.3. Объём памяти

Сколько гигабайт оперативной памяти установлено (`CsTotalPhysicalMemory` в байтах)? Запишите целое число гигабайт в `otvety\3`. В PowerShell есть константа `1GB`.

Заявка Windows 1.4. Сколько работает без перезагрузки

Определите время работы сервера в полных сутках: разница между текущим временем (`Get-Date`) и `OsLastBootUpTime`. Запишите целое число суток в `otvety\4`.

Заявка Windows 1.5. Свободное место на C:

Сколько гигабайт свободно на диске `C:` (`Get-Volume -DriveLetter C`, свойство `SizeRemaining`)? Запишите целое число (округление до ближайшего) в `otvety\5`.

Заявка Windows 1.6. Паспорт сервера

Составьте файл `pasport.txt` (в каталоге модуля) из четырёх строк: `HOST=` — имя компьютера (`CsName`), `CORES=` — процессоры, `RAMGB=` — память в гигабайтах, `OS=` — название системы.

Вопросы для размышления

1. Какие данные паспорта вы получили из файлов, а какие пришлось вычислять? Почему так?
2. Чем описание сервера в файлах `proc` отличается от настоящего Linux?

Глава 2

Модуль 2. Учётные записи и группы

Идея и словарь

Учётная запись — запись о том, кто может войти в систему и от чьего имени работают его программы. У каждой записи есть имя, числовой идентификатор (UID в Linux, SID в Windows), пароль (в виде необратимого хеша, а не открытого текста), срок действия и принадлежность к **группам**. Группа — роль: права выдаются группе, а человек добавляется в неё или исключается, и вместе с ним приходят и уходят все права роли. Такой порядок удобнее и надёжнее, чем выдача прав каждому отдельно.

Жизненный цикл записи. *Создание*: имя, домашний каталог, начальные группы, политика пароля. *Изменение*: переход сотрудника в другой отдел меняет его группы. *Блокировка*: запись остаётся, но войти в неё нельзя; это первое действие при увольнении или подозрении на компрометацию, потому что улики (файлы, история) сохраняются. *Удаление*: убирает запись, а иногда и данные, поэтому выполняется последним и после архивирования.

Что проверяют при аудите. Записи с **правами администратора** под безобидными именами (в Linux — вторая запись с UID 0, в Windows — лишний член группы Administrators). Записи **без пароля**. Записи **давно не используемые**. Записи **служебных программ** с возможностью интерактивного входа. Эти четыре проверки покрывают большую часть реальных находок.

Linux: инструменты

Базы учётных записей Linux — обычные текстовые файлы: `/etc/passwd` (имя, UID, GID, комментарий, домашний каталог, оболочка), `/etc/shadow` (хеш пароля, даты) и `/etc/group`.

```
getent passwd anna                          # запись пользователя
awk -F: '$3 == 0 {print $1}' /etc/passwd    # все записи с UID 0
awk -F: '$2 == "" {print $1}' /etc/shadow   # записи без пароля
sudo useradd -m -s /bin/bash -c "Никита Орлов" -G dev nikita
sudo usermod -aG docker nikita              # добавить в группу (без -a группы
заменятся!)
sudo usermod -L gleb                        # заблокировать пароль
sudo chage -M 90 -m 1 -W 14 nikita          # срок жизни пароля
sudo userdel -r temp                        # удалить вместе с домашним каталогом
```

Ключ `-a` у `usermod -G` запоминают сразу: без него список групп *заменяется*, и пользователь теряет прежние. Блокировка пароля (`-L`) не отзывает SSH-ключи: чтобы закрыть вход полностью, задают ещё оболочку `/usr/sbin/nologin` или срок действия записи (`usermod -e`). Вывод `chage -l имя` показывает политику пароля.

Windows: инструменты

```
Get-LocalUser | Format-Table Name, Enabled, PasswordRequired
Get-LocalGroupMember -Group Administrators
```

```
$pw = ConvertTo-SecureString 'Start-Passw0rd!' -AsPlainText -Force
New-LocalUser -Name nikita -Password $pw -FullName 'Никита Орлов' -Description
'разработчик'
Add-LocalGroupMember -Group dev -Member nikita
Disable-LocalUser -Name gleb
Set-LocalUser -Name gleb -AccountExpires (Get-Date).AddDays(-1)
Remove-LocalUser -Name temp
```

Пароль в PowerShell передаётся не строкой, а `SecureString`: так он не остаётся в истории команд в открытом виде (в учебных целях мы создаём его из текста, чего в настоящей работе стоит избегать). Имена членов группы возвращаются в виде `ИМЯ_КОМПЬЮТЕРА\пользователь`; чтобы сравнить с именем пользователя, часть до обратной косой отбрасывают оператором `-split`.

Что чаще всего идёт не так

Самая частая ошибка — удалить запись вместо блокировки: вместе с ней пропадает возможность выяснить, что было сделано от её имени. Вторая — забыть, что у службы тоже есть учётная запись и что её удаление «роняет» службу.

Практикум Linux: заявки модуля 2

Каталог полигона: `02_uchetnye`. Запуск: `lab 2`, затем `zayavki`.

Отдел кадров прислал список изменений: выходит новый разработчик, уволен дизайнер, Борис переходит в другой отдел. Одновременно служба безопасности просит проверить, нет ли на `srv01` «лишних» учётных записей. Работаете с учебной базой в `srv/etc` (`passwd`, `shadow`, `group`) с помощью учебных `useradd`, `usermod`, `passwd`, `chage`, `groupadd` — те же ключи, что в настоящем Linux. Перед работой: `source ~/sysadmin-lab/env.sh`, затем `lab 2`.

Заявка Linux 2.1. Кто у нас обычный пользователь

Запишите в `otvety/1` имена всех **обычных** пользователей (UID от 1000 до 65533) из `srv/etc/passwd`, по одному в строке, отсортировав по алфавиту.

Заявка Linux 2.2. Второй root

Служба безопасности подозревает, что кто-то создал учётную запись с полномочиями root под другим именем. Найдите в `srv/etc/passwd` учётную запись с UID 0, кроме `root`, и запишите её имя в `otvety/2`.

Заявка Linux 2.3. Обезвредить второй root

Не удаляя улик, заблокируйте найденную учётную запись (`usermod -L`) и замените ей оболочку входа на `/usr/sbin/nologin` (`usermod -s`).

Заявка Linux 2.4. Учётные записи без пароля

В `srv/etc/shadow` у некоторых записей поле пароля пустое: войти можно без пароля. Запишите в `otvety/4` их имена, по одному в строке, отсортировав.

Заявка Linux 2.5. Закрыть дыру

Заблокируйте командой `passwd -l` каждую запись из задания 4. (Допустимо, если запись вы уже удалили, как в задании 10.)

Заявка Linux 2.6. Новый сотрудник

Выходит разработчик Никита Орлов. Создайте учётную запись `nikita`: домашний каталог (`-m`), оболочка `/bin/bash`, комментарий `Никита Орлов`, дополнительная группа `dev`.

Заявка Linux 2.7. Политика паролей

Для `nikita` задайте политику: пароль живёт не более 90 дней, не чаще чем раз в 1 день, предупреждение за 14 дней, и потребуйте сменить пароль при первом входе. Используйте `chage`.

Заявка Linux 2.8. Борис меняет отдел

Борис переходит из разработки в дизайн. Уберите его из группы `dev`, добавьте в `design`; членство в `sudo` должно сохраниться. Пользуйтесь одним вызовом `usermod`.

Заявка Linux 2.9. Увольнение

Дизайнер `gleb` уволен. Заблокируйте его учётную запись и задайте дату окончания её действия — любой день в прошлом (`usermod -e`). Домашний каталог удалять нельзя: его заберёт архив.

Заявка Linux 2.10. Убрать временную запись

Удалите учётную запись `temp` вместе с домашним каталогом (`userdel -r`). Убедитесь, что исчезла и её личная группа.

Практикум Windows: заявки модуля 2

Каталог полигона: `02_uchetnye`. Запуск: `lab 2`, затем `zayavki`.

Отдел кадров прислал список изменений: выходит новый разработчик, уволен дизайнер. Одновременно служба безопасности просит проверить, нет ли на `SRV01` «лишних» учётных записей и групп. Работаете с учебной базой локальных пользователей (`srv\etc\localusers.csv` и `localgroups.csv`) теми же командлетами, что в настоящем Windows: `Get-LocalUser`, `New-LocalUser`, `Set-LocalUser`, `Disable-LocalUser`, `Remove-LocalUser`, `Get-LocalGroupMember`, `Add-LocalGroupMember` и другими. Перед работой: `. $HOME\sysadmin-lab\env.ps1`, затем `lab 2`.

Заявка Windows 2.1. Кто у нас обычный пользователь

Запишите в `otvety\1` имена всех **включённых** учётных записей, кроме встроенных (`Administrator`, `Guest`, `DefaultAccount`, `WDAGUtilityAccount`), по одному в строке, отсортировав по алфавиту.

Заявка Windows 2.2. Лишний администратор

Служба безопасности подозревает, что кто-то получил права администратора. Посмотрите состав группы `Administrators` (`Get-LocalGroupMember`). Кроме `Administrator` и `dmitri` там есть ещё одна учётная запись. Запишите её имя (без префикса `SRV01\`) в `otvety\2`.

Заявка Windows 2.3. Обезвредить

Не удаляя улик, отключите найденную учётную запись (`Disable-LocalUser`) и выведите её из группы `Administrators` (`Remove-LocalGroupMember`).

Заявка Windows 2.4. Учётные записи без пароля

Найдите **включённые** учётные записи, которым пароль не требуется (`PasswordRequired` равно `False`). Запишите их имена в `otvety\4`, по одному в строке, отсортировав.

Заявка Windows 2.5. Закрыть дыру

Отключите (`Disable-LocalUser`) каждую запись из задания 4.

Заявка Windows 2.6. Новый сотрудник

Выходит разработчик Никита Орлов. Создайте учётную запись `nikita`: полное имя `Никита Орлов`, описание `разработчик`, любой пароль (`-Password` принимает `SecureString`), пароль должен иметь срок действия. Добавьте её в группу `dev`.

Заявка Windows 2.7. Увольнение

Дизайнер `gleb` уволен. Отключите его учётную запись и задайте дату окончания действия — вчерашний день (`Set-LocalUser -AccountExpires`). Профиль `srv\Users\gleb` удалять нельзя: его заберёт архив.

Заявка Windows 2.8. Убрать временную запись

Удалите учётную запись `temp` (`Remove-LocalUser`) вместе с её профилем `srv\Users\temp`. Убедитесь, что `temp` исчезла из всех групп.

Вопросы для размышления

1. Почему важнее заблокировать учётную запись, чем сразу её удалить, когда сотрудник уволился?
2. Чем отличается `usermod -G` от `usermod -aG`? Приведите пример, когда ошибка в выборе ключа опасна.

Глава 3

Модуль 3. Права доступа: chmod, umask, setgid, ACL

Идея и словарь

Права доступа отвечают на вопрос «кто и что может сделать с этим файлом». Система хранит их вместе с каждым файлом и проверяет при каждой операции. Правильная настройка прав — принцип **минимальных полномочий**: каждому достаточно ровно того, что нужно для работы, и ничего больше.

В Linux у файла есть владелец, группа-владелец и три набора прав — для владельца, для группы и для остальных. Каждый набор состоит из трёх битов: чтение (**r**, 4), запись (**w**, 2), выполнение (**x**, 1). Сумма даёт цифру: 7 — всё, 6 — чтение и запись, 5 — чтение и выполнение, 4 — только чтение, 0 — ничего. Три цифры образуют права, например 640. Для каталога смысл битов другой: **r** — видеть список имён, **w** — создавать и удалять записи, **x** — входить в каталог и обращаться к вложенным объектам. Три особых бита: **setuid** и **setgid** (запуск от имени владельца или группы; на каталоге setgid заставляет новые файлы наследовать группу каталога — основа общей папки), и **sticky** (в каталоге с записью для всех удалять и переименовывать можно только свои файлы, как в `/tmp`). Если трёх наборов не хватает, добавляют **ACL** — список записей «пользователь или группа — права».

В Windows прав больше, и они устроены иначе. Каждый объект NTFS хранит **список контроля доступа** (ACL): набор записей (ACE), каждая из которых разрешает или запрещает определённые права конкретному пользователю или группе. Права **наследуются** от родительской папки, если не отключить наследование. Запрещающая запись (Deny) сильнее разрешающей, поэтому её применяют осторожно. Вместо имён групп в командах надёжнее использовать **SID**: имя группы Everyone в русской Windows называется «Все», а SID `S-1-1-0` одинаков везде.

Linux: инструменты

```
ls -l файл                       # -rw-r----- 1 anna dev ...  права, владелец,
группа
stat -c '%a %A %U:%G' файл       # 640 -rw-r----- anna:dev
chmod 640 файл                   # числовой вид
chmod u=rw,g=r,o= файл           # символьный: точно задать права
chmod g+s каталог                # setgid (общая папка)
chmod 1777 /tmp/exchange         # sticky + rwx для всех
umask 027                        # новые файлы 640, каталоги 750
setfacl -m u:nobody:r-- файл     # именная запись ACL
getfacl файл
find . -type f -perm -002        # файлы, доступные на запись всем
```

Права новых файлов определяет маска `umask`: файл создаётся с правами `666 & ~umask`, каталог — `777 & ~umask`. Права `x` у файла разрешают запуск как программы; у сценария их нужно выдать явно.

Windows: инструменты

```
icacls файл                                    # показать ACL
(Get-Acl файл).Access                          # то же объектами
icacls файл /grant '*S-1-5-32-545:(OI)(CI)RX'   # Users: чтение и выполнение, с
наследованием
icacls файл /remove '*S-1-1-0'                 # убрать все записи группы Все
icacls папка /inheritance:d                    # отключить наследование, сохранив
копии
icacls папка /deny '*S-1-5-32-546:(OI)(CI)W' # запретить запись группе Guests
```

Флаги `(OI)` и `(CI)` означают наследование файлами и вложенными папками. Права обозначаются буквами: `F` — полный доступ, `M` — изменение, `RX` — чтение и выполнение, `R` — чтение, `W` — запись. Звёздочка перед SID — принятая в `icacls` запись идентификатора безопасности.

Что чаще всего идёт не так

Типичная ошибка — «починить» отказ доступа командой `chmod 777` или записью для группы «Все». Это не исправление, а отмена защиты: после неё любой процесс на сервере может изменить файл. Правильный путь: выяснить, от чьего имени работает программа, и выдать *именно ему ровно те* права.

Практикум Linux: заявки модуля 3

Каталог полигона: `03_prava`. Запуск: `lab 3`, затем `zayavki`.

После проверки учётных записей служба безопасности обошла файлы сервера и прислала замечания: ключи SSH читаются всеми, в каталог загрузки сайта может писать любой, у команды разработчиков нет нормальной общей папки. В этом модуле права меняются **по-настоящему** (`chmod`, `umask`, `setfacl`) на файлах каталога `srv`: ломать здесь можно, система не пострадает. Задания выполняются от имени обычного пользователя.

Заявка Linux 3.1. Ключи SSH

Каталог `srv/home/anna/.ssh` и ключи в нём открыты всем. SSH отвергает такие ключи. Задайте права: каталог — только владельцу (`700`), закрытый ключ `id_ed25519` и файл `authorized_keys` — чтение и запись только владельцу (`600`), открытый ключ `id_ed25519.pub` — чтение всем, запись владельцу (`644`).

Заявка Linux 3.2. Отчёт бухгалтерии

Файл `srv/home/olga/otchety/year.pdf` доступен на запись всем. Оставьте владельцу чтение и запись, группе — только чтение, остальным — ничего. Используйте **символьную** запись `chmod`.

Заявка Linux 3.3. Сценарий резервного копирования

Сценарий `srv/opt/app/backup.sh` нельзя запустить. Разрешите запуск владельцу и группе, остальным запретите всё, владельцу оставьте чтение и запись, группе — чтение и выполнение (итог `750`).

Заявка Linux 3.4. Читать права

Определите права файла `srv/etc/app/secret.key`. В `otvety/4a` запишите их числовой вид (`stat -c %a`), в `otvety/4b` — строку вида `-rw-r-----` (`stat -c %A`).

Заявка Linux 3.5. Каталог загрузки сайта

В каталог `srv/srv/www/uploads` может писать и удалять файлы любой пользователь (777). Оставьте владельцу полный доступ, остальным — только просмотр и вход (755).

Заявка Linux 3.6. Общая папка команды

Для каталога `srv/srv/projects/team` нужна общая папка: владелец и группа имеют полный доступ, остальные — никакого, а новые файлы и подкаталоги автоматически наследуют группу каталога. Установите бит setgid и права `2770`.

Заявка Linux 3.7. Обменный каталог

В `srv/tmp/exchange` должны писать все, но удалять и переименовывать можно только свои файлы. Установите sticky-бит и права 1777.

Заявка Linux 3.8. umask

Маска `umask` определяет права новых файлов. Запустите подоболочку, в которой действует `umask 027`, создайте в пустом временном каталоге файл `f` и каталог `d` и выведите их права (`stat -c %a f d`). Сохраните вывод в `otvety/8`: он должен состоять из двух строк.

Заявка Linux 3.9. ACL для аудитора

Аудитору (учётная запись `nobody` есть в любой системе) нужно читать `srv/srv/projects/report.txt`, но не группа и не все. Добавьте именную запись ACL: `setfacl -m u:nobody:r-- файл`. Проверьте `getfacl`.

Заявка Linux 3.10. Файлы, открытые всем на запись

В каталоге `srv` спрятано несколько обычных файлов с правом записи для всех. Найдите их командой `find` (`-type f -perm -002`), запишите пути в `otvety/10`, отсортировав, а затем снимите право записи у остальных пользователей (`chmod o-w`).

Практикум Windows: заявки модуля 3

Каталог полигона: `03_prava`. Запуск: `lab 3`, затем `zayavki`.

Служба безопасности обошла общие папки сервера и прислала замечания: в папке бухгалтерии есть файлы, доступные на запись всем, права на папку `Finance` непонятно откуда наследуются, а общая папка `Public` не настроена нормально. В этом модуле права меняются **по-настоящему** (настоящие NTFS-списки доступа через `icacls`, `Get-Acl`) на файлах каталога `srv`: ломать здесь можно, система не пострадает. Выполняйте задания от имени обычного пользо-

вателя (права администратора не нужны). **Важно про имена групп.** В русской Windows группа `Everyone` называется «Все», `Users` — «Пользователи», и команды с английскими именами не сработают. Поэтому в заданиях группы указываются **идентификаторами безопасности (SID)**, одинаковыми во всех системах: `*S-1-1-0` — все (Everyone), `*S-1-5-32-545` — встроенная группа Users, `*S-1-5-32-546` — Guests, `*S-1-5-32-544` — Administrators (звёздочка перед SID нужна `icacls`). Учебная функция `ConvertTo-Sid` переводит имя из ACL в строку SID.

Заявка Windows 3.1. Файлы, открытые всем

В каталогах `srv\Finance` и `srv\Public` есть файлы, на которые группа «Все» (`Everyone`, SID `S-1-1-0`) имеет право записи. Найдите их (`Get-Acl`, свойство `Access`; сравнивайте `ConvertTo-Sid $_.IdentityReference` с `'S-1-1-0'`) и запишите имена файлов (без пути) в `otvety\1`, по одному в строке, отсортировав.

Заявка Windows 3.2. Закрыть лишнее

Уберите права «Все» (SID `*S-1-1-0`) на все файлы из задания 1 командой `icacls файл /remove *S-1-1-0`. Файлы должны остаться на месте, остальные права не трогайте.

Заявка Windows 3.3. Папка бухгалтерии: отключить наследование

На папке `srv\Finance` явно выдан доступ встроенной группе Users (`*S-1-5-32-545`). Бухгалтерия — закрытая зона: отключите наследование прав с сохранением копий унаследованных записей (`icacls папка /inheritance:d`) и уберите из ACL группу Users (`/remove`). Ваша учётная запись должна сохранить полный доступ.

Заявка Windows 3.4. Общая папка для чтения

На папку `srv\Public` выдайте группе Users (`*S-1-5-32-545`) право «чтение и выполнение» (`RX`) с наследованием на вложенные папки и файлы: `icacls папка /grant '*S-1-5-32-545:(OI)(CI)RX'`.

Заявка Windows 3.5. Запрет записи гостям

В папке `srv\Public\readonly` запретите группе Guests (`*S-1-5-32-546`) запись: `icacls папка /deny '*S-1-5-32-546:(OI)(CI)W'`. В ACL должна появиться запись типа `Deny`.

Заявка Windows 3.6. Где отключено наследование

В каталогах `srv\Shared` и `srv\Public` есть файлы с отключённым наследованием прав (свойство `AreAccessRulesProtected` объекта ACL равно `True`). Найдите их среди обычных файлов (`Get-ChildItem -Recurse -File`) и запишите имена (без пути) в `otvety\6`, по одному в строке, отсортировав.

Вопросы для размышления

1. Почему закрытому ключу нужны права 600, а открытому достаточно 644?
2. Чем sticky-бит на каталоге отличается от права записи на нём?

Глава 4

Модуль 4. sudo и минимальные полномочия

Идея и словарь

Работать от имени администратора постоянно — плохая привычка: любая опечатка и любая вредоносная программа получают полную власть над сервером. Поэтому в зрелых системах полные полномочия выдаются *узко и на время*: человеку разрешают выполнить конкретную команду с повышенными правами, а не превращают его в администратора.

В Linux это делает `sudo`. Правила хранятся в файле `/etc/sudoers` и каталоге `/etc/sudoers.d`. Правило называет, *кому* (пользователь или `%группа`), *на каких узлах*, *от чьего имени* и *какие команды* разрешено выполнять, и нужен ли пароль (`NOPASSWD`). Опасны правила с командой `ALL`, а также разрешающие запуск редакторов и оболочек: из `vim` или `less` можно выйти в оболочку и получить root. Для правки одного файла служит `sudoedit`. Ошибка синтаксиса в sudoers может лишить доступа ко всем административным действиям, поэтому файл всегда проверяют командой `visudo -c`.

В Windows главная граница — членство в группе **Administrators**. Узкие группы позволяют делегировать часть работы без полных прав: **Remote Desktop Users** (вход по RDP), **Event Log Readers** (чтение журналов), **Backup Operators** (резервное копирование). Дополнительно служит запрос **UAC**: даже администратор работает с урезанным токеном и подтверждает повышение. Полезная проверка при аудите — состав Administrators и таблица делегирования: нет ли правил «любая команда без пароля».

Linux: инструменты

```
getent group sudo                           # кто в группе sudo
sudo -l -U boris                            # что разрешено пользователю
grep -l 'NOPASSWD' /etc/sudoers.d/*         # подозрительные правила
visudo -c -f /etc/sudoers.d/60-webadmins    # проверка синтаксиса файла

# содержимое /etc/sudoers.d/60-webadmins (права файла 0440)
%webadmins ALL=(root) NOPASSWD: /usr/bin/systemctl reload nginx
```

Файлы в `sudoers.d` должны принадлежать root и иметь права 0440: при больших правах sudo отказывается читать их. Правило для одной команды указывает её полный путь и аргументы: `systemctl reload nginx` разрешено, `systemctl stop nginx` — нет. В учебном полигоне `sudo -l` умеет только показывать правила (никакие команды не выполняются).

Windows: инструменты

```
Get-LocalGroupMember -Group Administrators
Remove-LocalGroupMember -Group Administrators -Member olga
New-LocalGroup -Name webadmins -Description 'Администраторы веб-сервера'
```

```
Add-LocalGroupMember -Group 'Event Log Readers' -Member anna
Import-Csv delegation.csv | Where-Object { $_.Command -eq '*' }
```

В учебном полигоне делегирование представлено таблицей `srv\etc\delegation.csv`: в каждой строке — кто, какую команду и нужен ли пароль. Такие таблицы легко проверять автоматически — отбирать правила с командой «любая» или с запуском оболочки.

Что чаще всего идёт не так

«Временное» повышение прав имеет свойство становиться постоянным. При аудите ищут не только лишних членов группы администраторов, но и правила с `NOPASSWD: ALL`, доступ к оболочкам и редакторам, а также правила, выданные группе вместо людей без документации.

Практикум Linux: заявки модуля 4

Каталог полигона: `04_sudo`. Запуск: `lab 4`, затем `zayavki`.

Руководитель ИТ просит навести порядок с повышением полномочий. Разработчикам иногда нужно перезапустить веб-сервер, но давать им полный root нельзя. В `srv/etc/sudoers` и `srv/etc/sudoers.d` накопились правила разных лет, часть из них опасна, одно — с ошибкой. Учебные `sudo -l -U имя` и `visudo -c -f файл` разбирают эти файлы так же, как настоящие. Это последний модуль первого блока: последняя заявка (контрольная) использует приёмы модулей 2–4.

Заявка Linux 4.1. Кто в группе sudo

Запишите в `otvety/1` имена членов группы `sudo` из `srv/etc/group`, по одному в строке, отсортировав.

Заявка Linux 4.2. Что разрешено Борису

Сохраните в `otvety/2` вывод учебной команды `sudo -l -U boris`. В нём должна быть строка с полным доступом `(ALL : ALL) ALL`.

Заявка Linux 4.3. Опасное правило

Найдите в `srv/etc/sudoers.d` файл, выдающий пользователю полный доступ без пароля (`NOPASSWD: ALL`). Запишите имя файла (без пути) в `otvety/3`.

Заявка Linux 4.4. Только перезапуск nginx

Замените правило `anna` в `90-dev`: Анна должна без пароля перезапускать nginx командой `/usr/bin/systemctl restart nginx` и больше ничего не уметь. Проверьте `sudo -l -U anna`.

Заявка Linux 4.5. Ошибка в правиле

Файл `99-oops` содержит синтаксическую ошибку, из-за неё sudo на настоящем сервере отказался бы читать конфигурацию. Найдите ошибку командой `visudo -c -f srv/etc/sudoers.d/99-oops`, исправьте её. После ис-

правления Павел должен по-прежнему запускать `apt` без пароля.

Заявка Linux 4.6. Группа веб-администраторов

Создайте группу `webadmins`, добавьте в неё `vera` и создайте файл `srv/etc/sudoers.d/60-webadmins` с правилом: члены группы могут без пароля выполнять только `/usr/bin/systemctl reload nginx` от имени root. Права файла — `0440`.

Заявка Linux 4.7. Редакторы с побегом в оболочку

Из редактора `vim` и пейджера `less`, запущенных через sudo, можно выйти в оболочку root. Найдите в `srv/etc/sudoers.d` файлы с такими правилами (команды `vim`, `less`, `bash`, `sh`, `find`, `awk`) и запишите имена файлов в `otvety/7`, по одному в строке, отсортировав.

Заявка Linux 4.8. sudoedit вместо vim

Замените правило Веры в `92-editors`: вместо `vim` и `less` разрешите ей только `sudoedit /etc/nginx/nginx.conf` (правка через временную копию, без оболочки). Прочие права Веры из прежнего правила пропадают.

Заявка Linux 4.9. Журнал и защита терминала

Создайте `srv/etc/sudoers.d/00-defaults` (права `0440`) с двумя строками: `Defaults use_pty` и `Defaults logfile="/var/log/sudo.log"`. Файл должен проходить `visudo -c -f`.

Заявка Linux 4.10. Контрольная: кто остался с полным root

Составьте в файле `sudo-audit.txt` (в каталоге модуля) список всех учётных записей из `srv/etc/passwd`, которым sudo сейчас разрешает запустить `/bin/bash` (то есть полный доступ), по одному имени в строке, отсортировав. Список должен получиться «чистым»: после исправлений заданий 4, 5, 8 в нём остаются только `boris`, `dmitri` и `root`.

Практикум Windows: заявки модуля 4

Каталог полигона: `04_polnomochiya`. Запуск: `lab 4`, затем `zayavki`.

Руководитель ИТ просит навести порядок с повышением полномочий. В Windows права «админа» даёт членство в группе `Administrators`; кроме неё есть более узкие группы (`Remote Desktop Users`, `Event Log Readers`, `Backup Operators`), которые позволяют выполнять отдельные действия без полного контроля. На `SRV01` накопились лишние администраторы и таблица делегирования `srv\etc\delegation.csv` (какому пользователю какую команду разрешено запускать с повышением и нужен ли пароль), где часть правил опасна. Работаете с `Get-LocalGroupMember`, `Add-LocalGroupMember`, `Remove-LocalGroupMember`, `New-LocalGroup` и обычными `Import-Csv` / `Export-Csv`.

Заявка Windows 4.1. Кто администратор

Посмотрите состав группы `Administrators` (`Get-LocalGroupMember`). Кроме `Administrator` и `dmitri` там оказались двое «временных» администраторов. Запишите их имена (без `SRV01\`) в `otvety\1`, по одному в строке, отсортировав.

Заявка Windows 4.2. Лишить полномочий

Выведите этих двоих из группы `Administrators`. В группе должны остаться только `Administrator` и `dmitri`.

Заявка Windows 4.3. Узкие группы вместо администратора

Создайте локальную группу `webadmins` (описание `Администраторы веб-сервера`), добавьте в неё `vera`. Кроме того, чтобы `anna` могла читать журналы событий без прав администратора, добавьте её в группу `Event Log Readers`.

Заявка Windows 4.4. Опасные правила делегирования

В таблице `srv\etc\delegation.csv` правило с командой `*` разрешает запуск чего угодно. Найдите такие правила, где пароль **не требуется** (`NoPassword`), и запишите имена пользователей (столбец `Principal`) в `otvety\4`.

Заявка Windows 4.5. Сузить правило

Замените правило пользователя из задания 4: вместо `*` разрешите ему только команду `Restart-Service nginx` и оставьте `NoPassword`. Остальные правила не меняйте (таблицу перезапишите через `Export-Csv -NoTypeInformation -Encoding UTF8`).

Заявка Windows 4.6. Контрольная: акт по полномочиям

Подготовьте файл `audit-admin.txt` (в каталоге модуля) из двух строк: `ADMINS=` — имена членов группы `Administrators` через запятую без пробелов, отсортированные; `SHELL_RULES=` — число правил делегирования, где команда содержит `powershell.exe` или `cmd.exe` (запуск оболочки равносилен полному доступу).

Вопросы для размышления

1. Почему правило `NOPASSWD: ALL` считается опасным, даже если оно выдано доверенному сотруднику?
2. Как вы объясните разработчику, почему ему разрешили перезапуск службы, но не `vim` под sudo?

Глава 5

Модуль 5. Процессы и ресурсы

Идея и словарь

Процесс — запущенная программа вместе с её памятью, открытыми файлами и правами. У каждого процесса есть идентификатор (**PID**), владелец, родитель и приоритет. Именно процессы потребляют процессор и память, поэтому на вопрос «почему сервер тормозит» отвечает список процессов: кто занимает ресурсы, откуда запущен, чей он.

Что считать подозрительным. Программа, запущенная из временного или скрытого каталога (`/tmp`, `.cache`, `AppData`); процесс с именем, похожим на системный, но с другим путём; процесс, работающий от неожиданной учётной записи; программа, которая исчезла с диска, но продолжает работать. Путь файла программы — один из самых надёжных признаков.

Завершение процесса. В Linux это **сигнал**: **SIGTERM** (15) вежливо просит завершиться, давая программе время закрыть файлы; **SIGKILL** (9) обрывает процесс немедленно, и программа ничего не успевает сохранить. Правило: сначала SIGTERM, и лишь если процесс не реагирует, SIGKILL. В Windows принудительное завершение выполняет `Stop-Process -Force`; корректное завершение консольных программ обычно устроено как запрос окну или службе. Приоритет определяет, кому достаётся процессор в споре: в Linux — значение **nice** от -20 до 19 (чем больше, тем ниже приоритет), в Windows — класс приоритета (`Idle`, `BelowNormal`, `Normal`, `High`).

Фоновая работа. Программа, запущенная в терминале, зависит от него и завершится при закрытии. Чтобы она пережила выход, её запускают в фоне с защитой от сигнала разрыва (`nohup … &` в Linux, `Start-Process` в Windows) и перенаправляют вывод в файл.

Linux: инструменты

```
ps -ef | grep report-gen        # найти процесс; PID --- второй столбец
ps -o pid,ni,cmd -p 1234        # приоритет nice
kill 1234                       # SIGTERM (вежливо)
kill -9 1234                    # SIGKILL (принудительно)
pkill worker-pool               # все процессы с таким именем
renice -n 10 -p 1234            # понизить приоритет
sleep 300 &  echo $!            # запустить в фоне, PID последнего фонового
nohup ./long-job.sh > long.out 2>&1 &
```

Приём `grep '[r]eport-gen'` исключает из вывода сам `grep`: в строке команды grep встречается слово «[r]eport-gen», которое шаблону не соответствует. Обычный пользователь может только *повышать* nice своих процессов; понизить его обратно может лишь root.

Windows: инструменты

```
Get-Process -Name report-gen                    # свойства: Id, Path, CPU,
PriorityClass
```

```
Get-Process | Where-Object { $_.Path -like '*\Temp\*' }
Stop-Process -Id 1234 -Force
Stop-Process -Name worker-pool -Force
(Get-Process -Name nightly-backup).PriorityClass = 'BelowNormal'
Start-Process -FilePath ping.exe -ArgumentList '-n','300','127.0.0.1' -WindowStyle
Hidden -RedirectStandardOutput C:\log\ping.out
```

Свойство `Path` у процессов, запущенных от имени других пользователей или системных, может быть недоступно без прав администратора — это не ошибка. Файл запущенной программы Windows удалить не даёт: сначала процесс нужно остановить.

Что чаще всего идёт не так

Нельзя убивать процессы «по имени из головы». Сначала выясните PID, путь и владельца; затем решите, что делать. Завершение процесса базы данных сигналом SIGKILL — верный путь к повреждению файлов.

Практикум Linux: заявки модуля 5

Каталог полигона: `05_processy`. Запуск: `lab 5`, затем `zayavki`.

Пользователи жалуются, что сервер «подтормаживает». Вы заходите на `srv01` и находите несколько подозрительных процессов: один не реагирует на обычное завершение, второй запущен из временного каталога, третий мешает остальным. В этом модуле процессы **настоящие**: при подготовке полигон запускает несколько безобидных программ-«сонь» под служебными именами. Убить их можно только своими руками командами `ps`, `kill`, `renice`, `pkill`; на систему это не влияет.

Заявка Linux 5.1. Найти процесс

Пользователь жалуется, что программа `report-gen` «висит». Найдите её PID (идентификатор процесса) и запишите в `otvety/1`.

Заявка Linux 5.2. Процесс из временного каталога

Один из процессов запущен не из `srv/opt/app`, а из скрытого каталога `srv/tmp/.cache`: так обычно прячут вредоносные программы. Узнайте имя этого процесса (имя файла программы без пути) и запишите в `otvety/2`.

Заявка Linux 5.3. Вежливое завершение

Завершите процесс `report-gen` корректно — сигналом SIGTERM (`kill PID`): так программа успевает закрыть файлы.

Заявка Linux 5.4. Упрямый процесс

Процесс `stuck-import` игнорирует обычное завершение. Сначала убедитесь в этом: отправьте `kill PID` и проверьте, что процесс остался. Затем завершите его принудительно сигналом SIGKILL (`kill -9`).

Заявка Linux 5.5. Устранить вредителя

Завершите процесс из `srv/tmp/.cache` и удалите его файл (до удаления файла запомните имя из задания 2).

Заявка Linux 5.6. Понизить приоритет

Резервное копирование `nightly-backup` мешает остальным. Не завершая процесс, понизьте его приоритет до значения nice, равного 10 (`renice`).

Заявка Linux 5.7. Задача в фоне

Запустите в фоне команду `sleep 300` (знак `&` в конце) и запишите её PID в `otvety/7`. Процесс должен продолжать работать.

Заявка Linux 5.8. Пережить выход из системы

Сценарий `srv/opt/job/long-job.sh` пишет строку «tick» каждую секунду. Запустите его так, чтобы он продолжал работать после закрытия терминала: через `nohup`, в фоне, с выводом в файл `srv/var/log/long-job.out`. Файл должен продолжать расти.

Заявка Linux 5.9. Завершить группу процессов

Запущено три копии `worker-pool`. Завершите их все одной командой по имени (`pkill worker-pool` или `killall`).

Заявка Linux 5.10. Контрольная: что осталось

Составьте файл `processes.txt` (в каталоге модуля): имена всех запущенных программ из каталога `srv/opt/app`, по одному в строке, без повторов, отсортированные. После выполнения заданий 3–6, 9 должна остаться только одна.

Практикум Windows: заявки модуля 5

Каталог полигона: `05_processy`. Запуск: `lab 5`, затем `zayavki`.

Пользователи жалуются, что сервер «подтормаживает». Вы заходите на `SRV01` и находите несколько подозрительных процессов: один «завис», второй запущен из временного каталога, третий мешает остальным. В этом модуле процессы **настоящие**: при подготовке полигон запускает несколько безобидных программ-«сонь» (копии `ping.exe` под служебными именами). Убить их можно только своими руками командами `Get-Process`, `Stop-Process`, `Start-Process`; на систему это не влияет. Свойство `Path` процесса показывает, откуда запущена программа.

Заявка Windows 5.1. Найти процесс

Пользователь жалуется, что программа `report-gen` «висит». Найдите её PID (идентификатор процесса, свойство `Id`) командой `Get-Process` и запишите в `otvety\1`.

Заявка Windows 5.2. Процесс из временного каталога

Один из процессов запущен не из `srv\app`, а из скрытого каталога `srv\Temp\.cache`: так обычно прячут вредоносные программы. Найдите его по свойству `Path` и запишите имя процесса (свойство `Name`, без `.exe`) в `otvety\2`.

Заявка Windows 5.3. Завершить процесс

Завершите процесс `report-gen` командой `Stop-Process`.

Заявка Windows 5.4. Устранить вредителя

Завершите процесс из `srv\Temp\.cache` и удалите его файл. Удалить файл запущенной программы Windows не даст, так что порядок действий очевиден.

Заявка Windows 5.5. Понизить приоритет

Резервное копирование `nightly-backup` мешает остальным. Не завершая процесс, понизьте его приоритет до `BelowNormal` (свойство `PriorityClass`).

Заявка Windows 5.6. Завершить группу процессов

Запущено три копии `worker-pool`. Завершите их все одной командой.

Заявка Windows 5.7. Фоновая программа с выводом в файл

Запустите в фоне, без окна, команду `ping.exe -n 300 127.0.0.1` с перенаправлением вывода в файл `srv\var\log\ping.out` (`Start-Process` с параметрами `-WindowStyle Hidden`, `-RedirectStandardOutput`). Файл должен продолжать расти.

Заявка Windows 5.8. Контрольная: что осталось

Составьте файл `processes.txt` (в каталоге модуля): имена всех запущенных процессов, программы которых лежат в `srv\app`, без повторов, отсортированные. После заданий 3–6 должны остаться две: `nightly-backup` и `stuck-import`, который вы не трогали.

Вопросы для размышления

1. Почему сигнал SIGKILL считается крайней мерой, а SIGTERM — нормальной?
2. Чем опасно удалять файл запущенной программы до завершения процесса и зачем вообще нужен nice?

Глава 6

Модуль 6. Пакеты и репозитории

Идея и словарь

Пакет — архив с программой, её настройками по умолчанию и описанием: версия, зависимости, список файлов. **Пакетный менеджер** устанавливает, обновляет и удаляет пакеты, сам подтягивая то, без чего программа не работает (**зависимости**), и ведёт учёт, какой файл принадлежит какому пакету. Пакеты берутся из **репозитория** — каталога, который описывается в настройках источников; если адрес источника перестал работать, обновление и установка прекращаются.

Зачем всё это администратору. Установка из пакета повторяема и обратима: можно узнать, что и откуда установлено, и убрать установленное без остатка. Обновления закрывают уязвимости, поэтому устаревшие пакеты — прямая угроза. Но обновление может изменить поведение программы, поэтому версию важных пакетов иногда **закрепляют** и обновляют осознанно, после проверки.

Основные действия одинаковы в любой системе: обновить сведения об источниках, найти пакет, посмотреть сведения о нём, установить, посмотреть, что устарело, обновить, удалить. Различаются только команды. Отдельно различают *удаление* (программа убрана, настройки остались) и *полное удаление* (убраны и настройки). Пакеты, установленные ради зависимостей, помечаются как *автоматические*; когда они больше не нужны, их можно убрать вместе с зависимым пакетом.

Linux: инструменты

```
sudo apt update                  # обновить сведения об источниках
apt search nginx
apt show nginx
sudo apt install nginx tmux
dpkg -l | grep '^ii'             # что установлено
dpkg -S /usr/bin/htop            # какому пакету принадлежит файл
dpkg -L nginx-common             # какие файлы входят в пакет
apt list --upgradable            # что устарело
sudo apt-mark hold openssl       # закрепить версию
sudo apt upgrade
sudo apt purge telnet            # удалить вместе с настройками
apt-mark showmanual              # пакеты, установленные вручную
```

Файл `/etc/apt/sources.list` (и файлы в `sources.list.d`) перечисляют источники. Если `apt update` сообщает о недоступном источнике, причина почти всегда в его адресе: сначала прочитайте сообщение, потом правьте файл. Ключ `-y` отвечает «да» на вопросы, поэтому пользуйтесь им осознанно.

Windows: инструменты

```
winget search notepad
winget show --id Notepad++.Notepad++
winget install --id Notepad++.Notepad++
```

```
winget list                          # установленные программы
winget upgrade                       # что можно обновить
winget pin add --id Microsoft.PowerShell
winget upgrade --all
winget uninstall --id Oracle.JavaRuntimeEnvironment
```

Идентификатор вида `Издатель.Программа` — самый надёжный способ указать пакет: названия бывают неоднозначными. Менеджер `winget` встроен в современные Windows 10/11; на серверах Windows Server его может не быть, и тогда используют другие средства (`PackageManagement`, `msiexec`, групповые политики). В полигоне модель `winget` работает с локальным каталогом.

Что чаще всего идёт не так

Удаляя пакет, убедитесь, что от него ничего не зависит: менеджер предупредит о зависимых пакетах, но не знает о ваших сценариях и заданиях планировщика. Установка из непроверенного источника — прямой путь к вредоносной программе.

Практикум Linux: заявки модуля 6

Каталог полигона: `06_pakety`. Запуск: `lab 6`, затем `zayavki`.

На `srv01` пора навести порядок с программами: источник пакетов давно закрыт, часть пакетов устарела, остался небезопасный `telnet`, а на новый веб-сервер нужно поставить `nginx`. Учебные `apt`, `apt-get`, `apt-cache`, `apt-mark` и `dpkg` работают с локальной базой пакетов в `srv/var/lib` (`available` — что есть в репозитории, `status` — что установлено) и понимают привычные команды и ключи.

Заявка Linux 6.1. Сколько пакетов установлено

Определите, сколько пакетов установлено на `srv01` (учебная команда `dpkg -l`: у установленных пакетов строка начинается с `ii`). Запишите число в `otvety/1`.

Заявка Linux 6.2. Чей это файл

На сервере установлена одна из утилит `htop`, `tree`, `ncdu`. Командой `dpkg -S` найдите пакет по пути файла (`/usr/bin/htop`, `/usr/bin/tree` или `/usr/bin/ncdu`; подойдёт только тот, что установлен) и запишите имя пакета в `otvety/2`.

Заявка Linux 6.3. Починить источник

`apt update` завершается ошибкой: адрес в `srv/etc/apt/sources.list` ведёт на закрытый архив (`old-releases.example`). Замените его на `archive.example`, чтобы `apt update` проходил без ошибок.

Заявка Linux 6.4. Установить nginx и tmux

Установите пакеты `nginx` и `tmux` (`apt install`). Убедитесь, что оба появились в `dpkg -l`.

Заявка Linux 6.5. Что подтянулось само

При установке `nginx` apt добавил зависимости. Запишите в `otvety/5` имена пакетов, поставленных **автоматически** (`apt-mark showauto`), по одному в строке, отсортировав.

Заявка Linux 6.6. Убрать telnet

Протокол telnet передаёт пароли открытым текстом. Удалите пакет `telnet` вместе с конфигурацией (`apt purge`).

Заявка Linux 6.7. Что нужно обновить

Выясните, какие установленные пакеты имеют более новую версию в репозитории (`apt list --upgradable`). Запишите их имена в `otvety/7`, по одному в строке, отсортировав.

Заявка Linux 6.8. Обновить всё, кроме openssl

Руководитель просит обновить все пакеты, но `openssl` пока должен остаться прежней версии: на него завязано приложение, которое ещё не проверено на новой версии. Зафиксируйте `openssl` (`apt-mark hold`), затем выполните `apt upgrade`.

Заявка Linux 6.9. Файлы конфигурации пакета

Файлы каких путей, начинающихся с `/etc`, входят в пакет `nginx-common`? Получите список командой `dpkg -L` и запишите пути в `otvety/9`, отсортировав.

Заявка Linux 6.10. Контрольная: список установленного вручную

Подготовьте для аудита файл `packages.txt` (в каталоге модуля): имена пакетов, которые установлены **вручную** (то есть не как зависимости), по одному в строке, отсортированные. Файл должен соответствовать текущему состоянию системы: `telnet` в нём уже не должно быть, а `nginx` и `tmux` должны быть.

Практикум Windows: заявки модуля 6

Каталог полигона: `06_pakety`. Запуск: `lab 6`, затем `zayavki`.

На `SRV01` пора навести порядок с программами: часть устарела, остался небезопасный старый Java, а для работы нужны `Notepad++` и `7-Zip`. В Windows для этого служит менеджер пакетов `winget`. Учебный `winget` в полигоне работает с локальной базой `srv\var\lib\winget` (`available.csv` — что есть в каталоге, `installed.csv` — что установлено) и понимает привычные команды: `list`, `search`, `show`, `install`, `uninstall`, `upgrade`, `pin`. Ничего на вашем настоящем компьютере он не меняет.

Заявка Windows 6.1. Сколько программ установлено

Определите, сколько пакетов установлено на `SRV01` по выводу `winget list` (одна строка таблицы — один пакет, заголовок не считается). Запишите число в `otvety\1`.

Заявка Windows 6.2. Найти нужный пакет

Найдите в каталоге пакет редактора Notepad++ командой `winget search notepad` и запишите его **идентификатор** (первый столбец) в `otvety\2`.

Заявка Windows 6.3. Установить программы

Установите `Notepad++` и `7-Zip` по идентификаторам (`winget install --id …`). Убедитесь, что они появились в `winget list`.

Заявка Windows 6.4. Убрать старую Java

Старый Java 8 (`Oracle.JavaRuntimeEnvironment`) больше не нужен и небезопасен. Удалите его (`winget uninstall --id …`).

Заявка Windows 6.5. Что нужно обновить

Выясните, у каких установленных пакетов в каталоге есть более новая версия (`winget upgrade` без параметров). Запишите идентификаторы в `otvety\5`, по одному в строке, отсортировав.

Заявка Windows 6.6. Обновить всё, кроме PowerShell

Руководитель просит обновить все пакеты, но `Microsoft.PowerShell` пока остаётся прежней версии (на неё завязаны сценарии). Закрепите его (`winget pin add --id …`), затем выполните `winget upgrade --all`. После этого обновлений должно остаться только для закреплённого пакета.

Заявка Windows 6.7. Контрольная: список для аудита

Подготовьте файл `packages.txt` (в каталоге модуля): идентификаторы всех установленных пакетов, по одному в строке, отсортированные. `Notepad++` и `7-Zip` в нём должны быть, Java — нет.

Вопросы для размышления

1. Почему при удалении telnet важно ещё и проверить, не нужен ли он другим службам (скриптам, мониторингу)?
2. В каких случаях разумно зафиксировать версию пакета (`hold`), и чем это грозит безопасности?

Глава 7

Модуль 7. Службы и автозапуск (systemd)

Идея и словарь

Служба (в Linux — *демон*) — программа, работающая в фоне без участия пользователя: веб-сервер, база данных, планировщик. За запуск, остановку, перезапуск и автозапуск служб отвечает **менеджер служб**: в Linux это **systemd**, в Windows — **диспетчер служб**. Он знает, какую программу запускать, от чьего имени, какие службы должны быть запущены раньше и что делать при сбое.

Описание службы — маленький текстовый файл (в systemd это **unit-файл**, в Windows — запись в реестре, которой управляет `sc.exe`). В нём три смысловые группы: *что запускать* (путь к программе), *от чьего имени и с какими условиями* (пользователь, рабочий каталог, файл настроек, зависимости) и *что делать при сбое* (перезапуск через заданное время). Менеджер читает описание при загрузке или по команде перечитать конфигурацию; после изменения файла без такой команды работает *старое* описание — классическая причина «я же всё исправил, а оно не запускается».

Служба может быть *запущена* (сейчас работает), и независимо от этого *включена в автозапуск* (стартует при загрузке). Это два разных свойства. Когда служба не запускается, действуют строго в таком порядке: прочитать статус, прочитать журнал службы, найти в нём причину, исправить, перечитать конфигурацию, запустить, проверить статус. Типичные причины: неверный путь к программе, отсутствующий файл настроек, несуществующая учётная запись, незапущенная зависимость, нехватка места на диске.

Linux: инструменты

```
systemctl list-units --failed          # какие службы упали
systemctl status shop-api              # состояние и последние строки журнала
journalctl -u shop-api -n 20           # журнал службы
systemctl cat shop-api                 # показать unit-файл
sudo systemctl daemon-reload           # перечитать unit-файлы
sudo systemctl start shop-api
sudo systemctl enable --now hello      # автозапуск и запуск сразу
sudo systemctl disable --now bluetooth

# /etc/systemd/system/hello.service
[Unit]
Description=Hello
[Service]
ExecStart=/usr/local/bin/hello.sh
Restart=on-failure
RestartSec=5
[Install]
WantedBy=multi-user.target
```

Код вида `203/EXEC` в статусе означает, что systemd не смог выполнить программу из `ExecStart` (нет файла или формат неверен); `217/USER` — не удалось опре-

делить пользователя из `User=`. Раздел `[Install]` нужен только для автозапуска: без него `enable` сообщит об ошибке.

Windows: инструменты

```
Get-Service | Where-Object { $_.StartType -eq 'Automatic' -and $_.Status -eq
'Stopped' }
Get-WinEvent -FilterHashtable @{ LogName = 'System'; Level = 2 } -MaxEvents 10
sc.exe qc shop-api                                  # настройки службы
sc.exe config shop-api binPath= "C:\Program Files\Shop\shop-api.exe"
Start-Service shop-api
Stop-Service bluetooth
Set-Service bluetooth -StartupType Disabled
sc.exe failure flaky reset= 86400 actions= restart/5000
```

Синтаксис `sc.exe` необычен: после имени параметра идёт знак `=`, и значение пишется *после пробела*. События запуска и остановки служб пишутся в журнал System от источника `Service Control Manager`: код `7000` — служба не запустилась, `7001` — не запустилась зависимость, `7038` — не удалось войти в систему под учётной записью службы, `7036` — смена состояния.

Что чаще всего идёт не так

Перезапуск ради «а вдруг поможет» без чтения журнала — пустая трата времени и риск потерять важные сообщения. После изменения описания службы не забудьте перечитать конфигурацию (`daemon-reload`).

Практикум Linux: заявки модуля 7

Каталог полигона: `07_sluzhby`. Запуск: `lab 7`, затем `zayavki`.

После обновления на `srv01` не запустились три службы: интернет-магазин (`shop-api`), генератор отчётов (`report-worker`) и служба синхронизации (`legacy-sync`). Кроме того, нужно подключить новую небольшую службу и отключить лишнюю. Работа идёт с учебными `systemctl` и `journalctl`: они читают настоящие unit-файлы из `srv/etc/systemd/system` и ведут состояние в `srv/var/lib/systemd`, поэтому ошибки в unit-файле дают те же сообщения, что и в настоящем systemd (коды `203/EXEC`, `217/USER`, `205/EXEC`, напоминание о `daemon-reload`).

Заявка Linux 7.1. Какие службы упали

Выясните, какие службы находятся в состоянии `failed` (`systemctl list-units --failed`). Запишите их имена без суффикса `.service` в `otvety/1`, по одному в строке, отсортировав.

Заявка Linux 7.2. Код ошибки магазина

Службу `shop-api` не удаётся запустить. Командой `systemctl status shop-api` найдите код, с которым завершился процесс (в строке `Process:` в скобках `status=…`). Запишите его в `otvety/2`.

Заявка Linux 7.3. Исправить магазин

Запустите `shop-api`. Причина — опечатка в пути `ExecStart` в unit-файле. Исправьте файл `srv/etc/systemd/system/shop-api.service`, выполните `systemctl daemon-reload` и запустите службу. Убедитесь, что `systemctl is-active shop-api` выводит `active`.

Заявка Linux 7.4. Служба без пользователя

`report-worker` падает с кодом `217/USER`: в unit-файле указан `User=svc-report`, а такой учётной записи нет. Создайте системного пользователя `svc-report` без входа в систему (`useradd -r -s /usr/sbin/nologin svc-report`) и запустите службу. Менять unit-файл не нужно.

Заявка Linux 7.5. Нет файла настроек

`legacy-sync` не стартует (`205/EXEC`): не найден файл `EnvironmentFile`. Создайте `srv/etc/legacy-sync.env` с одной строкой `SYNC_HOST=backup.local` и запустите службу.

Заявка Linux 7.6. Неудачные входы по SSH

В журнале службы `ssh` есть записи о неудачных входах (`Failed password`). Командой `journalctl -u ssh` посчитайте, сколько их, и запишите число в `otvety/6`.

Заявка Linux 7.7. Новая служба hello

Создайте unit-файл `srv/etc/systemd/system/hello.service` для программы `/usr/local/bin/hello.sh`: описание `Hello`, `ExecStart=/usr/local/bin/hello.sh`, `Restart=on-failure`, раздел `[Install]` с `WantedBy=multi-user.target`. Загрузите изменения, включите автозапуск и запустите службу.

Заявка Linux 7.8. Перезапуск при сбое

Служба `flaky` иногда падает, а в unit-файле `Restart=no`. Задайте `Restart=on-failure` и `RestartSec=5` (пауза 5 секунд), затем примените изменения (`systemctl daemon-reload`).

Заявка Linux 7.9. Отключить лишнее

На сервере не нужен `bluetooth`. Остановите службу и уберите её из автозапуска (`systemctl disable --now bluetooth`).

Заявка Linux 7.10. Контрольная: что стартует при загрузке

Составьте файл `services.txt` (в каталоге модуля): имена служб, включённых в автозапуск (`systemctl list-unit-files --state=enabled`), без суффикса `.service`, по одному в строке, отсортированные.

Практикум Windows: заявки модуля 7

Каталог полигона: `07_sluzhby`. Запуск: `lab 7`, затем `zayavki`.

После обновления на SRV01 не запустились три службы: интернет-магазин (`shop-api`), генератор отчётов (`report-worker`) и служба синхронизации (`legacy-sync`). Кроме того, нужно настроить автоперезапуск нестабильной службы и отключить лишнюю. Работа идёт с учебными `Get-Service`, `Start-Service`, `Set-Service`, `sc.exe` и `Get-WinEvent`: они читают таблицу служб `srv\etc\services.csv` и журнал событий `srv\var\log\eventlog.csv` и выдают те же сообщения, что и настоящий диспетчер служб (коды событий 7000, 7001, 7038). Для служб, которые не удаётся запустить, причина записывается в журнал System.

Заявка Windows 7.1. Какие службы не работают

Выясните, какие службы настроены на автоматический запуск (`StartType` равен `Automatic`), но сейчас остановлены (`Status` равен `Stopped`). Запишите их имена в `otvety\1`, по одному в строке, отсортировав.

Заявка Windows 7.2. Код ошибки магазина

Службу `shop-api` не удаётся запустить. Найдите в журнале System (`Get-WinEvent -FilterHashtable @{LogName='System'; Level=2}`) последнюю ошибку, относящуюся к `shop-api`, и запишите её **код события** (`Id`) в `otvety\2`.

Заявка Windows 7.3. Исправить магазин

Причина в журнале: служба указывает на несуществующий файл `shop-api-v2.exe`, а файл называется `shop-api.exe`. Исправьте путь командой `sc.exe config shop-api binPath= "C:\Program Files\Shop\shop-api.exe"` (пробел после = обязателен) и запустите службу.

Заявка Windows 7.4. Служба без пользователя

`report-worker` не стартует (событие 7038): служба работает от имени `SRV01\svc-report`, а такой учётной записи нет. Создайте пользователя `svc-report` с произвольным паролем (`New-LocalUser`, `-Password` принимает `SecureString`) и запустите службу. Менять настройки службы не нужно.

Заявка Windows 7.5. Зависимая служба

`legacy-sync` не стартует (событие 7001): она зависит от службы `queue-svc`, которая остановлена. Запустите сначала `queue-svc`, затем `legacy-sync`. Обе должны работать.

Заявка Windows 7.6. Неудачные входы

В журнале Security есть записи о неудачных входах (событие с кодом 4625). Сосчитайте их (`Get-WinEvent -FilterHashtable @{LogName='Security'; Id=4625}`) и запишите число в `otvety\6`.

Заявка Windows 7.7. Перезапуск при сбое

Служба `flaky` иногда падает. Настройте восстановление: после сбоя перезапускать через 5 секунд, счётчик сбоев сбрасывать через сутки: `sc.exe failure flaky reset= 86400 actions= restart/5000`.

Заявка Windows 7.8. Отключить лишнее

На сервере не нужна служба `bluetooth`. Остановите её (`Stop-Service`) и запретите запуск (`Set-Service -StartupType Disabled`).

Заявка Windows 7.9. Контрольная: что стартует при загрузке

Составьте файл `services-auto.txt` (в каталоге модуля): имена служб с типом запуска `Automatic`, по одному в строке, отсортированные. `bluetooth` в нём быть не должно.

Вопросы для размышления

1. Почему после правки unit-файла нужен `daemon-reload`, и что произойдёт, если его забыть?
2. Чем `restart` отличается от `reload`, и когда лучше использовать каждый из них?

Глава 8

Модуль 8. Журналы и расследование

Идея и словарь

Журнал — хронологическая запись событий: какое обращение пришло, кто вошёл, что упало. Журналы — основной инструмент расследования: ни одна атака и ни один сбой не проходят бесследно. Умение быстро задавать журналу вопросы и получать числа — ядро навыка администратора.

Журнал доступа веб-сервера содержит по строке на запрос: адрес клиента, время, метод и путь, код ответа, размер ответа, время обработки. Коды ответа делят на классы: 2xx — успех, 3xx — перенаправление, **401** и **403** — отказ в доступе, **404** — страница не найдена, 5xx — ошибка сервера. По журналу видно два характерных признака атаки. **Подбор пароля**: много запросов `POST /login` с кодом 401 с одного адреса. **Сканирование**: много ответов 404 на несуществующие, но «интересные» пути (`/wp-login.php`, `/.env`, `/phpmyadmin`).

Типовые вопросы к журналу: сколько всего запросов, какой адрес самый активный, сколько ошибок, какой путь запрашивают чаще всего, какой запрос самый медленный, сколько байт отдано. Все они решаются одной схемой: *выбрать нужное поле, сгруппировать, посчитать, упорядочить*. Порог, после которого адрес считают подозрительным (в заданиях — 20 неудачных входов или 50 ответов 404), выбирают так, чтобы отделить забывчивого пользователя от автоматического перебора.

Журналы растут бесконечно, поэтому их **ротируют**: старые файлы сжимают и со временем удаляют. В Linux за это отвечает `logrotate`, в Windows подобную работу выполняют сценарии и политики журналов.

Linux: инструменты

```
wc -l access.log                                          # всего запросов
awk '{print $1}' access.log | sort | uniq -c | sort -rn | head -n 5    # топ
адресов
awk '$9 == 404' access.log | wc -l                        # ответов 404
awk '{s += $10} END {print s}' access.log                 # суммарно отдано байт
grep 'POST /login' access.log | awk '$9 == 401 {print $1}' | sort | uniq -c | awk
'$1 > 20'
sort -k11 -n -r access.log | head -n 1                    # самый медленный запрос
grep 'Failed password' auth.log | awk '{print $9}' | sort | uniq -c | sort -rn

# /etc/logrotate.d/shop
/var/log/shop/*.log {
    weekly
    rotate 8
    compress
    missingok
    notifempty
}
```

Номер поля в `awk` считается от единицы, а разделитель по умолчанию — пробел. Цепочка `sort | uniq -c | sort -rn` — классический способ посчитать повторы и получить частые значения первыми: `uniq` умеет считать только *соседние*

одинаковые строки, поэтому перед ним всегда `sort`.

Windows: инструменты

```
$l = Get-Content access.log | ForEach-Object {
    $f = $_ -split ' '
    [pscustomobject]@{ Ip = $f[2]; Method = $f[3]; Path = $f[4]; Status =
    [int]$f[5]; Bytes = [int64]$f[6]; Time = [double]$f[7] }
}
$l | Group-Object Ip | Sort-Object Count -Descending | Select-Object -First 5
@($l | Where-Object Status -eq 404).Count
($l | Measure-Object Bytes -Sum).Sum
$l | Where-Object { $_.Method -eq 'POST' -and $_.Status -eq 401 } | Group-Object
Ip | Where-Object Count -gt 20
$l | Sort-Object Time -Descending | Select-Object -First 1
Get-WinEvent -FilterHashtable @{ LogName = 'Security'; Id = 4625 } |
ForEach-Object Message
```

Разбор строки в объект с именованными полями делает последующие запросы читаемыми: вместо номеров полей — имена. Событие 4625 журнала Security — неудачный вход; 4624 — успешный. Сообщение события можно разбирать регулярным выражением: `if ($_.Message -match 'Имя: ([^;]+);') { $Matches[1] }`.

Что чаще всего идёт не так

Нельзя блокировать адрес по одной строке журнала: общий для офиса адрес может принадлежать десятку добросовестных сотрудников. Перед блокировкой сопоставьте частоту, типы запросов и коды ответа. И не удаляйте журналы до того, как закончено расследование.

Практикум Linux: заявки модуля 8

Каталог полигона: `08_zhurnaly`. Запуск: `lab 8`, затем `zayavki`.

Интернет-магазин ночью «тормозил», а утром служба безопасности нашла в журнале доступа странные запросы. Вам передали журнал веб-сервера `srv/var/log/nginx/access.log` и журнал входов `srv/var/log/auth.log`. Их содержимое **у каждого своё** (адреса, имена, количества генерируются по вашему `.seed`), поэтому чужие ответы не подойдут. Вся работа — командами `grep`, `awk`, `sort`, `uniq`, `wc`, `cut`. Формат строки журнала доступа: `IP - - [дата] "МЕТОД путь HTTP/1.1" код байты время_ответа`.

Заявка Linux 8.1. Сколько было запросов

Определите общее число запросов в `srv/var/log/nginx/access.log` (одна строка — один запрос). Запишите число в `otvety/1`.

Заявка Linux 8.2. Самый активный адрес

Найдите IP-адрес, с которого пришло больше всего запросов, и запишите его в `otvety/2`.

Заявка Linux 8.3. Сколько страниц не найдено

Сколько запросов завершилось кодом 404? Запишите число в `otvety/3`.

Заявка Linux 8.4. Самый популярный путь

Какой путь (седьмое поле) запрашивали чаще всего? Запишите его в `otvety/4`. Подумайте, нормально ли это для интернет-магазина.

Заявка Linux 8.5. Подбор паролей на сайте

Подбор пароля на странице входа оставляет много ответов `401` на `POST /login`. Найдите адреса, у которых таких запросов больше 20, и запишите их в `otvety/5`, по одному в строке, отсортировав.

Заявка Linux 8.6. Сканер уязвимостей

Другой адрес перебирает несуществующие страницы (`/wp-login.php`, `/.env`…): получает много ответов 404. Найдите адреса с более чем 50 ответами 404 и запишите их в файл `blocklist.txt` (в каталоге модуля), по одному в строке.

Заявка Linux 8.7. Самый медленный запрос

Последнее поле каждой строки — время ответа в секундах. Найдите самый медленный запрос и запишите его путь (седьмое поле) в `otvety/7`.

Заявка Linux 8.8. Объём трафика

Подсчитайте суммарное число отданных байт (десятое поле) по всем запросам и запишите это целое число в `otvety/8`.

Заявка Linux 8.9. Кого подбирали по SSH

В `srv/var/log/auth.log` найдите имя учётной записи, на которую приходится больше всего неудачных входов (`Failed password for ИМЯ from …`). Запишите имя в `otvety/9`.

Заявка Linux 8.10. Ротация журналов

Файл `srv/etc/logrotate.d/shop` настроен неполно. Допишите настройку для `/var/log/shop/*.log`: хранить 8 копий (`rotate 8`), ротировать еженедельно (`weekly`), сжимать (`compress`), не считать ошибкой отсутствие файла (`missingok`) и не ротировать пустые (`notifempty`). Проверьте командой `logrotate-check`.

Практикум Windows: заявки модуля 8

Каталог полигона: `08_zhurnaly`. Запуск: `lab 8`, затем `zayavki`.

Интернет-магазин ночью «тормозил», а утром служба безопасности нашла в журнале доступа странные запросы. Вам передали журнал веб-сервера `srv\var\log\iis\access.log` и журнал событий безопасности (`Get-WinEvent`, события 4625 — неудачные входы). Их содержимое **у каждого своё** (адреса, имена, количества генерируются по вашему `.seed`), чужие ответы не подойдут. Работа ведётся конвейерами PowerShell: `Get-Content`, `ForEach-Object`, `Where-Object`, `Group-Object`, `Sort-Object`, `Measure-Object`, `Select-String`. Формат строки жур-

нала, поля разделены пробелом: `дата время IP МЕТОД путь код байты время_ответа` (индексы после `-split ' '`: 2 — IP, 3 — метод, 4 — путь, 5 — код, 6 — байты, 7 — время ответа в секундах).

Заявка Windows 8.1. Сколько было запросов

Определите общее число запросов в `srv\var\log\iis\access.log` (одна строка — один запрос). Запишите число в `otvety\1`.

Заявка Windows 8.2. Самый активный адрес

Найдите IP-адрес, с которого пришло больше всего запросов, и запишите его в `otvety\2`. Удобно разобрать каждую строку в объект, например: `$l = Get-Content srv\\var\\log\\iis\\access.log | ForEach-Object { $f = $_ -split ' '; [pscustomobject]@{ Ip = $f[2]; Method = $f[3]; Path = $f[4]; Status = [int]$f[5]; Bytes = [int64]$f[6]; Time = [double]$f[7] } }`, и затем сгруппировать (`Group-Object Ip`).

Заявка Windows 8.3. Сколько страниц не найдено

Сколько запросов завершилось кодом 404? Запишите число в `otvety\3`.

Заявка Windows 8.4. Самый популярный путь

Какой путь запрашивали чаще всего? Запишите его в `otvety\4`. Подумайте, нормально ли это для интернет-магазина.

Заявка Windows 8.5. Подбор паролей на сайте

Подбор пароля на странице входа оставляет много ответов `401` на `POST /login`. Найдите адреса, у которых таких запросов больше 20, и запишите их в `otvety\5`, по одному в строке. Адрес с единичными ошибками — просто забывчивый пользователь.

Заявка Windows 8.6. Сканер уязвимостей

Другой адрес перебирает несуществующие страницы (`/wp-login.php`, `/.env`...): получает много ответов 404. Найдите адреса с более чем 50 ответами 404 и запишите их в файл `blocklist.txt` (в каталоге модуля), по одному в строке.

Заявка Windows 8.7. Самый медленный запрос

Последнее поле каждой строки — время ответа в секундах. Найдите самый медленный запрос и запишите его путь в `otvety\7`.

Заявка Windows 8.8. Объём трафика

Подсчитайте суммарное число отданных байт (шестое поле) по всем запросам и запишите это целое число в `otvety\8`.

Заявка Windows 8.9. Кого подбирали по паролю

Среди событий 4625 журнала Security найдите имя учётной записи (в тексте сообщения после `Имя:`), на которую приходится больше всего неудачных входов, и запишите имя в `otvety\9`.

Заявка Windows 8.10. Архив старых журналов

В `srv\var\log\old` лежат журналы разного возраста. Журналы старше 30 дней упакуйте в один архив `srv\var\log\old\archive.zip` (`Compress-Archive`) и удалите исходные файлы. Свежие журналы остаются как есть.

Вопросы для размышления

1. Почему адрес с пятью ошибками входа не стоит блокировать, а адрес со ста — стоит? Где проходит граница?
2. Какие ещё источники, кроме журнала веб-сервера, вы бы изучили при расследовании подбора паролей?

Глава 9

Модуль 9. Диски, место и файловые системы

Идея и словарь

Место на диске — ресурс, который неизбежно кончается. Когда оно кончается, программы перестают писать журналы и данные, службы падают, а база данных может повредиться. Поэтому администратор умеет быстро ответить на три вопроса: *какой диск заполнен*, *что на нём занимает место* и *что из этого можно безопасно удалить*.

Диск делят на **разделы**, на разделах создают **файловые системы**, а те подключают к дереву каталогов (в Linux — **монтируют** в каталог, в Windows — присваивают букву). Информация о том, что подключать при загрузке, хранится в `/etc/fstab`; в строке записывают устройство (надёжнее всего по **UUID**, а не по имени вроде `/dev/sdb1`, которое может измениться), точку монтирования, тип, параметры и два числа. Параметр `nofail` не даёт серверу «зависнуть» при загрузке, если диска нет.

Место может кончиться двумя разными способами. Первый — кончились *байты*: большие файлы, разросшиеся журналы, дампы и архивы. Второй — кончились **inode** (структуры, описывающие файлы): их число задаётся при создании файловой системы, и каталог с сотнями тысяч мелких файлов (например, почтовая очередь) исчерпывает их при свободном месте в гигабайты. Признак: запись невозможна, а `df -h` показывает свободное место; ответ даёт `df -i`. В NTFS аналогичной нехватки практически не бывает, но каталоги с очень большим числом файлов сильно замедляют работу.

Безопасная очистка идёт в таком порядке: найти крупнейшие каталоги и файлы, выяснить, что это и кому нужно, убедиться, что есть копия, удалять *самое очевидное* (разросшиеся отладочные журналы, архивы старше срока хранения), проверить результат. Удаление «чтобы освободить место» без понимания причины лишь откладывает следующую остановку.

Linux: инструменты

```
df -h                          # место по файловым системам
df -i                          # inode
du -sk /var/* | sort -rn | head       # крупнейшие подкаталоги
find /var -type f -exec stat -c '%s %n' {} + | sort -rn | head -n 3
ls /var/spool/mailq | wc -l
find /var/spool/mailq -type f -delete                      # безопасно при тысячах
файлов
find /var/log/app -name '*.gz' -mtime +30 -delete          # старше 30 суток
lsblk                          # диски и разделы
blkid                          # UUID разделов

# строка /etc/fstab
UUID=3b2c9d0f-2222-4b7f-8b22-0123456789cd /data ext4 defaults,nofail 0 2
```

Команда `rm *` при десятках тысяч файлов может завершиться ошибкой «слиш-

ком длинный список аргументов»: оболочка раскрывает маску до запуска `rm`. Конструкция `find … -delete` этого ограничения не имеет. Перед удалением всегда выполняйте ту же команду без `-delete` и просматривайте список.

Windows: инструменты

```
Get-Volume                                                  # тома: размер и свободное
место
Get-ChildItem C:\logs -Recurse -File | Measure-Object Length -Sum
Get-ChildItem C:\var -Directory | ForEach-Object {
    [pscustomobject]@{ Name = $_.Name; Size = (Get-ChildItem $_.FullName -Recurse
    -File | Measure-Object Length -Sum).Sum }
} | Sort-Object Size -Descending
Get-ChildItem C:\var -Recurse -File | Sort-Object Length -Descending |
Select-Object -First 3
@(Get-ChildItem C:\spool\mailq -File).Count
Get-ChildItem C:\logs -Filter *.zip | Where-Object { $_.LastWriteTime -lt
(Get-Date).AddDays(-30) } | Remove-Item
```

Не забывайте, что размер файлов (`Length`) и место, занимаемое на диске, различаются: каждый файл занимает целое число **кластеров**. Тысячи крошечных файлов занимают на диске заметно больше суммы их размеров.

Что чаще всего идёт не так

Опасное решение при нехватке места — удалить «всё, что выглядит старым». Журналы могут быть нужны для расследования, а крупный файл в каталоге базы данных — сама база. Сначала выясните, что это.

Практикум Linux: заявки модуля 9

Каталог полигона: `09_diski`. Запуск: `lab 9`, затем `zayavki`.

Мониторинг прислал тревогу: на `srv01` заканчивается место, а почтовая очередь перестала принимать письма, хотя «свободные гигабайты» есть. Нужно найти, что занимает диск, понять разницу между нехваткой места и нехваткой inode, почистить лишнее и подготовить подключение нового диска. Файлы в `srv/var` настоящие (измеряйте их `du`, `find`, `ls`), а таблицы `df`, `df -i`, `lsblk` и `blkid` в полигоне берутся из учебных данных `srv/var/lib/lab` и выглядят так же, как на настоящем сервере.

Заявка Linux 9.1. Что занимает место

В `srv/var` четыре крупных каталога: `cache`, `backups`, `crash`, `mail`. Выясните командой `du`, какой из них самый большой, и запишите его имя (без пути) в `otvety/1`.

Заявка Linux 9.2. Три самых больших файла

Найдите три самых больших файла в дереве `srv/var` (`find … -exec stat` или `ls -lS`). Запишите их имена (без путей) в `otvety/2`, по одному в строке, отсортировав.

Заявка Linux 9.3. Где кончается место

Откройте таблицу `df -h`. Какая файловая система, кроме `/var`, заполнена на 90 % и больше? Запишите точку монтирования (последний столбец) в `otvety/3`.

Заявка Linux 9.4. Свободное место есть, а писать нельзя

Почтовая очередь не принимает новые письма, хотя на `/var` ещё есть свободное место. Причина — исчерпаны inode. Командой `df -i` найдите точку монтирования с `IUse%`, равным 100 %, и запишите её в `otvety/4`.

Заявка Linux 9.5. Сколько файлов в очереди

Посчитайте, сколько файлов лежит в каталоге `srv/var/spool/mailq`, и запишите число в `otvety/5`.

Заявка Linux 9.6. Очистить очередь

Удалите все файлы из `srv/var/spool/mailq`, но сам каталог оставьте: без него почтовая служба не запустится.

Заявка Linux 9.7. Старые архивы

В `srv/var/log/app` лежат архивы журналов `*.gz`, часть которых старше 30 дней. Удалите только старые (`find -mtime +30`); свежие три файла должны остаться.

Заявка Linux 9.8. UUID нового диска

В сервер добавлен диск `/dev/sdb`, раздел `/dev/sdb1` ещё не используется (см. `lsblk`). Узнайте UUID этого раздела командой `blkid` и запишите только значение UUID в `otvety/8`.

Заявка Linux 9.9. Подключение по UUID

Подготовьте автоматическое подключение раздела `/dev/sdb1` (ext4) в каталог `/data`: создайте каталог `srv/data` и добавьте в `srv/etc/fstab` строку с его UUID, точкой монтирования `/data`, типом `ext4`, параметрами `defaults,nofail` и числами `0 2`. Остальные строки fstab не трогайте.

Заявка Linux 9.10. Контрольная: отчёт для руководителя

Подготовьте файл `disk-report.txt` (в каталоге модуля) из трёх строк вида `ПОЛЕ=значение` в таком порядке: `BIGGEST=` (самый большой каталог из задания 1), `FULL=` (заполненная на 90 % файловая система из задания 3), `INODES=` (точка монтирования из задания 4).

Практикум Windows: заявки модуля 9

Каталог полигона: `09_diski`. Запуск: `lab 9`, затем `zayavki`.

Мониторинг прислал тревогу: на `SRV01` заканчивается место на одном из дисков, а почтовая очередь превратилась в свалку мелких файлов. Нужно найти, что

занимает диск, почистить лишнее по правилам и составить отчёт. Файлы в `srv\var` настоящие (измеряйте их `Get-ChildItem`, `Measure-Object`, `Sort-Object`), а таблица томов `Get-Volume` в полигоне берётся из учебных данных `srv\var\lib\lab\volumes.csv` и выглядит так же, как на настоящем сервере.

Заявка Windows 9.1. Что занимает место

В `srv\var` четыре крупных каталога: `cache`, `backups`, `crash`, `mail`. Определите, какой из них самый большой (сумма размеров файлов внутри, `Get-ChildItem -Recurse -File | Measure-Object Length -Sum`), и запишите его имя в `otvety\1`.

Заявка Windows 9.2. Три самых больших файла

Найдите три самых больших файла в дереве `srv\var` (`Get-ChildItem -Recurse -File | Sort-Object Length -Descending`). Запишите их имена (без путей) в `otvety\2`, по одному в строке, отсортировав.

Заявка Windows 9.3. Где кончается место

Выведите таблицу томов (`Get-Volume`). Какой диск (не `C:`) заполнен почти полностью — свободно менее 10 % от размера? Запишите его букву в `otvety\3`.

Заявка Windows 9.4. Сколько файлов в очереди

Посчитайте, сколько файлов лежит в каталоге `srv\var\spool\mailq`, и запишите число в `otvety\4`.

Заявка Windows 9.5. Очистить очередь

Удалите все файлы из `srv\var\spool\mailq`, но сам каталог оставьте: без него почтовая служба не запустится.

Заявка Windows 9.6. Старые архивы

В `srv\var\log\app` лежат архивы журналов `*.zip`, часть которых старше 30 дней (свойство `LastWriteTime`). Удалите только старые; свежие три файла должны остаться.

Заявка Windows 9.7. Контрольная: отчёт для руководителя

Подготовьте файл `disk-report.txt` (в каталоге модуля) из двух строк вида `ПОЛЕ=значение`: `BIGGEST=` (самый большой каталог из задания 1) и `FULL=` (заполненный диск из задания 3).

Вопросы для размышления

1. Чем нехватка inode отличается от нехватки места, и по каким признакам вы бы их различили на незнакомом сервере?
2. Зачем в fstab используют UUID вместо имени вроде /dev/sdb1 и почему полезен параметр nofail?

Глава 10

Модуль 10. Резервное копирование и восстановление

Идея и словарь

Резервная копия, которую никогда не восстанавливали, — лишь надежда. Копию делают, чтобы пережить три ситуации: ошибку человека («удалил не то»), отказ оборудования и вредоносное шифрование. Поэтому копия должна быть *отдельной* (на другом носителе или узле), *проверяемой* (известно, что она не повреждена) и *восстановимой* (вы знаете, как и за какое время вернуть данные).

Архив объединяет много файлов в один и обычно сжимает их. **Полная копия** содержит всё; **инкрементальная** — только изменившееся после предыдущей, поэтому она быстрее и меньше, но для восстановления нужна цепочка копий. Простой способ узнать, что изменилось, — сравнить время изменения файлов с контрольной отметкой. Из архива исключают то, что не нужно хранить: временные файлы, кэш, журналы отладки.

Контрольная сумма (хеш) — число, вычисляемое по содержимому файла: при изменении хотя бы одного бита оно меняется. Сумму сохраняют вместе с копией; позже вычисляют заново и сравнивают, чтобы убедиться, что копия не повреждена при хранении или передаче. Сумма не защищает от подмены, если злоумышленник может изменить и файл, и сумму, и не говорит ничего о том, что копия *содержит нужное*.

Восстановление проверяют **в другое место**: распаковывают копию в отдельный каталог и сравнивают с рабочими данными, не трогая последние. Для восстановления одного файла архив нет смысла распаковывать целиком: распаковывают нужный путь. Старые копии удаляют по правилу хранения (например, «хранить три последние суточные»): имена с датой в формате ГГГГ-ММ-ДД упорядочиваются как обычные строки, что упрощает ротацию.

Linux: инструменты

```
tar czf backup/full.tar.gz -C srv/srv shop                    # -C: перейти в
каталог до архивации
tar czf backup/clean.tar.gz --exclude='*.tmp' --exclude='cache' -C srv/srv shop
tar tzf backup/nightly.tar.gz | grep -vc '/$'                # сколько файлов в
архиве
( cd backup && sha256sum clean.tar.gz > clean.sha256 )
( cd backup/set && sha256sum -c SHA256SUMS )                  # проверка сумм
mkdir -p srv/restore && tar xzf backup/clean.tar.gz -C srv/restore
tar xzf backup/nightly.tar.gz -C srv shop/config/settings.ini # один файл
find srv/srv/shop -type f -newer backup/marker                # изменилось после
отметки
ls backup/daily-*.tar.gz | sort | head -n -3 | xargs rm      # оставить три свежих
```

Контрольную сумму вычисляют из каталога, где лежит файл, и указывают имя без пути: иначе в файле сумм запишется путь, по которому при проверке файла не найдётся.

Windows: инструменты

```
Compress-Archive -Path srv\Shop -DestinationPath backup\full.zip
robocopy srv\Shop backup\clean /E /XF *.tmp /XD cache     # копия без мусора
$h = (Get-FileHash backup\full.zip -Algorithm SHA256).Hash
"$h  full.zip" | Set-Content -Encoding ASCII backup\full.sha256
Expand-Archive -Path backup\full.zip -DestinationPath srv\restore
$m = (Get-Item backup\marker).LastWriteTime
Get-ChildItem srv\Shop -Recurse -File | Where-Object { $_.LastWriteTime -gt $m }
Get-ChildItem backup -Filter 'daily-*.txt' | Sort-Object Name | Select-Object
-SkipLast 3 | Remove-Item
```

Код возврата `robocopy` необычен: значения 0–7 означают успех в разных вариантах («файлы скопированы», «нечего копировать»), а ошибкой считается 8 и больше. Для восстановления одного файла из ZIP-архива распакуйте его во временный каталог (`$env:TEMP`) и скопируйте один файл командой `Copy-Item`.

Что чаще всего идёт не так

Классическая ошибка — хранить копию на том же диске, что и оригинал: отказ диска уносит обе. Вторая — ни разу не проверить восстановление. Третья — копировать только файлы базы данных, пока та работает: без остановки или специального режима копия может оказаться неконсистентной.

Практикум Linux: заявки модуля 10

Каталог полигона: `10_kopii`. Запуск: `lab 10`, затем `zayavki`.

Резервная копия, которую никогда не восстанавливали, — это только надежда. В этом модуле вы строите резервное копирование каталога интернет-магазина `srv/srv/shop` настоящими средствами: `tar`, `gzip`, `sha256sum`, `find`. Результаты копий складываются в каталог `backup` рядом с `srv`. Часть работы — восстановление: пострадавший файл настроек, проверка архивов на повреждение и восстановление **в другое место**, чтобы убедиться, что копия действительно читается.

Заявка Linux 10.1. Полная копия

Создайте сжатый архив `backup/full.tar.gz` с каталогом `shop` (`tar czf`). Внутри архива пути должны начинаться с `shop/`, а не с `srv/srv/shop/`: для этого пригодится ключ `-C`.

Заявка Linux 10.2. Копия без мусора

Сделайте архив `backup/clean.tar.gz` того же каталога, но без временных файлов `*.tmp` и без каталога `cache` (ключи `--exclude`). Остальные файлы должны попасть в архив.

Заявка Linux 10.3. Что в ночном архиве

Готовый архив `backup/nightly.tar.gz` сделан ночью. Не распаковывая его, определите, сколько в нём файлов (каталоги не считаются), и запишите число в `otvety/3`.

Заявка Linux 10.4. Контрольная сумма

Запишите контрольную сумму архива `backup/clean.tar.gz` в файл `backup/clean.sha256` (`sha256sum`) так, чтобы проверка `sha256sum -c clean.sha256`, выполненная в каталоге `backup`, проходила успешно.

Заявка Linux 10.5. Какой архив испорчен

В каталоге `backup/set` лежат архивы `mon`, `tue`, `wed` и файл сумм `SHA256SUMS`. Один из архивов повреждён. Проверьте суммы (`sha256sum -c`) и запишите имя повреждённого архива (например, `tue.tar.gz`) в `otvety/5`.

Заявка Linux 10.6. Восстановление в другое место

Распакуйте `backup/clean.tar.gz` в новый каталог `srv/restore` (создайте его) — так проверяют, что копия читается, не трогая рабочие данные. В результате должен появиться каталог `srv/restore/shop` с теми же файлами, что в архиве.

Заявка Linux 10.7. Восстановить один файл

Файл `srv/srv/shop/config/settings.ini` оказался пустым. Из ночного архива `backup/nightly.tar.gz` восстановите **только** его на прежнее место; остальные файлы магазина не должны измениться.

Заявка Linux 10.8. Что изменилось после отметки

Для инкрементальной копии нужно знать, какие файлы изменились после контрольной отметки `backup/marker`. Найдите их (`find … -newer`) среди обычных файлов `srv/srv/shop` и запишите имена (без пути) в `otvety/8`, по одному в строке, отсортировав.

Заявка Linux 10.9. Ротация копий

В `backup` лежат семь суточных копий `daily-2026-10-01.tar.gz` … `daily-2026-10-07.tar.gz`. Оставьте три самых свежих, остальные удалите.

Заявка Linux 10.10. Контрольная: отчёт о копии

Подготовьте файл `backup-report.txt` (в каталоге модуля) из двух строк: `FILES=` — число файлов в `backup/clean.tar.gz` (без каталогов) и `SHA=` — его контрольная сумма SHA-256 (только 64 символа хеша).

Практикум Windows: заявки модуля 10

Каталог полигона: `10_kopii`. Запуск: `lab 10`, затем `zayavki`.

Резервная копия, которую никогда не восстанавливали, — это только надежда. В этом модуле вы строите резервное копирование каталога интернет-магазина `srv\Shop` настоящими средствами: `Compress-Archive`, `Expand-Archive`, `robocopy`, `Get-FileHash`, `Get-ChildItem`. Результаты копий складываются в каталог `backup` рядом с `srv`. Часть работы — восстановление: пострадавший файл настроек, проверка архивов на повреждение и восстановление **в другое место**, чтобы убедиться, что копия действительно читается.

Заявка Windows 10.1. Полная копия

Создайте архив `backup\full.zip` с каталогом `srv\Shop` (`Compress-Archive -Path srv\Shop -DestinationPath backup\full.zip`). В архиве должны быть все файлы каталога.

Заявка Windows 10.2. Копия без мусора

Скопируйте `srv\Shop` в каталог `backup\clean` без временных файлов `*.tmp` и без каталога `cache`: `robocopy srv\Shop backup\clean /E /XF *.tmp /XD cache`. Все остальные файлы должны попасть в копию. (Код возврата `robocopy` 1 означает «файлы скопированы» — это успех.)

Заявка Windows 10.3. Что в ночном архиве

Готовый архив `backup\nightly.zip` сделан ночью. Определите, сколько в нём файлов (каталоги не считаются), и запишите число в `otvety\3`. Можно открыть архив средствами .NET (`[IO.Compression.ZipFile]::OpenRead`) или распаковать во временный каталог.

Заявка Windows 10.4. Контрольная сумма

Запишите контрольную сумму SHA-256 архива `backup\full.zip` в файл `backup\full.sha256` одной строкой вида `СУММА full.zip` (два пробела; `Get-FileHash -Algorithm SHA256`).

Заявка Windows 10.5. Какой архив испорчен

В каталоге `backup\set` лежат архивы `mon`, `tue`, `wed` и файл сумм `SHA256SUMS.txt`. Один из архивов повреждён. Сравните хеши и запишите имя повреждённого архива (например, `tue.zip`) в `otvety\5`.

Заявка Windows 10.6. Восстановление в другое место

Распакуйте `backup\full.zip` в новый каталог `srv\restore` (`Expand-Archive`) — так проверяют, что копия читается, не трогая рабочие данные. В результате должен появиться каталог `srv\restore\Shop` со всеми файлами архива.

Заявка Windows 10.7. Восстановить один файл

Файл `srv\Shop\config\settings.ini` оказался пустым. Из ночного архива `backup\nightly.zip` восстановите **только** его на прежнее место (распакуйте архив во временный каталог и скопируйте один файл). Остальные файлы магазина (например, новые `changelog.txt` и `refunds.csv`) не должны измениться.

Заявка Windows 10.8. Что изменилось после отметки

Для инкрементальной копии нужно знать, какие файлы изменились после контрольной отметки `backup\marker`. Найдите их среди файлов `srv\Shop` (свойство `LastWriteTime` новее, чем у отметки) и запишите имена (без пути) в `otvety\8`, по одному в строке, отсортировав.

Заявка Windows 10.9. Ротация копий

В `backup` лежат семь суточных копий `daily-2026-10-01.txt` … `daily-2026-10-07.txt`. Оставьте три самых свежих, остальные удалите.

Заявка Windows 10.10. Контрольная: отчёт о копии

Подготовьте файл `backup-report.txt` (в каталоге модуля) из двух строк: `FILES=` — число файлов в каталоге `backup\clean` и `SHA=` — SHA-256 архива `backup\full.zip` (только 64 символа хеша).

Вопросы для размышления

1. Почему важно хранить копию отдельно от сервера и проверять восстановление, а не только создание архива?
2. В чём смысл контрольных сумм и что они не гарантируют?

Глава 11

Модуль 11. Автоматизация: сценарии и cron

Идея и словарь

Всё, что администратор делает больше двух раз, стоит превратить в **сценарий**: человек ошибается, устаёт и забывает, а сценарий выполняется одинаково. Сценарий — текстовый файл с командами; он принимает **параметры**, проверяет входные данные, сообщает о результате **кодом выхода** и пишет сообщения об ошибках в отдельный поток.

Три привычки хорошего сценария. Первая — *проверять входные данные*: нет аргумента или файла — понятное сообщение об использовании и ненулевой код выхода. Вторая — *возвращать правильный код*: 0 означает успех, любое другое число — ошибку; именно по коду планировщик, другие сценарии и мониторинг понимают, что произошло. Третья — *убирать за собой*: временные файлы и замки удаляются при любом завершении, даже аварийном (в Bash для этого служит `trap … EXIT`, в PowerShell — блок `finally`). Без замка два одновременно запущенных сценария резервного копирования испортят друг другу работу.

Чтобы сценарий выполнялся сам, его ставят на **расписание**: в Linux — **cron**, в Windows — **планировщик заданий**. Расписание cron записывается пятью полями (минута, час, день месяца, месяц, день недели) и командой; звёздочка означает «любое значение», запись `*/15` — «каждые 15», `1-5` — «с первого по пятый». Задание планировщика Windows состоит из *действия* (что запустить) и *триггера* (когда). Две распространённые ловушки: сценарий, запущенный по расписанию, работает в *другой среде* (другой `PATH`, другой рабочий каталог, часто другой пользователь), поэтому внутри сценария пишут полные пути; и запуск раньше, чем закончился предыдущий.

Linux: инструменты

```
#!/bin/bash
# top.sh [-n N] ФАЙЛ  --- N самых частых строк
n=3
while getopts "n:" o; do
  case $o in n) n=$OPTARG ;; *) exit 2 ;; esac
done
shift $((OPTIND - 1))
[ -n "$1" ] || { echo "Использование: $0 [-n N] ФАЙЛ" >&2; exit 2; }
sort "$1" | uniq -c | sort -rn | head -n "$n"

# замок, который снимается при любом завершении
trap 'rm -f "$LOCKFILE"' EXIT
touch "$LOCKFILE"

# расписание cron (crontab -l):  мин  час  день  мес  дн.нед.  команда
30 2 * * * /opt/scripts/backup-shop.sh
*/15 9-17 * * 1-5 /opt/scripts/check.sh
```

Код выхода последней команды лежит в `$?`. В полигоне учебный `cron-next`

`'выражение'` показывает, когда расписание сработает в следующий раз, а `crontab файл` проверяет и устанавливает расписание (с понятным сообщением об ошибке, если поле выходит за допустимые границы).

Windows: инструменты

```
# top.ps1 -Path ФАЙЛ [-N число]
param([int]$N = 3, [string]$Path)
if (-not $Path) { Write-Error "Использование: top.ps1 [-N число] -Path ФАЙЛ"; exit
2 }
Get-Content -LiteralPath $Path | Group-Object | Sort-Object Count -Descending |
    Select-Object -First $N | ForEach-Object { "$($_.Count) $($_.Name)" }

# замок, который снимается при любом завершении
try { New-Item -ItemType File -Path $env:LOCKFILE -Force | Out-Null; ... }
finally { Remove-Item $env:LOCKFILE -Force -ErrorAction SilentlyContinue }

# задание планировщика
$a = New-ScheduledTaskAction -Execute 'powershell.exe' -Argument '-File
C:\scripts\backup-shop.ps1'
$t = New-ScheduledTaskTrigger -Daily -At 02:30
Register-ScheduledTask -TaskName BackupShop -Action $a -Trigger $t
```

Код выхода сценария PowerShell доступен после запуска в переменной `$LASTEXITCODE`. Параметры сценария объявляются блоком `param(...)` в начале файла. Учебная команда `Get-LabNextRun -Trigger $триггер` показывает ближайшее срабатывание расписания.

Что чаще всего идёт не так

Самый частый отказ расписания — ошибка во времени (час 25, минута 60, «каждые 0 минут»), которая молча приводит к тому, что задание не выполняется никогда. После постановки задания проверяйте, когда оно сработает в следующий раз.

Практикум Linux: заявки модуля 11

Каталог полигона: `11_avtomatizaciya`. Запуск: `lab 11`, затем `zayavki`.

Администратор, который каждый день повторяет одни и те же команды вручную, делает ошибки. В этом модуле вы пишете небольшие рабочие сценарии bash (аргументы, коды выхода, `getopts`, `trap`) и ставите их на расписание. Сценарии выполняются **по-настоящему**; система расписания учебная: `crontab` хранит ваш файл в `srv/var/spool/cron`, `cron-next` показывает, когда сработает выражение (время «сейчас» фиксировано: воскресенье 2026-10-11 09:30, можно указать другое ключом `--from`), а `cron-run ЧАСЫ` перечисляет запуски в ближайшие часы. Сценарии кладите в `srv/opt/scripts`.

Заявка Linux 11.1. Сценарий-приветствие

Напишите сценарий `srv/opt/scripts/greet.sh`. Запущенный с аргументом (`bash greet.sh Анна`), он печатает `Привет, Анна!`. Без аргумента печатает в поток ошибок строку-подсказку с использованием и завершается с кодом 1.

Заявка Linux 11.2. Коды выхода

Напишите `srv/opt/scripts/check-file.sh ФАЙЛ`: код выхода 0, если файл существует и не пуст; 1, если файла нет; 2, если файл существует, но пуст. Сценарий ничего не печатает.

Заявка Linux 11.3. Починить count-lines.sh

Сценарий `srv/opt/scripts/count-lines.sh` должен печатать только число строк в файле, а он печатает ещё и имя файла и ломается на именах с пробелами. Исправьте его.

Заявка Linux 11.4. Замок и trap

Напишите `srv/opt/scripts/with-lock.sh`: в начале создаёт файл-замок, имя которого в переменной окружения `LOCKFILE`; затем, если задана `FAIL=1`, завершается с кодом 3, иначе выполняет работу (`sleep 1`) и завершается с кодом 0. Замок должен удаляться при **любом** завершении — для этого используйте `trap … EXIT`.

Заявка Linux 11.5. Сумма продаж

Напишите `srv/opt/scripts/sumcsv.sh ФАЙЛ`, который печатает сумму второго столбца CSV-файла (поля разделены запятой). Запустите его на `srv/var/lib/lab/sales.csv` и запишите результат в `otvety/5`.

Заявка Linux 11.6. Когда сработает

Используя `cron-next`, определите ближайшее срабатывание каждого из трёх выражений: `*/15 9-17 * * 1-5` (каждые 15 минут в рабочие часы будней), `0 3 * * 0` (воскресенье в 3:00) и `30 9 1 * *` (первого числа в 9:30). Время «сейчас» — по умолчанию. Запишите три строки вида `ГГГГ-ММ-ДД ЧЧ:ММ` в `otvety/6` в том же порядке.

Заявка Linux 11.7. Ночная копия

Поставьте на расписание резервное копирование: каждый день в 2:30 запускать `/opt/scripts/backup-shop.sh`. Подготовьте файл с одной строкой и установите его командой `crontab`. Результат проверьте `crontab -l`.

Заявка Linux 11.8. Ошибка в расписании

Файл `broken.cron` (в каталоге модуля) не устанавливается. Установите его командой `crontab` и прочитайте сообщение. Исправьте ошибку так, чтобы `nightly.sh` запускался в 3:00 каждую ночь, и установите расписание. В файле уже есть строка резервного копирования из задания 7 — не потеряйте её.

Заявка Linux 11.9. Свое выражение

Запишите в `otvety/9` выражение cron, по которому сценарий запускается каждые 10 минут с 8:00 до 19:50 включительно в дни с понедельника по субботу. Проверьте его командой `cron-next`.

Заявка Linux 11.10. Контрольная: частые слова

Напишите `srv/opt/scripts/top.sh [-n N] ФАЙЛ`, которое печатает N самых частых строк файла в виде «число строка» (по убыванию, по умолчанию N = 3). Разбирайте ключ через `getopts`. Без файла — сообщение об ошибке и код выхода 2. Запустите на `srv/var/lib/lab/words.txt`.

Практикум Windows: заявки модуля 11

Каталог полигона: `11_avtomatizaciya`. Запуск: `lab 11`, затем `zayavki`.

Администратор, который каждый день повторяет одни и те же команды вручную, делает ошибки. В этом модуле вы пишете небольшие рабочие сценарии PowerShell (параметры, коды выхода, `try/finally`) и ставите их на расписание. Сценарии выполняются **по-настоящему**; планировщик учебный: `Register-ScheduledTask`, `New-ScheduledTaskAction`, `New-ScheduledTaskTrigger`, `Get-ScheduledTask` хранят задания в `srv\etc\scheduled.csv`, а `Get-LabNextRun -Trigger $триггер` показывает, когда расписание сработает в следующий раз (время «сейчас» фиксировано: воскресенье 2026-10-11 09:30, другое можно задать ключом `-From 'ГГГГ-ММ-ДД ЧЧ:ММ'`). Сценарии кладите в `srv\scripts`. Если запуск сценариев запрещён: `Set-ExecutionPolicy -Scope Process Bypass`. Для вывода сообщений об ошибках используйте `Write-Error`.

Заявка Windows 11.1. Сценарий-приветствие

Напишите сценарий `srv\scripts\greet.ps1`. Запущенный с аргументом (`.\greet.ps1 Анна`), он выводит `Привет, Анна!`. Без аргумента выводит сообщение об использовании через `Write-Error` и завершается с кодом 1 (`exit 1`).

Заявка Windows 11.2. Коды выхода

Напишите `srv\scripts\check-file.ps1 ФАЙЛ`: код выхода 0, если файл существует и не пуст; 1, если файла нет; 2, если файл существует, но пуст. Сценарий ничего не выводит.

Заявка Windows 11.3. Починить count-lines.ps1

Сценарий `srv\scripts\count-lines.ps1` должен выводить только число строк в файле, а он выводит объект-таблицу `Measure-Object`. Исправьте его: результат — одно целое число. Имя файла может содержать пробелы.

Заявка Windows 11.4. Замок и finally

Напишите `srv\scripts\with-lock.ps1`: в начале создаёт файл-замок, имя которого в переменной окружения `$env:LOCKFILE`; затем, если `$env:FAIL` равна 1, завершается с кодом 3, иначе выполняет работу (`Start-Sleep 1`) и завершается с кодом 0. Замок должен удаляться при **любом** завершении — для этого используйте `try { … } finally { … }`.

Заявка Windows 11.5. Сумма продаж

Напишите `srv\scripts\sumcsv.ps1 ФАЙЛ`, который выводит сумму второго столбца CSV-файла без заголовка (поля разделены запятой). Запустите его на `srv\var\lib\lab\sales.csv` и запишите результат в `otvety\5`.

Заявка Windows 11.6. Когда сработает

Постройте три расписания и узнайте ближайшее срабатывание каждого командой `Get-LabNextRun -Trigger …`: ежедневно в 02:30 (`New-ScheduledTaskTrigger -Daily -At 02:30`); по понедельникам, средам и пятницам в 09:00 (`-Weekly -DaysOfWeek Monday,Wednesday,Friday -At 09:00`); по воскресеньям в 03:00. Время «сейчас» — по умолчанию. Запишите три строки вида `ГГГГ-ММ-ДД ЧЧ:ММ` в `otvety\6` в том же порядке.

Заявка Windows 11.7. Ночная копия

Поставьте на расписание резервное копирование: задание `BackupShop`, ежедневно в 02:30, запускает `powershell.exe -File C:\scripts\backup-shop.ps1`. Используйте `New-ScheduledTaskAction`, `New-ScheduledTaskTrigger`, `Register-ScheduledTask`. Проверьте `Get-ScheduledTask`.

Заявка Windows 11.8. Ошибка в расписании

Сценарий `register-all.ps1` (в каталоге модуля) регистрирует три задания, но на втором выдаёт ошибку. Запустите его, прочитайте сообщение и исправьте так, чтобы `Nightly` запускалось в 03:00 каждую ночь. После запуска должны существовать задания `Check5`, `Nightly` и `BackupShop` (последнее из задания 7 сценарий просто пересоздаёт).

Заявка Windows 11.9. Рабочая смена

Зарегистрируйте задание `Shift` (действие любое, например `powershell.exe -File C:\scripts\check.ps1`), которое срабатывает по понедельникам–субботам в 08:00 (воскресенье не входит). Используйте `-Weekly -DaysOfWeek …`.

Заявка Windows 11.10. Контрольная: частые слова

Напишите `srv\scripts\top.ps1 [-N число] -Path ФАЙЛ`, которое выводит N самых частых строк файла в виде «число строка» (по убыванию; по умолчанию N = 3). Без `-Path` — сообщение об ошибке (`Write-Error`) и код выхода 2. Запустите на `srv\var\lib\lab\words.txt`.

Вопросы для размышления

1. Почему сценарий, запущенный из cron, может работать иначе, чем тот же сценарий в вашем терминале?
2. Что случится, если сценарий резервного копирования запустят по расписанию дважды одновременно, и как от этого защищаются?

Глава 12

Модуль 12. Сеть, SSH и межсетевой экран

Идея и словарь

Сетевая служба доступна, пока кто-то слушает её **порт**. Чем меньше портов открыто наружу, тем меньше поверхность атаки. Поэтому аудит сети начинается с вопроса «кто что слушает»: для каждого слушающего порта нужно знать программу, адрес привязки и причину. Порт на адресе `0.0.0.0` (или `[::]`) доступен со всей сети, порт на `127.0.0.1` — только с самого сервера. База данных на локальном адресе безопасна; та же база на `0.0.0.0` открыта всем.

Межсетевой экран хоста решает, какие входящие соединения пропускать. Надёжный принцип — *запретить всё по умолчанию и разрешить только нужное*, как можно уже: администратору SSH можно разрешить доступ только из внутренней подсети, а вебу — со всей сети. В Linux правила **ufw** проверяются сверху вниз и срабатывает первое подходящее, поэтому запрещающее правило для конкретного адреса надо *вставить выше* разрешающих. В брандмауэре Windows наоборот: любое блокирующее правило сильнее любого разрешающего, порядок не важен. Эту разницу стоит запомнить: она ломает привычки при переходе между системами.

SSH — защищённая удалённая оболочка. Надёжный вход строится на **ключах**: пара из закрытого ключа (хранится только у владельца, права 600) и открытого, который дописывается в файл `authorized_keys` на сервере. Вход по паролю отключают (`PasswordAuthentication no`), вход под администратором — тоже (`PermitRootLogin no`), но только после того, как проверено, что вход по ключу работает: иначе можно потерять доступ. Клиентскому файлу `~/.ssh/config` задают короткие имена серверов, чтобы не вводить адрес, порт, пользователя и ключ каждый раз; итоговые параметры соединения показывает `ssh -G имя`.

Linux: инструменты

```
ss -tlnp                                          # кто слушает какой TCP-порт
ssh-keygen -t ed25519 -N '' -C deploy@srv01 -f ~/.ssh/id_ed25519_deploy
cat ~/.ssh/id_ed25519_deploy.pub >> ~/.ssh/authorized_keys     # >> --- дописать
ssh -G web                                        # итоговые параметры для имени web

# ~/.ssh/config
Host web
    HostName 10.0.0.5
    User deploy
    Port 2222
    IdentityFile ~/.ssh/id_ed25519_deploy

sudo ufw default deny incoming
sudo ufw allow from 192.168.1.0/24 to any port 22 proto tcp
sudo ufw allow 80/tcp
sudo ufw insert 1 deny from 198.51.100.7       # выше разрешающих
sudo ufw enable
```

В полигоне `ufw status` показывает текущие правила, а `fwtest tcp 22 from 8.8.8.8` отвечает, пропустит ли такой набор правил пакет. Закрытый ключ должен иметь права 600, иначе клиент SSH откажется им пользоваться.

Windows: инструменты

```
Get-NetTCPConnection -State Listen | Sort-Object LocalPort
ssh-keygen -t ed25519 -N '""' -C deploy@srv01 -f $HOME\.ssh\id_ed25519_deploy
Add-Content $HOME\.ssh\authorized_keys -Value (Get-Content
$HOME\.ssh\id_ed25519_deploy.pub)
ssh -G -F $HOME\.ssh\config web

Set-NetFirewallProfile -Profile Domain,Private,Public -Enabled True
-DefaultInboundAction Block
New-NetFirewallRule -DisplayName 'SSH из локальной сети' -Direction Inbound
-Action Allow `
    -Protocol TCP -LocalPort 22 -RemoteAddress 192.168.1.0/24
New-NetFirewallRule -DisplayName 'Блок атакующего' -Direction Inbound -Action
Block `
    -Protocol TCP -RemoteAddress 198.51.100.7
Remove-NetFirewallRule -DisplayName 'Telnet (23)'
```

Пустая парольная фраза в Windows PowerShell 5.1 передаётся в `ssh-keygen` как `'""'`: иначе PowerShell «съедает» пустые кавычки. В полигоне учебная `Test-LabFirewall -LocalPort 22 -RemoteAddress 8.8.8.8` показывает вердикт брандмауэра (`Allow` или `Block`).

Что чаще всего идёт не так

Открытая «на минутку» база данных или консоль администрирования — источник большинства громких утечек. Правила для SSH принято проверять с *второго* терминала, не закрывая рабочий: ошибка в правиле может отрезать вас от сервера.

Практикум Linux: заявки модуля 12

Каталог полигона: `12_set_ssh`. Запуск: `lab 12`, затем `zayavki`.

Аудит безопасности показал: на `srv01` открыто больше сетевых портов, чем нужно, вход по SSH разрешён всем и по паролю, межсетевой экран выключен. Нужно навести порядок. В этом модуле: таблица слушающих портов `ss -tlnp` берётся из учебных данных, `ssh-keygen` и `ssh -G` — **настоящие** (ключи и конфигурация клиента — реальные файлы), правила межсетевого экрана ведут учебные `ufw` и `fwtest` (синтаксис как у настоящего ufw; `fwtest tcp ПОРТ [from АДРЕС]` показывает, пропустит ли такой пакет ваш набор правил). Вне полигона `ss` и остальные команды работают как обычно.

Заявка Linux 12.1. Кто слушает порт

Изучите `ss -tlnp`. Какая программа слушает на этом сервере необычный порт из диапазона 8000–8999? Запишите её имя (в кавычках в столбце Process) в `otvety/1`.

Заявка Linux 12.2. Что видно из сети

Порты, слушающие на адресе `0.0.0.0`, доступны со всей сети; порты на `127.0.0.1` — только с самого сервера. Запишите номера портов, слушающих на `0.0.0.0`, через запятую без пробелов, по возрастанию числа, в одну строку в `otvety/2`.

Заявка Linux 12.3. Ключ для автоматизации

Для пользователя anna создайте ключ без парольной фразы: `srv/home/anna/.ssh/id_ed25519_deploy` (тип ed25519, комментарий `deploy@srv01`, `ssh-keygen`). Добавьте открытый ключ в конец `srv/home/anna/.ssh/authorized_keys`; уже имеющаяся там строка должна сохраниться.

Заявка Linux 12.4. Короткое имя для сервера

Создайте клиентский конфиг `srv/home/anna/.ssh/config` с записью `Host web: HostName 10.0.0.5, User deploy, Port 2222, IdentityFile ~/.ssh/id_ed25519_deploy`. Проверьте результат командой `ssh -G -F srv/home/anna/.ssh/config web`.

Заявка Linux 12.5. Закрутить гайки в sshd

В `srv/etc/ssh/sshd_config` запретите вход root (`PermitRootLogin no`), вход по паролю (`PasswordAuthentication no`) и перенаправление X11 (`X11Forwarding no`). Работает первое неприкомментированное вхождение директивы; закомментированные строки не учитываются.

Заявка Linux 12.6. Убрать лишние правила

В списке правил ufw остались разрешения для telnet (порт 23) и для открытого наружу MySQL (порт 3306). Посмотрите правила (`ufw status numbered` покажет их после включения; можно читать и `srv/etc/ufw/user.rules`) и удалите оба командой `ufw delete`.

Заявка Linux 12.7. Запрет по умолчанию

Установите политику по умолчанию для входящих соединений «запретить» (`ufw default deny incoming`).

Заявка Linux 12.8. Разрешить нужное и включить

Разрешите: SSH (порт 22/tcp) **только** из подсети `192.168.1.0/24`, веб-серверу — порты 80/tcp и 443/tcp для всех. Затем включите межсетевой экран (`ufw enable`).

Заявка Linux 12.9. Проверка правил

Проверьте результат командой `fwtest` для пяти случаев и запишите пять слов `ALLOW` или `DENY` в `otvety/9` в порядке: ssh из `192.168.1.50`; ssh из `8.8.8.8` (пакет из Интернета); порт 80; порт 3306; порт 23.

Заявка Linux 12.10. Контрольная: акт аудита

Подготовьте файл `netaudit.txt` (в каталоге модуля) из трёх строк: `OPEN=` — порты на `0.0.0.0` через запятую (как в задании 2), `FW=` — состояние межсетевого экрана (`active` или `inactive`, из `ufw status`), `ROOTLOGIN=` — значение `PermitRootLogin` из `sshd_config`.

Практикум Windows: заявки модуля 12

Каталог полигона: `12_set_ssh`. Запуск: `lab 12`, затем `zayavki`.

Аудит безопасности показал: на `SRV01` открыто больше сетевых портов, чем нужно, вход по SSH разрешён всем и по паролю, брандмауэр выключен. Нужно навести порядок. В этом модуле: таблицу слушающих портов `Get-NetTCPConnection -State Listen` берём из учебных данных (в объекте есть дополнительное свойство `ProcessName`), `ssh-keygen` и `ssh -G` — **настоящие** (ключи и конфигурация клиента — реальные файлы; нужен клиент OpenSSH: `Get-Command ssh, ssh-keygen`), правила брандмауэра ведут учебные `New-NetFirewallRule`, `Get-NetFirewallRule`, `Remove-NetFirewallRule`, `Set-NetFirewallProfile` (порт, протокол и адрес входят в сам объект правила), а `Test-LabFirewall -Protocol TCP -LocalPort N [-RemoteAddress IP]` показывает, пропустит ли брандмауэр такой пакет (блокирующее правило сильнее разрешающего, как в Windows).

Заявка Windows 12.1. Кто слушает порт

Изучите `Get-NetTCPConnection -State Listen`. Какая программа слушает необычный порт из диапазона 8000–8999? Запишите имя процесса (свойство `ProcessName`) в `otvety\1`.

Заявка Windows 12.2. Что видно из сети

Порты, слушающие на `0.0.0.0`, доступны со всей сети; порты на `127.0.0.1` — только с самого сервера. Запишите номера портов, слушающих на `0.0.0.0`, через запятую без пробелов, по возрастанию, в одну строку в `otvety\2`.

Заявка Windows 12.3. Ключ для автоматизации

Для пользователя anna создайте ключ без парольной фразы: `srv\Users\anna\.ssh\id_ed25519_deploy` (тип ed25519, комментарий `deploy@srv01`, команда `ssh-keygen`; пустая фраза в Windows PowerShell 5.1 задаётся как `-N '""'`). Добавьте открытый ключ в конец `srv\Users\anna\.ssh\authorized_keys`; уже имеющаяся там строка должна сохраниться.

Заявка Windows 12.4. Короткое имя для сервера

Создайте клиентский конфиг `srv\Users\anna\.ssh\config` с записью `Host web`: `HostName 10.0.0.5`, `User deploy`, `Port 2222`, `IdentityFile ~/.ssh/id_ed25519_deploy`. Проверьте результат командой `ssh -G -F (Resolve-Path srv\Users\anna\.ssh\config).Path web` (клиенту нужен полный путь к файлу).

Заявка Windows 12.5. Закрутить гайки в sshd

В `srv\ProgramData\ssh\sshd_config` запретите вход root/администратора (`PermitRootLogin no`), вход по паролю (`PasswordAuthentication no`) и перенаправление X11 (`X11Forwarding no`). Работает первое неприкомментированное вхождение директивы; закомментированные строки не учитываются.

Заявка Windows 12.6. Убрать лишние правила

В списке правил брандмауэра остались разрешения для telnet (порт 23) и для открытого наружу MySQL (порт 3306). Посмотрите правила (`Get-NetFirewallRule`) и удалите оба командой `Remove-NetFirewallRule`.

Заявка Windows 12.7. Запрет по умолчанию

Включите брандмауэр и установите действие по умолчанию для входящих соединений «блокировать»: `Set-NetFirewallProfile -Profile Domain,Private,Public -Enabled True -DefaultInboundAction Block`.

Заявка Windows 12.8. Разрешить нужное

Разрешите входящие: SSH (порт 22/TCP) **только** из подсети `192.168.1.0/24`, веб-серверу — порты 80/TCP и 443/TCP для всех (`New-NetFirewallRule -DisplayName … -Direction Inbound -Action Allow -Protocol TCP -LocalPort … [-RemoteAddress …]`).

Заявка Windows 12.9. Проверка правил

Проверьте результат командой `Test-LabFirewall` для пяти случаев и запишите пять слов `Allow` или `Block` в `otvety\9` в порядке: ssh (22) из `192.168.1.50`; ssh из `8.8.8.8` (пакет из Интернета); порт 80; порт 3306; порт 23.

Заявка Windows 12.10. Контрольная: акт аудита

Подготовьте файл `netaudit.txt` (в каталоге модуля) из трёх строк: `OPEN=` — порты на `0.0.0.0` через запятую (как в задании 2), `FW=` — состояние брандмауэра (`True` или `False`, свойство `Enabled`), `ROOTLOGIN=` — значение `PermitRootLogin` из `sshd_config`.

Вопросы для размышления

1. Почему принцип «запретить всё и разрешить нужное» надёжнее, чем «разрешить всё и запретить опасное»?
2. Что нужно сделать до отключения входа по паролю, чтобы не потерять доступ к серверу?

Глава 13

Модуль 13. Аудит безопасности

Идея и словарь

Аудит безопасности — осмотр системы глазами атакующего: где лежит то, что не должно лежать на виду, и что осталось от прежних работ. Он состоит из серии небольших проверок, каждая отвечает на один вопрос и даёт конкретный список. Результат — не «всё плохо», а перечень находок с описанием, как исправить.

Что искать в первую очередь. **Секреты в открытом виде**: пароли, токены и ключи API в файлах настроек и в сценариях; найденное заменяют заглушкой и выпускают новый секрет, потому что старый считается скомпрометированным. **Секреты в истории команд**: пароль, переданный в командной строке (`mysql -u root -pПАРОЛЬ`), остаётся в истории оболочки. **Чужие задания в расписании**: запуск программы из временного или скрытого каталога — типичный способ закрепиться в системе. **Ключи уволенных**: открытый ключ ушедшего сотрудника в `authorized_keys` даёт ему доступ, пока вы его не удалите. **Лишние службы**: устаревшие протоколы (telnet, rsh) и ненужные демоны. **Служебные учётные записи с интерактивным входом**: служба не должна иметь оболочку (Linux) или право удалённого входа (Windows). **Политика паролей**: срок жизни, минимальная длина, сложность, блокировка после неудачных попыток.

Порядок действий при находке: зафиксировать (записать, где и что), исправить, проверить, что исправление сработало, задокументировать. Итог оформляется *актом*: короткий документ с числами, которые проверяемы повторным запуском тех же команд.

Linux: инструменты

```
grep -rliE '(password|passwd|secret|token)[[:space:]]*=[[:space:]]*[^[:space:]]+'
/home /opt
grep -l -- ' -p[A-Za-z]' /home/*/.bash_history
grep -l '/tmp/' /var/spool/cron/crontabs/*
sed -i '/oleg@/d' /home/*/.ssh/authorized_keys
awk -F: '$3 > 0 && $3 < 1000 && $7 ~ /sh$/ {print $1}' /etc/passwd   # служебные
записи с оболочкой
sudo usermod -s /usr/sbin/nologin svc-ci
sudo systemctl disable --now telnet-server
sed -i -E
's/((password|token)[[:space:]]*=[[:space:]]*)[^[:space:]]+/\1CHANGEME/I' файл
```

Ключ `-l` у `grep` печатает только имена файлов, `-r` ищет рекурсивно, `-i` игнорирует регистр. Замена значения через `sed` с группой в скобках сохраняет левую часть строки и меняет только значение.

Windows: инструменты

```
Get-ChildItem C:\Users, 'C:\Program Files' -Recurse -File |
    Select-String -Pattern '(password|passwd|secret|token)\s*=\s*\S+' |
    Select-Object -Expand Path -Unique
```

```
Get-ChildItem C:\Users -Recurse -Filter ConsoleHost_history.txt | Select-String
-Pattern ' -p[A-Za-z]'
Get-ScheduledTask | Where-Object { $_.Actions[0].Execute -like '*\Temp\*' }
Unregister-ScheduledTask -TaskName Updater -Confirm:$false
(Get-Content authorized_keys | Where-Object { $_ -notmatch 'oleg@' }) |
Set-Content authorized_keys
Get-LocalGroupMember -Group 'Remote Desktop Users'
(Get-Content f.cfg) -replace '((password|token)\s*=\s*)\S+', '$1CHANGEME' |
Set-Content f.cfg
```

История команд PowerShell хранится в файле `ConsoleHost_history.txt` в профиле пользователя (модуль PSReadLine). Группа `Remote Desktop Users` определяет, кто может заходить по RDP: служебным учётным записям там делать нечего.

Что чаще всего идёт не так

Замена пароля на заглушку не отзывает его: если он уже мог быть прочитан кем-то посторонним, его надо сменить. Аудит, который только находит и не исправляет, и исправление без повторной проверки — одинаково неполные.

Практикум Linux: заявки модуля 13

Каталог полигона: `13_audit`. Запуск: `lab 13`, затем `zayavki`.

Перед проверкой службы безопасности вам поручено самому найти слабые места на `srv01`: пароли в открытом виде в файлах и в истории команд, подозрительное задание в расписании, ключи уволенного сотрудника, лишние службы, служебные учётные записи с оболочкой входа. Всё это уже **внесено** в сервер при подготовке модуля (и у каждого немного по-своему). Ваша задача — найти, исправить и составить акт. Работайте командами `grep -r`, `find`, `awk`, `sed`, а также знакомыми `usermod`, `systemctl`.

Заявка Linux 13.1. Пароли в файлах

Найдите файлы в `srv/home` и `srv/opt`, содержащие строку вида `password = значение` (также `passwd`, `secret`, `token`; регистр не важен; после слова обязательно знак `=` и значение). Запишите пути (в том виде, как их выводит `grep`, например `srv/opt/app/app.conf`) в `otvety/1`, по одному в строке, отсортировав. Строки без знака `=` (простые упоминания слова) не считаются.

Заявка Linux 13.2. Пароль в истории команд

В `.bash_history` одного из пользователей `anna`, `boris`, `pavel` остался пароль, переданный в командной строке (`mysql … -pПАРОЛЬ`). Найдите этого пользователя и запишите его имя в `otvety/2`.

Заявка Linux 13.3. Убрать секреты

В каждом файле из задания 1 замените значение секрета на `CHANGEME`, сохранив остальные строки. Число строк в файлах не должно измениться.

Заявка Linux 13.4. Политика паролей

В `srv/etc/login.defs` задайте срок жизни пароля 90 дней (`PASS_MAX_DAYS`), минимальный интервал смены 1 день (`PASS_MIN_DAYS`) и предупреждение за 14 дней (`PASS_WARN_AGE`).

Заявка Linux 13.5. Чужое расписание

В каталоге `srv/var/spool/cron/crontabs` лежат расписания пользователей. В одном из них есть задание, запускающее программу из `/tmp` — это типичный признак взлома. Запишите имя пользователя (имя файла) в `otvety/5`.

Заявка Linux 13.6. Убрать задание

Удалите подозрительную строку из расписания этого пользователя; остальные строки сохраните.

Заявка Linux 13.7. Ключи уволенного

Бывший сотрудник `oleg` оставил свои открытые ключи (комментарий `oleg@home-pc`) в `authorized_keys` нескольких пользователей (`srv/home/*/.ssh/authorized_keys`). Удалите все его ключи, оставив ключи остальных.

Заявка Linux 13.8. Лишние службы

На сервере включены в автозапуск устаревшие и ненужные службы `telnet-server`, `rsh-server` и `cups`. Отключите и остановите их (`systemctl disable --now`). Служба `nginx` должна остаться.

Заявка Linux 13.9. Служебные учётные записи с оболочкой

Служебные учётные записи (UID меньше 1000, кроме root) не должны иметь оболочку входа. Найдите те, у которых оболочка заканчивается на `sh` (`/bin/bash`, `/bin/sh`...), и запишите их имена в `otvety/9`, по одному в строке, отсортировав.

Заявка Linux 13.10. Контрольная: исправить и составить акт

Запретите вход для найденных в задании 9 учётных записей (`usermod -s /usr/sbin/nologin`). Затем составьте файл `audit.txt` (в каталоге модуля) из трёх строк: `SECRETS=` — число файлов из задания 1, `CRON=` — пользователь из задания 5, `SHELLS=` — число исправленных учётных записей.

Практикум Windows: заявки модуля 13

Каталог полигона: `13_audit`. Запуск: `lab 13`, затем `zayavki`.

Перед проверкой службы безопасности вам поручено самому найти слабые места на `SRV01`: пароли в открытом виде в файлах и в истории команд PowerShell, подозрительное задание в планировщике, ключи уволенного сотрудника, лишние службы, служебные учётные записи с правом удалённого входа. Всё это уже **внесено** в сервер при подготовке модуля (и у каждого немного по-

своему). Ваша задача — найти, исправить и составить акт. Работайте командами `Get-ChildItem -Recurse`, `Select-String`, `Where-Object`, `Get-ScheduledTask`, `Unregister-ScheduledTask`, `Set-Service`, `Remove-LocalGroupMember`.

Заявка Windows 13.1. Пароли в файлах

Найдите файлы в `srv\Users` и `srv\Program Files`, содержащие строку вида `password = значение` (также `passwd`, `secret`, `token`; регистр не важен; после слова обязательно знак `=` и значение). Запишите **имена** файлов (без пути) в `otvety\1`, по одному в строке, отсортировав. Строки без знака `=` (простые упоминания слова) не считаются.

Заявка Windows 13.2. Пароль в истории команд

В истории команд PowerShell (`srv\Users\ИМЯ\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt`) одного из пользователей `anna`, `boris`, `pavel` остался пароль, переданный в командной строке (`mysql … -pПАРОЛЬ`). Найдите этого пользователя и запишите его имя в `otvety\2`.

Заявка Windows 13.3. Убрать секреты

В каждом файле из задания 1 замените значение секрета на `CHANGEME`, сохранив остальные строки. Число строк в файлах не должно измениться.

Заявка Windows 13.4. Политика паролей

В `srv\etc\secpol.ini` задайте срок жизни пароля 90 дней (`MaximumPasswordAge`), минимальный интервал смены 1 день (`MinimumPasswordAge`), минимальную длину 12 символов (`MinimumPasswordLength`) и включите сложность (`PasswordComplexity=1`). Прочие параметры не меняйте.

Заявка Windows 13.5. Чужое задание планировщика

Среди заданий `Get-ScheduledTask` есть одно, запускающее программу из скрытого каталога `Public\Temp\.x` — типичный признак взлома. Запишите его имя (`TaskName`) в `otvety\5`.

Заявка Windows 13.6. Убрать задание

Удалите подозрительное задание командой `Unregister-ScheduledTask`; задание `BackupShop` должно остаться.

Заявка Windows 13.7. Ключи уволенного

Бывший сотрудник `oleg` оставил свои открытые ключи (комментарий `oleg@home-pc`) в `authorized_keys` нескольких пользователей (`srv\Users\*\.ssh\authorized_keys`). Удалите все его ключи, оставив ключи остальных.

Заявка Windows 13.8. Лишние службы

На сервере включены в автозапуск устаревшие и ненужные службы `telnet-server`, `rsh-server` и `cups`. Остановите их и запретите запуск (`Stop-Service`, `Set-Service -StartupType Disabled`). Служба `nginx` должна остаться.

Заявка Windows 13.9. Служебные записи с удалённым входом

Служебные учётные записи (`svc-ci`, `ftp-user`, `mysql`) не должны входить в группу `Remote Desktop Users`: служба не должна входить на сервер интерактивно. Найдите в этой группе записи, у которых включён флаг `PasswordNeverExpires` (признак служебной записи), и запишите их имена в `otvety\9`, по одному в строке, отсортировав.

Заявка Windows 13.10. Контрольная: исправить и составить акт

Выведите найденные в задании 9 учётные записи из группы `Remote Desktop Users` (`Remove-LocalGroupMember`); `dmitri` в группе остаётся. Затем составьте файл `audit.txt` (в каталоге модуля) из трёх строк: `SECRETS=` — число файлов из задания 1, `TASK=` — имя задания из задания 5, `RDP=` — число исправленных учётных записей.

Вопросы для размышления

1. Чем опасен пароль в аргументах команды, даже если файл конфигурации защищён?
2. Какие признаки отличают взлом от обычной ошибки администратора в найденных вами находках?

Глава 14

Модуль 14. Итоговый инцидент: «сервер тормозит»

Идея и словарь

Реальные проблемы редко приходят с табличкой «здесь диск». Директор сообщает: «всё тормозит». Задача администратора — превратить расплывчатую жалобу в последовательность проверок, каждая из которых даёт факт, и найти причины (иногда несколько сразу). Итоговый модуль собирает приёмы всего курса в одну историю, где признаки наслаиваются: посторонний процесс нагружает сервер, диск заполнен журналом, служба магазина упала из-за ошибки в настройках, в журналах видна атака подбором пароля, а нужный файл настроек потерян.

Рабочий порядок расследования. **Сначала наблюдение, потом действие**: список процессов, места на дисках, состояния служб, последние ошибки в журналах. **Сначала сохранить следы, потом чистить**: перед удалением файла или процесса запишите, что это, откуда и когда. **Один шаг — одна проверка**: после каждого изменения убедитесь, что оно сработало, и только потом переходите дальше. **Отделять причину от следствия**: служба упала из-за отсутствия места или из-за ошибки настройки? И то и другое может быть верным, и исправлять надо оба. **Закончить отчётом**: что обнаружено, что сделано, что осталось и что предлагается сделать, чтобы не повторилось (мониторинг места, ротация журналов, блокировка перебора, регулярные копии).

В отличие от предыдущих модулей, здесь заявки не называют инструментов. Если вы справились без подсказок, значит, цель курса достигнута: вы умеете выбирать команду, а не просто выполнять названную.

Linux: инструменты

```
ps -ef | grep 'tmp'                                  # процессы из временных
каталогов
du -sk /var/* | sort -rn | head -n 3                 # куда делось место
journalctl -u shop-api -n 20                         # почему упала служба
grep 'Failed password' /var/log/auth.log | awk '{print $11}' | sort | uniq -c |
sort -rn | head
sudo ufw insert 1 deny from 198.51.100.7             # заблокировать атакующего выше
разрешающих
tar xzf backup/nightly.tar.gz -C srv etc/shop.conf   # вернуть потерянный файл
```

Цепочка действий: найти и остановить посторонний процесс, очистить диск от разросшихся журналов, прочитать журнал службы, исправить причину и запустить, найти атакующего, заблокировать, восстановить файл из копии, написать отчёт.

Windows: инструменты

```
Get-Process | Where-Object { $_.Path -like '*\Temp\*' }
```

```
Get-Volume
Get-WinEvent -FilterHashtable @{ LogName = 'System'; Level = 2 } -MaxEvents 5
Get-WinEvent -FilterHashtable @{ LogName = 'Security'; Id = 4625 } |
    ForEach-Object { if ($_.Message -match 'адрес источника: (\S+)') { $Matches[1]
    } } | Group-Object | Sort-Object Count -Descending
New-NetFirewallRule -DisplayName 'Блок' -Direction Inbound -Action Block -Protocol
TCP -RemoteAddress 198.51.100.7
Expand-Archive backup\nightly.zip -DestinationPath $env:TEMP\n; Copy-Item
$env:TEMP\n\etc\shop.conf srv\etc\
```

Те же шаги в другой последовательности: найти процесс, остановить, освободить место, прочитать журнал событий службы, создать недостающую учётную запись, запустить службу, найти атакующего, заблокировать, восстановить файл и составить отчёт.

Что чаще всего идёт не так

Типичная ошибка на реальных инцидентах — остановиться на первой найденной причине. Если после исправления проблема не ушла полностью, ищите вторую; если ушла — проверьте, не вернётся ли она. И не забывайте итоговый отчёт: без него опыт остаётся только в голове.

Практикум Linux: заявки модуля 14

Каталог полигона: `14_incident`. Запуск: `lab 14`, затем `zayavki`.

Утром директор сообщает: интернет-магазин не открывается, а сервер «тормозит». У вас нет подсказок, только заявка. Реальная картина такова (вы её сами раскроете): в системе завёлся посторонний процесс, на модуле `/var` кончается место, служба магазина упала, в журналах видна атака подбором пароля, а файл настроек магазина потерян. Здесь используются все приёмы курса: процессы, диски, службы, журналы, межсетевой экран, резервные копии. Порядок заданий соответствует разумному порядку расследования; последнее задание — ваш отчёт руководителю.

Заявка Linux 14.1. Подозрительный процесс

Сервер перегружен. Среди запущенных процессов найдите тот, программа которого лежит во временном скрытом каталоге `srv/tmp/.cache`. Запишите имя программы (без пути) в `otvety/1`.

Заявка Linux 14.2. Остановить и удалить

Завершите этот процесс (сначала вежливо, затем при необходимости принудительно) и удалите его файл. Файл удалять следует после остановки процесса.

Заявка Linux 14.3. Куда делось место

Таблица `df -h` показывает, что `/var` заполнен на 97 %. Определите, какой подкаталог `srv/var` занимает больше всего места, и запишите его имя (без пути) в `otvety/3`.

Заявка Linux 14.4. Освободить место

Удалите файл `srv/var/log/app/debug.log` (отладочный журнал, разросшийся без ротации) и архивы журналов `*.gz` в `srv/var/log/app`, которым больше 30 дней. Файл `srv/var/cache/pkg.cache` не трогайте.

Заявка Linux 14.5. Почему упал магазин

Служба `shop-api` не запущена. Прочитайте её журнал (`journalctl -u shop-api`) и найдите файл, который не удалось загрузить. Запишите его полный путь в `otvety/5`.

Заявка Linux 14.6. Запустить магазин

Создайте недостающий файл `srv/etc/shop.env` с одной строкой `SHOP_ENV=prod` и запустите службу `shop-api`.

Заявка Linux 14.7. Кто ломится

В `srv/var/log/auth.log` найдите IP-адрес, с которого пришло больше всего неудачных входов (`Failed password`), и запишите его в `otvety/7`.

Заявка Linux 14.8. Заблокировать атакующего

Заблокируйте найденный адрес: правило должно срабатывать раньше разрешающих (`ufw insert 1 deny from АДРЕС`), чтобы атакующий не прошёл и на порт 80. Остальные пользователи (подсеть `192.168.1.0/24` на 22-й порт, все на 80-й) должны сохранить доступ.

Заявка Linux 14.9. Вернуть потерянный файл

Файл `srv/etc/shop.conf` потерян. Он есть в ночном архиве `backup/nightly.tar.gz` (путь внутри архива — `etc/shop.conf`). Восстановите **только** его на прежнее место (в каталог `srv/etc`).

Заявка Linux 14.10. Контрольная: отчёт директору

Составьте файл `incident.txt` (в каталоге модуля) из четырёх строк: `HOG=` — имя постороннего процесса (задание 1), `DISK=` — каталог, занимавший место (задание 3), `ATTACKER=` — адрес из задания 7, `SERVICE=` — текущее состояние службы `shop-api` по `systemctl is-active`.

Практикум Windows: заявки модуля 14

Каталог полигона: `14_incident`. Запуск: `lab 14`, затем `zayavki`.

Утром директор сообщает: интернет-магазин не открывается, а сервер «тормозит». У вас нет подсказок, только заявка. Реальная картина такова (вы её сами раскроете): в системе завёлся посторонний процесс, на диске кончается место, служба магазина не запускается, в журнале безопасности видна атака подбором пароля, а файл настроек магазина потерян. Здесь используются все приёмы курса: процессы, диски, службы, журналы, брандмауэр, резервные копии. Порядок заданий соответствует разумному порядку расследования; последнее задание — ваш отчёт руководителю.

Заявка Windows 14.1. Подозрительный процесс

Сервер перегружен. Среди запущенных процессов найдите тот, программа которого лежит во временном скрытом каталоге `srv\Temp\.cache` (свойство `Path`). Запишите имя процесса (свойство `Name`) в `otvety\1`.

Заявка Windows 14.2. Остановить и удалить

Завершите этот процесс (`Stop-Process`) и удалите его файл. Файл запущенной программы Windows удалить не даст.

Заявка Windows 14.3. Куда делось место

Таблица `Get-Volume` показывает, что диск `D:` (том `Var`) заполнен почти полностью. Определите, какой подкаталог `srv\var` занимает больше всего места, и запишите его имя в `otvety\3`.

Заявка Windows 14.4. Освободить место

Удалите файл `srv\var\log\app\debug.log` (отладочный журнал, разросшийся без ротации) и архивы `*.zip` в `srv\var\log\app`, которым больше 30 дней. Файл `srv\var\cache\pkg.cache` не трогайте.

Заявка Windows 14.5. Почему не запускается магазин

Служба `shop-api` не запущена. Найдите в журнале System (`Get-WinEvent`) ошибку этой службы и выясните, какой учётной записи не существует. Запишите имя этой записи (без `SRV01\`) в `otvety\5`.

Заявка Windows 14.6. Запустить магазин

Создайте недостающую учётную запись (`New-LocalUser`, любой пароль) и запустите службу `shop-api`.

Заявка Windows 14.7. Кто ломится

В журнале Security найдите IP-адрес, с которого пришло больше всего неудачных входов (событие 4625, адрес — после слов `адрес источника:` в сообщении), и запишите его в `otvety\7`.

Заявка Windows 14.8. Заблокировать атакующего

Заблокируйте найденный адрес: создайте правило `New-NetFirewallRule -Direction Inbound -Action Block -Protocol TCP -RemoteAddress АДРЕС` (все порты). Остальные пользователи (подсеть `192.168.1.0/24` на 22-й порт, все на 80-й) должны сохранить доступ. В Windows блокирующее правило сильнее любого разрешающего, поэтому порядок правил не важен.

Заявка Windows 14.9. Вернуть потерянный файл

Файл `srv\etc\shop.conf` потерян. Он есть в ночном архиве `backup\nightly.zip` (путь внутри архива — `etc\shop.conf`). Восстановите **только**

его на прежнее место (в `srv\etc`): распакуйте архив во временный каталог и скопируйте один файл. Файл `other.conf` в `srv\etc` появляться не должен.

Заявка Windows 14.10. Контрольная: отчёт директору

Составьте файл `incident.txt` (в каталоге модуля) из четырёх строк: `HOG=` — имя постороннего процесса (задание 1), `DISK=` — каталог, занимавший место (задание 3), `ATTACKER=` — адрес из задания 7, `SERVICE=` — текущее состояние службы `shop-api` (`Running` или `Stopped`).

Вопросы для размышления

1. В каком порядке вы бы действовали при настоящем инциденте и почему сначала нужно сохранить следы (журналы), а потом чистить?
2. Какие два-три мероприятия не позволят повторить такой инцидент (мониторинг, ротация журналов, блокировка перебора)?

Глава 15

Модуль 15. Сетевые настройки хоста

Идея и словарь

Прежде чем звать сетевого администратора, системный администратор обязан сам прочитать сетевые настройки хоста. Их пять: **адрес** с длиной префикса (какой сети принадлежит сервер), **шлюз по умолчанию** (куда уходят пакеты для всех остальных сетей), **таблица маршрутов** (исключения из правила «всё через шлюз»), **DNS-серверы** (кто переводит имена в адреса) и локальный файл имён `hosts` (проверяется раньше DNS). Большинство жалоб «сервер не видит ресурс» сводятся к ошибке в одном из этих пяти мест. Модуль даёт навык быстро найти его: сначала адрес и маршрут, потом имя, потом доступность.

Порядок поиска имени важен: сначала файл `hosts`, затем DNS. Поэтому запись в `hosts` годится для быстрой проверки и аварийного обхода, но опасна как постоянное решение: она устаревает незаметно для всех, кроме одного сервера.

Linux: инструменты

Адреса и маршруты читают командой `ip addr` и `ip route`; шлюз по умолчанию — строка `default via …`. DNS-серверы перечислены в `/etc/resolv.conf` (строки `nameserver`), порядок разрешения имён задаёт `/etc/nsswitch.conf`, а проверить имя так, как его увидит система, позволяет `getent hosts имя`. Постоянные настройки в Ubuntu лежат в `/etc/netplan/*.yaml`: адреса, шлюз (`routes`) и серверы имён (`nameservers`). В полигоне `ip` печатает учебный снимок; файлы `hosts`, `resolv.conf` и netplan — обычные текстовые, их правят по-настоящему.

Windows: инструменты

Адреса показывает `Get-NetIPAddress`, маршруты — `Get-NetRoute` (шлюз по умолчанию — префикс `0.0.0.0/0`, свойство `NextHop`), DNS-серверы — `Get-DnsClientServerAddress`; менять их позволяют `New-NetRoute` и `Set-DnsClientServerAddress`. Имя проверяет `Resolve-DnsName`. Файл имён находится в `C:\textbackslash Windows\textbackslash System32\textbackslash drivers\textbackslash etc\textbackslash hosts`. В полигоне перечисленные командлеты читают учебные таблицы, а `hosts` — обычный текстовый файл.

Что чаще всего идёт не так

Ошибиться в длине префикса (`/24` вместо `/16`) — значит «потерять» полсети. Два шлюза по умолчанию дают нестабильную работу. Запись в `hosts` перекрывает DNS и забывается.

Практикум Linux: заявки модуля 15

Каталог полигона: `15_set_hosta`. Запуск: `lab 15`, затем `zayavki`.

Новый сервер `web01` подключили к сети, но он «не видит» часть ресурсов: неясно, какой у него адрес, куда он отправляет пакеты, к каким DNS-серверам об-

ращается и почему имя `files` не находится. Прежде чем звать сетевого администратора, системный администратор должен сам прочитать сетевые настройки хоста и привести их в порядок. В модуле используются команды `ip addr`, `ip route` (в полигоне они читают учебные таблицы из `srv/var/lib/lab`), файлы `etc/hosts`, `etc/hostname`, `etc/resolv.conf`, описание сети `etc/netplan/01-net.yaml` и таблица счётчиков интерфейсов `proc/net/dev`. Вне полигона `ip` — настоящая программа.

Заявка Linux 15.1. Адрес сервера

Выясните адрес интерфейса `eth0` (команда `ip -br addr` или `ip addr`). Запишите адрес вместе с длиной маски (вида `10.1.2.3/24`) в `otvety/1`.

Заявка Linux 15.2. Шлюз по умолчанию

Определите адрес шлюза по умолчанию (маршрут `default via …`, команда `ip route`). Запишите адрес шлюза в `otvety/2`.

Заявка Linux 15.3. Серверы DNS

Из файла `srv/etc/resolv.conf` выберите адреса DNS-серверов (строки `nameserver`) и запишите их в `otvety/3`, по одному в строке, отсортировав.

Заявка Linux 15.4. Сколько маршрутов

Сколько маршрутов в таблице (`ip route`)? Запишите число в `otvety/4`.

Заявка Linux 15.5. Непосредственно подключённая сеть

Запишите в `otvety/5` адрес сети (вида `10.1.2.0/24`), в которой находится интерфейс `eth0`: маршрут с `dev eth0 proto kernel`.

Заявка Linux 15.6. Имя для файлового сервера

Имя `files` не находится. Добавьте в `srv/etc/hosts` строку: адрес `10.20.30.50`, полное имя `files.sever.local` и короткое `files`. Проверьте: `getent hosts files`.

Заявка Linux 15.7. Статический адрес в netplan

Файл `srv/etc/netplan/01-net.yaml` описывает интерфейс `eth0` с получением адреса по DHCP. Перепишите его так: `dhcp4: false`, адрес — тот, что вы нашли в задании 1, шлюз — из задания 2 (маршрут `to: default`, `via: …`), DNS — адреса из задания 3. Сохраните структуру YAML (отступы пробелами).

Заявка Linux 15.8. Имя сервера

Переименуйте сервер в `web01`: запишите имя в `srv/etc/hostname` и замените в `srv/etc/hosts` имя `srv01` в строке `127.0.1.1` на `web01`. Строка `localhost` должна остаться.

Заявка Linux 15.9. Интерфейс с потерями

В `srv/proc/net/dev` четвёртое число после имени интерфейса в блоке приёма (`drop`) показывает отброшенные пакеты. Найдите интерфейс с наибольшим числом `drop` и запишите его имя (без двоеточия) в `otvety/9`.

Заявка Linux 15.10. Контрольная: сетевой паспорт хоста

Составьте файл `netreport.txt` (в каталоге модуля) из четырёх строк: `IP=` (адрес с маской из задания 1), `GW=` (шлюз), `DNS=` (два адреса через запятую по возрастанию) и `HOST=` (имя из `srv/etc/hostname`).

Практикум Windows: заявки модуля 15

Каталог полигона: `15_set_hosta`. Запуск: `lab 15`, затем `zayavki`.

Новый сервер `WEB01` подключили к сети, но он «не видит» часть ресурсов: неясно, какой у него адрес, куда он отправляет пакеты, к каким DNS-серверам обращается и почему имя `files` не находится. Прежде чем звать сетевого администратора, системный администратор должен сам прочитать сетевые настройки и привести их в порядок. В модуле используются команды `Get-NetIPAddress`, `Get-NetRoute`, `New-NetRoute`, `Get-DnsClientServerAddress`, `Set-DnsClientServerAddress` и `Resolve-DnsName` (учебные версии читают таблицы `srv\etc\ipaddr.csv`, `routes.csv`, `dns.csv`, `dnszone.csv`) и файл `hosts` (`srv\Windows\System32\drivers\etc\hosts`).

Заявка Windows 15.1. Адрес сервера

Выясните адрес интерфейса `Ethernet` (`Get-NetIPAddress -InterfaceAlias Ethernet`). Запишите адрес вместе с длиной префикса (вида `10.1.2.3/24`) в `otvety\1`.

Заявка Windows 15.2. Шлюз по умолчанию

Определите адрес шлюза по умолчанию: маршрут с префиксом назначения `0.0.0.0/0` (`Get-NetRoute`), свойство `NextHop`. Запишите адрес в `otvety\2`.

Заявка Windows 15.3. Серверы DNS

Какие DNS-серверы использует интерфейс `Ethernet` (`Get-DnsClientServerAddress`, свойство `ServerAddresses`)? Запишите адреса в `otvety\3`, по одному в строке, отсортировав.

Заявка Windows 15.4. Сколько маршрутов

Сколько маршрутов в таблице (`Get-NetRoute`)? Запишите число в `otvety\4`.

Заявка Windows 15.5. Имя для файлового сервера

Имя `files` не находится. Добавьте в файл `srv\Windows\System32\drivers\etc\hosts` строку: адрес `10.20.30.50`, имена `files.sever.local` и `files` (через пробел). Проверьте командой `Resolve-DnsName files`; старые строки сохраните.

Заявка Windows 15.6. Другие DNS-серверы

Задайте для интерфейса `Ethernet` DNS-серверы `10.20.30.2` и `10.20.30.3` (`Set-DnsClientServerAddress -InterfaceAlias Ethernet -ServerAddresses …`).

Заявка Windows 15.7. Маршрут в удалённую сеть

Добавьте статический маршрут в сеть `172.16.0.0/16` через шлюз по умолчанию (адрес из задания 2) на интерфейсе `Ethernet` (`New-NetRoute -DestinationPrefix … -NextHop … -InterfaceAlias …`).

Заявка Windows 15.8. Контрольная: сетевой паспорт хоста

Составьте файл `netreport.txt` (в каталоге модуля) из четырёх строк: `IP=` (адрес с префиксом из задания 1), `GW=` (шлюз), `DNS=` (текущие DNS-серверы интерфейса `Ethernet` через запятую по возрастанию), `ROUTES=` (текущее число маршрутов).

Вопросы для размышления

1. Чем маршрут `proto kernel` отличается от `proto static`, и что произойдёт с доступом к соседним сетям, если удалить шлюз по умолчанию?
2. Почему расхождение между `hostname` и `/etc/hosts` может замедлять запуск служб?

Глава 16

Модуль 16. Хранилище: разделы, LVM, монтирование, swap

Идея и словарь

Диск, раздел, файловая система, том — четыре разных слова. **Диск** — физическое устройство; **раздел** — его часть; **файловая система** — способ хранить файлы внутри раздела; **том** — то, что система монтирует и показывает как каталог или букву. В Linux между разделом и файловой системой часто стоит **LVM**: физические тома объединяются в группу, из неё нарезаются логические тома, которые можно увеличивать на лету. Администратор должен уметь прочитать эту картину, найти свободное место и подготовить новый диск.

Главная ловушка — **таблица монтирования**. Если в `/etc/fstab` опечатка или имя устройства поменялось после перезагрузки, сервер не поднимется. Поэтому диски в `fstab` указывают по `UUID`, а изменения проверяют до перезагрузки.

Linux: инструменты

`lsblk` показывает дерево дисков, разделов и томов; `blkid` — UUID и тип файловой системы; `df -h` — занятость смонтированных систем; `pvs`, `vgs`, `lvs` — физические тома, группы и логические тома LVM; `swapon --show` — подкачку. Файл `/etc/fstab` содержит шесть полей: устройство, точка монтирования, тип, параметры, `dump`, порядок проверки. В полигоне вывод дисковых команд берётся из учебных таблиц, а `fstab` и `mounts` правятся по-настоящему.

Windows: инструменты

`Get-Disk` перечисляет диски, `Initialize-Disk` задаёт стиль таблицы разделов (`GPT` или `MBR`), `Get-Partition` и `New-Partition` работают с разделами, `Get-Volume` и `Format-Volume` — с томами и файловыми системами (`NTFS`, `ReFS`). Новый диск приходит со стилем `RAW`: сначала инициализация, потом раздел, потом форматирование. В полигоне командлеты читают и пишут учебные таблицы.

Что чаще всего идёт не так

Форматирование и удаление раздела уничтожают данные: перед выполнением сверяйте номер диска. Имя устройства `/dev/sdX` в `fstab` может измениться; UUID — нет.

Практикум Linux: заявки модуля 16

Каталог полигона: `16_hranilische`. Запуск: `lab 16`, затем `zayavki`.

К серверу подключили новые диски. Нужно понять, что где лежит: какие диски есть, какой из них свободен, как устроен том LVM, какая файловая система смонтирована в `/var`, и дописать таблицу монтирования `fstab` так, чтобы сер-

вер загружался и после перезагрузки. Ошибка в `fstab` — классическая причина «сервер не поднялся после перезагрузки». Вывод `lsblk`, `blkid`, `lvs`, `vgs`, `pvs` и `swapon --show` в полигоне берётся из учебных таблиц `srv/var/lib/lab`; файлы `srv/etc/fstab` и `srv/proc/mounts` — обычные текстовые, их правят и читают по-настоящему.

Заявка Linux 16.1. Размер диска sdb

Определите размер диска `sdb` по выводу `lsblk` (строка типа `disk`). Запишите значение (например, `30G`) в `otvety/1`.

Заявка Linux 16.2. Свободный диск

Один из дисков (`sda`, `sdb`, `sdc`) не содержит разделов. Запишите его имя в `otvety/2`.

Заявка Linux 16.3. Свободное место в группе томов

Сколько свободного места в группе томов `vg0`? Найдите значение в столбце `VFree` вывода `vgs` и запишите в `otvety/3` (например, `7.00g`).

Заявка Linux 16.4. Крупнейший логический том

Какой логический том (столбец `LV` вывода `lvs`) самый большой? Запишите его имя в `otvety/4`.

Заявка Linux 16.5. Тип файловой системы /var

Определите, какая файловая система смонтирована в `/var`, по `srv/proc/mounts` (третье поле). Запишите тип в `otvety/5`.

Заявка Linux 16.6. Надёжное имя диска в fstab

В `srv/etc/fstab` раздел `/dev/sdc1` подключён по имени устройства: оно может измениться после перезагрузки. Замените его на `UUID=…` (значение из `blkid`) в строке с точкой монтирования `/data`; остальные поля не меняйте.

Заявка Linux 16.7. Опечатка в параметрах

В строке монтирования `/home` в `fstab` вместо `defaults` написано `default`: система не смонтирует том. Исправьте.

Заявка Linux 16.8. Файл подкачки

Размер раздела подкачки (swap) покажет `swapon --show`. Запишите его (например, `2G`) в `otvety/8`. Затем добавьте в `fstab` строку для файла подкачки `/swapfile`: `/swapfile none swap sw 0 0`.

Заявка Linux 16.9. Сколько останется

Планируется создать ещё один логический том размером 4G в группе `vg0`. Сколько свободного места (в гигабайтах, целым числом) останется в группе? Запишите число в `otvety/9`.

Заявка Linux 16.10. Контрольная: паспорт хранилища

Составьте файл `storage.txt` (в каталоге модуля) из четырёх строк: `FREEDISK=` (диск из задания 2), `VG=` (имя группы томов), `VFREE=` (свободное место из задания 3) и `FSVAR=` (тип ФС `/var`).

Практикум Windows: заявки модуля 16

Каталог полигона: `16_hranilische`. Запуск: `lab 16`, затем `zayavki`.

К серверу подключили новые диски. Нужно понять, что где лежит: какие диски есть, какой из них новый и неинициализированный, какие тома и с какой файловой системой уже работают, сколько на них места, и подготовить новый диск: инициализировать, разбить, отформатировать. Работа ведётся командами `Get-Disk`, `Initialize-Disk`, `Get-Partition`, `New-Partition`, `Get-Volume`, `Format-Volume`; учебные версии читают таблицы `srv\etc\disks.csv`, `partitions.csv` и `srv\var\lib\lab\volumes.csv`.

Заявка Windows 16.1. Сколько дисков

Сколько физических дисков видит система (`Get-Disk`)? Запишите число в `otvety\1`.

Заявка Windows 16.2. Новый диск

Один из дисков не инициализирован (стиль раздела `RAW`, статус `Offline`). Запишите его номер (свойство `Number`) в `otvety\2`.

Заявка Windows 16.3. Инициализировать

Инициализируйте новый диск с таблицей разделов GPT (`Initialize-Disk -Number … -PartitionStyle GPT`).

Заявка Windows 16.4. Создать раздел

Создайте на новом диске один раздел на всё свободное место с буквой `E` (`New-Partition -DiskNumber … -UseMaximumSize -DriveLetter E`).

Заявка Windows 16.5. Форматировать том

Отформатируйте том `E:` в NTFS с меткой `Data2` (`Format-Volume -DriveLetter E -FileSystem NTFS -NewFileSystemLabel Data2`). Проверьте `Get-Volume`.

Заявка Windows 16.6. Файловая система тома D

Какая файловая система на томе `D:` (свойство `FileSystem`)? Запишите название в `otvety\6`.

Заявка Windows 16.7. Сколько свободно на D

Сколько процентов тома `D:` свободно (`SizeRemaining / Size * 100`, целое число с округлением)? Запишите в `otvety\7`.

Заявка Windows 16.8. Контрольная: паспорт хранилища

Составьте файл `storage.txt` (в каталоге модуля) из четырёх строк: `DISKS=` (число дисков), `NEWDISK=` (номер нового диска), `VOLUMES=` (число томов с файловой системой после форматирования) и `FREE_D=` (процент свободного места на `D:`).

Вопросы для размышления

1. Почему ошибка в `fstab` опаснее, чем ошибка в обычном конфигурационном файле, и как её безопасно проверить до перезагрузки?
2. Чем логический том LVM удобнее раздела диска, когда место заканчивается?

Глава 17

Модуль 17. Мониторинг и производительность

Идея и словарь

Жалоба «сервер тормозит» не говорит ничего. Задача администратора — заменить ощущение измерением и найти **узкое место**: процессор, память, диск или сеть. Если процессор занят на 90 %, но нагрузка (`load average`) ниже числа ядер, всё хорошо; если очередь процессов длиннее числа ядер, процессам приходится ждать. Если свободной памяти нет и растёт использование подкачки, система «пробуксовывает». Если высок показатель ожидания ввода-вывода (`wa`) и диск загружен почти на 100 %, виноват диск. Один снимок мало что значит: смотрят динамику и сравнивают с обычным состоянием.

Измерения полезны, когда превращены в **пороги и оповещения**: сценарий проверяет нагрузку, перечень процессов-нарушителей и свободное место, а мониторинг сообщает дежурному до того, как пожалуются пользователи. Хорошее оповещение редкое, точное и требует действия.

Linux: инструменты

`top -b -n 1` — процессы по загрузке; `free -m` — память и подкачка; `vmstat` — очередь, подкачка, ввод-вывод и проценты процессора (`us`, `sy`, `id`, `wa`); `iostat` — загрузка дисков (`\%util`, `await`); `df -h` — место на томах; файл `/proc/loadavg` — нагрузка за 1, 5 и 15 минут, `/proc/cpuinfo` — число процессоров. В полигоне эти программы печатают учебные снимки из `srv/var/lib/lab`.

Windows: инструменты

`Get-Process` показывает процессы (свойства `CPU`, `WorkingSet`), `Get-Counter` — счётчики производительности, `Get-Volume` — место на томах. Для анализа результата служат `Sort-Object`, `Where-Object`, `Group-Object` и `Measure-Object`. В полигоне снимок процессов лежит в таблице `procs.csv`, и те же конвейеры применяются к ней через `Import-Csv`.

Что чаще всего идёт не так

Сравнивать числа из текста как строки (`9` больше `50`!) — классическая ошибка: преобразуйте в число. Один всплеск не повод для тревоги; порог без интервала наблюдения даёт ложные срабатывания.

Практикум Linux: заявки модуля 17

Каталог полигона: `17_monitoring`. Запуск: `lab 17`, затем `zayavki`.

Пользователи пожаловались, что сервер «тормозит», но подробностей нет. Администратор отвечает не догадкой, а измерением: кто занимает процессор и память, не уходит ли система в подкачку, не упирается ли диск, достаточно ли

ядер для нагрузки. В модуле вы читаете вывод `top`, `free`, `vmstat`, `iostat`, `df` и `/proc/loadavg` (в полигоне эти программы выводят учебные снимки из `srv/var/lib/lab`, вне полигона работают настоящие), находите «узкое место», пишете проверку нагрузки и список процессов-нарушителей, которые затем можно подключить к настоящему мониторингу.

Заявка Linux 17.1. Кто занимает процессор

В снимке `top` (`top -b -n 1`) найдите процесс с наибольшим значением `%CPU` и запишите его имя (последний столбец) в `otvety/1`.

Заявка Linux 17.2. Кто занимает память

Найдите процесс с наибольшим значением `%MEM` (десятый столбец) и запишите его имя в `otvety/2`.

Заявка Linux 17.3. Доля занятой памяти

По выводу `free -m` вычислите, сколько процентов памяти занято (`used / total * 100`, округлить до целого). Запишите число в `otvety/3`.

Заявка Linux 17.4. Нагрузка за 15 минут

Из файла `srv/proc/loadavg` возьмите среднюю нагрузку за 15 минут (третье число) и запишите в `otvety/4`.

Заявка Linux 17.5. Ожидание диска

В выводе `vmstat` столбец `wa` показывает долю времени ожидания ввода-вывода. Запишите наибольшее значение `wa` из снимка в `otvety/5`.

Заявка Linux 17.6. Загруженный диск

В выводе `iostat` найдите диск с наибольшей загрузкой (последний столбец `%util`) и запишите его имя в `otvety/6`.

Заявка Linux 17.7. Проверка нагрузки

Напишите сценарий `check-load.sh КАТАЛОГ`, который читает `КАТАЛОГ/proc/loadavg` и `КАТАЛОГ/proc/cpuinfo`: если нагрузка за 1 минуту **меньше** числа процессоров (строк `processor`), печатает `OK` и завершается с кодом 0; иначе печатает `ALERT` и завершается с кодом 1.

Заявка Linux 17.8. Процессы-нарушители

Составьте файл `alert.txt` (в каталоге модуля): имена процессов с `%CPU` больше 50 и их значения в виде `имя значение`, по убыванию значения.

Заявка Linux 17.9. Заполненные файловые системы

По `df -h` найдите файловые системы, занятые на 85 % и более, и запишите их точки монтирования (последний столбец) в `otvety/9`, по одной в строке, отсортировав.

Заявка Linux 17.10. Контрольная: сводка для дежурного

Составьте файл `monitor.txt` (в каталоге модуля) из четырёх строк: `TOPCPU=`, `TOPMEM=` (имена из заданий 1–2), `MEMPCT=` (доля памяти из задания 3), `LOAD15=` (нагрузка из задания 4).

Практикум Windows: заявки модуля 17

Каталог полигона: `17_monitoring`. Запуск: `lab 17`, затем `zayavki`.

Пользователи жалуются, что сервер «тормозит». Администратор отвечает измерением, а не догадкой: какие процессы занимают процессор и память, сколько их, не заканчивается ли место на томах. В полигоне снимок процессов лежит в файле `srv\var\lib\lab\procs.csv` (столбцы `Name`, `Id`, `CPU`, `WorkingSetMB`), тома описаны в `volumes.csv` (читает `Get-Volume`). Для анализа служат `Import-Csv`, `Sort-Object`, `Group-Object`, `Measure-Object` — те же приёмы работают с настоящим `Get-Process`.

Заявка Windows 17.1. Кто занимает процессор

В снимке `srv\var\lib\lab\procs.csv` найдите процесс с наибольшим значением `CPU` и запишите его имя в `otvety\1`.

Заявка Windows 17.2. Кто занимает память

Найдите процесс с наибольшим `WorkingSetMB` и запишите его имя в `otvety\2`.

Заявка Windows 17.3. Суммарная память

Сколько мегабайт памяти занимают все процессы снимка вместе? Запишите число в `otvety\3`.

Заявка Windows 17.4. Сколько экземпляров svchost

Сколько процессов с именем `svchost` в снимке? Запишите число в `otvety\4`.

Заявка Windows 17.5. Сколько «горячих» процессов

Сколько процессов используют `CPU` больше 50? Запишите число в `otvety\5`.

Заявка Windows 17.6. Процессы-нарушители

Составьте файл `alert.txt` (в каталоге модуля): строки `имя значение` для процессов с `CPU` больше 50, по убыванию значения.

Заявка Windows 17.7. Тома с малым запасом

Найдите тома, на которых свободно **меньше 15 %** (`Get-Volume`, `SizeRemaining / Size`). Запишите буквы дисков в `otvety\7`, по одной в строке, отсортировав; если таких нет, оставьте файл пустым.

Заявка Windows 17.8. Контрольная: сводка для дежурного

Составьте файл `monitor.txt` (в каталоге модуля) из четырёх строк: `TOPCPU=`, `TOPMEM=` (имена из заданий 1–2), `MEMTOTAL=` (сумма из задания 3) и `HOT=` (число из задания 5).

Вопросы для размышления

1. Почему одной цифры «загрузка процессора 90 %» недостаточно, чтобы объяснить медленную работу сервера?
2. Какие пороги вы бы настроили для оповещений и почему важно избегать ложных тревог?

Глава 18

Модуль 18. Хеши, шифрование и сертификаты

Идея и словарь

Три понятия часто путают. **Кодирование** (base64) — только другая запись тех же данных, раскодировать может любой. **Хеш** (SHA-256) — «отпечаток» данных фиксированной длины: изменение одного бита меняет его до неузнаваемости, а восстановить данные по хешу нельзя; хеш отвечает на вопрос «файл тот же?». **Шифрование** и **электронная подпись** используют ключи: шифр скрывает содержимое, подпись доказывает автора и неизменность. Контрольная сумма, лежащая рядом с файлом, защищает от случайной порчи, но не от злоумышленника, который подменит и файл, и сумму: для этого сумму подписывают.

Сертификат связывает имя сервера с открытым ключом и подписан удостоверяющим центром. Администратору важны четыре поля: кому выдан (Subject), какие имена покрывает (Subject Alternative Name), кем подписан (Issuer) и когда истекает (Not After). Истёкший сертификат останавливает сайт так же, как поломка сервера, поэтому срок действия берут под мониторинг заранее.

Linux: инструменты

sha256sum файл печатает хеш, sha256sum -c список проверяет файлы по списку и помечает несовпадения словом FAILED; base64 и base64 -d кодируют и раскодируют; openssl x509 -in cert.pem -noout -text показывает поля сертификата, а openssl x509 -noout -enddate — срок действия. В полигоне описание сертификата лежит в текстовом файле в том виде, в каком его печатает openssl.

Windows: инструменты

Get-FileHash -Algorithm SHA256 вычисляет хеш (свойство Hash, буквы заглавные), [Convert]::ToBase64String и FromBase64String кодируют и раскодируют, Get-ChildItem Cert:\textbackslash LocalMachine\textbackslash My перечисляет сертификаты хранилища (свойства Subject, NotAfter). В полигоне описание сертификата лежит в текстовом файле.

Что чаще всего идёт не так

Сумму нельзя брать из того же источника, что и файл. Сравнивать хеши нужно целиком. base64 не скрывает пароль.

Практикум Linux: заявки модуля 18

Каталог полигона: 18_crypto. Запуск: lab 18, затем zayavki.

Из хранилища скачали набор файлов и список контрольных сумм SHA256SUMS, но один файл по дороге повредился. Нужно доказать это числом, а не на глаз: пересчитать хеши, найти расхождение, расшифровать секрет, закодированный

в base64, и прочитать описание сертификата веб-сервера (поля, срок действия, альтернативные имена), чтобы вовремя заметить, что он скоро истечёт. Файлы лежат в srv/var/www/files, текстовое описание сертификата — в srv/etc/ssl/cert.txt (в таком виде его печатает openssl x509 -text).

Заявка Linux 18.1. Хеш файла

Вычислите SHA-256 файла srv/var/www/files/f1.dat (sha256sum) и запишите **только хеш** (64 символа) в otvety/1.

Заявка Linux 18.2. Какой файл повреждён

Проверьте файлы по списку SHA256SUMS (sha256sum -c, запускать в каталоге srv/var/www/files). Запишите имя файла, не прошедшего проверку, в otvety/2.

Заявка Linux 18.3. Раскодировать секрет

Файл srv/var/lib/lab/secret.b64 закодирован в base64. Раскодируйте его (base64 -d) и запишите результат в otvety/3.

Заявка Linux 18.4. Имя сертификата

Из srv/etc/ssl/cert.txt возьмите имя сервера (CN= в строке Subject, без O=). Запишите в otvety/4.

Заявка Linux 18.5. Срок действия

Узнайте дату окончания действия сертификата (строка Not After) и запишите её в виде ГГГГ-ММ-ДД в otvety/5.

Заявка Linux 18.6. Сколько имён

Сколько альтернативных имён (DNS:) перечислено в сертификате? Запишите число в otvety/6.

Заявка Linux 18.7. Сценарий контрольных сумм

Напишите сценарий mkhash.sh КАТАЛОГ, который создаёт в каталоге файл SHA256SUMS с хешами всех остальных файлов (сам SHA256SUMS в список не входит); проверка sha256sum -c должна проходить.

Заявка Linux 18.8. Контрольная: акт проверки

Составьте файл crypto.txt (в каталоге модуля) из четырёх строк: BAD= (повреждённый файл), SECRET= (раскодированный секрет), CN= (имя сертификата), EXPIRES= (дата окончания).

Практикум Windows: заявки модуля 18

Каталог полигона: 18_crypto. Запуск: lab 18, затем zayavki.

Из хранилища скачали набор файлов и список контрольных сумм SHA256SUMS, но один файл по дороге повредился. Нужно доказать это числом: пересчитать

хеши командой `Get-FileHash`, найти расхождение, раскодировать секрет из base64 и прочитать описание сертификата веб-сервера, чтобы вовремя заметить скорое истечение срока. Файлы лежат в `srv\var\www\files`, описание сертификата — в `srv\etc\ssl\cert.txt` (в таком виде его печатает `openssl x509 -text`).

Заявка Windows 18.1. Хеш файла

Вычислите SHA-256 файла `srv\var\www\files\f1.dat` (`Get-FileHash -Algorithm SHA256`) и запишите **только хеш** (свойство `Hash`) в `otvety\1`.

Заявка Windows 18.2. Какой файл повреждён

Сравните хеши файлов со списком `srv\var\www\files\SHA256SUMS` (формат `хеш имя`). Запишите имя файла, у которого хеш не совпал, в `otvety\2`.

Заявка Windows 18.3. Раскодировать секрет

Файл `srv\var\lib\lab\secret.b64` закодирован в base64. Раскодируйте (`[Convert]::FromBase64String`, затем `[Text.Encoding]::UTF8.GetString`) и запишите результат в `otvety\3`.

Заявка Windows 18.4. Имя сертификата

Из `srv\etc\ssl\cert.txt` возьмите имя сервера (`CN=` в строке `Subject`, без `O=`) и запишите в `otvety\4`.

Заявка Windows 18.5. Срок действия

Запишите дату окончания действия сертификата (строка `Not After`) в виде `ГГГГ-ММ-ДД` в `otvety\5`.

Заявка Windows 18.6. Сколько имён

Сколько альтернативных имён (`DNS:`) перечислено в сертификате? Запишите число в `otvety\6`.

Заявка Windows 18.7. Список контрольных сумм

Составьте файл `hashes.txt` (в каталоге модуля): для каждого файла `f*.dat` из `srv\var\www\files` строка `имя ХЕШ` (хеш как выдаёт `Get-FileHash`), по возрастанию имени.

Заявка Windows 18.8. Контрольная: акт проверки

Составьте файл `crypto.txt` (в каталоге модуля) из четырёх строк: `BAD=` (повреждённый файл), `SECRET=` (раскодированный секрет), `CN=` (имя сертификата), `EXPIRES=` (дата окончания).

Вопросы для размышления

1. Чем контрольная сумма отличается от электронной подписи, и от чего защищает каждая?
2. Почему base64 нельзя считать шифрованием, а истекший сертификат — мелочью?

Глава 19

Модуль 19. Веб-сервер: конфигурация и журнал доступа

Идея и словарь

Веб-сервер — обычная служба с текстовой конфигурацией и журналом. Администратору нужно уметь три вещи: прочитать конфигурацию (на каком порту и для какого имени слушает сервер, откуда отдаёт файлы), безопасно её изменить и по журналу доступа понять, что происходит на самом деле. Конфигурация nginx состоит из **блоков** в фигурных скобках и **директив**, каждая из которых заканчивается точкой с запятой; пропущенная точка с запятой или лишняя скобка не дают серверу запуститься. Поэтому перед перезагрузкой конфигурацию проверяют (`nginx -t`), а не «смотрят глазами».

Журнал доступа — строка на запрос: адрес клиента, время, запрос, **код ответа** и размер. Коды группируются: 2xx — успех, 3xx — перенаправление, 4xx — ошибка клиента (часто 404: страницы нет), 5xx — ошибка сервера. Резкий рост 5xx — повод для тревоги, рост 404 — чаще неверные ссылки или сканирование сайта. Подсчёт и группировка строк журнала — те же приёмы, что в модуле 5.

Linux: инструменты

Конфигурация: `/etc/nginx/nginx.conf` и каталог `sites-enabled`; проверка `nginx -t`, применение `systemctl reload nginx`. Журнал: `/var/log/nginx/access.log`. Разбор: `awk '\$9 == 404'`, `sort | uniq -c | sort -rn`, `grep -n`.

Windows: инструменты

Для IIS и nginx под Windows принципы те же: конфигурация — текстовый файл (`web.config` или `nginx.conf`), журнал — текстовый файл. Разбор: `Get-Content`, `-split`, `Group-Object`, `Where-Object`.

Что чаще всего идёт не так

Поиск `grep 404` находит и другие поля строки: проверяйте нужный столбец. Перезагрузка без проверки конфигурации может остановить сайт.

Практикум Linux: заявки модуля 19

Каталог полигона: `19_webserver`. Запуск: `lab 19`, затем `zayavki`.

Корпоративный портал отвечает медленно, а у разработчиков есть жалоба: «часть страниц не открывается». Администратору нужно сделать две вещи: привести конфигурацию веб-сервера в порядок (порт, корневой каталог, скрытие версии сервера, отсутствие синтаксических ошибок) и по журналу доступа понять, что происходит: сколько запросов завершилось ошибкой, какой клиент шлёт больше всего запросов. Конфигурация лежит в `srv/etc/nginx/sites/portal.conf` (формат nginx), журнал — в `srv/var/log/nginx/access.log`.

Заявка Linux 19.1. Порт сайта

В `srv/etc/nginx/sites/portal.conf` найдите порт, который слушает сервер (директива `listen`). Запишите номер в `otvety/1`.

Заявка Linux 19.2. Корневой каталог

Какой каталог указан в директиве `root`? Запишите путь (без `;`) в `otvety/2`.

Заявка Linux 19.3. Скрыть версию сервера

Добавьте в блок `server` директиву `server_tokens off;` (сервер не будет сообщать свою версию в заголовках и страницах ошибок). Остальные строки не удаляйте.

Заявка Linux 19.4. Страницы не найдены

Сколько запросов в `srv/var/log/nginx/access.log` завершилось кодом `404`? Код ответа — девятое поле строки. Запишите число в `otvety/4`.

Заявка Linux 19.5. Самый активный клиент

Какой адрес клиента (первое поле) встречается в журнале чаще всего? Запишите в `otvety/5`.

Заявка Linux 19.6. Ошибки сервера

Сколько запросов завершилось ошибкой сервера (коды `5xx`)? Запишите число в `otvety/6`.

Заявка Linux 19.7. Пропущенные точки с запятой

В старой конфигурации `srv/etc/nginx/old.conf` у двух директив пропущена `;` в конце строки (строки с `{` и `}` не считаются). Запишите номера этих строк в `otvety/7`, по одному в строке, по возрастанию. Файл не правьте.

Заявка Linux 19.8. Проверка конфигурации

Напишите сценарий `check-conf.sh ФАЙЛ`, который печатает `GOOD` и завершается с кодом 0, если число `{` равно числу `}` и каждая непустая строка (кроме комментариев `#`) оканчивается на `;`, `{` или `}`; иначе печатает `BAD` и завершается с кодом 1.

Заявка Linux 19.9. Контрольная: справка по порталу

Составьте файл `web.txt` (в каталоге модуля) из четырёх строк: `PORT=`, `ROOT=` (из заданий 1–2), `NOTFOUND=` (число 404) и `TOPCLIENT=` (адрес из задания 5).

Практикум Windows: заявки модуля 19

Каталог полигона: `19_webserver`. Запуск: `lab 19`, затем `zayavki`.

Корпоративный портал отвечает медленно, а у разработчиков есть жалоба: «часть страниц не открывается». Администратору нужно привести конфигура-

цию веб-сервера в порядок (порт, корневой каталог, скрытие версии сервера, отсутствие синтаксических ошибок) и по журналу доступа понять, что происходит: сколько запросов завершилось ошибкой, какой клиент шлёт больше всего запросов. Конфигурация лежит в `srv\etc\nginx\sites\portal.conf` (формат nginx), журнал — в `srv\var\log\nginx\access.log`.

Заявка Windows 19.1. Порт сайта

В `srv\etc\nginx\sites\portal.conf` найдите порт, который слушает сервер (директива `listen`). Запишите номер в `otvety\1`.

Заявка Windows 19.2. Корневой каталог

Какой каталог указан в директиве `root`? Запишите путь (без `;`) в `otvety\2`.

Заявка Windows 19.3. Скрыть версию сервера

Добавьте в блок `server` директиву `server_tokens off;`, чтобы сервер не сообщал свою версию. Остальные строки сохраните.

Заявка Windows 19.4. Страницы не найдены

Сколько запросов в `srv\var\log\nginx\access.log` завершилось кодом `404`? Код ответа — девятое слово строки. Запишите число в `otvety\4`.

Заявка Windows 19.5. Самый активный клиент

Какой адрес клиента (первое слово) встречается в журнале чаще всего? Запишите в `otvety\5`.

Заявка Windows 19.6. Ошибки сервера

Сколько запросов завершилось ошибкой сервера (коды `5xx`)? Запишите число в `otvety\6`.

Заявка Windows 19.7. Пропущенные точки с запятой

В старой конфигурации `srv\etc\nginx\old.conf` у двух директив пропущена `;` (строки с `{` и `}` не считаются). Запишите номера этих строк в `otvety\7`, по одному в строке, по возрастанию. Файл не правьте.

Заявка Windows 19.8. Контрольная: справка по порталу

Составьте файл `web.txt` (в каталоге модуля) из четырёх строк: `PORT=`, `ROOT=` (задания 1–2), `NOTFOUND=` (число 404) и `TOPCLIENT=` (адрес из задания 5).

Вопросы для размышления

1. Почему перед перезагрузкой веб-сервера конфигурацию сначала проверяют отдельной командой (`nginx -t`)?
2. Что можно узнать о работе сайта по одному только журналу доступа, а чего нельзя?

Глава 20

Модуль 20. Конфигурации под контролем версий

Идея и словарь

Конфигурация сервера — такой же результат труда, как программа, и обращаться с ней следует так же: хранить в системе контроля версий. **Репозиторий** `git` запоминает последовательность **коммитов**; каждый содержит снимок файлов, автора, время и сообщение. Благодаря этому можно ответить на вопросы «что изменилось», «кто и зачем» и «как вернуть, как было». **Ветка** позволяет готовить исправление, не трогая основную линию. Хорошее сообщение коммита отвечает на вопрос «зачем», а не «что» (что видно в различиях).

Вторая идея — **идемпотентность**: действие, повторное выполнение которого не меняет результат. «Добавить строку» неидемпотентно (при повторе будет дубль), «убедиться, что строка есть» идемпотентно. Именно такие сценарии можно безопасно запускать по расписанию и после сбоя; на этом принципе построены системы управления конфигурацией (Ansible, Puppet, DSC).

Linux: инструменты

`git status`, `git log --oneline`, `git diff`, `git add`, `git commit -m`, `git switch -c ветка`, `git log -S строка` (поиск коммита по тексту). Идемпотентная вставка строки: `grep -qxF -- "\$line" файл || echo "\$line" >> файл`.

Windows: инструменты

Команды `git` те же. Идемпотентная вставка строки: условие `if (-not (Select-String -Path файл -Pattern '\^{}строка\$' -Quiet)) \{ Add-Content … \}`. Для настройки Windows средствами конфигурации служит PowerShell DSC, который устроен по тому же принципу.

Что чаще всего идёт не так

Коммит «Исправления» ничего не объясняет. Секреты (пароли, ключи) в репозиторий не кладут. Повторяемый сценарий без проверки «уже сделано» создаёт дубли.

Практикум Linux: заявки модуля 20

Каталог полигона: `20_git_iac`. Запуск: `lab 20`, затем `zayavki`.

После очередной «правки на живом сервере» сайт перестал работать, и никто не помнит, что именно меняли. Решение — хранить конфигурации в репозитории `git`: каждое изменение имеет автора, время и пояснение, любое состояние можно восстановить, а сравнение двух версий показывает, что сломалось. Вторая половина практики — **идемпотентность**: сценарий настройки можно запускать сколько угодно раз, результат один и тот же. В модуле используется

настоящий git; репозиторий лежит в srv/opt/infra.

Заявка Linux 20.1. Сколько коммитов

Сколько коммитов в истории репозитория srv/opt/infra (git rev-list --count HEAD)? Запишите число в otvety/1.

Заявка Linux 20.2. Первое сообщение

Какое сообщение у самого первого (старейшего) коммита? Запишите его в otvety/2.

Заявка Linux 20.3. Кто включил отладку

Строка debug=true попала в конфигурацию одним из коммитов. Найдите его (git log -S debug=true) и запишите его сообщение в otvety/3.

Заявка Linux 20.4. Что не закоммичено

В рабочем каталоге есть незафиксированное изменение. Какой это файл (git status --porcelain)? Запишите имя в otvety/4.

Заявка Linux 20.5. Зафиксировать изменение

Зафиксируйте это изменение коммитом с сообщением Update ИМЯ_ФАЙЛА (имя из задания 4). После коммита рабочий каталог должен быть чистым.

Заявка Linux 20.6. Ветка для срочной правки

Создайте ветку hotfix и переключитесь на неё (git switch -c hotfix или git checkout -b hotfix).

Заявка Linux 20.7. Идемпотентная настройка

Напишите сценарий ensure-line.sh ФАЙЛ СТРОКА: если строки нет в файле (точное совпадение целой строки), он дописывает её в конец; если есть — ничего не меняет. Повторные запуски не должны давать дублей.

Заявка Linux 20.8. Контрольная: паспорт репозитория

Составьте файл infra.txt (в каталоге модуля) из четырёх строк: FIRST= (сообщение первого коммита), DEBUG= (сообщение из задания 3), CHANGED= (файл из задания 4), BRANCH= (текущая ветка).

Практикум Windows: заявки модуля 20

Каталог полигона: 20_git_iac. Запуск: lab 20, затем zayavki.

После очередной «правки на живом сервере» сайт перестал работать, и никто не помнит, что именно меняли. Решение — хранить конфигурации в репозитории git: каждое изменение имеет автора, время и пояснение, любое состояние можно восстановить, а сравнение двух версий показывает, что сломалось. Вторая половина практики — **идемпотентность**: настройку можно выполнять

сколько угодно раз, результат один и тот же. В модуле используется настоящий `git`; репозиторий лежит в `srv\opt\infra`.

Заявка Windows 20.1. Сколько коммитов

Сколько коммитов в истории репозитория `srv\opt\infra` (`git rev-list --count HEAD`)? Запишите число в `otvety\1`.

Заявка Windows 20.2. Первое сообщение

Какое сообщение у самого первого (старейшего) коммита? Запишите его в `otvety\2`.

Заявка Windows 20.3. Кто включил отладку

Строка `debug=true` попала в конфигурацию одним из коммитов. Найдите его (`git log -S debug=true`) и запишите его сообщение в `otvety\3`.

Заявка Windows 20.4. Что не закоммичено

В рабочем каталоге есть незафиксированное изменение. Какой это файл (`git status --porcelain`)? Запишите имя в `otvety\4`.

Заявка Windows 20.5. Зафиксировать изменение

Зафиксируйте это изменение коммитом с сообщением `Update ИМЯ_ФАЙЛА` (имя из задания 4). После коммита рабочий каталог должен быть чистым.

Заявка Windows 20.6. Ветка для срочной правки

Создайте ветку `hotfix` и переключитесь на неё (`git switch -c hotfix` или `git checkout -b hotfix`).

Заявка Windows 20.7. Идемпотентная настройка

В файл `srv\opt\settings.conf` нужно добавить строку `retries=3`, но только если её там нет. Выполните это так, чтобы повторный запуск вашей команды не создавал дубль (проверьте условием `Select-String`); остальные строки сохраните.

Заявка Windows 20.8. Контрольная: паспорт репозитория

Составьте файл `infra.txt` (в каталоге модуля) из четырёх строк: `FIRST=` (сообщение первого коммита), `DEBUG=` (сообщение из задания 3), `CHANGED=` (файл из задания 4), `BRANCH=` (текущая ветка).

Вопросы для размышления

1. Чем коммит лучше копии файла с припиской `_old` и почему полезны содержательные сообщения коммитов?
2. Почему идемпотентность — главное свойство сценария настройки, который запускают по расписанию?

Глава 21

Модуль 21. Копии: ротация, инкременты и восстановление

Идея и словарь

Резервная копия имеет ценность только тогда, когда из неё можно восстановить данные. Поэтому политика копирования состоит не из одного действия, а из четырёх: **создавать** копии по расписанию, **проверять** их читаемость, **ограничивать** количество (ротация: старые удаляются, чтобы не заполнять диск) и **восстанавливать** на пробу. Классическое правило «3-2-1» требует три копии данных на двух разных носителях, одну из них — в другом месте.

Копии бывают **полные** (всё содержимое) и **инкрементные** (только изменения с момента прошлой копии). Чтобы найти изменения, нужна точка отсчёта — метка времени последней копии: файлы новее метки попадают в инкремент. Имена архивов с датой в формате ГГГГ-ММ-ДД удобны тем, что сортировка по имени совпадает с сортировкой по времени, и ротация сводится к «оставить последние N».

Linux: инструменты

`tar -czf архив -C каталог .` создаёт архив, `tar -tzf архив` показывает содержимое (и проверяет читаемость), `tar -xzf архив -C цель ./файл` восстанавливает один файл; `gzip -t` проверяет целостность сжатия. Ротация: `ls … | sort | head -n -N | xargs rm`. Изменения с момента метки: `find каталог -type f -newer метка`. Программы `rsync` и `restic` решают те же задачи удобнее (инкрементная передача, шифрование, дедупликация); полигон использует только стандартные средства.

Windows: инструменты

`Compress-Archive` и `Expand-Archive` создают и распаковывают архивы zip; читаемость проверяют открытием архива классом `[IO.Compression.ZipFile]::OpenRead` в блоке `try`. Ротация: `Sort-Object Name | Select-Object -SkipLast N | Remove-Item`. Изменения с момента метки: сравнение `LastWriteTime` с временем метки. Для больших объёмов служат `robocopy` (см. модуль 10) и теневые копии.

Что чаще всего идёт не так

Непроверенная копия может оказаться пустой или повреждённой. Ротация без проверки числа файлов может удалить всё. Копии на том же диске не защищают от отказа диска.

Практикум Linux: заявки модуля 21

Каталог полигона: `21_backup_policy`. Запуск: `lab 21`, затем `zayavki`.

Копии делаются каждую ночь, но никто не проверял, что их можно прочитать. Накопилось несколько архивов, один из них, судя по жалобам, «битый»; диск заполняется старыми копиями; а восстановить нужно один-единственный файл из последней **исправной** копии. Копия, которую ни разу не проверили, — не копия. В модуле вы пересчитываете и проверяете архивы `data-ГГГГ-ММ-ДД.tar.gz` в `srv/backup`, восстанавливаете один файл (`tar -x`), оставляете только нужное число свежих копий и находите файлы, изменившиеся с момента последней копии (основа инкрементной копии).

Заявка Linux 21.1. Сколько копий

Сколько архивов `data-*.tar.gz` лежит в `srv/backup`? Запишите число в `otvety/1`.

Заявка Linux 21.2. Самая свежая копия

Какая копия самая свежая? Дата входит в имя файла. Запишите имя (без каталога) в `otvety/2`.

Заявка Linux 21.3. Битый архив

Один архив повреждён. Найдите его проверкой `gzip -t` (или `tar -tzf`) и запишите имя в `otvety/3`.

Заявка Linux 21.4. Восстановить один файл

Восстановите файл `report.txt` из самой свежей **исправной** копии в каталог `restore` (`tar -xzf … -C restore ./report.txt`).

Заявка Linux 21.5. Оставить пять копий

Диск заполняется. Удалите самые старые архивы так, чтобы в `srv/backup` осталось ровно пять самых свежих `data-*.tar.gz` (метку `.last` не трогайте).

Заявка Linux 21.6. Что изменилось с прошлой копии

Метка последней копии — файл `srv/backup/.last`. Выведите имена файлов из `srv/data`, изменённых позже метки (`find -newer`), в `otvety/6`, по одному в строке, по возрастанию.

Заявка Linux 21.7. Сценарий ротации

Напишите сценарий `rotate.sh КАТАЛОГ N`: в каталоге остаются N самых свежих файлов `data-*.tar.gz` (по имени), остальные удаляются; другие файлы не затрагиваются.

Заявка Linux 21.8. Контрольная: отчёт о копиях

Составьте файл `backup.txt` (в каталоге модуля) из четырёх строк: `COUNT=` (число копий в начале, задание 1), `BAD=` (битый архив), `RESTORED=` (первая строка восстановленного `restore/report.txt`) и `CHANGED=` (сколько файлов изменилось, задание 6).

Практикум Windows: заявки модуля 21

Каталог полигона: `21_backup_policy`. Запуск: `lab 21`, затем `zayavki`.

Копии делаются каждую ночь, но никто не проверял, что их можно прочитать. Накопилось несколько архивов, один из них, судя по жалобам, «битый»; диск заполняется старыми копиями; а восстановить нужно один-единственный файл из последней **исправной** копии. Копия, которую ни разу не проверили, — не копия. В модуле вы проверяете архивы `data-ГГГГ-ММ-ДД.zip` в `srv\backup`, восстанавливаете один файл (`Expand-Archive`), оставляете только нужное число свежих копий и находите файлы, изменившиеся с момента последней копии (основа инкрементной копии).

Заявка Windows 21.1. Сколько копий

Сколько архивов `data-*.zip` лежит в `srv\backup`? Запишите число в `otvety\1`.

Заявка Windows 21.2. Самая свежая копия

Какая копия самая свежая? Дата входит в имя файла. Запишите имя (без каталога) в `otvety\2`.

Заявка Windows 21.3. Битый архив

Один архив повреждён. Найдите его проверкой открытия архива (`[IO.Compression.ZipFile]::OpenRead`, перехват ошибки) и запишите имя в `otvety\3`.

Заявка Windows 21.4. Восстановить один файл

Восстановите файл `report.txt` из самой свежей **исправной** копии в каталог `restore` (`Expand-Archive` целиком во временный каталог и копирование файла, либо чтение записи архива).

Заявка Windows 21.5. Оставить пять копий

Диск заполняется. Удалите самые старые архивы так, чтобы в `srv\backup` осталось ровно пять самых свежих `data-*.zip` (файл `last.txt` не трогайте).

Заявка Windows 21.6. Что изменилось с прошлой копии

Метка последней копии — время изменения файла `srv\backup\last.txt`. Выведите имена файлов из `srv\data`, изменённых позже метки, в `otvety\6`, по одному в строке, по возрастанию.

Заявка Windows 21.7. Контрольная: отчёт о копиях

Составьте файл `backup.txt` (в каталоге модуля) из четырёх строк: `COUNT=` (число копий в начале, задание 1), `BAD=` (битый архив), `RESTORED=` (первая строка `restore\report.txt`) и `CHANGED=` (сколько файлов изменилось, задание 6).

Вопросы для размышления

1. Почему копию нужно не только создавать, но и регулярно пробовать восстанавливать?
2. Как выбрать, сколько копий хранить, и чем правило «3-2-1» защищает от потери данных?

Часть III

Базы данных и язык SQL (SQLite)

О части: базы данных и язык SQL

Зачем системному администратору база данных

Почти любая служба, за которой присматривает администратор, хранит данные в базе: сайт — заказы и пользователей, система мониторинга — метрики, почтовый сервер — очереди и адресные книги, браузер и мессенджер на рабочем месте — историю и настройки. Когда что-то ломается, администратору приходится заглянуть в базу, понять, что в ней лежит, сделать копию, восстановить, перенести, исправить ошибочные данные и объяснить разработчику, что не так. Для этого нужно знать язык баз данных SQL и устройство самой базы.

В этой части вы изучаете **SQLite** — базу данных в одном файле, без сервера. Ей удобно учиться: установка не нужна, базу можно копировать и ломать, а язык SQL — тот же, что в больших системах (PostgreSQL, MySQL, Microsoft SQL Server), с небольшими отличиями в деталях. Часть состоит из 18 тем и 590 заданий. Темы идут от первого запроса к проектированию, индексам, триггерам, транзакциям, копиям и итоговому проекту; тема 18 посвящена JSON, полнотекстовому поиску и подключению баз. Оформление то же, что в практикуме по командам Linux: тема даёт общие сведения, а затем идут *разделы* — по одной команде или конструкции. В каждом разделе сначала описана команда (назначение, синтаксис, правила, пример), потом даны задания на неё. В темах 1–8 на каждую команду приходится по десять заданий; в темах 9–18 задания идут цепочкой и охватывают несколько команд.

Что нужно установить

Что	Как
Python 3	нужен для создания полигона и проверки (модуль `sqlite3` входит в стандартную поставку). Ubuntu: `python3` уже установлен; Windows: python.org или `winget install Python.Python.3`
Программа `sqlite3`	консоль для работы с базой. Ubuntu: `sudo apt install sqlite3`. Windows: `winget install SQLite.SQLite` либо архив sqlite-tools с sqlite.org. Вместо неё допустима консоль `python3 -m sqlite3 файл.db` (Python 3.12 и новее): базовые команды те же
Текстовый редактор	любой; сценарии сохраняйте в UTF-8

Полигон

Как и в предыдущих частях, ученик получает набор учебных баз и программу проверки. Полигон не зависит от операционной системы; создаёт его сценарий на Python.

```
python3 setup.py                          # создать ~/sql-lab (Windows: python
setup.py)
cd ~/sql-lab/01_pervye_zaprosy
```

```
sqlite3 shop.db                          # работа с базой
python3 ~/sql-lab/proverka.py 1          # проверка темы 1
python3 ~/sql-lab/proverka.py            # сводка и уровень
python3 ~/sql-lab/proverka.py -p 1.3     # подсказка к заданию 3 темы 1
python3 setup.py 5                       # пересоздать тему 5 (если база
испорчена)
```

Каждая тема — отдельный каталог полигона с учебными базами (`*.db`), файлом заданий `ZAYAVKI.md` (тот же текст, что в книге) и каталогом ответов `otvety`. Базы создаются заново при пересоздании темы, поэтому ошибаться можно спокойно. Данные индивидуальны (случайное число из `.seed`): ответы нельзя списать.

Учебные базы

База	Содержание
`shop.db`	интернет-магазин: клиенты (`customers`), товары (`products`), поставщики (`suppliers`), заказы (`orders`) и их строки (`order_items`)
`hr.db`	отделы (`departments`) и сотрудники (`employees`) с руководителями
`logs.db`	журнал запросов к сайту (`access`): время, адрес, метод, путь, код ответа, размер, время обработки; те же данные, что разбирались утилитами в модуле «Журналы и расследование»
Остальные	`sklad.db`, `uchet.db`, `bank.db`, `security.db`, `helpdesk.db`: создаются в соответствующих темах

Как выполнять задания

Ответ на вопрос записывается в файл `otvety/N`: результат запроса (для задания 3 — файл `otvety/3`). Есть три равноправных способа.

```
sqlite3 shop.db "SELECT COUNT(*) FROM customers;" > otvety/2     # из оболочки
(bash)

sqlite3 shop.db                                                  # внутри консоли
sqlite3
sqlite> .output otvety/2
sqlite> SELECT COUNT(*) FROM customers;
sqlite> .output stdout
```

Третий способ — сценарий на Python (появляется в теме о сценариях). В Windows PowerShell 5.1 оператор `>` записывает файл в кодировке UTF-16; проверка умеет её читать, но надёжнее пользоваться `.output` внутри `sqlite3`. Формат вывода по умолчанию (значения через вертикальную черту `|`, без заголовков) подходит для проверки; `.mode column`, `.mode csv`, `.headers on` тоже допустимы: программа проверки разбирает основные форматы и пропускает строку заголовков. Задания, где порядок строк существен, требуют `ORDER BY` и указывают порядок явно; остальные проверяются без учёта порядка.

Задания тем 9–15, 17 и 18 **изменяют** базу и проверяются по её состоянию. Они выполняются *по порядку*: каждое опирается на результат предыдущих. Если база испорчена, тему пересоздают командой `python3 setup.py N`. Перед из-

менением данных всегда делайте копию файла базы (`cp shop.db shop.db.bak`) — это тоже привычка администратора.

О диалектах SQL

Примеры и задания рассчитаны на SQLite версии 3.35 и новее (нужны `RETURNING`, оконные функции, `ALTER TABLE ... DROP COLUMN`). Основные конструкции (`SELECT`, `JOIN`, `GROUP BY`, подзапросы, CTE, транзакции) одинаковы в других базах; отличаются типы данных, функции работы с датами и строками, права доступа (в SQLite их нет: доступом к базе управляет операционная система через права на файл) и способ подключения. Версию SQLite показывает `sqlite3 --version`.

Тема 1. Первое знакомство: база данных в одном файле

Общие сведения

База данных — организованное хранилище данных, с которым работают по строгим правилам. В отличие от обычного текстового файла, база знает, из каких **таблиц** она состоит, какие в них **столбцы** и какого **типа** значения, не позволяет записать заведомо неверное, умеет быстро искать и одновременно обслуживать несколько программ. Таблица — набор **строк** (записей) одинакового устройства; столбец (поле) описывает одно свойство, например название товара или цену.

Большинство баз данных управляется языком **SQL** (Structured Query Language). Его команды делят на четыре группы: *запросы* (`SELECT`) читают данные; *изменение данных* (`INSERT`, `UPDATE`, `DELETE`); *описание структуры* (`CREATE`, `ALTER`, `DROP`); *управление транзакциями* (`BEGIN`, `COMMIT`, `ROLLBACK`). Язык *декларативный*: вы описываете, *какой результат* нужен, а не *как* его добыть; способ выбирает сама база.

SQLite — самая распространённая в мире встраиваемая база данных. Её особенность: нет отдельного сервера и настроек, вся база — *один файл*, который читает и пишет подключённая программа. Поэтому SQLite используют браузеры, мобильные приложения, программы мониторинга и множество служб, которые администратору приходится обслуживать: проверять, копировать, чинить, переносить. Для учебного курса это идеальный выбор: установка не требуется, базу можно скопировать и сломать, а язык SQL — тот же, что у PostgreSQL, MySQL и других (с небольшими отличиями).

Работают с SQLite программой командной строки `sqlite3`. Помимо SQL, она понимает собственные команды, которые начинаются с точки и не заканчиваются точкой с запятой: `.tables` (таблицы), `.schema` (описание структуры), `.headers on` (показывать названия столбцов), `.mode` (формат вывода), `.output файл` (писать вывод в файл), `.read файл` (выполнить SQL из файла), `.quit`. Команды SQL завершаются точкой с запятой; без неё `sqlite3` ждёт продолжения ввода.

Примеры.

```
sqlite3 shop.db                       # открыть базу (файл создаётся, если его
нет)
sqlite> .tables                       # какие есть таблицы
sqlite> .schema orders                # как создана таблица orders
sqlite> SELECT COUNT(*) FROM customers;
sqlite> .headers on
sqlite> .mode column
sqlite> SELECT * FROM customers LIMIT 3;
sqlite> .output otvety/4              # дальше вывод пойдёт в файл
sqlite> SELECT * FROM customers LIMIT 5;
sqlite> .output stdout                # вернуть вывод на экран
sqlite> .quit

sqlite3 shop.db "SELECT COUNT(*) FROM products;" > otvety/6      # один запрос из
оболочки
```

Служебная таблица `sqlite_master` описывает все объекты базы (таблицы, индексы, представления, триггеры): запрос `SELECT name FROM sqlite_master WHERE type = 'table'` — то же, что `.tables`, но результат можно сохранять и обрабатывать, как любой другой. Команда `PRAGMA table_info(таблица)` показывает столбцы: имя, тип, обязательность, значение по умолчанию, признак первичного ключа.

Что чаще всего идёт не так

Начинающие забывают точку с запятой в конце запроса и принимают приглашение `...>` за зависание: достаточно набрать `;`. Вторая типичная ошибка — записывать ответ в файл и не проверить его содержимое: после `.output файл` вывод на экран не показывается, пока не выполнено `.output stdout`.

Рабочий каталог и проверка

Каталог полигона: `~/sql-lab/01_pervye_zaprosy`. Проверка: `python3 proverka.py 1`.

Руководитель просит: «Посмотри, что за база данных у интернет-магазина, и скажи, что в ней лежит». Вы получили файл `shop.db` в формате SQLite. SQLite — это база данных в одном файле, без отдельного сервера: файл можно открыть программой `sqlite3`, скопировать, отправить по почте. В этой теме вы учитесь открывать базу, смотреть её устройство и выполнять первые запросы. Все задания выполняются в каталоге модуля: `sqlite3 shop.db`. Результат запроса записывается в файл ответа командой `.output otvety/N` (после запроса `.output stdout` возвращает вывод на экран) либо перенаправлением оболочки: `sqlite3 shop.db "запрос" > otvety/N`. В PowerShell 5.1 перенаправление `>` пишет файл в UTF-16: проверка это понимает, но лучше пользоваться `.output`.

Раздел 1. Команда sqlite3 — запуск консоли

1.1. Теоретические сведения

Назначение. Программа sqlite3 открывает файл базы данных и даёт командную строку для SQL-запросов. Если файла нет, он создаётся при первой записи.

Синтаксис:

```
sqlite3 файл.db
sqlite3 файл.db "SQL-запрос;"
sqlite3 --version
SELECT COUNT(*) FROM таблица;
SELECT выражение;
.output файл
.output stdout
```

Правила:

- вся база — один файл; копия файла — это копия базы, её делают командой оболочки (cp база.db копия.db, в PowerShell Copy-Item); чтобы работать с копией, откройте её: sqlite3 копия.db;
- первый запрос: SELECT COUNT(*) FROM таблица; печатает число строк таблицы (COUNT(*) — «посчитать строки»); SELECT выражение; без таблицы просто вычисляет выражение (арифметика +, -, *, /); подробно SELECT разбирается ниже;
- результат запроса можно сохранить в файл ответа: либо перенаправлением оболочки (> otvety/N), либо в консоли командой .output otvety/N, затем запрос, затем .output stdout (возврат вывода на экран); подробно — в записи про .output;
- с указанным запросом программа выполняет его и сразу завершается: так удобно записывать результат в файл оболочкой (> otvety/1);
- в приглашении sqlite> вводят SQL и служебные команды; приглашение ...> значит, что запрос не закончен — не хватает ;;
- выход из консоли — .quit или Ctrl+D.

Пример:

```
sqlite3 shop.db                          # открыть базу для работы
sqlite3 shop.db "SELECT 2 + 2;"          # один запрос из оболочки
sqlite3 --version                        # версия SQLite
sqlite3 shop.db "SELECT COUNT(*) FROM customers;"   # число строк в таблице
sqlite3 shop.db "SELECT 6 * 7;" > otvety/1          # результат — в файл ответа
```

1.2. Задания

Рабочий каталог: ~/sql-lab/01_pervye_zaprosy. Ответ каждого задания записывайте в файл otvety/N, где N — номер задания.

1. Сколько клиентов записано в таблице `customers`? Откройте `sqlite3 shop.db`, посчитайте строки запросом `SELECT COUNT(*) FROM таблица;` и запишите число в `otvety/1` командой `.output otvety/1` (затем `.output stdout`).
2. Откройте `shop.db` и выясните запросом с `COUNT(*)`, сколько поставщиков записано в таблице `suppliers`. Число запишите в `otvety/2`.
3. Сколько строк в таблице `products`? Посчитайте их запросом с `COUNT(*)` и запишите число в `otvety/3`.
4. Сколько строк в таблице `orders`? Посчитайте их запросом с `COUNT(*)` и запишите число в `otvety/4`.
5. Каждая строка таблицы `order_items` — одна позиция заказа. Сколько всего позиций? Посчитайте их запросом с `COUNT(*)`: выполните его одной командой из оболочки (`sqlite3 shop.db "запрос;"`) и запишите число в `otvety/5` перенаправлением `>`.
6. Сколько строк всего в двух таблицах: `customers` и `suppliers`? Посчитайте строки каждой таблицы запросом `SELECT COUNT(*) FROM …;` по отдельности, сложите два числа и запишите сумму в `otvety/6`.
7. На сколько товаров в таблице `products` больше, чем поставщиков в `suppliers`? Посчитайте строки каждой таблицы запросом `SELECT COUNT(*) FROM …;`, вычтите и запишите разность в `otvety/7`.
8. Сколько строк во всех пяти таблицах базы вместе (`suppliers`, `products`, `customers`, `orders`, `order_items`)? Посчитайте строки каждой таблицы запросом `SELECT COUNT(*) FROM …;`, сложите пять чисел и запишите сумму в `otvety/8`.
9. База — это один файл. Сделайте копию `shop.db` под именем `shop_copy.db` в каталоге темы (командой оболочки `cp` или `copy`), откройте **копию** (`sqlite3 shop_copy.db`) и запросом `SELECT COUNT(*) FROM customers;` узнайте число клиентов в ней; запишите его в `otvety/9`.
10. Запрос `SELECT` может вычислять выражение и без таблицы. В неделе 7 дней: вычислите запросом `SELECT выражение;`, сколько дней в 52 неделях (умножение — `*`), и запишите результат в `otvety/10`.

Раздел 2. Команда `.tables, .schema` — устройство базы

2.1. Теоретические сведения

Назначение. Служебные команды консоли показывают, какие в базе таблицы и как они созданы. Они начинаются с точки и **не** заканчиваются точкой с запятой.

Синтаксис:

```
.tables
.schema [таблица]
SELECT name FROM sqlite_master WHERE type = 'table';
PRAGMA table_info(таблица);
SELECT ... FROM pragma_table_info('таблица') [WHERE условие] [ORDER BY столбец];
```

Правила:

- `pk` равен 0 для обычного столбца и номеру столбца в составном первичном ключе (1, 2, ...) для ключевого;
- имя столбца `notnull` в запросе нужно брать в квадратные скобки: `[notnull]` (это ещё и служебное слово SQLite);
- `WHERE условие` оставляет только подходящие строки: сравнения `=`, `>`, текст берут в одинарные кавычки (`name = 'price'`), несколько условий соединяют словом `AND`; `ORDER BY столбец` упорядочивает строки результата (подробно — в следующих модулях);
- число столбцов таблицы можно получить так: `SELECT COUNT(*) FROM pragma_table_info('таблица');` — одна строка описания на каждый столбец.

Что показывает:

Элемент	Описание
`.tables`	список таблиц и представлений
`.schema`	операторы `CREATE`, которыми создана вся база (или одна таблица)
`sqlite_master`	служебная таблица со всеми объектами базы: тип, имя, текст `CREATE`; её можно запрашивать как обычную
`PRAGMA table_info(t)`	столбцы таблицы: номер `cid`, имя `name`, тип `type`, признак `notnull`, значение по умолчанию `dflt_value`, признак первичного ключа `pk`
`pragma_table_info('t')`	то же самое, но в виде таблицы, которую можно выбирать в `SELECT`, отбирать и считать

Пример:

```
.tables
.schema suppliers
SELECT type, name FROM sqlite_master;
```

```
SELECT name, type FROM pragma_table_info('suppliers');
SELECT name FROM pragma_table_info('suppliers') WHERE pk > 0;
SELECT type FROM pragma_table_info('suppliers') WHERE name = 'country';
SELECT COUNT(*) FROM pragma_table_info('suppliers');
```

2.2. Задания

Рабочий каталог: `~/sql-lab/01_pervye_zaprosy`. Ответ каждого задания записывайте в файл `otvety/N`, где `N` — номер задания.

11. Выясните, какие таблицы есть в базе `shop.db` (команда `.tables` или запрос к служебной таблице `sqlite_master`: имена — столбец `name`, отобрать таблицы — `WHERE type = 'table'`). Запишите имена таблиц в `otvety/11`, **по одному в строке**.
12. Запишите в `otvety/12` описание столбцов таблицы `products`: результат команды `PRAGMA table_info(products);` (после `PRAGMA …` ставится точка с запятой, как у любого SQL).
13. Сохраните в `otvety/13` команду `CREATE TABLE`, которой создана таблица `orders` (команда `.schema orders` или запрос к `sqlite_master`, столбец `sql`).
14. Запишите в `otvety/14` имена столбцов таблицы `customers`, по одному в строке, в том порядке, в котором они идут в таблице (столбец `name` результата `PRAGMA table_info(customers);` либо `SELECT name FROM pragma_table_info('customers');`).
15. Сколько столбцов в таблице `order_items`? Посчитайте строки описания: `SELECT COUNT(*) FROM pragma_table_info('таблица');`. Число запишите в `otvety/15`.
16. Первичный ключ таблицы `order_items` составной. Запишите в `otvety/16` имена входящих в него столбцов, по одному в строке, в порядке их номеров в ключе (отберите строки описания с `pk > 0` условием `WHERE`, упорядочите по `pk`).
17. В таблице `customers` не все столбцы обязательны. Запишите в `otvety/17` имена столбцов, которые **допускают** `NULL` (без `NOT NULL`), исключая первичный ключ `id`; по одному в строке, в порядке следования. Отбирайте по полю `[notnull]` и `pk` условием `WHERE … AND …`.
18. Какой тип данных объявлен у столбца `price` таблицы `products`? Запишите название типа в `otvety/18` (поле `type` описания столбца, отбор — `WHERE name = 'price'`).
19. Какое значение по умолчанию задано столбцу `vip` таблицы `customers`? Запишите его в `otvety/19` (поле `dflt_value` описания столбца или слово `DEFAULT` в `.schema customers`).
20. Сколько столбцов во всех таблицах базы вместе? Запишите число в `otvety/20`. Подсказка: для каждой таблицы посчитайте столбцы запросом `SELECT COUNT(*) FROM pragma_table_info('таблица');` и сложите пять чисел.

Раздел 3. Команда SELECT … FROM — первый запрос

3.1. Теоретические сведения

Назначение. Команда SELECT читает данные из таблицы: вы называете столбцы и таблицу, база возвращает строки. Данные при этом не меняются.

Синтаксис:

```
SELECT столбцы FROM таблица;
SELECT * FROM таблица LIMIT n [OFFSET m];
SELECT DISTINCT столбец FROM таблица;
SELECT COUNT(*) | COUNT(столбец) | COUNT(DISTINCT столбец) FROM таблица;
```

Правила:

- * — все столбцы таблицы; столбцы можно перечислять через запятую в любом порядке;
- LIMIT n ограничивает число строк, OFFSET m пропускает первые m строк;
- DISTINCT убирает повторяющиеся значения;
- COUNT(*) считает все строки, а COUNT(столбец) — только строки, где значение **не** NULL; COUNT(DISTINCT столбец) — число разных значений;
- запрос заканчивается точкой с запятой; регистр ключевых слов не важен, но их принято писать прописными; имена таблиц и столбцов в SQLite тоже не чувствительны к регистру.

Пример:

```
SELECT product_id, qty FROM order_items LIMIT 3;    -- два столбца, три строки
SELECT email FROM customers LIMIT 3 OFFSET 2;       -- пропустить две строки,
взять три
SELECT DISTINCT country FROM suppliers;             -- страны без повторов
SELECT COUNT(DISTINCT product_id) FROM order_items; -- сколько разных товаров
заказывали
```

3.2. Задания

Рабочий каталог: ~/sql-lab/01_pervye_zaprosy. Ответ каждого задания записывайте в файл otvety/N, где N — номер задания.

21. Выведите первые пять строк таблицы customers (все столбцы *, ограничение LIMIT) и запишите результат в otvety/21.
22. Выведите названия и цены (столбцы name, price) первых трёх товаров по порядку id (LIMIT 3) и запишите результат в otvety/22.
23. Выведите все столбцы таблицы suppliers, все строки, и запишите результат в otvety/23.
24. Выведите страну и название поставщика (именно в таком порядке: country, name) для всех поставщиков. Результат — в otvety/24.
25. Выведите из таблицы orders столбцы id, order_date и status для первых четырёх заказов (по порядку id, LIMIT). Результат — в otvety/25.

26. Выведите столбцы `id`, `name`, `stock` таблицы `products` для товаров с шестого по десятый включительно (по порядку `id`): пропустите пять строк и возьмите пять (`LIMIT … OFFSET …`). Результат — в `otvety/26`.
27. У части клиентов город не указан (`NULL`). Сколько клиентов, у которых город **указан**? Используйте `COUNT(столбец)`. Запишите число в `otvety/27`.
28. Сколько клиентов оставили адрес электронной почты (`email` не `NULL`)? Используйте `COUNT(столбец)`. Запишите число в `otvety/28`.
29. Сколько **разных** городов указано у клиентов? Пустые значения (`NULL`) городом не считаются. Используйте `COUNT(DISTINCT столбец)`. Запишите число в `otvety/29`.
30. Какие значения встречаются в столбце `status` таблицы `orders`? Выведите каждое значение один раз (`DISTINCT`) и запишите результат в `otvety/30`.

Раздел 4. Команда .headers, .mode — вид вывода

4.1. Теоретические сведения

Назначение. Определяют, как консоль печатает результат: с заголовками столбцов или без, в виде значений через черту, таблицы, CSV.

Синтаксис:

```
.headers on|off
.mode list|column|csv|box|line
```

Правила:

- настройки действуют до конца сеанса консоли;
- NULL печатается как пустое поле;
- для файлов ответов подходит формат по умолчанию; .headers on допустим: проверка пропускает строку заголовков.

Режимы вывода:

Элемент	Описание
list	значения через вертикальную черту, без выравнивания (по умолчанию)
column	выровненные столбцы; удобно читать глазами
csv	значения через запятую; годится для обмена с таблицами
box, line	таблица в рамке и по одному полю в строке

Пример:

```
.headers on
.mode column
SELECT id, name FROM customers LIMIT 3;
.mode csv
SELECT order_id, product_id, qty FROM order_items LIMIT 3;
```

4.2. Задания

Рабочий каталог: ~/sql-lab/01_pervye_zaprosy. Ответ каждого задания записывайте в файл otvety/N, где N — номер задания.

31. Подготовьте файл report.txt (в каталоге темы): первая строка — названия столбцов name и city, далее — имя и город первых пяти клиентов (по id). Используйте .headers on, .output report.txt и затем запрос.
32. Включите заголовки (.headers on) и запишите в otvety/32 столбцы id, name, country всех поставщиков: первая строка файла будет названиями столбцов.
33. Переключите вывод в формат CSV (.mode csv) и запишите в otvety/33 столбцы id, name, price первых пяти товаров (по id).

34. Включите заголовки и режим `.mode column`. Запишите в `otvety/34` столбцы `id`, `name`, `category`, `price` первых четырёх товаров (по `id`).
35. Подготовьте файл `suppliers.csv` (в каталоге темы) в формате CSV с первой строкой-заголовком: столбцы `id`, `name`, `country` всех поставщиков. Нужны `.headers on`, `.mode csv` и `.output suppliers.csv`.
36. Верните вывод в исходный вид: `.mode list` и `.headers off`. Запишите в `otvety/36` столбцы `id` и `status` первых шести заказов (по `id`).
37. В режиме `.mode csv` запишите в `otvety/37` столбцы `id`, `name`, `vip` первых шести клиентов (по `id`).
38. В режиме `.mode csv` запишите в `otvety/38` столбцы `id`, `city`, `email` первых восьми клиентов (по `id`). Обратите внимание, как выглядят отсутствующие значения (`NULL`): пустым полем между запятыми.
39. Включите заголовки и `.mode column`. Запишите в `otvety/39` столбцы `id`, `name`, `registered` шести клиентов, начиная с седьмого (`LIMIT 6 OFFSET 6`, порядок по `id`).
40. Подготовьте файл `statuses.csv` (в каталоге темы): первая строка — заголовок `status`, далее каждый встречающийся в таблице `orders` статус по одному разу. Нужны `.headers on`, `.mode csv`, `.output statuses.csv` и `DISTINCT`.

Раздел 5. Команда .output — вывод в файл

5.1. Теоретические сведения

Назначение. Перенаправляет результат запросов в файл вместо экрана. Так записывают ответы заданий в каталог otvety.

Синтаксис:

```
.output файл
.output stdout
.once файл
```

Правила:

- после .output файл вывод на экран не показывается, пока не выполнено .output stdout;
- файл при каждом .output файл создаётся заново, прежнее содержимое теряется; несколько запросов подряд пишутся в один файл одну за другой;
- .once файл перенаправляет вывод только следующего запроса, дальше вывод снова идёт на экран;
- после записи проверьте содержимое файла командой оболочки (cat, type);
- то же получится в оболочке: sqlite3 база.db "запрос;" > файл; в Windows PowerShell 5.1 так получается UTF-16, надёжнее .output.

Пример:

```
.output spisok.txt
SELECT name FROM products LIMIT 5;
.output stdout
.once chisla.txt
SELECT MAX(id) FROM orders;
```

5.2. Задания

Рабочий каталог: ~/sql-lab/01_pervye_zaprosy. Ответ каждого задания записывайте в файл otvety/N, где N — номер задания.

41. Запишите командой .output otvety/41 первые пять строк таблицы order_items (SELECT * с LIMIT). Не забудьте вернуть вывод на экран: .output stdout.
42. Запишите в otvety/42 столбцы name и vip первых восьми клиентов (по id, LIMIT 8) командой .output.
43. Сколько **разных** клиентов оформляли заказы (разные значения customer_id в orders, COUNT(DISTINCT …))? Запишите число в otvety/43 командой .output.
44. Из скольких разных стран работают поставщики? Посчитайте COUNT(DISTINCT …) и запишите число в otvety/44.
45. Запишите в otvety/45 столбцы name и stock первых шести товаров (по id, LIMIT 6) командой .output.

46. Запишите в `otvety/46` столбцы `id` и `order_date` пяти заказов, начиная с двадцать первого (`LIMIT 5 OFFSET 20`, порядок по `id`).
47. Не переключая `.output`, выполните два запроса подряд: сначала число клиентов, затем число товаров. В файле `otvety/47` должно оказаться две строки: сначала первое число, затем второе.
48. Запишите в текстовый файл `names.txt` (в каталоге темы, не в `otvety`) названия всех поставщиков, по одному в строке, без других столбцов. Используйте `.once names.txt` или `.output names.txt`.
49. Не заходя в консоль, одной командой оболочки запишите в `otvety/49` список категорий товаров (`category` таблицы `products`), каждая категория один раз.
50. Запишите в `otvety/50` столбцы `id`, `name`, `price` десяти товаров, начиная с одиннадцатого (`LIMIT 10 OFFSET 10`, порядок по `id`).

Вопросы для размышления

1. Чем база данных SQLite отличается от обычного текстового файла, в который можно записать те же строки?
2. Почему результат запроса полезно записывать в файл, а не только читать с экрана?

Тема 2. Выбор столбцов, выражения и сортировка

Общие сведения

Запрос SELECT читает данные. Его основная форма: SELECT *что выбрать* FROM *откуда*. Вместо списка столбцов можно написать * (все столбцы), но в рабочих запросах столбцы перечисляют явно: так запрос не «сломается» при добавлении столбца и не вытянет лишнее.

В списке столбцов можно использовать **выражения**: арифметику (price * 100), склейку строк (name || ' / ' || category), функции. Результату выражения дают имя ключевым словом AS: price * 100 AS price_kop. Такое имя (*псевдоним*) можно использовать в ORDER BY.

Порядок строк в результате **не определён**, пока не указан ORDER BY. Сортировка бывает по возрастанию (ASC, по умолчанию) и убыванию (DESC); ключей может быть несколько. Если у нескольких строк значения ключа совпадают, их взаимный порядок тоже неопределён, поэтому добавляют ещё один ключ, например id.

DISTINCT убирает повторяющиеся строки результата. LIMIT n оставляет первые n строк, OFFSET m пропускает первые m (вместе они дают постраничный вывод). Порядок записи предложений фиксирован: SELECT — FROM — WHERE — GROUP BY — HAVING — ORDER BY — LIMIT; а выполняются они в другом порядке (сначала FROM, затем отбор, потом выбор столбцов, сортировка и ограничение), поэтому, например, WHERE не видит псевдонимов из списка столбцов.

Примеры.

```
SELECT name, email FROM customers ORDER BY name;
SELECT name, price * 100 AS price_kop FROM products ORDER BY id;
SELECT DISTINCT category FROM products ORDER BY category;
SELECT name, price FROM products ORDER BY price DESC LIMIT 5;
SELECT category, name, price FROM products ORDER BY category, price DESC;
SELECT id, name FROM products ORDER BY id LIMIT 10 OFFSET 10;   -- страница 2 по
10 строк
SELECT name || ' / ' || category FROM products ORDER BY id LIMIT 5;
```

Оператор || склеивает строки. Если хотя бы одно из склеиваемых значений равно NULL, результат тоже NULL: об этом полезно помнить при работе с необязательными столбцами.

Что чаще всего идёт не так

Нельзя полагаться на «обычный» порядок строк: он зависит от внутреннего устройства таблицы и может измениться после изменения данных или создания индекса. Если порядок нужен, его всегда задают явно.

Рабочий каталог и проверка

Каталог полигона: ~/sql-lab/02_select. Проверка: python3 proverka.py 2.

Бухгалтерия и склад просят разные «срезы» одних и тех же данных: одним нужны только имена и адреса, другим — цены, пересчитанные в копейки, третьим — рейтинг самых дорогих товаров. Все эти просьбы решает команда `SELECT`: вы выбираете **столбцы**, придумываете **вычисляемые выражения**, задаёте **порядок** строк (`ORDER BY`), убираете повторы (`DISTINCT`) и ограничиваете число строк (`LIMIT`, `OFFSET`). Базы: `shop.db` (магазин) и `hr.db` (сотрудники и отделы). Результат каждого запроса записывайте в файл ответа (`.output otvety/N`). Если порядок строк указан в задании, он проверяется; при равных значениях ключа добавляйте последним ключом `id`, чтобы порядок был однозначным. Пустое значение (`NULL`) печатается пустым полем.

Раздел 6. Команда SELECT … FROM — выбор столбцов

6.1. Теоретические сведения

Назначение. Команда SELECT читает данные из таблицы и возвращает результат в виде новой таблицы. Сама база при этом не меняется.

Синтаксис:

```
SELECT столбец1, столбец2, ... FROM таблица;
SELECT * FROM таблица;
```

Правила:

- после SELECT перечисляют столбцы через запятую, после FROM — таблицу, из которой они берутся;
- * означает «все столбцы»; в рабочих запросах столбцы перечисляют явно, чтобы запрос не «сломался» после добавления столбца;
- в теме две базы: shop.db (магазин: customers, products, suppliers, orders, order_items) и hr.db (employees, departments); нужную открывают так: sqlite3 hr.db;
- если порядок строк в задании не оговорён, строки идут в порядке добавления (по id); как задать порядок явно, описано ниже;
- запрос завершается точкой с запятой; без неё sqlite3 ждёт продолжения ввода (приглашение ...>);
- ключевые слова можно писать в любом регистре, но принято прописными.

Пример:

```
SELECT name, registered FROM customers;     -- два столбца всех клиентов
SELECT * FROM departments;                  -- все столбцы таблицы
SELECT stock, name FROM products;            -- порядок столбцов задаёте вы
```

6.2. Задания

Рабочий каталог: ~/sql-lab/02_select. Ответ каждого задания записывайте в файл otvety/N, где N — номер задания.

1. Выведите имена и адреса электронной почты всех клиентов (name, email) в порядке id. У клиентов без адреса поле останется пустым. Результат — в otvety/1.
2. Выведите все столбцы всех строк таблицы suppliers (поставщики). Результат — в otvety/2.
3. В базе hr.db выведите название и город каждого отдела (таблица departments). Результат — в otvety/3.
4. Выведите цену, а затем название каждого товара (именно в таком порядке столбцов). Результат — в otvety/4.
5. Выведите id, название и остаток на складе (stock) всех товаров. Результат — в otvety/5.

6. Выведите имя, город и признак `vip` всех клиентов. У клиентов без города поле будет пустым. Результат — в `otvety/6`.
7. В `hr.db` выведите имя, должность и оклад (`salary`) всех сотрудников. Результат — в `otvety/7`.
8. Выведите номер заказа, номер товара и количество (`order_id`, `product_id`, `qty`) для всех позиций таблицы `order_items`. Результат — в `otvety/8`.
9. Выведите `id`, `customer_id` и `status` всех заказов. Результат — в `otvety/9`.
10. В `hr.db` выведите все столбцы таблицы `employees`. Обратите внимание на пустые значения в `dept_id` и `manager_id`. Результат — в `otvety/10`.

Раздел 7. Команда AS — выражения и псевдонимы

7.1. Теоретические сведения

Назначение. В списке столбцов можно писать не только имена, но и **выражения**: арифметику, склейку строк, вызовы функций. Ключевое слово AS даёт результату выражения имя (псевдоним).

Синтаксис:

```
SELECT выражение AS псевдоним FROM таблица;
SELECT ... FROM таблица ORDER BY ключ [DESC];
SELECT ... FROM таблица ORDER BY ключ LIMIT n;
```

Правила:

- арифметика: +, -, *, /, % (остаток от деления); деление целых чисел даёт целое;
- склейка строк — оператор ||; если хотя бы одно значение равно NULL, результат тоже NULL;
- чтобы строки шли в нужном порядке, в конец запроса дописывают ORDER BY ключ (по возрастанию) или ORDER BY ключ DESC (по убыванию), а LIMIT n оставляет первые n строк; подробно обе конструкции разобраны в следующих записях;
- псевдоним можно использовать в ORDER BY, но **нельзя** в WHERE (отбор выполняется раньше, чем вычисляется список столбцов);
- строковые литералы заключают в одинарные кавычки: ' / ';
- полезные функции: UPPER(s), LENGTH(s), SUBSTR(s, начало, длина), COALESCE(x, замена) — первое значение, не равное NULL.

Пример:

```
SELECT name, price / 100 AS rubles FROM products;           -- арифметика и
псевдоним
SELECT category || ': ' || name FROM products;             -- склейка строк
SELECT name, COALESCE(email, 'нет') FROM customers;        -- функция: подстановка
вместо NULL
SELECT name, LENGTH(name) FROM customers;                  -- функция: длина
строки
SELECT name, UPPER(name) FROM suppliers ORDER BY id;       -- порядок строк: ORDER
BY
SELECT name, price * 2 AS dbl FROM products ORDER BY dbl DESC, id LIMIT 3;   -- по
псевдониму, три строки
```

7.2. Задания

Рабочий каталог: ~/sql-lab/02_select. Ответ каждого задания записывайте в файл otvety/N, где N — номер задания.

11. Выведите названия товаров и их цену в копейках (цена, умноженная на 100) под именем price_kop; порядок — ORDER BY id. Результат — в otvety/11.

12. Выведите название и цену с НДС 20% под именем `price_vat`, вычисленную в целых числах как `price * 120 / 100`; порядок — `ORDER BY id`. Результат — в `otvety/12`.
13. В `hr.db` выведите имя сотрудника и годовую зарплату (`salary * 12`) под именем `year_salary`; порядок — `ORDER BY id`. Результат — в `otvety/13`.
14. Для каждой позиции заказа выведите `order_id`, `product_id` и сумму `qty * price` под именем `total`; порядок — `ORDER BY order_id, product_id`. Результат — в `otvety/14`.
15. Выведите для первых пяти товаров по `id` (`ORDER BY id LIMIT 5`) одну строку вида `название / категория` (оператор конкатенации `||` и литерал `' / '`). Результат — в `otvety/15`.
16. Для клиентов выведите `id` и остаток от деления `id` на 3 под именем `grp`; порядок — `ORDER BY id`. Результат — в `otvety/16`.
17. Выведите название поставщика заглавными буквами (`UPPER`) под именем `up` и длину названия (`LENGTH`) под именем `len`; порядок — `ORDER BY id`. Результат — в `otvety/17`.
18. Выведите `id`, имя и город клиента; если город не задан, вместо пустого значения выведите `не указан` (функция `COALESCE`). Столбец назовите `city`; порядок — `ORDER BY id`. Результат — в `otvety/18`.
19. Выведите `id` и строку `имя <почта>` (имя, пробел, угловые скобки с адресом, всё через `||`) под именем `contact`; порядок — `ORDER BY id`. У клиентов без почты результат будет пустым. Результат — в `otvety/19`.
20. Для каждого товара вычислите стоимость остатка на складе (`price * stock`) под именем `value`. Выведите название и `value` пяти товаров с наибольшей стоимостью остатка (`ORDER BY value DESC, id` и `LIMIT 5`). Результат — в `otvety/20`.

Раздел 8. Команда ORDER BY — сортировка

8.1. Теоретические сведения

Назначение. Задаёт порядок строк результата. Без ORDER BY порядок **не определён**, даже если «в прошлый раз» строки шли правильно.

Синтаксис:

```
SELECT ... FROM таблица ORDER BY ключ1 [ASC | DESC], ключ2 [ASC | DESC], ...;
```

Правила:

- ASC — по возрастанию (по умолчанию), DESC — по убыванию;
- ключей может быть несколько: строки с равным первым ключом упорядочиваются по второму и так далее;
- для однозначного порядка добавляйте последним ключом уникальный столбец, например id;
- NULL в SQLite считается меньше любого значения: при ASC идёт первым.

Что можно указать в качестве ключа:

Элемент	Описание
имя столбца	обычная сортировка по столбцу
псевдоним	имя, заданное через AS в списке столбцов
номер	порядковый номер столбца в списке (ORDER BY 2); удобно, но менее читаемо
выражение	например ORDER BY price * stock DESC

Пример:

```
SELECT name, salary FROM employees ORDER BY salary DESC;
SELECT dept_id, name, salary FROM employees ORDER BY dept_id, salary DESC;
SELECT name, salary / 12 AS month_pay FROM employees ORDER BY month_pay DESC, id;
SELECT name, stock FROM products ORDER BY 2 DESC, 1;     -- по номерам столбцов
```

8.2. Задания

Рабочий каталог: ~/sql-lab/02_select. Ответ каждого задания записывайте в файл otvety/N, где N — номер задания.

21. Выведите названия всех товаров в алфавитном порядке (ORDER BY name). Обратите внимание, как строки сравниваются «по буквам»: …-10 идёт раньше …-2. Результат — в otvety/21.
22. Выведите название и цену всех товаров, от самого дорогого к самому дешёвому (ORDER BY price DESC; при равной цене — по id). Результат — в otvety/22.

23. Выведите имя и дату регистрации (`registered`) клиентов, начиная с самых недавно зарегистрированных (`DESC`); при одинаковой дате — по возрастанию `id`. Результат — в `otvety/23`.
24. В `hr.db` выведите имя и оклад сотрудников, начиная с наименьшего оклада (по возрастанию); при равенстве — по `id`. Результат — в `otvety/24`.
25. Выведите категорию, название и цену всех товаров, упорядочив сначала по категории (по алфавиту), а внутри категории — по убыванию цены (`DESC`). Результат — в `otvety/25`.
26. Выведите название и остаток на складе (`stock`) товаров, упорядочив по остатку по убыванию, а при равенстве — по названию. В `ORDER BY` используйте номера столбцов. Результат — в `otvety/26`.
27. Выведите `id` и название товаров по убыванию стоимости остатка `price * stock`, не создавая для неё столбца в результате (выражение — прямо в `ORDER BY`); при равенстве — по `id`. Результат — в `otvety/27`.
28. Выведите `id`, имя и город клиентов, упорядочив по городу по алфавиту, при равенстве — по `id`. Клиенты без города (`NULL`) окажутся первыми. Результат — в `otvety/28`.
29. В `hr.db` выведите `id`, имя, `dept_id` и оклад всех сотрудников: по возрастанию `dept_id` (сотрудники без отдела — первыми), внутри отдела — по убыванию оклада, затем по `id`. Результат — в `otvety/29`.
30. Выведите `order_id`, `product_id` и сумму `qty * price` (псевдоним `total`) пяти самых крупных позиций заказов (`ORDER BY total DESC` и `LIMIT 5`); при равенстве — по `order_id`, затем по `product_id`. Результат — в `otvety/30`.

Раздел 9. Команда DISTINCT — без повторов

9.1. Теоретические сведения

Назначение. Убирает из результата повторяющиеся строки: остаются только различные.

Синтаксис:

```
SELECT DISTINCT столбец1, столбец2, ... FROM таблица;
```

Правила:

- DISTINCT относится ко **всей строке** результата, а не к первому столбцу: для списка `a, b` различными считаются пары `(a, b)`;
- все NULL считаются одинаковыми, поэтому в результате будет не более одной строки с NULL;
- на больших таблицах DISTINCT требует сортировки или хеширования — не добавляйте его «на всякий случай».

Пример:

```
SELECT DISTINCT dept_id FROM employees ORDER BY dept_id;
SELECT DISTINCT vip, city FROM customers ORDER BY vip, city;
```

9.2. Задания

Рабочий каталог: `~/sql-lab/02_select`. Ответ каждого задания записывайте в файл `otvety/N`, где `N` — номер задания.

31. Выведите без повторов (DISTINCT) все категории товаров (category) в алфавитном порядке, по одной в строке. Результат — в otvety/31.
32. В hr.db выведите без повторов (DISTINCT) все должности сотрудников (position) по алфавиту. Результат — в otvety/32.
33. Выведите без повторов (DISTINCT) все статусы заказов (status) по алфавиту. Результат — в otvety/33.
34. Выведите без повторов (DISTINCT) страны поставщиков (country) по алфавиту. Результат — в otvety/34.
35. Выведите без повторов (DISTINCT) пары supplier_id, category из таблицы products, упорядочив по supplier_id, затем по category. Результат — в otvety/35.
36. Выведите без повторов (DISTINCT) пары city, vip из таблицы customers, упорядочив по городу, затем по vip. Пустой город (NULL) тоже образует пару. Результат — в otvety/36.
37. Выведите без повторов (DISTINCT) номера клиентов (customer_id), сделавших хотя бы один заказ, по возрастанию. Результат — в otvety/37.
38. Выведите без повторов (DISTINCT) годы оформления заказов: первые четыре символа даты, SUBSTR(order_date, 1, 4), под именем year; по возрастанию. Результат — в otvety/38.

39. Выведите без повторов (`DISTINCT`) номера товаров (`product_id`), которые встречаются в позициях заказов, по возрастанию. Результат — в `otvety/39`.
40. Выведите три наибольших различных значения остатка (`stock`) среди товаров, по убыванию (`DISTINCT`, `ORDER BY … DESC`, `LIMIT 3`). Результат — в `otvety/40`.

Раздел 10. Команда LIMIT, OFFSET — часть результата

10.1. Теоретические сведения

Назначение. LIMIT оставляет в результате только первые строки, OFFSET пропускает указанное число строк с начала. Вместе они дают постраничный вывод.

Синтаксис:

```
SELECT ... FROM таблица ORDER BY ключ LIMIT n [OFFSET m];
```

Правила:

- LIMIT n — взять не более n строк; OFFSET m — предварительно пропустить m строк;
- LIMIT без ORDER BY возвращает «какие-то» строки, поэтому на практике их используют только вместе;
- страница номер p по k строк: LIMIT k OFFSET (p-1)*k;
- ограничение выполняется последним, после сортировки;
- в SQLite OFFSET без ограничения записывают как LIMIT -1 OFFSET m.

Пример:

```
SELECT name, hired FROM employees ORDER BY hired DESC, id LIMIT 4;     -- четыре
последних принятых
SELECT id, name FROM suppliers ORDER BY id LIMIT 2 OFFSET 2;          --
пропустить две строки, взять две
SELECT id, name FROM customers ORDER BY id LIMIT -1 OFFSET 35;        -- всё,
кроме первых 35
```

10.2. Задания

Рабочий каталог: ~/sql-lab/02_select. Ответ каждого задания записывайте в файл otvety/N, где N — номер задания.

41. Выведите название и цену пяти самых дорогих товаров, от самого дорогого к самому дешёвому (ORDER BY price DESC и LIMIT 5). Результат — в otvety/41.
42. Выведите название и цену трёх самых дешёвых товаров, начиная с самого дешёвого (LIMIT 3; при равной цене — по id). Результат — в otvety/42.
43. Выведите id и имя первых шести клиентов по порядку id (LIMIT 6). Результат — в otvety/43.
44. Выведите id и название товаров с 11-го по 20-й по порядку id (LIMIT 10 OFFSET 10: пропустите первые десять, возьмите десять). Результат — в otvety/44.
45. Клиентов выводят по 7 на страницу. Выведите id и имя клиентов третьей страницы (по порядку id; OFFSET считайте по правилу «страница p по k строк»). Результат — в otvety/45.
46. Найдите цену самого дорогого товара без агрегатных функций: отсортируйте и возьмите одну строку. Запишите цену в otvety/46.

47. Выведите `id` и название товаров, пропустив первые 25 по порядку `id` и взяв все остальные (в SQLite для этого пишут `LIMIT -1 OFFSET ...`). Результат — в `otvety/47`.
48. В `hr.db` выведите имя и оклад сотрудника с вторым по величине окладом (`ORDER BY salary DESC`, `LIMIT 1 OFFSET 1`). Результат — в `otvety/48`.
49. Выведите название и цену товаров, занявших места с 6-го по 10-е по убыванию цены (`ORDER BY price DESC LIMIT … OFFSET …`). Результат — в `otvety/49`.
50. Выведите три последние различные даты оформления заказов (`order_date`), от самой поздней (`DISTINCT`, `ORDER BY … DESC`, `LIMIT 3`). Результат — в `otvety/50`.

Вопросы для размышления

1. Почему без `ORDER BY` нельзя рассчитывать на определённый порядок строк, даже если «в прошлый раз всё было правильно»?
2. Чем `DISTINCT` отличается от группировки, и когда без него не обойтись?

Тема 3. Условия отбора: WHERE

Общие сведения

Предложение WHERE отбирает строки, для которых условие истинно. В условиях применяют операторы сравнения (=, <> или !=, <, <=, >, >=), логические связки AND, OR, NOT, проверку попадания в список (IN (...)), в диапазон (BETWEEN a AND b, границы включаются) и поиск по шаблону (LIKE). В шаблоне LIKE символ % заменяет любое число символов, _ — ровно один. Для латинских букв LIKE не различает регистр, для кириллицы различает; более строгий поиск даёт GLOB (звёздочки и вопросы, регистр учитывается).

Особое место занимает **NULL** — «значение неизвестно или отсутствует». Это не число и не пустая строка. Любое сравнение с NULL даёт не «ложь», а «неизвестно», поэтому email = NULL никогда не выберет ни одной строки. Для проверки служат IS NULL и IS NOT NULL. Из-за логики с тремя значениями (истина, ложь, неизвестно) строка попадает в результат, только если условие *точно* истинно.

Приоритет операций: NOT сильнее AND, а AND сильнее OR. Выражение a OR b AND c означает a OR (b AND c); чтобы не ошибиться, скобки ставят всегда, когда смешиваются AND и OR.

Даты в SQLite нет специального типа: их хранят текстом в формате ГГГГ-ММ-ДД (или ГГГГ-ММ-ДД ЧЧ:ММ:СС), числом суток или числом секунд. Текстовый формат ГГГГ-ММ-ДД удобен тем, что при сравнении строк даты упорядочиваются верно. Формат ДД.ММ.ГГГГ этого свойства лишён: его сортировка и сравнение дают неверные результаты.

Примеры.

```
SELECT id, name, price FROM products WHERE price > 5000 ORDER BY id;
SELECT name FROM customers WHERE city IN ('Москва', 'Казань') ORDER BY name;
SELECT COUNT(*) FROM products WHERE price BETWEEN 1000 AND 3000;
SELECT name FROM products WHERE name LIKE 'Книг%';
SELECT name FROM customers WHERE email IS NULL;              -- не "= NULL"!
SELECT id FROM products WHERE stock = 0 OR price > 9000;
SELECT COUNT(*) FROM orders WHERE order_date >= '2026-01-01';
```

Что чаще всего идёт не так

Классическая ошибка: WHERE city <> 'Москва' не выберет клиентов, у которых город не указан (NULL), хотя по смыслу они «не из Москвы». Если такие строки нужны, условие дополняют: ... OR city IS NULL.

Рабочий каталог и проверка

Каталог полигона: ~/sql-lab/03_where. Проверка: python3 proverka.py 3.

Склад просит не «всё подряд», а только то, что подходит под условие: дорогие товары, книги дешевле тысячи, клиенты без электронной почты, заказы текущего года. Условия записываются в предложении WHERE. В этой теме вы освоите операторы сравнения, логику AND, OR, NOT, проверку попадания в список (IN) и диапазон (BETWEEN), поиск по шаблону (LIKE) и особенности пустого значения

`NULL`, которое **не равно ничему**, даже самому себе. Даты в базе хранятся текстом в формате ГГГГ-ММ-ДД, поэтому сравниваются как строки. Каждой конструкции посвящено ровно десять заданий.

Раздел 11. Команда WHERE — отбор строк и сравнения

11.1. Теоретические сведения

Назначение. Предложение WHERE оставляет в результате только те строки таблицы, для которых условие истинно; остальные отбрасываются до формирования списка столбцов. Условие строится из операторов сравнения, которые работают с числами, текстом и датами (даты в учебных базах хранятся текстом ГГГГ-ММ-ДД).

Синтаксис:

```
SELECT столбцы FROM таблица WHERE выражение оператор выражение [ORDER BY ...]
[LIMIT ...];
```

Правила:

- WHERE пишется после FROM и перед ORDER BY и LIMIT;
- строка попадает в результат, только если условие **истинно** (а не «ложно» и не «неизвестно»);
- в условии нельзя использовать псевдонимы из списка SELECT: отбор выполняется раньше, чем вычисляются столбцы результата;
- текстовые значения заключают в одинарные кавычки: 'Книги'; сравнение строк через = учитывает регистр; число пишут без кавычек;
- равенство — **один** знак =; «не равно» — <> или != (одно и то же);
- строки с NULL не попадают ни в city = 'Москва', ни в city <> 'Москва';
- даты ГГГГ-ММ-ДД сравниваются как строки и дают верный хронологический порядок: registered < '2024-01-01';
- слева и справа от оператора могут стоять выражения: WHERE qty * price > 1000;
- WHERE работает и с COUNT(*) (тема 1): считаются только отобранные строки.

Операторы сравнения:

Элемент	Описание
=	равно
<> или !=	не равно
<, <=	меньше, меньше или равно
>, >=	больше, больше или равно

Пример:

```
SELECT name, price FROM products WHERE price <= 500;                  -- не дороже
500
SELECT name, city FROM customers WHERE city <> 'Тверь';           -- строки с city
NULL сюда не попадут!
SELECT id, order_date FROM orders WHERE order_date < '2025-06-01'; -- дата как
строка
SELECT name FROM products WHERE stock >= 100 ORDER BY price;     -- условие стоит
до сортировки
```

```
SELECT COUNT(*) FROM orders WHERE status = 'paid';                -- сколько
оплаченных заказов
SELECT name FROM products WHERE stock > 0 ORDER BY price DESC, id LIMIT 3;   --
WHERE, ORDER BY, LIMIT — в таком порядке
SELECT order_id, qty * price AS total FROM order_items WHERE qty * price > 20000;
-- выражение в условии
```

11.2. Задания

Рабочий каталог: `~/sql-lab/03_where`. Ответ каждого задания записывайте в файл `otvety/N`, где `N` — номер задания.

1. Выведите `id`, название и цену товаров дороже 5000 (оператор `>`), по возрастанию `id`. Результат — в `otvety/1`.
2. Выведите `id` и название товаров, остаток которых равен нулю (`stock = 0`), по возрастанию `id`. Результат — в `otvety/2`.
3. Сколько заказов имеют статус `paid` (`WHERE status = ...`, считайте функцией `COUNT(*)` из темы 1)? Запишите число в `otvety/3`.
4. Выведите `id`, имя и город клиентов, чей город не равен `Москва` (`<>`), по возрастанию `id`. Клиенты без города в результат не попадут. Результат — в `otvety/4`.
5. Выведите `id`, название и цену товаров, которые стоят не дороже 3000 (`<=`), по возрастанию `id`. Результат — в `otvety/5`.
6. Выведите `id`, название и остаток товаров, которых на складе меньше 10 штук (оператор `<`), по возрастанию `id`. Результат — в `otvety/6`.
7. Выведите `id`, имя и дату регистрации клиентов, зарегистрированных раньше 1 января 2024 года (дата-строка и оператор `<`), по возрастанию `id`. Результат — в `otvety/7`.
8. Сколько заказов оформлено в 2026 году (дата не раньше `2026-01-01`, оператор `>=`; считайте `COUNT(*)`)? Запишите число в `otvety/8`.
9. Сколько заказов имеют статус, отличный от `cancelled` (оператор `<>`)? Запишите число в `otvety/9`.
10. Выведите название и цену трёх самых дорогих товаров, которые есть на складе (`stock > 0`): сначала по убыванию цены, затем по `id`. Сочетайте `WHERE`, `ORDER BY` и `LIMIT` (в таком порядке). Результат — в `otvety/10`.

Раздел 12. Команда AND, OR, NOT — логические операции

12.1. Теоретические сведения

Назначение. Логические операторы объединяют несколько условий в одно. AND требует выполнения обоих условий, OR — хотя бы одного, NOT меняет истину на ложь.

Синтаксис:

```
условие1 AND условие2
условие1 OR условие2
NOT условие
```

Правила:

- условий можно соединять сколько угодно: a AND b AND c;
- приоритет: сначала NOT, затем AND, затем OR; запись a OR b AND c означает a OR (b AND c);
- как только в условии встречаются и AND, и OR, ставьте скобки — это защищает от ошибки и делает запрос читаемым;
- если одно из условий «неизвестно» (из-за NULL), то AND с ним даёт «неизвестно» или «ложь», а OR — «истину», если второе условие истинно;
- NOT (x = 1) для NULL тоже «неизвестно»: отрицание не превращает «неизвестно» в «истину».

Таблица истинности (И — истина, Л — ложь):

Элемент	Описание
И AND И	истина
И AND Л или Л AND Л	ложь
Л OR Л	ложь
И OR Л или И OR И	истина
NOT И, NOT Л	ложь, истина

Пример:

```
SELECT name, price FROM products
WHERE category = 'Канцтовары' AND stock > 10;                    -- оба условия
сразу

SELECT name, category, price FROM products
WHERE (category = 'Продукты' OR category = 'Мебель')
  AND price < 1500;                                              -- скобки
  обязательны

SELECT id, status FROM orders WHERE NOT (status = 'new');        -- отрицание
условия

SELECT COUNT(*) FROM orders
WHERE customer_id < 6 AND status <> 'cancelled' AND order_date >= '2026-01-01';
-- три условия через AND
```

12.2. Задания

Рабочий каталог: `~/sql-lab/03_where`. Ответ каждого задания записывайте в файл `otvety/N`, где `N` — номер задания.

11. Выведите название и цену товаров категории `Книги`, цена которых меньше 8000 (оба условия сразу, `AND`), по возрастанию `id`. Результат — в `otvety/11`.
12. Выведите `id` товаров, которых нет на складе (`stock = 0`) **или** (оператор `OR`) которые дороже 9000, по возрастанию. Результат — в `otvety/12`.
13. Выведите `id`, название и остаток товаров категории `Мебель`, остаток которых меньше 50 штук (два условия через `AND`), по возрастанию `id`. Результат — в `otvety/13`.
14. Сколько заказов имеют статус `shipped` и оформлены не раньше `2026-01-01` (два условия через `AND`, считайте `COUNT(*)`)? Запишите число в `otvety/14`.
15. Выведите `id`, название и категорию товаров из категорий `Книги` или `Продукты` (через `OR`), по возрастанию `id`. Результат — в `otvety/15`.
16. Сколько товаров относится не к категории `Электроника`? Решите через `NOT`. Запишите число в `otvety/16`.
17. Выведите `id`, название, категорию и цену товаров из категорий `Канцтовары` или `Мебель`, которые стоят меньше 5000. Категории объедините через `OR` в скобках и свяжите с ценой через `AND`: без скобок условие поймут неверно. Порядок — по `id`. Результат — в `otvety/17`.
18. Выведите `id`, название, категорию и цену товаров, которые стоят больше 8000 **или** относятся к категории `Книги` и стоят меньше 2000 (используйте `OR` и `AND`). Порядок — по `id`. Результат — в `otvety/18`.
19. Сколько заказов имеют статус `new` или `paid` (`OR` в скобках) и оформлены раньше `2026-01-01` (`AND`)? Запишите число в `otvety/19`.
20. Выведите `id`, `customer_id` и дату заказов клиентов с `customer_id` меньше 6, которые не отменены (`<>`) и оформлены не раньше `2026-01-01`: три условия через `AND`. Порядок — по `id`. Результат — в `otvety/20`.

Раздел 13. Команда IN — проверка по списку

13.1. Теоретические сведения

Назначение. IN проверяет, входит ли значение в заданный список; NOT IN — что не входит.

Синтаксис:

```
выражение [NOT] IN (значение1, значение2, ...)
```

Правила:

- x IN (a, b, c) равносильно x = a OR x = b OR x = c, но короче и надёжнее;
- значения списка разделяют запятыми; текст — в одинарных кавычках;
- NOT IN не возвращает строки, где значение NULL: city NOT IN ('Омск') не найдёт клиентов без города;
- если в списке NOT IN есть NULL, условие никогда не бывает истинным — не включайте NULL в список.

Пример:

```
SELECT name, category FROM products WHERE category IN ('Книги', 'Мебель');
SELECT name, category FROM products WHERE category NOT IN ('Книги', 'Мебель');
SELECT id, name FROM customers WHERE id IN (2, 4, 6);              -- числа — без кавычек
SELECT COUNT(*) FROM orders
WHERE status IN ('paid', 'shipped') AND order_date >= '2026-01-01';   -- IN вместе с AND
```

13.2. Задания

Рабочий каталог: ~/sql-lab/03_where. Ответ каждого задания записывайте в файл otvety/N, где N — номер задания.

21. Выведите имена клиентов из Москвы или Казани (оператор IN), по алфавиту. Результат — в otvety/21.
22. Сколько заказов имеют статус new или paid? Используйте IN. Запишите число в otvety/22.
23. Выведите id и название товаров, категория которых не Книги и не Продукты (NOT IN), по возрастанию id. Результат — в otvety/23.
24. Выведите id, название и supplier_id товаров поставщиков с номерами 1 и 3 (IN со списком чисел), по возрастанию id. Результат — в otvety/24.
25. Выведите id заказов клиентов с номерами 1, 2 и 3 (IN), по возрастанию. Результат — в otvety/25.
26. Выведите id, имя и город клиентов, живущих не в Москве, Казани и Самаре (NOT IN), по возрастанию id. Клиенты без города в результат не попадут. Результат — в otvety/26.

27. Сколько товаров имеют остаток на складе ровно 0, 3 или 5 штук (оператор `IN`)? Запишите число в `otvety/27`.
28. Выведите `id` и имя клиентов с номерами 3, 7, 11 и 15 (`IN`), по алфавиту имён. Результат — в `otvety/28`.
29. Сколько заказов имеют статус `paid` или `shipped` (`IN`) и оформлены не раньше `2026-01-01` (`AND`)? Запишите число в `otvety/29`.
30. Выведите название и цену пяти самых дорогих товаров из категорий `Электроника` и `Мебель` (`IN`): по убыванию цены, затем по `id`. Результат — в `otvety/30`.

Раздел 14. Команда BETWEEN — диапазон

14.1. Теоретические сведения

Назначение. BETWEEN проверяет, лежит ли значение в диапазоне между двумя границами; NOT BETWEEN — что лежит вне диапазона.

Синтаксис:

```
выражение [NOT] BETWEEN нижняя AND верхняя
```

Правила:

- BETWEEN включает **обе** границы: x BETWEEN 10 AND 20 то же, что x >= 10 AND x <= 20;
- нижняя граница должна стоять первой: BETWEEN 20 AND 10 не вернёт ни одной строки;
- AND внутри BETWEEN — часть самого оператора, а не логическая связка;
- работает и с датами-строками: registered BETWEEN '2024-01-01' AND '2024-12-31' — весь 2024 год;
- строки с NULL не попадают ни в BETWEEN, ни в NOT BETWEEN.

Пример:

```
SELECT name, price FROM products WHERE price BETWEEN 100 AND 500;
SELECT name, price FROM products WHERE price NOT BETWEEN 100 AND 500;
SELECT id, order_date FROM orders
WHERE order_date BETWEEN '2025-07-01' AND '2025-09-30';        -- третий квартал
2025 года
SELECT order_id, qty * price AS total FROM order_items
WHERE qty * price BETWEEN 10000 AND 20000;                     -- слева может
стоять выражение
SELECT id FROM orders
WHERE order_date BETWEEN '2026-01-01' AND '2026-06-30' AND status IN ('paid',
'shipped');   -- BETWEEN, AND и IN вместе
```

14.2. Задания

Рабочий каталог: ~/sql-lab/03_where. Ответ каждого задания записывайте в файл otvety/N, где N — номер задания.

31. Сколько товаров стоит от 1000 до 4000 включительно (оператор BETWEEN)? Запишите число в otvety/31.
32. Выведите id, название и остаток товаров, остаток которых от 3 до 12 штук включительно (BETWEEN), по возрастанию id. Результат — в otvety/32.
33. Сколько заказов оформлено в 2025 году (с 2025-01-01 по 2025-12-31, BETWEEN)? Запишите число в otvety/33.
34. Выведите id, имя и дату регистрации клиентов, зарегистрированных в 2024 году (BETWEEN для двух дат-строк), по возрастанию id. Результат — в otvety/34.

35. Выведите `id`, название и цену товаров, цена которых **не** лежит в диапазоне от 500 до 8000 (`NOT BETWEEN`), по возрастанию `id`. Результат — в `otvety/35`.
36. Сколько позиций заказов (таблица `order_items`) имеют количество `qty` от 2 до 3 включительно (`BETWEEN`)? Запишите число в `otvety/36`.
37. Выведите `id` и дату заказов, оформленных в первом квартале 2026 года (`BETWEEN` от `2026-01-01` до `2026-03-31`), по возрастанию `id`. Результат — в `otvety/37`.
38. Выведите `id`, название и цену товаров, которые стоят от 2000 до 8000 включительно (`BETWEEN`) и есть на складе (`stock > 0`, связка `AND`). Порядок — по `id`. Результат — в `otvety/38`.
39. Из `order_items` выведите `order_id`, `product_id` и сумму позиции `qty * price` (назовите столбец `total`) там, где сумма от 10000 до 20000 включительно (`BETWEEN` с выражением `qty * price`; псевдоним в `WHERE` недоступен). Порядок — по `order_id`, затем по `product_id`. Результат — в `otvety/39`.
40. Выведите `id` и `customer_id` заказов, оформленных с `2026-01-01` по `2026-06-30`, со статусом `paid` или `shipped` (`BETWEEN` по дате, `IN` по статусу, связка `AND`), по возрастанию `id`. Результат — в `otvety/40`.

Раздел 15. Команда LIKE — поиск по шаблону

15.1. Теоретические сведения

Назначение. LIKE сравнивает текст не с точным значением, а с шаблоном, в котором специальные символы заменяют части строки.

Синтаксис:

```
текстовый_столбец [NOT] LIKE 'шаблон'
```

Правила:

- в шаблоне % заменяет любое количество символов (в том числе ни одного), _ — ровно один символ;
- для латиницы LIKE **не** различает регистр, а для кириллицы различает: 'книг%' не найдёт 'Книги';
- шаблон без % и _ работает как обычное равенство;
- чтобы найти сам символ % или _, его экранируют: LIKE '50!%' ESCAPE '!';
- строки с NULL не подходят ни под LIKE, ни под NOT LIKE;
- GLOB — родственный оператор с регистрозависимым сравнением и знаками * и ? вместо % и _.

Элементы шаблона:

Элемент	Описание
%	любая последовательность символов, в том числе пустая
_	ровно один любой символ
'Эле%'	начинается на Эле
'%-1_'	содержит -1 и после него ещё ровно один символ в конце строки
'%@sever%'	содержит @sever в любом месте

Пример:

```
SELECT name FROM products WHERE name LIKE 'Эле%';              -- названия,
начинающиеся на «Эле»
SELECT name FROM products WHERE name LIKE '%-1_';             -- например,
«Мебе-13», «Прод-15»
SELECT name, email FROM customers WHERE email LIKE '%@sever%';
SELECT name FROM products WHERE name NOT LIKE '%-2%';
SELECT id, name FROM customers WHERE name LIKE 'Б%';          -- заглавная
кириллическая Б: регистр важен
SELECT id, email FROM customers WHERE email LIKE 'user1_@%'; -- user10@…, user11@…
и т. д.
SELECT id, name FROM suppliers WHERE name LIKE '%Торг%' OR name LIKE '%Опт%';   --
два шаблона через OR
```

15.2. Задания

Рабочий каталог: `~/sql-lab/03_where`. Ответ каждого задания записывайте в файл `otvety/N`, где `N` — номер задания.

41. Выведите названия товаров, которые начинаются на `Книг` (`LIKE 'Книг%'`), по возрастанию `id`. Результат — в `otvety/41`.
42. Выведите `id` и название товаров, название которых заканчивается на `-2` и ещё один символ (шаблон `'%-2_'`), по возрастанию `id`. Результат — в `otvety/42`.
43. Сколько клиентов имеют адрес электронной почты, содержащий `@sever` (`LIKE` с шаблоном `'%@sever%'`)? Запишите число в `otvety/43`.
44. Выведите `id` и имя клиентов, чьё имя (фамилия стоит первой) начинается на заглавную `Б` (`LIKE 'Б%'`; для кириллицы регистр важен), по возрастанию `id`. Результат — в `otvety/44`.
45. Выведите названия товаров, у которых после `Книг-` стоит ровно один символ (шаблон `'Книг-_'`), по возрастанию `id`. Результат — в `otvety/45`.
46. Сколько товаров имеют название, не содержащее `-1` (`NOT LIKE`)? Запишите число в `otvety/46`.
47. Сколько клиентов имеют адрес, содержащий `@SEVER` (заглавными буквами; `LIKE '%@SEVER%'`)? Для латиницы `LIKE` регистр не учитывает. Запишите число в `otvety/47`.
48. Выведите `id`, имя и город клиентов, название города которых оканчивается на `а` (шаблон `'%а'`), по возрастанию `id`. Результат — в `otvety/48`.
49. Выведите `id` и название поставщиков, в названии которых есть `Торг` или `Опт` (два условия `LIKE` через `OR`), по возрастанию `id`. Результат — в `otvety/49`.
50. Выведите `id`, имя и почту клиентов, у которых почта подходит под шаблон `'user1_@%'` и город известен. Порядок — по `id`. Результат — в `otvety/50`.

Раздел 16. Команда IS NULL — пустое значение

16.1. Теоретические сведения

Назначение. NULL означает «значение неизвестно или отсутствует». Это не ноль и не пустая строка, и сравнивать его обычными операторами нельзя: для проверки есть специальные формы IS NULL и IS NOT NULL.

Синтаксис:

```
выражение IS NULL
выражение IS NOT NULL
```

Правила:

- NULL = NULL даёт «неизвестно», а не «истину»; поэтому WHERE city = NULL не вернёт ни одной строки — это ошибка, о которой SQL не сообщает;
- условие city <> 'Москва' **не** вернёт строки с city IS NULL; если они нужны, пишут city <> 'Москва' OR city IS NULL;
- пустая строка '' и NULL — разные значения: email = '' не находит NULL;
- у столбцов с ограничением NOT NULL пустых значений нет, проверять их бессмысленно;
- IS NULL и IS NOT NULL комбинируются с AND, OR, NOT IN и скобками, как любые другие условия.

Пример:

```
SELECT name FROM customers WHERE city IS NULL;                        -- город
неизвестен
SELECT name, city FROM customers WHERE city IS NOT NULL;           -- город указан
SELECT name FROM customers WHERE city <> 'Тверь' OR city IS NULL;  -- «не из
Твери», включая неизвестных
SELECT name FROM customers WHERE city IS NULL OR email IS NULL;    -- не хватает
хотя бы одного поля
SELECT name FROM customers WHERE vip = 1 AND (city IS NOT NULL OR email IS NOT
NULL);
SELECT name FROM customers WHERE city NOT IN ('Москва', 'Казань') OR city IS NULL;
```

16.2. Задания

Рабочий каталог: ~/sql-lab/03_where. Ответ каждого задания записывайте в файл otvety/N, где N — номер задания.

51. Выведите имена клиентов, у которых адрес электронной почты не задан (NULL), по алфавиту. Результат — в otvety/51.
52. Сколько клиентов не указали город (IS NULL)? Запишите число в otvety/52.
53. Сколько клиентов имеют заданный адрес электронной почты (IS NOT NULL)? Запишите число в otvety/53.
54. Выведите id, имя и город клиентов, у которых город не Москва **или не указан** (<> и IS NULL, связка OR), по возрастанию id. Результат — в otvety/54.

55. Выведите `id` и имя клиентов, у которых не указан город **или** не указана почта (`IS NULL` для каждого поля, связка `OR`), по возрастанию `id`. Результат — в `otvety/55`.
56. Выведите `id` и имя клиентов, чей город не `Москва` и не `Казань`, включая клиентов без города (`NOT IN` и `IS NULL`, связка `OR`). Порядок — по `id`. Результат — в `otvety/56`.
57. Выведите `id`, имя и город клиентов, у которых город указан, а почта — нет (`IS NOT NULL` и `IS NULL`, связка `AND`). Порядок — по `id`. Результат — в `otvety/57`.
58. Выведите `id` и имя клиентов, не являющихся VIP (`vip = 0`) и не имеющих почты (`IS NULL`). Порядок — по `id`. Результат — в `otvety/58`.
59. Сколько клиентов имеют и город, и почту (оба поля не `NULL`)? Запишите число в `otvety/59`.
60. Выведите имена VIP-клиентов (`vip = 1`), у которых известен город или почта (хотя бы одно из полей не `NULL`). Порядок — по алфавиту. Результат — в `otvety/60`.

Вопросы для размышления

1. Почему `NULL = NULL` не истинно, и как это связано с проверкой «значение неизвестно»?
2. Какие ошибки возможны, если сравнивать даты, хранящиеся в тексте, не в формате ГГГГ-ММ-ДД?

Тема 4. Функции, выражения CASE и работа с NULL

Общие сведения

Данные в таблицах редко лежат в том виде, в котором нужны отчёту. Для преобразования служат **функции**. Текстовые: `LENGTH` (длина в символах), `SUBSTR(строка, начало, длина)`, `INSTR(строка, подстрока)` (позиция), `REPLACE`, `TRIM`, `UPPER`/`LOWER` (в SQLite они преобразуют только латинские буквы). Числовые: `ROUND(x, n)`, `ABS`, `MIN`/`MAX` от нескольких значений. Даты: `DATE`, `TIME`, `DATETIME`, `STRFTIME(формат, дата)`, `JULIANDAY`; модификаторы вроде `'+14 days'`, `'-1 month'`, `'start of month'` сдвигают дату.

Выражение `CASE` — условный выбор: `CASE WHEN условие THEN значение ... ELSE значение END`. Оно позволяет раскладывать значения по группам прямо в запросе. Функции `COALESCE(a, b, ...)` (первое значение, не равное `NULL`), `NULLIF(a, b)` (`NULL`, если значения равны) и `IFNULL` помогают работать с пустыми значениями.

SQLite отличается от многих баз **динамической типизацией**: у столбца есть «склонность» (*type affinity*) — INTEGER, REAL, TEXT, BLOB, NUMERIC, — но у каждого *значения* есть собственный тип, и в столбец INTEGER можно записать текст, если он не похож на число. Тип значения показывает функция `TYPEOF`. Это гибко, но означает, что проверку правильности данных нужно задавать ограничениями (`CHECK`), иначе мусор попадёт в базу без возражений. Деление двух целых чисел даёт целое (`7 / 2 = 3`); чтобы получить дробный результат, одно из чисел делают вещественным (`7 / 2.0`).

Примеры.

```
SELECT name, LENGTH(name) AS len FROM customers ORDER BY len DESC, id LIMIT 5;
SELECT id, SUBSTR(email, 1, INSTR(email, '@') - 1) FROM customers WHERE email IS
NOT NULL LIMIT 5;
SELECT id, STRFTIME('%Y', order_date) FROM orders LIMIT 5;
SELECT name, CASE WHEN price < 1000 THEN 'дёшево' WHEN price < 5000 THEN 'средне'
ELSE 'дорого' END
FROM products LIMIT 10;
SELECT id, COALESCE(email, 'нет') FROM customers LIMIT 5;
SELECT TYPEOF(1), TYPEOF(1.5), TYPEOF('a'), TYPEOF(NULL);
SELECT DATE('2026-10-11', '+14 days'), JULIANDAY('2026-10-11') -
JULIANDAY('2026-01-01');
```

Что чаще всего идёт не так

Неверный формат даты не вызывает ошибку: `DATE('11.10.2026')` просто вернёт `NULL`. Если результат функции даты внезапно пуст, прежде всего проверьте формат исходного значения.

Рабочий каталог и проверка

Каталог полигона: `~/sql-lab/04_funktsii`. Проверка: `python3 proverka.py 4`.

Аналитик просит подготовить данные для отчёта, но «сырые» значения неудобны: логин надо отделить от почтового адреса, дату заказа пересчитать в год, цену разбить на группы «дёшево / средне / дорого», пустую почту заменить словом «нет». Для этого в SQL есть **функции** (`LENGTH`, `SUBSTR`, `INSTR`, `REPLACE`, `ROUND`, `STRFTIME`, `DATE`), выражение `CASE` и средства для работы с пустыми значениями (`COALESCE`, `NULLIF`). SQLite хранит значения с **динамической типизацией**: тип есть у значения, а не только у столбца, и его показывает функция `TYPEOF`. Раздел посвящён семи группам средств, и по каждой дано ровно десять заданий — от прямого применения к ловушкам (`NULL`, регистр, целочисленное деление, формат дат). Во всех заданиях порядок строк задан, при равенстве ключа добавляется `id`.

Раздел 17. Команда LENGTH, SUBSTR, INSTR — разбор строки

17.1. Теоретические сведения

Назначение. Эти функции измеряют строку, находят в ней подстроку и вырезают нужную часть. Их используют в списке SELECT, в WHERE и в ORDER BY.

Синтаксис:

```
LENGTH(s)
SUBSTR(s, начало [, длина])
INSTR(s, подстрока)
```

Правила:

- позиции в строке нумеруются **с единицы**: первый символ имеет номер 1;
- INSTR возвращает 0, если подстрока не найдена; поэтому INSTR(s, '@') - 1 у строки без @ даст -1;
- если любой аргумент равен NULL, результат тоже NULL;
- LENGTH для текста считает символы, а не байты;
- вызовы можно вкладывать друг в друга: вычисление идёт изнутри наружу;
- SUBSTR(s, n, 1) берёт один символ с номера n; часть после символа x начинается с INSTR(s, x) + 1, часть до него имеет длину INSTR(s, x) - 1;
- функции можно использовать и в WHERE, и в ORDER BY; псевдоним из SELECT в WHERE недоступен, выражение повторяют.

Основные функции:

Элемент	Описание
LENGTH(s)	длина строки в символах
SUBSTR(s, начало, длина)	подстрока; без третьего аргумента — до конца строки; отрицательное начало отсчитывается с конца
INSTR(s, подстрока)	позиция первого вхождения подстроки или 0

Пример:

```
SELECT name, LENGTH(city) FROM customers LIMIT 3;                -- NULL, если
города нет
SELECT name, SUBSTR(name, 1, INSTR(name, ' ') - 1) AS surname FROM customers LIMIT
3;   -- слово до первого пробела
SELECT SUBSTR('Северный ветер', -5);                             -- последние пять
символов: ветер
SELECT INSTR('abc', 'z');                                        -- 0
SELECT email, SUBSTR(email, INSTR(email, '@') + 1) FROM customers WHERE email IS
NOT NULL LIMIT 3;   -- после @
SELECT name, SUBSTR(name, INSTR(name, ' ') + 1, 1) FROM customers LIMIT 3;   --
первая буква второго слова
SELECT id FROM customers WHERE LENGTH(name) > 15 ORDER BY id LIMIT 3;     --
функция в WHERE
```

17.2. Задания

Рабочий каталог: `~/sql-lab/04_funktsii`. Ответ каждого задания записывайте в файл `otvety/N`, где `N` — номер задания.

1. Выведите имя и длину имени (`LENGTH`) под именем `len` у пяти клиентов с самыми длинными именами; при равной длине — по возрастанию `id`. Результат — в `otvety/1`.
2. Для первых пяти клиентов по `id`, у которых задана почта, выведите `id` и длину адреса (`LENGTH`); отбор — `email IS NOT NULL`. Результат — в `otvety/2`.
3. Для первых пяти клиентов по `id`, у которых задана почта, выведите `id` и часть адреса до символа `@` (`SUBSTR` и `INSTR`). Результат — в `otvety/3`.
4. Для первых пяти клиентов по `id` с заданной почтой выведите `id` и часть адреса после символа `@` (`SUBSTR` без третьего аргумента и `INSTR`). Результат — в `otvety/4`.
5. Имя клиента записано как «Фамилия Имя». Для первых восьми клиентов по `id` выведите `id` и фамилию (слово до пробела; `SUBSTR` и `INSTR(name, ' ')`). Результат — в `otvety/5`.
6. Для первых восьми клиентов по `id` выведите `id` и имя (слово после пробела; `SUBSTR` и `INSTR`). Результат — в `otvety/6`.
7. Название товара имеет вид `Канц-12`. Для первых восьми товаров по `id` выведите `id` и часть названия после дефиса (`SUBSTR` и `INSTR(name, '-')`). Результат — в `otvety/7`.
8. Для первых шести клиентов по `id` выведите `id`, позицию символа `@` в имени (`INSTR(name, '@')`) и позицию `@` в почте. Обратите внимание, чем различаются «не найдено» и «нет данных». Результат — в `otvety/8`.
9. Выведите `id` и фамилию первых шести клиентов по `id`, у которых фамилия содержит не менее 7 символов (`LENGTH` от выражения для фамилии, в `WHERE`). Результат — в `otvety/9`.
10. Для первых шести клиентов по `id` выведите `id` и строку вида `Иванов А.`: фамилия, пробел, первая буква имени и точка (склейка `||`; первая буква — `SUBSTR` с длиной 1). Результат — в `otvety/10`.

Раздел 18. Команда REPLACE, TRIM, UPPER, LOWER — преобразование строки

18.1. Теоретические сведения

Назначение. Эти функции возвращают изменённую копию строки: заменяют подстроку, убирают лишние пробелы, меняют регистр.

Синтаксис:

```
REPLACE(s, что, на_что)
TRIM(s) / LTRIM(s) / RTRIM(s)
UPPER(s) / LOWER(s)
```

Правила:

- `REPLACE` заменяет **все** вхождения; замена на пустую строку `''` удаляет подстроку;
- `UPPER` и `LOWER` в SQLite преобразуют регистр только латинских букв; кириллица остаётся без изменений;
- если любой аргумент равен `NULL`, результат тоже `NULL`;
- функции можно вкладывать: `UPPER(TRIM(s))`;
- оригинал в таблице не меняется: функция работает только с результатом запроса.

Основные функции:

Элемент	Описание
`REPLACE(s, что, на_что)`	заменяет все вхождения подстроки
`TRIM(s)`, `LTRIM(s)`, `RTRIM(s)`	убирает пробелы с двух сторон, слева, справа
`UPPER(s)`, `LOWER(s)`	верхний и нижний регистр (только латиница)

Пример:

```
SELECT name, UPPER(name) FROM suppliers LIMIT 3;                -- кириллица не меняется
SELECT name, REPLACE(name, ' ', '_') FROM customers LIMIT 3;
SELECT REPLACE('a-b-c', '-', '');                               -- abc
SELECT LENGTH(TRIM('  abc  '));                                 -- 3
```

18.2. Задания

Рабочий каталог: `~/sql-lab/04_funktsii`. Ответ каждого задания записывайте в файл `otvety/N`, где `N` — номер задания.

11. Для первых пяти клиентов по `id` с заданной почтой выведите `id` и почту, в которой `sever.example` заменено на `example.org` (`REPLACE`). Результат — в `otvety/11`.
12. Для первых пяти клиентов по `id` выведите `id` и имя, где пробел заменён символом `_` (`REPLACE`). Результат — в `otvety/12`.
13. Для первых пяти клиентов по `id` с заданной почтой выведите `id` и почту, из которой удалён текст `@sever.example` (замена на пустую строку). Результат — в `otvety/13`.
14. Для первых пяти клиентов по `id` с заданной почтой выведите `id` и почту заглавными буквами (`UPPER`). Результат — в `otvety/14`.
15. Для первых пяти товаров по `id` выведите `id`, категорию и категорию, переведённую в нижний регистр (`LOWER`). Убедитесь, что кириллица не изменилась. Результат — в `otvety/15`.
16. Допишите к имени по два пробела с каждой стороны (`'  ' || name || '  '`) и примените `TRIM`. Для первых пяти клиентов по `id` выведите `id` и длину результата. Результат — в `otvety/16`.
17. Сколько символов в строке `'  Привет  '` после обрезки пробелов только слева (`LTRIM`)? Результат — в `otvety/17`.
18. Для первых пяти клиентов по `id` с заданной почтой выведите `id` и число букв `e` в адресе: длина строки минус длина строки без букв `e` (`REPLACE`). Результат — в `otvety/18`.
19. Для первых пяти клиентов по `id` с заданной почтой выведите `id` и логин (часть до `@`) заглавными буквами (`UPPER` от `SUBSTR`/`INSTR`). Результат — в `otvety/19`.
20. Для первых пяти клиентов по `id` с заданной почтой выведите `id` и адрес, в котором `@` заменено на `[at]`, а затем точка заменена на `[dot]` (вложенные `REPLACE`). Результат — в `otvety/20`.

Раздел 19. Числовые функции

19.1. Теоретические сведения

Назначение. Числовые функции округляют значение, берут модуль и позволяют выбрать наименьшее или наибольшее из нескольких значений в одной строке.

Синтаксис:

```
ROUND(число [, знаков])
ABS(число)
MIN(a, b, ...) / MAX(a, b, ...)
a % b
```

Правила:

- деление двух целых чисел даёт целое (`7 / 2 = 3`); чтобы получить дробный результат, один из операндов делают вещественным: `7 / 2.0`;
- `ROUND(x)` без второго аргумента округляет до целого, но возвращает вещественное число (`3.0`);
- функция `MIN(a, b, ...)` с несколькими аргументами выбирает меньшее из значений **одной строки**; с одним аргументом это агрегат (изучается в теме 5);
- если любой аргумент `NULL`, результат `NULL`;
- оператор `||` превращает число в текст сам: `name || ': ' || ROUND(price / 1000.0, 1) || ' тыс.'`;
- `price / 1000 * 1000` (целое деление, затем умножение) округляет вниз до тысяч; `x % 2 = 0` — условие чётности;
- дробные числа хранятся приблизительно, поэтому возможны результаты вроде `0.30000000000000004`: для вывода применяйте `ROUND`.

Основные функции и операции:

Элемент	Описание
`ROUND(x, n)`	округляет до `n` знаков после запятой
`ABS(x)`	модуль числа
`MIN(a, b, ...)`, `MAX(a, b, ...)`	наименьшее и наибольшее из нескольких значений
`%`	остаток от деления: `10 % 3` равно 1

Пример:

```
SELECT 7 / 2, 7 / 2.0, 7 % 2;                                    -- 3, 3.5, 1
SELECT name, price, ROUND(price * 1.2, 1) AS with_tax FROM products LIMIT 3;
SELECT name, ABS(stock - 10) AS diff FROM products LIMIT 3;      -- насколько
остаток отличается от 10
SELECT name, MIN(price, 1000) FROM products LIMIT 3;             -- не больше 1000
в каждой строке
SELECT id, price / 1000 * 1000 FROM products LIMIT 3;            -- вниз до тысяч
SELECT id, name FROM products WHERE id % 2 = 0 LIMIT 3;          -- чётные id
SELECT name || ': ' || ROUND(price / 1000.0, 1) || ' тыс.' FROM products LIMIT 3;
```

19.2. Задания

Рабочий каталог: `~/sql-lab/04_funktsii`. Ответ каждого задания записывайте в файл `otvety/N`, где `N` — номер задания.

21. Для первых пяти товаров по `id` выведите `id` и треть цены, округлённую до двух знаков: `ROUND(price / 3.0, 2)`. Результат — в `otvety/21`.
22. Для первых пяти товаров по `id` выведите `id`, `price / 100` (целая часть) и `price % 100` (остаток). Результат — в `otvety/22`.
23. Для первых пяти товаров по `id` выведите `id`, остаток `stock` и модуль разности остатка и 20 (`ABS`). Результат — в `otvety/23`.
24. Для первых пяти товаров по `id` выведите `id` и цену с наценкой 20 %, округлённую до одного знака: `ROUND(price * 1.2, 1)`. Результат — в `otvety/24`.
25. Для первых пяти товаров по `id` выведите `id`, `MIN(price, 1000)` и `MAX(stock, 5)`. Результат — в `otvety/25`.
26. Для первых пяти товаров по `id` выведите `id` и цену, округлённую вниз до тысяч (`price / 1000 * 1000`). Результат — в `otvety/26`.
27. Выведите `id` и название первых пяти товаров с чётным `id` (оператор `%`: остаток от деления на 2 равен нулю). Результат — в `otvety/27`.
28. Для первых шести позиций заказов (`order_items`), упорядоченных по `order_id`, затем `product_id`, выведите `order_id`, `product_id` и стоимость строки со скидкой 10 %: `ROUND(qty * price * 0.9, 1)`. Результат — в `otvety/28`.
29. Для первых пяти товаров по `id` выведите `id`, `price / 7` и `ROUND(price / 7.0, 2)`. Сравните результаты. Результат — в `otvety/29`.
30. Для первых пяти товаров по `id` выведите строку вида `Канц-1: 3.4 тыс.`: название, двоеточие с пробелом, цена в тысячах, округлённая до одного знака (`ROUND(price / 1000.0, 1)`), пробел и слово `тыс.`; части соединяйте `||`. Результат — в `otvety/30`.

Раздел 20. Функции даты и времени

20.1. Теоретические сведения

Назначение. В SQLite нет отдельного типа «дата»: даты хранятся текстом `ГГГГ-ММ-ДД` (или `ГГГГ-ММ-ДД ЧЧ:ММ:СС`), а для работы с ними служат функции, разбирающие такой текст.

Синтаксис:

```
DATE(дата [, модификатор, ...])
STRFTIME('формат', дата [, модификатор, ...])
JULIANDAY(дата)
```

Правила:

- дата должна быть в формате `ГГГГ-ММ-ДД`; запись `11.10.2026` функции не понимают и возвращают `NULL`;
- `STRFTIME` всегда возвращает **текст**, даже для чисел: `STRFTIME('%Y', d) = '2026'` истинно, а `= 2026` — нет;
- модификаторы применяются по порядку слева направо: `DATE(d, 'start of month', '+1 month', '-1 day')` — последний день месяца;
- разность `JULIANDAY(a) - JULIANDAY(b)` — число суток между датами (дробное);
- `STRFTIME('%m', d)` тоже текст из двух цифр (`'03'`), поэтому с ним сравнивают строкой: `STRFTIME('%m', d) <= '03'`;
- функции дат можно использовать в `WHERE`: `WHERE STRFTIME('%Y', order_date) = '2026'`;
- `CAST(число AS INTEGER)` отбрасывает дробную часть (подробнее в последней записи раздела): `CAST(JULIANDAY(a) - JULIANDAY(b) AS INTEGER)` — полные сутки;
- самую раннюю дату проще получить сортировкой: `ORDER BY order_date, id LIMIT 1` (агрегат `MIN` ещё не изучен);
- текущую дату возвращает `DATE('now')`, но результат зависит от дня запуска, поэтому в заданиях с проверкой его избегают.

Функции, коды формата и модификаторы:

Элемент	Описание
`DATE(d, ...)`, `TIME(d, ...)`, `DATETIME(d, ...)`	дата, время, дата и время в стандартном виде
`STRFTIME('%Y', d)`	год: `2026`
`'%m'`, `'%d'`	месяц и день месяца (две цифры)
`'%w'`	день недели: 0 — воскресенье, 6 — суббота
`'%Y-%m'`	формат можно составлять из кодов и обычных символов
`'+N days'`, `'-N months'`, `'+N years'`	сдвиг даты
`'start of month'`, `'start of year'`	перейти к началу месяца или года

Элемент	Описание
JULIANDAY(d)	дата в виде числа суток; нужна для разности дат

Пример:

```
SELECT registered, STRFTIME('%m', registered) AS mon FROM customers LIMIT 3;
SELECT registered, DATE(registered, 'start of month') FROM customers LIMIT 3;
SELECT registered, DATE(registered, '+1 month') FROM customers LIMIT 3;
SELECT registered, STRFTIME('%w', registered) FROM customers LIMIT 3;
-- день недели
SELECT CAST(JULIANDAY('2026-12-31') - JULIANDAY(registered) AS INTEGER) AS
days_left
FROM customers LIMIT 3;
SELECT order_date, STRFTIME('%Y-%m', order_date) FROM orders LIMIT 3;
-- год и месяц
SELECT order_date, DATE(order_date, 'start of month', '+1 month', '-1 day') FROM
orders LIMIT 3;   -- конец месяца
SELECT id, order_date FROM orders WHERE STRFTIME('%Y', order_date) = '2026' LIMIT
3;   -- год — текст
SELECT CAST(JULIANDAY('2026-10-11') - JULIANDAY(order_date) AS INTEGER)
FROM orders ORDER BY order_date, id LIMIT 1;
-- возраст самого раннего заказа
```

20.2. Задания

Рабочий каталог: `~/sql-lab/04_funktsii`. Ответ каждого задания записывайте в файл `otvety/N`, где `N` — номер задания.

31. Выведите `id` и год заказа (`STRFTIME('%Y', order_date)`) для первых пяти заказов по `id`. Результат — в `otvety/31`.
32. Выведите `id` и номер месяца (`STRFTIME('%m', ...)`, две цифры) для первых пяти заказов по `id`. Результат — в `otvety/32`.
33. Выведите `id` и строку вида `2026-03` (год и месяц, `STRFTIME('%Y-%m', ...)`) для первых пяти заказов по `id`. Результат — в `otvety/33`.
34. Для первых пяти заказов по `id` выведите `id`, дату заказа и дату через 14 дней (`DATE(order_date, '+14 days')`). Результат — в `otvety/34`.
35. Для первых шести заказов по `id` выведите `id` и день недели (`'%w'`: 0 — воскресенье, 6 — суббота). Результат — в `otvety/35`.
36. Для первых пяти заказов по `id` выведите `id` и дату первого числа месяца заказа (`DATE` с модификатором `'start of month'`). Результат — в `otvety/36`.
37. Для первых пяти заказов по `id` выведите `id` и дату последнего дня месяца заказа (`DATE` с цепочкой модификаторов: начало месяца, `'+1 month'`, `'-1 day'`). Результат — в `otvety/37`.
38. Выведите `id` и дату первых пяти заказов по `id`, сделанных в 2026 году (`STRFTIME('%Y', ...)` в `WHERE`). Помните, что `STRFTIME` возвращает текст. Результат — в `otvety/38`.
39. Для первых пяти заказов по `id` выведите `id` и число полных суток от даты заказа до `2026-10-11` (`CAST(JULIANDAY(...) - JULIANDAY(...) AS INTEGER)`). Результат — в `otvety/39`.
40. Сколько полных суток прошло от самой ранней даты заказа до `2026-10-11`? Отсортируйте заказы по `order_date`, затем по `id`, возьмите первый (`LIMIT 1`) и вычислите разность `JULIANDAY` с приведением `CAST(... AS INTEGER)`. Результат — в `otvety/40`.

Раздел 21. Команда CASE — условное выражение

21.1. Теоретические сведения

Назначение. CASE возвращает одно из нескольких значений в зависимости от условий — это «если … то … иначе» внутри запроса.

Синтаксис:

```
CASE WHEN условие1 THEN значение1
     WHEN условие2 THEN значение2
     ELSE значение_по_умолчанию
END

CASE выражение WHEN значение1 THEN результат1 ... ELSE ... END
```

Правила:

- условия проверяются сверху вниз, срабатывает первое истинное; поэтому порядок WHEN важен: сначала более узкое условие;
- если ни одно условие не истинно и ELSE нет, результат — NULL; ELSE лучше писать всегда;
- все ветви должны возвращать значения совместимых типов — иначе столбец получится «разнотипным»;
- CASE — выражение, а не команда: его можно использовать в SELECT, WHERE, ORDER BY, GROUP BY и внутри агрегатов;
- после END удобно задать имя через AS;
- CASE возвращает обычное значение, поэтому его можно вставить в арифметику (qty * price * CASE ... END) и в ORDER BY как ключ сортировки;
- проверку на NULL пишут отдельной первой веткой WHEN city IS NULL: сравнение city = '...' с NULL не истинно.

Пример:

```
SELECT name,
       CASE WHEN stock = 0 THEN 'нет'
            WHEN stock < 10 THEN 'мало'
            ELSE 'много' END AS level
FROM products LIMIT 5;

SELECT id, CASE status WHEN 'new' THEN 'Новый' WHEN 'paid' THEN 'Оплачен' ELSE
'Другой' END
FROM orders LIMIT 5;                                          -- простая форма:
сравнение с одним выражением

SELECT id, stock FROM products
ORDER BY CASE WHEN stock = 0 THEN 0 ELSE 1 END, id LIMIT 5;    -- CASE как ключ
сортировки

SELECT order_id, qty * price * CASE WHEN qty >= 4 THEN 0.9 ELSE 1 END FROM
order_items LIMIT 5;  -- CASE в арифметике

SELECT id, CASE WHEN city IS NULL THEN 'не указан' WHEN city = 'Москва' THEN
'столица' ELSE 'регион' END
```

```
FROM customers LIMIT 5;                                    -- NULL проверяется
первым
```

21.2. Задания

Рабочий каталог: `~/sql-lab/04_funktsii`. Ответ каждого задания записывайте в файл `otvety/N`, где `N` — номер задания.

41. Для первых десяти товаров по `id` выведите название и группу цены: `дёшево`, если цена меньше 1000, `средне`, если меньше 5000, иначе `дорого` (выражение `CASE`). Результат — в `otvety/41`.
42. Для первых шести заказов по `id` выведите `id` и статус по-русски: `new` — `Новый`, `paid` — `Оплачен`, `shipped` — `Отправлен`, иначе `Отменён` (простая форма `CASE status WHEN ...`). Результат — в `otvety/42`.
43. Для первых восьми товаров по `id` выведите `id` и уровень остатка: `нет` при остатке 0, `мало` при остатке меньше 10, иначе `много` (`CASE WHEN`). Результат — в `otvety/43`.
44. Для первых шести клиентов по `id` выведите `id` и слово `VIP`, если `vip` равно 1, иначе `обычный` (простая форма `CASE vip WHEN ...`). Результат — в `otvety/44`.
45. Для первых шести товаров по `id` выведите `id` и слово `дёшево` для цены меньше 1000; ветку `ELSE` не пишите. Посмотрите, что получилось у остальных строк. Результат — в `otvety/45`.
46. Для первых шести заказов по `id` выведите `id`, дату и слово `2025`, если дата раньше `2026-01-01`, иначе `2026` (`CASE WHEN` со сравнением дат-строк). Результат — в `otvety/46`.
47. Выведите `id`, название и остаток первых шести товаров, отсортированных так: сначала товары с нулевым остатком, затем остальные; внутри групп — по `id` (ключ сортировки — выражение `CASE`). Результат — в `otvety/47`.
48. Для первых восьми клиентов по `id` выведите `id` и слово: `не указан`, если город `NULL`; `столица`, если `Москва`; иначе `регион`. Выражение `CASE WHEN`; проверку `city IS NULL` ставьте первой. Результат — в `otvety/48`.
49. Для первых шести позиций заказов, упорядоченных по `order_id`, `product_id`, выведите `order_id`, `product_id` и стоимость строки (`qty * price`) с коэффициентом 0.9 при `qty >= 4`, 0.95 при `qty >= 2` и 1 иначе; коэффициент задайте выражением `CASE` внутри умножения, итог округлите (`ROUND`) до двух знаков. Результат — в `otvety/49`.
50. Для первых шести заказов по `id` выведите `id`, дату и квартал: `I`, `II`, `III` или `IV` по месяцу (`STRFTIME('%m', ...)` не больше `'03'`, `'06'`, `'09'`). Результат — в `otvety/50`.

Раздел 22. Команда COALESCE, NULLIF, IFNULL — работа с NULL

22.1. Теоретические сведения

Назначение. Эти функции подставляют значение вместо NULL или, наоборот, превращают подходящее значение в NULL. Они нужны, чтобы пустые значения не «заражали» вычисления.

Синтаксис:

```
COALESCE(a, b, ...)
IFNULL(a, b)
NULLIF(a, b)
```

Правила:

- COALESCE возвращает первый аргумент, не равный NULL; если все NULL, то NULL;
- IFNULL(a, b) — упрощённый вариант COALESCE на два аргумента;
- NULLIF(a, b) возвращает NULL, если a = b, иначе a; классическое применение — защита от деления на ноль: x / NULLIF(y, 0);
- в SQLite деление на ноль даёт NULL, а в других СУБД — ошибку; NULLIF делает защиту явной и переносимой;
- COALESCE работает и с числами: COALESCE(ROUND(x / NULLIF(y, 0), 2), 0);
- арифметика с NULL даёт NULL (5 + NULL), склейка || — тоже; замена через COALESCE восстанавливает вычисление;
- подстановка скрывает проблему данных, а не решает её: следите, чтобы заменённое значение не попало в статистику как настоящее.

Пример:

```
SELECT name, COALESCE(city, 'не указан') FROM customers LIMIT 5;
SELECT name, 1000 / NULLIF(stock, 0) FROM products LIMIT 5;   -- при stock = 0
вернётся NULL
SELECT name, IFNULL(city, '-') || ' / ' || name FROM customers LIMIT 3;
SELECT id, COALESCE(email, city, 'нет данных') FROM customers LIMIT 5;   --
цепочка: первое не NULL
SELECT id, NULLIF(status, 'cancelled') FROM orders LIMIT 5;              --
'cancelled' превращается в NULL
SELECT name || ' (' || COALESCE(city, 'город не указан') || ')' FROM customers
LIMIT 5;
SELECT id, COALESCE(ROUND(price * 1.0 / NULLIF(stock, 0), 2), 0) FROM products
LIMIT 5;
```

22.2. Задания

Рабочий каталог: ~/sql-lab/04_funktsii. Ответ каждого задания записывайте в файл otvety/N, где N — номер задания.

51. Для первых пяти клиентов по id выведите id и почту, а если почта не задана — слово нет (COALESCE). Результат — в otvety/51.

52. Для первых восьми клиентов по id выведите id, имя и город; если города нет — не указан (COALESCE). Результат — в otvety/52.
53. Для первых восьми клиентов по id выведите id и город, а при отсутствии города — дефис - (IFNULL). Результат — в otvety/53.
54. Для первых восьми клиентов по id выведите id и строку имя (город) через || без обработки NULL. Посмотрите, что будет у клиентов без города. Результат — в otvety/54.
55. Повторите предыдущий запрос, но для клиентов без города подставьте текст город не указан (COALESCE). Результат — в otvety/55.
56. Для первых восьми клиентов по id выведите id и контакт: почта, а если её нет — город, а если нет и его — нет данных. Результат — в otvety/56.
57. Для первых восьми товаров по id выведите id и 1000 / NULLIF(stock, 0). Результат — в otvety/57.
58. Для первых восьми заказов по id выведите id и статус; статус cancelled замените на NULL (NULLIF). Результат — в otvety/58.
59. Выведите id, имя и город первых восьми клиентов по id, чей город не Москва; клиенты без города тоже подходят. Используйте COALESCE(city, '') <> 'Москва'. Результат — в otvety/59.
60. Для первых восьми товаров по id выведите id и price / stock, округлённую до двух знаков (ROUND(price * 1.0 / NULLIF(stock, 0), 2)); если остаток равен нулю, выведите 0 (COALESCE). Результат — в otvety/60.

Раздел 23. Команда CAST и TYPEOF — типы значений

23.1. Теоретические сведения

Назначение. SQLite хранит тип у каждого **значения**, а не только у столбца. TYPEOF показывает этот тип, а CAST явно преобразует значение в нужный.

Синтаксис:

```
TYPEOF(выражение)
CAST(выражение AS тип)
```

Правила:

- TYPEOF возвращает одно из слов: integer, real, text, blob, null;
- типы для CAST: INTEGER, REAL, TEXT; приведение к INTEGER **отбрасывает** дробную часть (CAST(3.9 AS INTEGER) равно 3), а не округляет;
- CAST(-3.9 AS INTEGER) равно -3: отбрасывание идёт к нулю;
- из текста берётся числовое начало: CAST('12abc' AS INTEGER) равно 12; если начала-числа нет ('abc'), результат 0 — без сообщения об ошибке;
- CAST(число AS TEXT) даёт текст, который можно склеивать через ||; SUBSTR и STRFTIME всегда возвращают текст;
- строки сравниваются посимвольно: '10' < '9' истинно, а CAST('10' AS INTEGER) < CAST('9' AS INTEGER) — ложно; результат сравнения в SQLite — число 1 (истина) или 0 (ложь);
- в столбец типа INTEGER можно записать и текст, если его нельзя превратить в число: база не откажет, поэтому данные нужно проверять;
- для вещественного деления достаточно CAST(x AS REAL) / y или x / 2.0.

Пример:

```
SELECT TYPEOF(1), TYPEOF(2.5), TYPEOF('x'), TYPEOF(NULL);   -- integer, real,
text, null
SELECT CAST('42' AS INTEGER) + 1, CAST(3.9 AS INTEGER);        -- 43, 3
SELECT name, CAST(price AS REAL) / 3 FROM products LIMIT 3;
SELECT DISTINCT TYPEOF(price) FROM products;                   -- проверка: все ли
цены числа
SELECT '10' < '9', CAST('10' AS INTEGER) < CAST('9' AS INTEGER);   -- 1, 0
SELECT CAST(-3.9 AS INTEGER), CAST('12abc' AS INTEGER), CAST('abc' AS INTEGER);
-- -3, 12, 0
SELECT id, CAST(STRFTIME('%Y', order_date) AS INTEGER) + 1 FROM orders LIMIT 3;
-- год как число
SELECT id, CAST(price AS TEXT) || ' руб.' FROM products LIMIT 3;
```

23.2. Задания

Рабочий каталог: ~/sql-lab/04_funktsii. Ответ каждого задания записывайте в файл otvety/N, где N — номер задания.

61. Выведите одной строкой типы значений 1, 1.5, 'a', NULL (функция TYPEOF). Результат — в otvety/61.

62. Для товара с `id` 1 выведите одной строкой `TYPEOF(price)`, `TYPEOF(price / 2)`, `TYPEOF(price / 2.0)` и `TYPEOF(name)`. Результат — в `otvety/62`.
63. Чему равно `CAST('42' AS INTEGER) + 1`? Результат — в `otvety/63`.
64. Выведите одной строкой `CAST(3.9 AS INTEGER)` и `CAST(-3.9 AS INTEGER)`. Результат — в `otvety/64`.
65. Выведите одной строкой `CAST('abc' AS INTEGER)` и `CAST('12abc' AS INTEGER)`. Результат — в `otvety/65`.
66. Для первых пяти товаров по `id` выведите `id` и строку цены с припиской `руб.`: `CAST(price AS TEXT) || ' руб.'`. Результат — в `otvety/66`.
67. Для первых пяти товаров по `id` выведите `id` и `stock` в процентах от цены, округлённые до двух знаков: `ROUND(CAST(stock AS REAL) / price * 100, 2)`. Результат — в `otvety/67`.
68. Для первых пяти товаров по `id` выведите `id`, тип части названия после дефиса (`TYPEOF(SUBSTR(...))`) и тип той же части после `CAST(... AS INTEGER)`. Результат — в `otvety/68`.
69. Выведите одной строкой результаты сравнений `'10' < '9'` и `CAST('10' AS INTEGER) < CAST('9' AS INTEGER)`. Результат — в `otvety/69`.
70. Для первых пяти заказов по `id` выведите `id`, тип года (`TYPEOF(STRFTIME('%Y', order_date))`) и следующий год как число: `CAST(STRFTIME('%Y', order_date) AS INTEGER) + 1`. Результат — в `otvety/70`.

Вопросы для размышления

1. Чем динамическая типизация SQLite удобна и в чём опасна, если в столбец «цена» по ошибке попал текст?
2. Когда `COALESCE` скрывает проблему данных, а не решает её?

Тема 5. Агрегатные функции и группировка

Общие сведения

Агрегатные функции сворачивают набор строк в одно значение: `COUNT` (число строк; `COUNT(*)` считает все, `COUNT(столбец)` — только непустые), `SUM`, `AVG`, `MIN`, `MAX`, `GROUP_CONCAT` (склейка значений). Все они игнорируют `NULL`: среднее считается только по известным значениям. Для подсчёта различных значений служит `COUNT(DISTINCT столбец)`.

Группировка `GROUP BY` делит строки на группы с одинаковым значением указанных столбцов и применяет агрегат к каждой группе отдельно. В списке столбцов запроса с группировкой допустимы только столбцы из `GROUP BY` и агрегаты. Условие на *строки* (до группировки) пишут в `WHERE`, а условие на *группы* (по результату агрегата) — в `HAVING`: `WHERE price > 100` отбирает товары, а `HAVING COUNT(*) > 5` отбирает категории, где больше пяти товаров.

Типовые задачи: «сколько в каждой категории», «какая сумма по каждому клиенту», «какие значения встречаются слишком часто» (отбор по порогу в `HAVING`). Группировать можно по нескольким столбцам и по выражениям, например по месяцу: `GROUP BY STRFTIME('%Y-%m', order_date)`. Итоговые отчёты строят, комбинируя агрегаты с соединениями (следующий модуль).

Примеры.

```
SELECT SUM(stock) FROM products;
SELECT ROUND(AVG(price), 2) FROM products;
SELECT MIN(price), MAX(price) FROM products;
SELECT category, COUNT(*) FROM products GROUP BY category ORDER BY category;
SELECT status, COUNT(*) AS n FROM orders GROUP BY status ORDER BY n DESC, status;
SELECT customer_id, COUNT(*) FROM orders GROUP BY customer_id HAVING COUNT(*) >=
3;
SELECT STRFTIME('%Y-%m', order_date) AS ym, COUNT(*) FROM orders GROUP BY ym ORDER
BY ym;
```

Что чаще всего идёт не так

Среднее по столбцу с `NULL` относится только к заполненным строкам; если нужно считать пустые нулями, используют `AVG(COALESCE(x, 0))`. Вторая ловушка — ставить условие на агрегат в `WHERE`: SQLite ответит ошибкой.

Рабочий каталог и проверка

Каталог полигона: `~/sql-lab/05_agregaty`. Проверка: `python3 proverka.py 5`.

Директору нужны не строки, а **цифры**: сколько всего товаров на складе, какова средняя цена, сколько заказов в каждом статусе, какая категория дороже всего. Такие вопросы решают **агрегатные функции** (`COUNT`, `SUM`, `AVG`, `MIN`, `MAX`), которые сворачивают много строк в одно значение, и группировка `GROUP BY`, которая делает это отдельно для каждой группы. Условие на группы записывается

не в `WHERE`, а в `HAVING`: `WHERE` отбирает строки **до** группировки, `HAVING` — группы **после** неё. Задания идут по конструкциям: `COUNT`; `SUM` и `AVG`; `MIN`, `MAX` и `GROUP_CONCAT`; `GROUP BY`; `HAVING` — по десять на каждую. Часть заданий выполняется на кадровой базе `hr.db`.

Раздел 24. Команда COUNT — подсчёт строк

24.1. Теоретические сведения

Назначение. COUNT сворачивает множество строк в одно число — количество строк или количество заполненных значений. Это самая частая агрегатная функция.

Синтаксис:

```
COUNT(*)
COUNT(столбец)
COUNT(DISTINCT столбец)
```

Правила:

- COUNT(*) считает **все** строки, в том числе с NULL;
- COUNT(столбец) считает только строки, где столбец не NULL;
- COUNT(DISTINCT столбец) считает различные значения (без NULL);
- WHERE выполняется до агрегата: считаются только отобранные строки;
- запрос с агрегатом без GROUP BY возвращает ровно одну строку, даже если подходящих строк нет (COUNT даёт 0).

Варианты подсчёта:

Элемент	Описание
COUNT(*)	число строк
COUNT(x)	число строк, где x не NULL
COUNT(DISTINCT x)	число различных значений x

Пример:

```
SELECT COUNT(*) FROM suppliers;                    -- всего поставщиков
SELECT COUNT(email) FROM customers;                -- у скольких указан email
(NULL не считается)
SELECT COUNT(DISTINCT supplier_id) FROM products;  -- сколько разных поставщиков
у товаров
SELECT COUNT(*) FROM orders WHERE status = 'new';  -- считаются только строки,
прошедшие WHERE
```

24.2. Задания

Рабочий каталог: ~/sql-lab/05_agregaty. Ответ каждого задания записывайте в файл otvety/N, где N — номер задания.

1. Сколько клиентов в таблице customers? Запишите число в otvety/1.
2. Сколько товаров имеется в наличии (stock больше нуля)? Запишите число в otvety/2.

3. У скольких клиентов указан город? Используйте `COUNT(city)`. Запишите число в `otvety/3`.
4. Сколько клиентов не указали email? Вычислите как разность `COUNT(*)` и `COUNT(email)`. Запишите число в `otvety/4`.
5. Сколько различных городов встречается у клиентов (`COUNT(DISTINCT city)`)? Запишите число в `otvety/5`.
6. Сколько различных клиентов сделали хотя бы один заказ (`COUNT(DISTINCT customer_id)`)? Запишите в `otvety/6`.
7. Сколько заказов в статусе `shipped` оформлено в 2025 году (дата начинается с `2025`, используйте `LIKE`)? Запишите число в `otvety/7`.
8. Сколько различных товаров хотя бы раз попадало в заказы (таблица `order_items`, столбец `product_id`)? Запишите число в `otvety/8`.
9. Одной строкой выведите по таблице `customers`: `COUNT(*)`, `COUNT(city)` и `COUNT(DISTINCT city)`. Результат — в `otvety/9`.
10. Сколько клиентов зарегистрировались не раньше `2024-01-01` и при этом указали и город, и email? Запишите число в `otvety/10`.

Раздел 25. Команда SUM и AVG — сумма и среднее

25.1. Теоретические сведения

Назначение. SUM складывает значения столбца или выражения, AVG вычисляет их среднее арифметическое. Обе функции работают с числами.

Синтаксис:

```
SUM(выражение)
AVG(выражение)
```

Правила:

- обе функции **пропускают** NULL: AVG делит на количество непустых значений, а не на число строк; чтобы считать NULL нулём, пишут AVG(COALESCE(x, 0));
- SUM пустого набора даёт NULL, а не 0 (в отличие от COUNT);
- агрегат можно считать от выражения: SUM(price * qty);
- AVG возвращает вещественное число; для красивого вывода его округляют: ROUND(AVG(x), 2);
- агрегаты нельзя вкладывать друг в друга: SUM(COUNT(*)) — ошибка;
- в WHERE агрегат использовать нельзя;
- деление целых чисел даёт целое (7 / 2 = 3); чтобы получить дробь или процент, умножьте на вещественное число: 100.0 * SUM(vip) / COUNT(*).

Функции:

Элемент	Описание
SUM(x)	сумма значений
AVG(x)	среднее арифметическое (результат вещественный)

Пример:

```
SELECT SUM(qty) FROM order_items;                              -- сколько штук продано
всего
SELECT ROUND(AVG(price), 1) FROM products WHERE stock > 0;     -- ROUND — из темы 4
SELECT AVG(vip), SUM(vip) FROM customers;                      -- у флага 0/1
среднее — доля единиц
SELECT SUM(salary) / COUNT(*) FROM employees;                  -- то же, что
среднее, но вручную (база hr.db)
```

25.2. Задания

Рабочий каталог: ~/sql-lab/05_agregaty. Ответ каждого задания записывайте в файл otvety/N, где N — номер задания.

11. Сколько единиц товара лежит на складе всего (сумма столбца stock)? Запишите число в otvety/11.

12. Какова средняя цена товара, округлённая до двух знаков (`ROUND(AVG(price), 2)`)? Запишите в `otvety/12`.
13. Сколько штук товара продано по всем позициям заказов (сумма `qty` в `order_items`)? Запишите число в `otvety/13`.
14. Вычислите общую сумму всех позиций заказов: сумму произведений `qty * price` в `order_items`. Запишите число в `otvety/14`.
15. Какова средняя цена товаров категории `Электроника`, округлённая до двух знаков? Запишите в `otvety/15`.
16. Какова стоимость остатков (`SUM(price * stock)`) только по товарам с ценой больше 1000? Запишите число в `otvety/16`.
17. Одной строкой выведите по `customers`: число VIP-клиентов `SUM(vip)`, число всех клиентов и долю VIP в процентах, округлённую до одного знака (`ROUND(100.0 * SUM(vip) / COUNT(*), 1)`). Результат — в `otvety/17`.
18. В базе `hr.db` найдите суммарную зарплату всех сотрудников (`SUM(salary)` по `employees`). Запишите число в `otvety/18`.
19. В базе `hr.db` выведите одной строкой `ROUND(AVG(dept_id), 3)` и `ROUND(AVG(COALESCE(dept_id, 0)), 3)` по `employees`. Два числа различаются: у двух сотрудников нет отдела (`NULL`). Результат — в `otvety/19`.
20. В базе `hr.db` найдите среднюю зарплату сотрудников, принятых не раньше `2022-01-01`, кроме директора (`position <> 'директор'`); округлите до целого `ROUND(..., 0)`. Запишите в `otvety/20`.

Раздел 26. MIN, MAX и GROUP_CONCAT

26.1. Теоретические сведения

Назначение. MIN и MAX находят наименьшее и наибольшее значение (числа, текст, даты), а GROUP_CONCAT склеивает значения столбца в одну строку.

Синтаксис:

```
MIN(выражение)   MAX(выражение)
GROUP_CONCAT(выражение [, разделитель])
```

Правила:

- MIN и MAX пропускают NULL и работают и с числами, и с текстом, и с датами в формате ГГГГ-ММ-ДД: MAX(order_date) — самая поздняя дата;
- особенность SQLite: если рядом с единственным MIN или MAX стоит обычный столбец, он берётся из той же строки, где найден минимум или максимум: SELECT name, MAX(price) FROM products;
- GROUP_CONCAT по умолчанию разделяет значения запятой без пробела; свой разделитель задают вторым аргументом;
- порядок элементов в GROUP_CONCAT не гарантирован;
- NULL в GROUP_CONCAT пропускаются; для пустого набора результат — NULL.

Функции:

Элемент	Описание
MIN(x), MAX(x)	наименьшее и наибольшее значение
GROUP_CONCAT(x)	значения через запятую
GROUP_CONCAT(x, ' | ')	значения, склеенные с заданным разделителем

Пример:

```
SELECT MIN(stock), MAX(stock) FROM products;             -- наименьший и
наибольший остаток
SELECT MAX(registered) FROM customers WHERE vip = 1;     -- самая поздняя
регистрация среди VIP
SELECT name, MIN(salary) FROM employees;                 -- кто получает меньше
всех (база hr.db)
SELECT GROUP_CONCAT(DISTINCT country) FROM suppliers;    -- список стран одной
строкой
```

26.2. Задания

Рабочий каталог: ~/sql-lab/05_agregaty. Ответ каждого задания записывайте в файл otvety/N, где N — номер задания.

21. Выведите одной строкой наименьшую и наибольшую цену товара (MIN, MAX). Запишите в otvety/21.

22. Выведите одной строкой самую раннюю и самую позднюю дату регистрации клиентов (`registered`). Результат — в `otvety/22`.
23. Какова наибольшая цена (`MAX(price)`) среди товаров категории `Мебель`? Запишите число в `otvety/23`.
24. Какова самая поздняя дата заказа (`MAX(order_date)`) в статусе `shipped`? Запишите дату в `otvety/24`.
25. Выведите одной строкой первое и последнее по алфавиту имя клиента (`MIN(name)`, `MAX(name)`). Результат — в `otvety/25`.
26. В базе `hr.db` выведите наименьшую и наибольшую зарплату сотрудников отдела с `dept_id = 2`. Результат — в `otvety/26`.
27. Склейте названия всех поставщиков в одну строку через разделитель ` / ` (пробел, косая черта, пробел): `GROUP_CONCAT(name, ' / ')`; порядок — как в таблице, без сортировки. Запишите строку в `otvety/27`.
28. В базе `hr.db` склейте функцией `GROUP_CONCAT(name, ' / ')` названия отделов, расположенных в Москве (`WHERE city = 'Москва'`), в одну строку; порядок — как в таблице. Запишите строку в `otvety/28`.
29. Выведите название и цену самого дорогого товара одним запросом: `SELECT name, MAX(price) FROM products`. Результат — в `otvety/29`.
30. Выведите название и цену самого дешёвого товара (`MIN(price)`, обычный столбец `name` рядом) среди имеющихся в наличии (`stock > 0`). Результат — в `otvety/30`.

Раздел 27. Команда GROUP BY — группировка

27.1. Теоретические сведения

Назначение. GROUP BY разбивает строки на группы с одинаковыми значениями ключа и считает агрегаты отдельно для каждой группы. В результате одна строка соответствует одной группе.

Синтаксис:

```
SELECT ключ, агрегат(...) FROM таблица [WHERE ...]
GROUP BY ключ1 [, ключ2, ...] [ORDER BY ...] [LIMIT ...];
```

Правила:

- в списке SELECT допускаются столбцы из GROUP BY и агрегаты; любой другой «голый» столбец SQLite не отвергает, но берёт его значение из произвольной строки группы — это ловушка;
- порядок предложений жёсткий: WHERE → GROUP BY → HAVING → ORDER BY → LIMIT;
- группировать можно по нескольким столбцам (группа — каждая пара значений) и по выражению, например GROUP BY STRFTIME('%Y', registered);
- в GROUP BY и ORDER BY SQLite допускает псевдоним из списка столбцов;
- NULL образует отдельную группу;
- сортировать результат можно по агрегату или его псевдониму: ORDER BY COUNT(*) DESC или ORDER BY n DESC; для повторяемого порядка при равенстве добавляют второй ключ (ORDER BY n DESC, category);
- LIMIT выполняется последним, после группировки и сортировки: ORDER BY n DESC LIMIT 5 даёт пять самых больших групп;
- порядок групп без ORDER BY не определён.

Пример:

```
SELECT vip, COUNT(*) FROM customers GROUP BY vip;                    -- две группы:
обычные и VIP
SELECT supplier_id, category, AVG(price) AS avg_price
FROM products GROUP BY supplier_id, category ORDER BY supplier_id, category;   --
группа = пара значений
SELECT STRFTIME('%Y', order_date) AS y, COUNT(*) FROM orders GROUP BY y ORDER BY
y;   -- по выражению через псевдоним
SELECT order_id, SUM(qty) FROM order_items WHERE price > 1000 GROUP BY order_id
ORDER BY 2 DESC LIMIT 3;   -- WHERE работает до группировки
```

27.2. Задания

Рабочий каталог: ~/sql-lab/05_agregaty. Ответ каждого задания записывайте в файл otvety/N, где N — номер задания.

31. Сколько товаров в каждой категории? Выведите категорию и число (GROUP BY), по алфавиту категорий. Результат — в otvety/31.

32. Сгруппируйте заказы по статусу (`GROUP BY status`) и посчитайте их в каждом статусе. Выведите статус и число `n`, начиная с самых частых; при равенстве — по алфавиту статуса. Результат — в `otvety/32`.
33. Сгруппируйте клиентов по городу (`GROUP BY city`) и посчитайте их в каждом. Выведите город и число, по возрастанию города; клиенты без города образуют отдельную группу (в файле у неё пустое значение города). Результат — в `otvety/33`.
34. Сгруппируйте товары по категории (`GROUP BY category`) и для каждой вычислите стоимость остатков (`SUM(price * stock)`) под именем `v` и выведите категорию и `v` от большей к меньшей; при равенстве — по алфавиту категории. Результат — в `otvety/34`.
35. Сгруппируйте товары по поставщику (`GROUP BY supplier_id`) и выведите число его товаров и среднюю цену, округлённую до двух знаков; порядок — по `supplier_id`. Результат — в `otvety/35`.
36. Сгруппируйте заказы по месяцу (`GROUP BY` по выражению или его псевдониму) и посчитайте их. Выведите месяц в виде `ГГГГ-ММ` (`STRFTIME('%Y-%m', order_date)`) и число, по возрастанию месяца. Результат — в `otvety/36`.
37. Сгруппируйте товары по двум столбцам сразу (`GROUP BY supplier_id, category`) и посчитайте товары в каждой паре. Выведите обе колонки и число, порядок — по `supplier_id`, затем по категории. Результат — в `otvety/37`.
38. Отберите строки условием `WHERE` (заказы с `2026-01-01`, не `cancelled`), затем сгруппируйте по статусу (`GROUP BY status`) и посчитайте заказы. Выведите статус и число, по алфавиту статуса. Результат — в `otvety/38`.
39. Сгруппируйте клиентов по году регистрации (`GROUP BY` по `STRFTIME('%Y', registered)`) и выведите год, число клиентов и число VIP-клиентов (`SUM(vip)`), по возрастанию года. Результат — в `otvety/39`.
40. Сгруппируйте `order_items` по товару (`GROUP BY product_id`) и найдите, сколько штук продано всего (`SUM(qty)` под именем `total`). Выведите `product_id` и `total` для пяти самых продаваемых; при равенстве — по возрастанию `product_id`. Результат — в `otvety/40`.

Раздел 28. Команда HAVING — условие на группы

28.1. Теоретические сведения

Назначение. HAVING отбирает группы после того, как они сформированы и агрегаты посчитаны. Это «WHERE для групп».

Синтаксис:

```
... GROUP BY ключ HAVING условие_на_агрегат ...
```

Правила:

- WHERE отбирает **строки** до группировки и не может содержать агрегатов; HAVING отбирает **группы** и обычно содержит агрегат;
- условие на обычный столбец быстрее и правильнее писать в WHERE: меньше строк попадёт в группировку;
- в HAVING SQLite разрешает псевдоним агрегата из списка столбцов (в других СУБД так нельзя): `SELECT MAX(price) - MIN(price) AS spread ... GROUP BY category HAVING spread > 3000`;
- вместе в одном запросе: WHERE отсекает строки, GROUP BY формирует группы, HAVING отсекает группы; пустой результат — нормальный ответ, если ни одна группа не подошла;
- условия в HAVING объединяют через AND и OR так же, как в WHERE.

Пример:

```
SELECT supplier_id, COUNT(*) AS n FROM products
GROUP BY supplier_id HAVING COUNT(*) > 5 ORDER BY supplier_id;   -- поставщики с
более чем пятью товарами

SELECT category, AVG(price) FROM products
WHERE stock > 0                                                  -- строки
отбираются до группировки
GROUP BY category
HAVING AVG(price) > 3000;                                        -- группы
отбираются после
```

28.2. Задания

Рабочий каталог: `~/sql-lab/05_agregaty`. Ответ каждого задания записывайте в файл `otvety/N`, где `N` — номер задания.

41. Сгруппируйте заказы по клиенту (`GROUP BY customer_id`) и условием на группу (`HAVING`) оставьте клиентов, оформивших не менее трёх заказов: выведите `customer_id` и число заказов, по возрастанию `customer_id`. Результат — в `otvety/41`.
42. Сгруппируйте товары по категории и условием на группу (`HAVING AVG(price) > 3000`) оставьте категории, где средняя цена больше 3000: категорию и среднюю цену, округлённую до двух знаков; порядок — по алфавиту категории. Результат — в `otvety/42`.

43. Для каждой категории вычислите разницу между наибольшей и наименьшей ценой под именем `spread`; условием на группу (`HAVING`) оставьте категории, где `spread` больше 3000; порядок — по алфавиту категории. Результат — в `otvety/43`.
44. Сгруппируйте заказы по статусу и условием `HAVING` оставьте статусы, у которых больше 20 заказов; выведите статус и число заказов; порядок — по алфавиту статуса. Результат — в `otvety/44`.
45. Сгруппируйте клиентов по городу и условием `HAVING` оставьте города, где не менее четырёх клиентов (клиентов без города отсеките в `WHERE`): город и число клиентов, по алфавиту города. Результат — в `otvety/45`.
46. Среди товаров, имеющихся в наличии (`stock > 0`), отсеките остальные в `WHERE`, сгруппируйте по поставщику и условием `HAVING` оставьте тех, у кого таких товаров не менее трёх: выведите `supplier_id` и число товаров, по возрастанию `supplier_id`. Результат — в `otvety/46`.
47. Сгруппируйте `order_items` по товару и условием `HAVING` оставьте товары, проданных штук которых суммарно не менее 30 (`SUM(qty)`): выведите `product_id` и сумму `total`, по возрастанию `product_id`. Результат — в `otvety/47`.
48. Отберите в `WHERE` заказы в статусе `shipped`, сгруппируйте по клиенту и условием `HAVING` оставьте тех, у кого таких заказов не менее двух: выведите `customer_id` и число таких заказов, по возрастанию `customer_id`. Результат — в `otvety/48`.
49. Среди заказов, оформленных до `2026-01-01`, сгруппируйте их по месяцу и условием `HAVING` оставьте месяцы, в которые было не менее пяти заказов: месяц в виде `ГГГГ-ММ` и число заказов, по возрастанию месяца. Результат — в `otvety/49`.
50. В базе `hr.db` сгруппируйте сотрудников по `dept_id` (сотрудников без отдела отсеките в `WHERE`) и условием `HAVING` оставьте отделы, в которых не менее трёх сотрудников. Выведите `dept_id`, число сотрудников и среднюю зарплату, округлённую до целого; порядок — по `dept_id`. Результат — в `otvety/50`.

Вопросы для размышления

1. Чем `WHERE` отличается от `HAVING`, и почему условие по `COUNT(*)` нельзя записать в `WHERE`?
2. Что произойдёт со средним значением, если среди чисел есть `NULL`, и чем `COUNT(*)` отличается от `COUNT(столбец)`?

Тема 6. Соединение таблиц: JOIN

Общие сведения

Принцип хорошей схемы — каждый факт хранится в одном месте: сведения о клиенте — в таблице клиентов, заказы — в таблице заказов, строки заказа — в своей. Таблицы связывают **ключами**: *первичный ключ* (`PRIMARY KEY`) однозначно определяет строку, *внешний ключ* (`REFERENCES`) — ссылка на первичный ключ другой таблицы (`orders.customer_id` ссылается на `customers.id`). Чтобы получить осмысленный результат (заказ вместе с именем клиента), таблицы **соединяют** (`JOIN`) по условию связи.

`INNER JOIN` (или просто `JOIN`) оставляет только пары строк, для которых условие истинно: клиент без заказов в результат не попадёт. `LEFT JOIN` сохраняет все строки левой таблицы, а недостающие значения правой заменяет на `NULL`; именно он отвечает на вопрос «у кого *нет* заказов» (отбор по `правая.ключ IS NULL`). `CROSS JOIN` даёт все возможные пары, `NATURAL JOIN` соединяет по одинаковым именам столбцов (в рабочих запросах его избегают). Таблицу можно соединить *с самой собой*, подключив её дважды под разными **псевдонимами**: так получают «сотрудник — его руководитель».

Короткие псевдонимы таблиц (`orders o`, `customers c`) делают запрос читаемым, а имена столбцов с префиксом (`o.id`, `c.id`) снимают неоднозначность. Соединения вместе с группировкой дают основные отчёты: выручка по клиентам, число позиций в заказе, продажи по товарам. При соединении трёх таблиц следите за кратностью: если соединить заказ и с его строками, и с платежами, суммы удвоятся.

Примеры.

```
SELECT o.id, c.name FROM orders o JOIN customers c ON c.id = o.customer_id ORDER
BY o.id LIMIT 10;

SELECT c.name FROM customers c LEFT JOIN orders o ON o.customer_id = c.id
WHERE o.id IS NULL ORDER BY c.name;                          -- клиенты без заказов

SELECT c.name, SUM(i.qty * i.price) AS total
FROM customers c JOIN orders o ON o.customer_id = c.id JOIN order_items i ON
i.order_id = o.id
WHERE o.status <> 'cancelled' GROUP BY c.id ORDER BY total DESC LIMIT 5;

SELECT e.name, m.name FROM employees e LEFT JOIN employees m ON m.id =
e.manager_id;   -- самосоединение
```

Что чаще всего идёт не так

Забытое условие соединения превращает `JOIN` в декартово произведение: из двух таблиц по 1000 строк получится миллион. Если результат неожиданно огромен, первым делом проверьте предложение `ON`.

Рабочий каталог и проверка

Каталог полигона: `~/sql-lab/06_soedineniya`. Проверка: `python3 proverka.py 6`.

Данные магазина разложены по нескольким таблицам: клиенты отдельно, заказы отдельно, строки заказов отдельно. Чтобы получить осмысленный отчёт («какой клиент на какую сумму заказал»), таблицы нужно **соединить** по общему столбцу. Внутреннее соединение (`JOIN`) оставляет только совпавшие строки, левое (`LEFT JOIN`) сохраняет все строки левой таблицы и показывает `NULL` там, где пары нет. Цепочка соединений проходит через три и больше таблиц; шаблон `LEFT JOIN … IS NULL` находит строки без пары; таблица может соединяться сама с собой (`employees` и её руководители). Соединения вместе с группировкой дают большинство отчётов. Базы: `shop.db` (магазин) и `hr.db` (сотрудники и отделы). Тема построена по конструкциям: на каждую — десять заданий.

Раздел 29. Ключи и связи таблиц

29.1. Теоретические сведения

Назначение. Данные в учебных базах разложены по таблицам, чтобы не повторять одно и то же. Таблицы связаны через ключи: значение первичного ключа одной таблицы хранится в столбце другой.

Синтаксис:

```
id INTEGER PRIMARY KEY                          -- первичный ключ
supplier_id INTEGER REFERENCES suppliers(id)    -- внешний ключ
```

Правила:

- **первичный ключ** (PRIMARY KEY) однозначно определяет строку своей таблицы; он бывает составным (PRIMARY KEY (order_id, product_id));
- **внешний ключ** (REFERENCES) хранит значение первичного ключа другой таблицы — это «ссылка»; по ней строки сопоставляются при соединении;
- связи базы магазина: products.supplier_id → suppliers.id, orders.customer_id → customers.id, order_items.order_id → orders.id, order_items.product_id → products.id;
- связи базы кадров: employees.dept_id → departments.id, employees.manager_id → employees.id (ссылка на ту же таблицу);
- служебная таблица sqlite_master содержит по строке на каждый объект базы; её столбцы: type ('table', 'index'…), name (имя объекта), tbl_name, rootpage, sql (текст CREATE); список таблиц — SELECT name FROM sqlite_master WHERE type = 'table';
- столбцы таблицы отдаёт табличная функция pragma_table_info('таблица') — её используют в FROM как обычную таблицу, имя таблицы пишут в кавычках; у результата столбцы cid (номер столбца с нуля), name (имя), type (тип), notnull (1 — NOT NULL), dflt_value (значение по умолчанию), pk (0 — не ключ, иначе номер в первичном ключе);
- внешние ключи таблицы отдаёт pragma_foreign_key_list('таблица'); у результата столбцы id, seq, table (на какую таблицу ссылка), from (столбец-ссылка в этой таблице), to (столбец, на который ссылаются), on_update, on_delete, match;
- from, table, to и notnull совпадают со служебными словами SQL, поэтому такие имена берут в двойные кавычки ("from") или в квадратные скобки ([from]); name, type, cid, pk в кавычках не нуждаются;
- если у таблицы нет внешних ключей, pragma_foreign_key_list возвращает пустой результат;
- SQLite по умолчанию не проверяет внешние ключи при вставке, поэтому «висячие» ссылки (на несуществующую строку) возможны.

Пример:

```
SELECT sql FROM sqlite_master WHERE name = 'order_items';       -- как задана
таблица
SELECT id, name, dept_id FROM employees LIMIT 3;                -- dept_id —
просто число, ссылка на departments.id
```

```
SELECT name, type, pk FROM pragma_table_info('products');       -- столбцы таблицы
и признак ключа
SELECT name FROM pragma_table_info('customers') WHERE [notnull] = 1 ORDER BY cid;
-- обязательные столбцы, в порядке следования
SELECT COUNT(*) FROM pragma_table_info('orders');               -- сколько в
таблице столбцов
SELECT "from", "table", "to" FROM pragma_foreign_key_list('products');   -- на что
ссылается products
SELECT [from], [table] FROM pragma_foreign_key_list('orders');  -- то же с
квадратными скобками
```

29.2. Задания

Рабочий каталог: `~/sql-lab/06_soedineniya`. Ответ каждого задания записывайте в файл `otvety/N`, где `N` — номер задания.

1. В `shop.db` определите, сколько в базе таблиц (запрос к `sqlite_master`, `type = 'table'`). Запишите одно число. Результат — в `otvety/1`.
2. Выведите имена всех таблиц `shop.db` по алфавиту. Результат — в `otvety/2`.
3. Выведите имена столбцов таблицы `orders` (функция `pragma_table_info('orders')`, столбец `name`) в порядке их следования в таблице (сортировка по `cid`). Результат — в `otvety/3`.
4. Выведите имена столбцов первичного ключа таблицы `order_items` (`pragma_table_info('order_items')`, условие `pk > 0`) по порядку их номеров в ключе (сортировка по `pk`). Результат — в `otvety/4`.
5. Для таблицы `order_items` выведите из `pragma_foreign_key_list('order_items')` три поля: столбец-ссылку (`from`), таблицу (`table`) и её ключ (`to`) — по алфавиту столбца-ссылки. Имена полей берите в квадратные скобки: `[from]`. Результат — в `otvety/5`.
6. В `hr.db` выведите из `pragma_foreign_key_list('employees')` поля `[from]` и `[table]` (столбец-ссылка и таблица, на которую он ссылается) — по алфавиту столбца-ссылки. Результат — в `otvety/6`.
7. В `hr.db` выясните, у скольких сотрудников указан отдел: `COUNT(dept_id)` не считает `NULL`. Запишите одно число. Результат — в `otvety/7`.
8. Сколько столбцов в таблице `products`? Посчитайте строки результата `pragma_table_info('products')` функцией `COUNT(*)`. Запишите одно число. Результат — в `otvety/8`.
9. Выведите имя и тип (`name`, `type`) столбцов таблицы `customers`, объявленных с `NOT NULL` (в `pragma_table_info` у них `[notnull] = 1`; слово `notnull` в SQLite служебное, поэтому имя берут в квадратные скобки), в порядке следования в таблице (по `cid`). Результат — в `otvety/9`.
10. В `hr.db` найдите сотрудника без руководителя (`manager_id IS NULL`): выведите его имя и должность. Результат — в `otvety/10`.

Раздел 30. Команда JOIN … ON — внутреннее соединение

30.1. Теоретические сведения

Назначение. JOIN (то же, что INNER JOIN) объединяет строки двух таблиц по условию связи. В результат попадают только те пары, для которых условие истинно.

Синтаксис:

```
SELECT ...
FROM таблица1 [AS] п1
JOIN таблица2 [AS] п2 ON п2.столбец = п1.столбец
[WHERE ...] [ORDER BY ...];
```

Правила:

- после ON записывают условие связи; как правило, это равенство внешнего ключа и первичного ключа;
- псевдонимы таблиц (orders o, customers c) делают запись короче; если они заданы, столбцы записывают с ними: o.id, c.name;
- если столбец с таким именем есть в обеих таблицах (id, name), без префикса SQLite выдаст ошибку ambiguous column name;
- строки без пары (клиент без заказов) в результат **не** попадают; если они нужны — LEFT JOIN;
- соединение можно сочетать с группировкой и агрегатами: ... JOIN ... ON ... GROUP BY d.id (подробнее — в последней записи модуля);
- первые N строк результата берут связкой ORDER BY … LIMIT N, ключ сортировки — по какой-либо таблице соединения (ORDER BY o.id);
- забытое ON превращает соединение в декартово произведение (каждая строка с каждой): для 40 клиентов и 120 заказов это 4800 строк мусора;
- условия отбора пишут в WHERE как обычно, они вычисляются после соединения; при внутреннем соединении условие в ON и в WHERE даёт один результат.

Виды соединений:

Элемент	Описание
JOIN или INNER JOIN	только совпавшие пары строк
LEFT JOIN	все строки левой таблицы; если пары нет — NULL в столбцах правой
CROSS JOIN	все возможные пары (без условия)

Пример:

```
SELECT o.id, o.status, c.city
FROM orders o JOIN customers c ON c.id = o.customer_id
WHERE o.status = 'new' ORDER BY o.id LIMIT 5;           -- заказ и город его
клиента
```

```
SELECT i.order_id, p.name, i.qty
FROM order_items i JOIN products p ON p.id = i.product_id
WHERE i.order_id = 2 ORDER BY p.name;                    -- что лежит во втором
заказе

SELECT e.name, d.city FROM employees e JOIN departments d ON d.id = e.dept_id
WHERE d.name = 'Склад' ORDER BY e.id;                    -- база hr.db: сотрудники
склада и город отдела
```

30.2. Задания

Рабочий каталог: `~/sql-lab/06_soedineniya`. Ответ каждого задания записывайте в файл `otvety/N`, где `N` — номер задания.

11. Выведите номер заказа и имя клиента для первых десяти заказов по `id` внутренним соединением `JOIN … ON` таблиц `orders` и `customers` (связь `orders.customer_id = customers.id`). Результат — в `otvety/11`.
12. Соедините `products` и `suppliers` (`JOIN … ON`, связь по `supplier_id`) и выведите название товара и название поставщика для первых десяти товаров по `id`. Результат — в `otvety/12`.
13. В `hr.db` соедините `employees` и `departments` (связь `employees.dept_id = departments.id`) и выведите имя сотрудника и название его отдела для первых десяти сотрудников по `id`. Результат — в `otvety/13`.
14. Соедините `order_items` и `products` (связь `order_items.product_id = products.id`) и выведите названия товаров и количество (`qty`) в заказе номер 1, по возрастанию `id` товара. Результат — в `otvety/14`.
15. Соедините `orders` и `customers` и выведите номер заказа, имя клиента и дату заказа для первых восьми заказов со статусом `paid` (условие в `WHERE`), по возрастанию номера заказа. Результат — в `otvety/15`.
16. Соедините `products` и `suppliers` и выведите название товара и поставщика для товаров поставщиков из страны `Россия` (столбец `suppliers.country`, условие в `WHERE`), первые десять по `id` товара. Результат — в `otvety/16`.
17. В `hr.db` выведите имена сотрудников, отдел которых находится в городе `Москва` (город — столбец таблицы `departments`), с названием отдела; по возрастанию `id` сотрудника, первые двенадцать. Результат — в `otvety/17`.
18. Соедините `order_items` и `products`. Для каждой строки заказа вычислите сумму `i.qty * i.price` под именем `line`. Выведите номер заказа, название товара и `line` для пяти самых дорогих строк; при равенстве — по номеру заказа, затем по `id` товара. Результат — в `otvety/18`.
19. Сколько всего строк (позиций) лежит в заказах со статусом `cancelled`? Соедините `orders` и `order_items`. Запишите одно число. Результат — в `otvety/19`.
20. В `hr.db` соедините `employees` и `departments`, сгруппируйте по отделу (`GROUP BY d.id`) и выведите название отдела и среднюю зарплату сотрудников, округлённую до целых (`ROUND(AVG(e.salary), 0)`), по возрастанию `id` отдела. Результат — в `otvety/20`.

Раздел 31. Цепочка из трёх и более таблиц

31.1. Теоретические сведения

Назначение. Соединения можно цеплять друг за другом: каждое следующее `JOIN` связывает новую таблицу с уже подключённой. Так получают отчёт, в котором есть данные из нескольких таблиц сразу.

Синтаксис:

```
FROM а
JOIN б ON б.столбец = а.столбец
JOIN в ON в.столбец = б.столбец
JOIN г ON г.столбец = в.столбец
```

Правила:

- цепочка строится по связям: `customers → orders → order_items → products → suppliers`; можно взять любой её участок;
- у каждой таблицы свой псевдонимный префикс (`c`, `o`, `i`, `p`, `s`), иначе `id` и `name` будут неоднозначны;
- каждое `JOIN` обязано иметь собственное `ON`; пропущенное условие снова даёт декартово произведение;
- связь «один ко многим» размножает строки: после присоединения `order_items` на заказ приходится столько строк, сколько в нём позиций;
- чтобы не получить повторов, когда нужны именно клиенты или товары, а не строки заказов, используют `DISTINCT` или `COUNT(DISTINCT …)`;
- условия по любой таблице цепочки пишут в общем `WHERE`; порядок таблиц в цепочке на результат не влияет;
- отчёт «по клиентам», «по категориям», «по странам» получают так: соедините нужную цепочку, добавьте `GROUP BY` по ключу группы и агрегат (`SUM(i.qty * i.price)`); в списке `SELECT` рядом с ключом группы можно вывести имя, которое для всей группы одинаково;
- строку заказа считают как `i.qty * i.price`: цена продажи хранится в самой строке `order_items`, а не берётся из `products`.

Пример:

```
SELECT o.id, p.name, s.name
FROM orders o JOIN order_items i ON i.order_id = o.id
              JOIN products p ON p.id = i.product_id
              JOIN suppliers s ON s.id = p.supplier_id
WHERE o.id = 4 ORDER BY p.id;                              -- товары заказа и их
поставщики

SELECT DISTINCT c.name FROM customers c
JOIN orders o ON o.customer_id = c.id
JOIN order_items i ON i.order_id = o.id
WHERE i.qty = 5 ORDER BY c.name LIMIT 5;                   -- кто хотя бы раз купил
пять штук одного товара

SELECT p.category, SUM(i.qty * i.price) AS total
```

```
FROM orders o JOIN order_items i ON i.order_id = o.id
              JOIN products p ON p.id = i.product_id
WHERE o.status = 'paid' GROUP BY p.category ORDER BY p.category;   -- выручка
оплаченных заказов по категориям
```

31.2. Задания

Рабочий каталог: `~/sql-lab/06_soedineniya`. Ответ каждого задания записывайте в файл `otvety/N`, где `N` — номер задания.

21. Выведите номер заказа, название товара и количество для заказов с номерами 1–3 (`orders → order_items → products`), по номеру заказа и `id` товара. Результат — в `otvety/21`.
22. Для заказа номер 1 выведите название каждого товара и название его поставщика (`order_items → products → suppliers`), по `id` товара. Результат — в `otvety/22`.
23. Для заказа номер 5 выведите имя клиента, название товара и количество (четыре таблицы: `customers → orders → order_items → products`), по `id` товара. Результат — в `otvety/23`.
24. Цепочкой `customers → orders → order_items` и группировкой по клиенту (`GROUP BY c.id`) вычислите выручку (`SUM(i.qty * i.price)` по заказам, **кроме отменённых** — условие в `WHERE`) под именем `total` и выведите имена и выручку пяти лучших клиентов; при равенстве — по `id` клиента. Результат — в `otvety/24`.
25. Выведите имена (без повторов) первых десяти по алфавиту клиентов, которые покупали товары категории `Электроника` (`customers → orders → order_items → products`). Результат — в `otvety/25`.
26. Для заказов со статусом `shipped` выведите номер заказа, имя клиента и название товара, но только товары категории `Книги`; по номеру заказа и `id` товара, первые десять. Результат — в `otvety/26`.
27. Выведите номер заказа, название товара и название поставщика для строк заказов с товарами поставщиков из страны `Китай` (`order_items → products → suppliers`); по номеру заказа и `id` товара, первые десять. Результат — в `otvety/27`.
28. Для клиентов-VIP (`c.vip = 1`) выведите номер заказа, имя клиента и название товара (цепочка `customers → orders → order_items → products`), по номеру заказа и `id` товара, первые десять строк. Результат — в `otvety/28`.
29. Цепочкой `orders → order_items → products` и группировкой по категории (`GROUP BY p.category`) вычислите выручку (`SUM(i.qty * i.price)` по заказам, **кроме отменённых**) под именем `total`; выведите категорию и выручку по алфавиту категорий. Результат — в `otvety/29`.
30. Цепочкой `orders → order_items → products → suppliers` и группировкой по стране поставщика (`GROUP BY s.country`) вычислите выручку (`SUM(i.qty * i.price)`, заказы **кроме отменённых**) под именем `total`. Выведите страну и выручку, сначала самые большие; при равенстве — по алфавиту страны. Результат — в `otvety/30`.

Раздел 32. Команда LEFT JOIN — сохранение строк левой таблицы

32.1. Теоретические сведения

Назначение. LEFT JOIN возвращает все строки левой таблицы. Если для строки нет пары в правой таблице, столбцы правой таблицы заполняются NULL.

Синтаксис:

```
FROM левая л LEFT JOIN правая п ON п.столбец = л.столбец
```

Правила:

- левая таблица — та, что записана **до** слов LEFT JOIN;
- условие в WHERE по столбцу правой таблицы отбрасывает строки с NULL и превращает LEFT JOIN в обычный JOIN; условие, которое должно ограничивать только правую таблицу, пишут в ON;
- если у одной строки левой таблицы несколько пар справа, она повторяется в результате столько раз;
- NULL из правой таблицы заменяют на понятное значение функцией COALESCE(выражение, замена);
- LEFT JOIN можно цеплять цепочкой: customers c LEFT JOIN orders o ON … LEFT JOIN order_items i ON i.order_id = o.id; клиент без заказов сохранится, а в столбцах o и i у него будут NULL;
- агрегат от пустой группы: COUNT(п.id) даёт 0, а SUM и MAX — NULL, который заменяют на понятное значение: COALESCE(MAX(o.order_date), 'нет');
- после LEFT JOIN с группировкой считайте COUNT(п.id), а не COUNT(*): COUNT(*) даст 1 даже для строки без пары, COUNT(п.id) — 0.

Пример:

```
SELECT s.name, p.name FROM suppliers s LEFT JOIN products p ON p.supplier_id =
s.id
WHERE s.country = 'Китай' ORDER BY s.id, p.id LIMIT 4;     -- поставщик со своими
товарами

SELECT c.name, o.id FROM customers c
LEFT JOIN orders o ON o.customer_id = c.id AND o.status = 'new'   -- условие в ON:
клиенты остаются в результате
ORDER BY c.id, o.id LIMIT 5;

SELECT d.name, COUNT(e.id) FROM departments d
LEFT JOIN employees e ON e.dept_id = d.id AND e.position = 'дизайнер'
GROUP BY d.id ORDER BY d.id;                                -- hr.db: дизайнеры по
отделам, нули сохраняются
```

32.2. Задания

Рабочий каталог: ~/sql-lab/06_soedineniya. Ответ каждого задания записывайте в файл otvety/N, где N — номер задания.

31. Левым соединением `suppliers LEFT JOIN products` выведите название поставщика и название товара, а там, где товара нет, — текст `нет товаров` (`COALESCE`). Только поставщики с `id` 1–2, по `id` поставщика и `id` товара, первые двенадцать строк. Результат — в `otvety/31`.
32. Левым соединением `customers LEFT JOIN orders` выведите имя клиента и номер заказа для клиентов с `id` от 31 до 33 включительно, по `id` клиента и номеру заказа. У клиента без заказов номер остаётся пустым (`NULL`). Результат — в `otvety/32`.
33. В `hr.db` выведите левым соединением `employees LEFT JOIN departments` имя сотрудника и название отдела; если отдела нет — текст `нет отдела`. Только сотрудники с `id` от 22 до 25, по `id`. Результат — в `otvety/33`.
34. Левым соединением `customers LEFT JOIN orders` и группировкой по клиенту (`GROUP BY c.id`) выведите имя клиента и число его заказов (`COUNT(o.id)`) для клиентов с `id` от 30 до 36 включительно, по `id` клиента. Клиенты без заказов должны остаться в результате с нулём. Результат — в `otvety/34`.
35. В `hr.db` левым соединением `departments LEFT JOIN employees` и группировкой по отделу (`GROUP BY d.id`) выведите название каждого отдела и число сотрудников (`COUNT(e.id)`), по возрастанию `id` отдела; отделы без сотрудников тоже должны попасть в результат. Результат — в `otvety/35`.
36. Левым соединением `customers LEFT JOIN orders` и группировкой по клиенту (`GROUP BY c.id`) выведите для клиентов с `id` от 30 до 34 имя и дату самого позднего заказа (`MAX(o.order_date)`); если заказов нет — текст `нет`. По `id` клиента. Результат — в `otvety/36`.
37. Для клиентов с `id` от 1 до 8 выведите (`LEFT JOIN`, `GROUP BY c.id`, `COUNT(o.id)`) имя и число заказов со статусом `paid`. Клиенты без оплаченных заказов должны остаться с нулём, поэтому условие по статусу пишите в `ON`, а не в `WHERE`. Результат — в `otvety/37`.
38. Для каждого поставщика (`suppliers LEFT JOIN products`, `GROUP BY s.id`, `COUNT(p.id)`) выведите название и число его товаров, **имеющихся на складе** (`stock > 0`; условие — в `ON`), по `id` поставщика. Поставщик без таких товаров должен остаться с нулём. Результат — в `otvety/38`.
39. Левым соединением `products LEFT JOIN order_items` и группировкой по товару (`GROUP BY p.id`) выведите для товаров с `id` от 1 до 10 название и сколько штук продано всего (`SUM(i.qty)`); если не продавался — 0 (`COALESCE(SUM(i.qty), 0)`). По `id` товара. Результат — в `otvety/39`.
40. Для клиентов с `id` от 30 до 34 (`GROUP BY c.id`) выведите имя, число заказов (`COUNT(DISTINCT o.id)`) и сумму `qty * price` по всем их заказам (`COALESCE(SUM(...), 0)`). Соединяйте `customers LEFT JOIN orders LEFT JOIN order_items`; по `id` клиента. Результат — в `otvety/40`.

Раздел 33. Поиск строк без пары: LEFT JOIN … IS NULL

33.1. Теоретические сведения

Назначение. Шаблон «найти те, у кого нет связанных записей»: сделать LEFT JOIN и оставить строки, где ключ правой таблицы равен NULL.

Синтаксис:

```
SELECT л.* FROM левая л
LEFT JOIN правая п ON п.внешний_ключ = л.id
WHERE п.id IS NULL;
```

Правила:

- проверяют именно ключ правой таблицы (п.id) — столбец, который в настоящих парах не бывает NULL;
- IS NULL пишется в WHERE, а не в ON: после соединения результат нужно отфильтровать;
- условие «без пары с таким-то свойством» кладут в ON: LEFT JOIN … ON … AND п.статус = '…' WHERE п.id IS NULL;
- обратная задача — «у кого есть хотя бы одна запись» — решается обычным JOIN (с DISTINCT или группировкой, чтобы не получить повторов);
- сравнение = NULL ничего не вернёт, как и в теме 3;
- «нет ни одного заказа с датой от 2026-01-01», «нет товара дороже 9500» — условие со значением кладут в ON вместе с условием связи, а в WHERE проверяют п.id IS NULL;
- пересчитать найденные строки можно обычным COUNT(*) поверх этого же запроса.

Пример:

```
SELECT p.name FROM products p
LEFT JOIN order_items i ON i.product_id = p.id AND i.qty >= 5
WHERE i.order_id IS NULL ORDER BY p.id LIMIT 5;           -- товары, которых ни
разу не брали по пять штук

SELECT d.name FROM departments d
LEFT JOIN employees e ON e.dept_id = d.id AND e.position = 'кладовщик'
WHERE e.id IS NULL;                                        -- отделы без
кладовщиков (hr.db)
```

33.2. Задания

Рабочий каталог: ~/sql-lab/06_soedineniya. Ответ каждого задания записывайте в файл otvety/N, где N — номер задания.

41. Найдите клиентов, у которых нет ни одного заказа (customers LEFT JOIN orders и проверка o.id IS NULL в WHERE). Выведите имена по алфавиту. Результат — в otvety/41.

42. В `hr.db` найдите сотрудников, у которых отдел не задан или не существует (`employees LEFT JOIN departments`, проверка `d.id IS NULL`). Выведите имена по возрастанию `id`. Результат — в `otvety/42`.
43. Найдите клиентов, у которых нет ни одного заказа со статусом `paid` (условие по статусу — в `ON`). Выведите первые двенадцать по `id` клиента: `id` и имя. Результат — в `otvety/43`.
44. Выведите названия первых десяти товаров по `id`, которых **нет** в заказе номер 1 (`products LEFT JOIN order_items` с условием `i.order_id = 1` в `ON`). Результат — в `otvety/44`.
45. Сколько клиентов не сделали ни одного заказа? Используйте `LEFT JOIN … IS NULL` и `COUNT(*)`. Запишите одно число. Результат — в `otvety/45`.
46. Найдите клиентов, у которых нет ни одного заказа со статусом `cancelled` (`LEFT JOIN` и `IS NULL`, условие по статусу — в `ON`). Выведите первые двенадцать по `id` клиента: `id` и имя. Результат — в `otvety/46`.
47. В `hr.db` выведите названия отделов, в которых нет сотрудников с должностью `тестировщик` (должность — условие в `ON`), по `id` отдела. Результат — в `otvety/47`.
48. Выведите названия поставщиков, у которых нет ни одного товара дороже 9500 (`suppliers LEFT JOIN products`, условие `p.price > 9500` — в `ON`, проверка `p.id IS NULL`), по `id` поставщика. Результат — в `otvety/48`.
49. Сколько клиентов сделали хотя бы один заказ? Здесь нужен обычный `JOIN` и `COUNT(DISTINCT c.id)` — без `DISTINCT` клиент считается по числу заказов. Запишите одно число. Результат — в `otvety/49`.
50. Найдите клиентов, у которых нет заказов с датой `2026-01-01` и позже (условие в `ON`; даты в формате `ГГГГ-ММ-ДД`). Выведите первые двенадцать по `id`: `id` и имя. Результат — в `otvety/50`.

Раздел 34. Самосоединение таблицы

34.1. Теоретические сведения

Назначение. Таблицу можно соединить с самой собой, если одна её строка ссылается на другую (сотрудник — руководитель). Для этого ту же таблицу подключают дважды под разными псевдонимами.

Синтаксис:

```
FROM таблица a JOIN таблица б ON б.id = a.ссылка_на_родителя
```

Правила:

- псевдонимы обязательны: без них SQLite не поймёт, о какой «копии» таблицы речь;
- роли копий удобно отражать в псевдонимах: `e` — сотрудник (employee), `m` — руководитель (manager);
- у строки верхнего уровня (директора) ссылка равна `NULL`; обычный `JOIN` её потеряет, а `LEFT JOIN` сохранит;
- одинаковые имена столбцов различают псевдонимами и `AS`: `e.name AS emp`, `m.name AS boss`;
- пары «равных» строк получают условием по общему значению и `a.id < б.id`: так каждая пара встретится один раз и строка не сравнится сама с собой;
- для сотрудников, подчинённых одному руководителю, общее значение — `manager_id`: `JOIN employees b ON b.manager_id = a.manager_id AND a.id < b.id`;
- сгруппировав по руководителю (`GROUP BY m.id`), считают его подчинённых: `COUNT(*)` — число соединённых строк `e`;
- псевдонимы пригодятся и в `LEFT JOIN` одной таблицы с самой собой: `employees e LEFT JOIN employees s ON s.manager_id = e.id` — к каждому сотруднику его подчинённые `s`, `s.id IS NULL` — подчинённых нет;
- соединение можно повторить: `e → m → b` даёт сотрудника, его руководителя и руководителя руководителя.

Пример:

```
SELECT e.name AS emp, m.name AS boss
FROM employees e JOIN employees m ON m.id = e.manager_id
WHERE e.position = 'бухгалтер' ORDER BY e.id;              -- бухгалтеры и их
начальники (hr.db)

SELECT m.position, COUNT(*) AS subordinates
FROM employees e JOIN employees m ON m.id = e.manager_id
GROUP BY m.position ORDER BY m.position;                   -- подчинённые,
сгруппированные по должности руководителя

SELECT e.name FROM employees e LEFT JOIN employees s ON s.manager_id = e.id
WHERE s.id IS NULL ORDER BY e.id LIMIT 3;                   -- сотрудники без
подчинённых
```

34.2. Задания

Рабочий каталог: `~/sql-lab/06_soedineniya`. Ответ каждого задания записывайте в файл `otvety/N`, где `N` — номер задания.

51. В базе `hr.db` выведите имя каждого сотрудника и имя его руководителя (соединение таблицы `employees` с самой собой под псевдонимами `e` и `m`, `LEFT JOIN`; у директора руководителя нет), по возрастанию `id` сотрудника. Результат — в `otvety/51`.
52. В `hr.db` то же самое, но обычным `JOIN` (`employees e JOIN employees m`): имя сотрудника и имя руководителя, первые десять по `id` сотрудника. Директор в результате не появится. Результат — в `otvety/52`.
53. В `hr.db` соедините `employees e` с `employees m` (руководитель) и выведите имя и должность сотрудников, у которых руководитель `m` имеет должность `директор`, по `id` сотрудника. Результат — в `otvety/53`.
54. В `hr.db` вычислите для каждого сотрудника с руководителем разницу `m.salary - e.salary` под именем `gap`. Выведите имя сотрудника и `gap` для пяти наибольших разниц; при равенстве — по `id` сотрудника. Результат — в `otvety/54`.
55. В `hr.db` соедините `employees e` с `employees m` (руководитель), сгруппируйте по руководителю (`GROUP BY m.id`) и выведите его имя и число подчинённых (`COUNT(*)` под именем `n`); сначала самые большие, при равенстве — по `id` руководителя. Результат — в `otvety/55`.
56. В `hr.db` соединив `employees e` с `employees m`, сгруппируйте по руководителю и условием на группу `HAVING COUNT(*) > 3` оставьте тех, у кого больше трёх подчинённых; выведите имена руководителей по `id`. Результат — в `otvety/56`.
57. В `hr.db` соедините `employees e` с `employees m` (руководитель) и выведите имя каждого сотрудника `e` с должностью `начальник отдела` и имя его руководителя `m`, по `id` сотрудника. Результат — в `otvety/57`.
58. В `hr.db` выведите пары коллег, у которых руководитель — директор (`manager_id = 1`): имя первого и имя второго, причём `id` первого меньше `id` второго (соединение по общему `manager_id`). Сортировка по `id` первого, затем второго; первые десять пар. Результат — в `otvety/58`.
59. В `hr.db` выведите имя сотрудника, имя его руководителя и имя руководителя руководителя (`employees e → m → b`), по `id` сотрудника, первые десять. Результат — в `otvety/59`.
60. В `hr.db` соедините `employees e` с `employees m` (руководитель), сгруппируйте по руководителю (`GROUP BY m.id`) и выведите его имя и среднюю зарплату его подчинённых `e`, округлённую до целых (`ROUND(AVG(e.salary), 0)`), по `id` руководителя. Результат — в `otvety/60`.

Раздел 35. Агрегаты по соединению

35.1. Теоретические сведения

Назначение. К результату соединения применяют `GROUP BY` и агрегатные функции. Так строятся отчёты: суммы по клиентам, по товарам, по отделам.

Синтаксис:

```
SELECT ключ, COUNT(...), SUM(...)
FROM a JOIN б ON ... [JOIN в ON ...]
[WHERE ...] GROUP BY ключ [HAVING ...] ORDER BY ... [LIMIT n];
```

Правила:

- после соединения `COUNT(*)` считает строки соединения, а не объекты главной таблицы: позиции, а не заказы;
- для верного подсчёта главных объектов используют `COUNT(DISTINCT a.id)`; суммы по столбцу главной таблицы после размножения строк завышаются;
- группировать нужно по ключу (`c.id`), а имя выводить как добавочный столбец: если имена могут совпасть, группировка по имени склеит разных людей;
- после `LEFT JOIN` считайте `COUNT(п.id)`, а не `COUNT(*)`; сумму пустой группы заменяйте через `COALESCE(SUM(…), 0)`;
- условие на группу пишут в `HAVING`, на строки — в `WHERE`; дробные результаты оборачивайте в `ROUND(…, 2)`;
- порядок обработки: `JOIN` и `ON` → `WHERE` → `GROUP BY` → `HAVING` → `ORDER BY` → `LIMIT`; для повторяемого порядка сортируйте и по уникальному ключу.

Пример:

```
SELECT p.category, ROUND(AVG(i.price), 0) AS avg_price
FROM order_items i JOIN products p ON p.id = i.product_id
GROUP BY p.category ORDER BY p.category;                    -- средняя цена продажи
по категориям

SELECT s.name, COUNT(p.id) AS n
FROM suppliers s LEFT JOIN products p ON p.supplier_id = s.id
GROUP BY s.id ORDER BY s.id;                                -- поставщики и число
их товаров (0 допустим)

SELECT o.status, COUNT(DISTINCT o.id) AS orders, COUNT(*) AS rows_joined
FROM orders o JOIN order_items i ON i.order_id = o.id
GROUP BY o.status HAVING orders > 5 ORDER BY o.status;      -- сравните два числа:
заказы и строки после JOIN
```

35.2. Задания

Рабочий каталог: `~/sql-lab/06_soedineniya`. Ответ каждого задания записывайте в файл `otvety/N`, где `N` — номер задания.

61. Сколько строк (позиций) в каждом из первых десяти заказов по `id`? Соедините `orders` с `order_items`, сгруппируйте по заказу (`GROUP BY o.id`) и выведите номер заказа и число (`COUNT(*)`). Результат — в `otvety/61`.

62. Соедините `order_items` с `products`, сгруппируйте по категории (`GROUP BY p.category`) и посчитайте, сколько штук продано (`SUM(i.qty)` под именем `pieces`). Выведите категорию и число, по алфавиту категории. Результат — в `otvety/62`.
63. В `hr.db` соедините `departments` с `employees`, сгруппируйте по отделу (`GROUP BY d.id`) и выведите название отдела и суммарную зарплату его сотрудников (`SUM(e.salary)` под именем `fund`), по `id` отдела. Результат — в `otvety/63`.
64. Сгруппируйте строки заказов по заказу (`orders JOIN order_items`, `GROUP BY o.id`), вычислите сумму каждого заказа (`SUM(i.qty * i.price)`) под именем `total` и выведите номер и сумму пяти самых крупных заказов; при равенстве — по возрастанию номера. Результат — в `otvety/64`.
65. Соедините `products` с `order_items`, сгруппируйте по товару (`GROUP BY p.id`) и выведите названия и суммарное проданное количество (`SUM(i.qty)` под именем `q`) трёх самых продаваемых товаров; при равенстве — по `id` товара. Результат — в `otvety/65`.
66. Соедините `orders` с `order_items`, сгруппируйте по статусу (`GROUP BY o.status`) и выведите статус, число заказов (`COUNT(DISTINCT o.id)` под именем `orders`) и число строк после соединения с `order_items` (`COUNT(*)` под именем `rows_joined`), по алфавиту статуса. Результат — в `otvety/66`.
67. Соедините `orders` с `order_items`, сгруппируйте по заказу и условием на группу `HAVING COUNT(*) > 2` оставьте заказы, в которых больше двух позиций; выведите номера по возрастанию, первые десять. Результат — в `otvety/67`.
68. Сгруппируйте выручку по месяцам заказа (`orders JOIN order_items`, `strftime('%Y-%m', o.order_date)` под именем `ym`, `SUM(qty * price)` под именем `total`, отменённые заказы не считать). Выведите первые шесть месяцев по возрастанию `ym`. Результат — в `otvety/68`.
69. Сгруппируйте выручку по городам клиентов (`customers → orders → order_items`, `GROUP BY c.city`; отменённые заказы не считать, клиентов с пустым городом отсеките в `WHERE`, `SUM(qty * price)` под именем `total`). Выведите три города с наибольшей выручкой; при равенстве — по алфавиту города. Результат — в `otvety/69`.
70. Соедините `orders` с `order_items` и вычислите среднюю сумму строки заказа `i.qty * i.price` по заказам **кроме отменённых** (`WHERE o.status <> 'cancelled'`), округлив до двух знаков (`ROUND(AVG(...), 2)`). Запишите одно число. Результат — в `otvety/70`.

Вопросы для размышления

1. Чем `JOIN` отличается от `LEFT JOIN`, и как по результату `LEFT JOIN` отличить «пары нет» от «значение пустое»?
2. Почему при соединении трёх таблиц легко получить завышенные суммы, и как этого избежать?
3. Чем отличается условие в `ON` от того же условия в `WHERE` при `LEFT JOIN`?

Тема 7. Подзапросы и операции над множествами

Общие сведения

Подзапрос — запрос, вложенный в другой запрос. Он может вернуть одно значение (тогда его используют вместо числа: `WHERE price > (SELECT AVG(price) FROM products)`), столбец значений (для `IN (SELECT ...)`) или таблицу (в предложении `FROM`). *Коррелированный* подзапрос ссылается на столбцы внешнего запроса и выполняется для каждой его строки; типичный пример — `EXISTS`, который проверяет, существует ли хотя бы одна подходящая строка (что именно выбирает подзапрос, неважно).

Выбор между подзапросом и соединением — вопрос читаемости и результата. Условие «есть хотя бы один» естественнее выражается через `EXISTS` или `IN`, потому что не размножает строки; условие «получить данные из другой таблицы» — через `JOIN`. Отдельное предостережение для `NOT IN`: если подзапрос возвращает хотя бы один `NULL`, выражение `x NOT IN (...)` никогда не бывает истинным. Поэтому для «нет ни одной связанной строки» надёжнее `NOT EXISTS` или `LEFT JOIN ... IS NULL`.

Операции над множествами объединяют результаты двух запросов с одинаковым числом и типом столбцов: `UNION` (объединение без повторов), `UNION ALL` (с повторами, быстрее), `INTERSECT` (общие строки), `EXCEPT` (строки первого, которых нет во втором). `ORDER BY` пишется один раз в конце всего составного запроса. Этими операциями удобно сравнивать списки: «у кого заказы были в 2025 году, но не в 2026».

Примеры.

```
SELECT name, price FROM products WHERE price > (SELECT AVG(price) FROM products)
ORDER BY price DESC;

SELECT name FROM customers
WHERE id IN (SELECT customer_id FROM orders WHERE status = 'cancelled') ORDER BY
name;

SELECT c.name FROM customers c
WHERE EXISTS (SELECT 1 FROM orders o WHERE o.customer_id = c.id AND o.order_date
>= '2026-01-01');

SELECT ROUND(AVG(t), 2) FROM (SELECT SUM(qty * price) AS t FROM order_items GROUP
BY order_id);

SELECT customer_id FROM orders WHERE order_date LIKE '2025%'
INTERSECT
SELECT customer_id FROM orders WHERE order_date LIKE '2026%';
```

Что чаще всего идёт не так

Коррелированный подзапрос в списке столбцов выполняется для каждой строки и на больших таблицах может быть очень медленным; такой запрос часто переписывают соединением с группировкой.

Рабочий каталог и проверка

Каталог полигона: `~/sql-lab/07_podzaprosy`. Проверка: `python3 proverka.py 7`.

Некоторые вопросы нельзя задать одним простым запросом: «товары дороже **среднего**», «клиенты, у которых **есть** отменённый заказ», «клиенты, которые покупали и в 2025, и в 2026 году». Для них запрос вкладывают в запрос: **подзапрос** возвращает значение или список, который используется во внешнем запросе (`> (SELECT ...)`, `IN (SELECT ...)`, `EXISTS (SELECT ...)`). Результаты нескольких запросов объединяют операторами множеств: `UNION` (объединение без повторов), `UNION ALL`, `INTERSECT` (пересечение), `EXCEPT` (разность). Тема состоит из семи конструкций, по десять заданий на каждую; базу данных задания не изменяют.

Раздел 36. Подзапрос, возвращающий одно значение

36.1. Теоретические сведения

Назначение. Подзапрос — это SELECT в круглых скобках внутри другого запроса. Если он возвращает ровно одно значение (одну строку и один столбец), его можно подставить туда, где ожидается число или строка, например в сравнение в WHERE.

Синтаксис:

```
SELECT ... FROM таблица WHERE столбец оператор (SELECT агрегат FROM таблица2
[WHERE ...]);
```

Правила:

- подзапрос всегда заключают в скобки; сначала вычисляется он, затем внешний запрос;
- для сравнения с одним значением подзапрос должен вернуть одну строку: обычно это агрегат (AVG, MAX, MIN, SUM, COUNT);
- если подзапрос вернёт несколько строк, SQLite молча возьмёт **первую** — ошибки не будет, а результат окажется случайным; ставьте агрегат или LIMIT 1;
- если подзапрос не вернул ни одной строки, его значение равно NULL, и сравнение с ним не выполняется ни для одной строки;
- такое сравнение заменяет два шага: узнать значение отдельным запросом и вписать его в основной запрос руками;
- внутри подзапроса работают WHERE, JOIN, выражения (AVG(qty * price)) и даже ORDER BY ... LIMIT 1: так берут значение из «первой» или «последней» строки ((SELECT customer_id FROM orders ORDER BY order_date DESC, id DESC LIMIT 1));
- подзапрос не зависит от строки внешнего запроса и вычисляется один раз; его можно сочетать с другими условиями через AND (category IN (...) AND price > (SELECT AVG(price) FROM products));
- даты вида ГГГГ-ММ-ДД — это текст, но сравниваются и дают MIN/MAX в правильном порядке.

Пример:

```
SELECT name, stock FROM products
WHERE stock < (SELECT AVG(stock) FROM products);        -- остаток ниже среднего

SELECT name, price FROM products
WHERE price = (SELECT MIN(price) FROM products);        -- все самые дешёвые
товары

SELECT name FROM customers
WHERE id = (SELECT customer_id FROM orders ORDER BY order_date DESC, id DESC LIMIT
1);   -- клиент последнего заказа
```

36.2. Задания

Рабочий каталог: `~/sql-lab/07_podzaprosy`. Ответ каждого задания записывайте в файл `otvety/N`, где `N` — номер задания.

1. Выведите название и цену товаров, цена которых выше средней по всем товарам (подзапрос в `WHERE`). Сначала самые дорогие. Результат — в `otvety/1`.
2. Выведите название и цену товаров дешевле средней цены, от самых дешёвых. Результат — в `otvety/2`.
3. Выведите названия товаров, цена которых равна наибольшей цене в таблице (подзапрос с `MAX`), по возрастанию `id`. Результат — в `otvety/3`.
4. Выведите название и цену самого дешёвого товара (`price = (SELECT MIN(...) ...)`; по `id`). Результат — в `otvety/4`.
5. Выведите название и остаток (`stock`) товаров, остаток которых больше среднего остатка. Сначала с наибольшим остатком, при равенстве — по `id`. Результат — в `otvety/5`.
6. Выведите `id` и дату заказов, дата которых равна самой ранней дате в таблице `orders`. По возрастанию `id`. Результат — в `otvety/6`.
7. Сколько товаров стоит дороже средней цены? Запишите число в `otvety/7`.
8. Выведите имя клиента, сделавшего самый поздний заказ (по дате; при равенстве дат — с наибольшим `id` заказа). Подзапрос должен вернуть одно значение: `customer_id` из заказа, найденного через `ORDER BY ... LIMIT 1`. Результат — в `otvety/8`.
9. Выведите по названию товары категорий `Электроника` и `Мебель` (`category IN (...)`), цена которых выше **средней цены по всем товарам**. Столбцы: название, цена. Результат — в `otvety/9`.
10. Выведите название товара и стоимость позиции (`qty * price` из `order_items`) под именем `s` для пяти позиций, стоимость которых больше средней стоимости позиции. Сначала самые дорогие; при равенстве — по `order_id`, затем по `product_id`. Соедините `order_items` с `products`. Результат — в `otvety/10`.

Раздел 37. Команда IN, NOT IN — значение в списке из подзапроса

37.1. Теоретические сведения

Назначение. Проверяет, входит ли значение в список, который выдал подзапрос с **одним столбцом**. Так находят строки, связанные с другой таблицей, не соединяя её.

Синтаксис:

```
... WHERE столбец IN (SELECT столбец2 FROM таблица2 [WHERE ...]);
... WHERE столбец NOT IN (SELECT столбец2 FROM таблица2 [WHERE ...]);
```

Правила:

- подзапрос должен возвращать один столбец, того же смысла, что и столбец слева;
- повторы в списке не мешают: значение либо входит в список, либо нет;
- **ловушка `NOT IN`:** если в списке есть хотя бы одно `NULL`, то `x NOT IN (...)` не будет истинным ни для одной строки, и результат окажется пустым;
- защита от ловушки: добавить в подзапрос `WHERE столбец2 IS NOT NULL` или использовать `NOT EXISTS`;
- для небольших списков значения можно писать прямо: `IN ('new', 'paid')`; `NOT IN ('Москва', 'Казань')` с `NULL` слева тоже не истинно: строки с `city IS NULL` в результат не попадут;
- в подзапросе работают `WHERE`, `GROUP BY` и `HAVING`: `IN (SELECT customer_id FROM orders GROUP BY customer_id HAVING COUNT(*) >= 5)`;
- подзапросы вкладываются друг в друга: результат внутреннего `IN (SELECT ...)` служит условием для среднего, и так по цепочке таблиц;
- условия `IN` и `NOT IN` можно соединять через `AND`/`OR`: клиенты, которые есть в одном списке, но не в другом.

Пример:

```
SELECT name FROM products
WHERE supplier_id IN (SELECT id FROM suppliers WHERE country = 'Китай');   --
товары китайских поставщиков

SELECT name FROM customers
WHERE id NOT IN (SELECT customer_id FROM orders WHERE status = 'paid');    -- ни
разу не платили

SELECT name FROM customers
WHERE id IN (SELECT customer_id FROM orders GROUP BY customer_id HAVING COUNT(*)
>= 5);   -- пять заказов и больше

SELECT name FROM products WHERE id IN
  (SELECT product_id FROM order_items WHERE order_id IN
     (SELECT id FROM orders WHERE status = 'cancelled'));                  -- два
     уровня вложенности
```

37.2. Задания

Рабочий каталог: `~/sql-lab/07_podzaprosy`. Ответ каждого задания записывайте в файл `otvety/N`, где `N` — номер задания.

11. Выведите по алфавиту имена клиентов, у которых есть хотя бы один заказ со статусом `cancelled` (`IN (SELECT ...)`). Результат — в `otvety/11`.
12. Выведите по алфавиту имена клиентов, у которых нет заказов, с помощью `NOT IN (SELECT ...)`. Результат — в `otvety/12`.
13. Выведите названия товаров, поставщики которых находятся в стране `Германия` (`supplier_id IN (SELECT ...)`). По возрастанию `id`. Результат — в `otvety/13`.
14. Выведите названия товаров, которые хотя бы раз заказывали в количестве 5 штук (`qty = 5` в `order_items`). По возрастанию `id`. Результат — в `otvety/14`.
15. Выведите по алфавиту имена клиентов, среди заказов которых нет ни одного со статусом `paid` (включая клиентов без заказов). Используйте `NOT IN`. Результат — в `otvety/15`.
16. Выведите по алфавиту имена клиентов, заказывавших товары категории `Мебель`. Вложите друг в друга три подзапроса `IN (SELECT ...)`: товары категории → позиции заказов → заказы → клиенты. Результат — в `otvety/16`.
17. Сколько клиентов живёт **не** в Москве и **не** в Казани, если считать только клиентов с указанным городом? Используйте `city NOT IN ('Москва', 'Казань')`. Запишите число в `otvety/17`.
18. Выведите по алфавиту имена клиентов, у которых есть заказы, но среди них нет ни одного со статусом `paid`. Условие — `id IN (...) AND id NOT IN (...)`. Результат — в `otvety/18`.
19. Сколько заказов имеют статус `new` или `paid`? Используйте `status IN ('new', 'paid')`. Запишите число в `otvety/19`.
20. Выведите по алфавиту имена клиентов, у которых пять и более заказов. Подзапрос — с `GROUP BY customer_id HAVING COUNT(*) >= 5`. Результат — в `otvety/20`.

Раздел 38. Команда EXISTS — есть ли связанная строка

38.1. Теоретические сведения

Назначение. EXISTS (подзапрос) истинно, если подзапрос вернул хотя бы одну строку. Обычно подзапрос **коррелированный**: он ссылается на столбцы внешнего запроса и вычисляется для каждой его строки.

Синтаксис:

```
SELECT ... FROM таблица1 а
WHERE [NOT] EXISTS (SELECT 1 FROM таблица2 б WHERE б.ключ = а.ключ [AND ...]);
```

Правила:

- что именно выбирает подзапрос (1, *, столбец), не имеет значения — важно лишь, есть ли строки;
- таблицам дают короткие псевдонимы (FROM customers c), чтобы различать внешние и внутренние столбцы: o.customer_id = c.id;
- NOT EXISTS выбирает строки, для которых связанных строк **нет**; в отличие от NOT IN, он не боится NULL;
- условие связи (б.ключ = а.ключ) нельзя забывать: без него подзапрос не зависит от внешней строки, и результат одинаков для всех строк;
- внутри подзапроса можно добавлять любые дополнительные условия: даты, статусы, суммы (LIKE '2025%', >= '2026-01-01');
- в подзапросе допустимы JOIN, а также GROUP BY ... HAVING: тогда EXISTS истинно, если нашлась хотя бы одна группа, прошедшая HAVING (EXISTS (SELECT 1 FROM orders o JOIN order_items i ON i.order_id = o.id WHERE o.customer_id = c.id GROUP BY o.id HAVING SUM(i.qty * i.price) > 30000));
- несколько условий связи с разными таблицами соединяют через AND: WHERE EXISTS (...) AND NOT EXISTS (...); в каждом подзапросе псевдоним o свой.

Пример:

```
SELECT s.name FROM suppliers s
WHERE EXISTS (SELECT 1 FROM products p
              WHERE p.supplier_id = s.id AND p.stock = 0);     -- у поставщика
              есть товар, которого нет на складе

SELECT p.name FROM products p
WHERE NOT EXISTS (SELECT 1 FROM order_items i WHERE i.product_id = p.id);   --
товар ни разу не продавался

SELECT c.name FROM customers c
WHERE EXISTS (SELECT 1 FROM orders o WHERE o.customer_id = c.id AND o.status =
'paid')
  AND NOT EXISTS (SELECT 1 FROM orders o WHERE o.customer_id = c.id AND o.status =
  'cancelled');   -- платили и ни разу не отменяли
```

38.2. Задания

Рабочий каталог: `~/sql-lab/07_podzaprosy`. Ответ каждого задания записывайте в файл `otvety/N`, где `N` — номер задания.

21. Выведите по алфавиту имена клиентов, у которых есть заказ с датой не раньше `2026-01-01` (коррелированный подзапрос `EXISTS`). Результат — в `otvety/21`.
22. Выведите по алфавиту названия поставщиков, у которых есть товар с нулевым остатком (`stock = 0`). Используйте `EXISTS`. Результат — в `otvety/22`.
23. Выведите по алфавиту имена клиентов, у которых нет ни одного заказа со статусом `shipped` (включая клиентов без заказов). Используйте `NOT EXISTS`. Результат — в `otvety/23`.
24. Выведите названия товаров, которые попадали хотя бы в один отменённый заказ (`status = 'cancelled'`). В подзапросе `EXISTS` соедините `order_items` и `orders`. По возрастанию `id`. Результат — в `otvety/24`.
25. Выведите по алфавиту имена VIP-клиентов (`vip = 1`), у которых есть хотя бы один заказ (условие `vip = 1` и `EXISTS` через `AND`). Результат — в `otvety/25`.
26. Выведите `id` первых десяти заказов (по возрастанию), в которых есть позиция с `qty = 5` (`EXISTS` по `order_items`, затем `ORDER BY ... LIMIT 10`). Результат — в `otvety/26`.
27. Выведите по алфавиту имена клиентов, у которых есть и заказ со статусом `paid`, и заказ со статусом `cancelled`. Используйте два `EXISTS`, соединённых `AND`. Результат — в `otvety/27`.
28. Выведите по алфавиту имена клиентов, у которых есть заказ в 2025 году (`order_date LIKE '2025%'`), но нет ни одного отменённого заказа: `EXISTS (...) AND NOT EXISTS (...)`. Результат — в `otvety/28`.
29. Выведите по алфавиту названия поставщиков, у которых нет ни одного товара дороже 9500 (`NOT EXISTS` по `products`). Результат — в `otvety/29`.
30. Выведите по алфавиту имена клиентов, у которых есть хотя бы один заказ на сумму больше 30000 (сумма заказа — `SUM(qty * price)`). В подзапросе `EXISTS` используйте `GROUP BY` и `HAVING`. Результат — в `otvety/30`.

Раздел 39. Подзапрос в списке столбцов

39.1. Теоретические сведения

Назначение. Подзапрос в SELECT вычисляет **отдельное значение для каждой строки** внешнего запроса и выводится как обычный столбец. Так добавляют к строке сведения из другой таблицы без JOIN и GROUP BY.

Синтаксис:

```
SELECT столбцы, (SELECT агрегат FROM таблица2 б WHERE б.ключ = а.ключ) AS
псевдоним
FROM таблица1 а;
```

Правила:

- подзапрос обязан вернуть одно значение, поэтому в нём стоит агрегат;
- связь с внешней строкой делается через псевдоним внешней таблицы (а.ключ);
- агрегат без строк даёт NULL (SUM, MAX; только COUNT даёт 0): если нужен ноль, оберните в COALESCE(..., 0);
- столбцу из подзапроса обязательно давайте имя через AS, иначе в заголовке окажется текст всего подзапроса;
- на больших таблицах такой подзапрос выполняется для каждой строки; для отчётов по всей таблице обычно быстрее JOIN с GROUP BY;
- в одном SELECT может быть сколько угодно подзапросов, у каждого свой AS; внутри подзапроса допустимы JOIN и выражения (SUM(i.qty * i.price));
- значение-текст берут через ORDER BY ... LIMIT 1: (SELECT p.name FROM products p WHERE p.supplier_id = s.id ORDER BY p.price DESC LIMIT 1);
- подзапрос, не связанный с внешней строкой, тоже допустим и участвует в выражении: price - (SELECT AVG(price) FROM products);
- в ORDER BY внешнего запроса можно использовать имя, данное подзапросу через AS (ORDER BY n DESC).

Пример:

```
SELECT c.name,
       (SELECT COUNT(*) FROM orders o WHERE o.customer_id = c.id) AS n_orders
FROM customers c ORDER BY n_orders DESC, c.id LIMIT 5;

SELECT s.name,
       (SELECT COALESCE(MAX(p.price), 0) FROM products p WHERE p.supplier_id =
       s.id) AS max_price
FROM suppliers s;

SELECT name, ROUND(price - (SELECT AVG(price) FROM products), 1) AS diff
FROM products ORDER BY price DESC LIMIT 3;

SELECT s.name,
       (SELECT p.name FROM products p WHERE p.supplier_id = s.id ORDER BY p.price
       DESC LIMIT 1) AS top_item
FROM suppliers s;
```

39.2. Задания

Рабочий каталог: `~/sql-lab/07_podzaprosy`. Ответ каждого задания записывайте в файл `otvety/N`, где `N` — номер задания.

31. Выведите название товара и суммарно проданное количество (`SUM(qty)` из `order_items`, а если продаж не было — 0) под именем `q`, для пяти товаров с наибольшим `q`; при равенстве — по `id`. Используйте подзапрос в списке столбцов. Результат — в `otvety/31`.
32. Выведите имя клиента и число его заказов под именем `n` для пяти клиентов с наибольшим `n`; при равенстве — по `id` клиента. Результат — в `otvety/32`.
33. Выведите название поставщика и наибольшую цену его товара под именем `mx` (0, если товаров нет). Все поставщики, по возрастанию `id`. Результат — в `otvety/33`.
34. Выведите имя клиента и дату его последнего заказа под именем `last_date`. Покажите пять клиентов с самой поздней датой; при равенстве — по `id` клиента. Результат — в `otvety/34`.
35. Выведите имя клиента и сумму всех его заказов (`SUM(qty * price)`; 0, если заказов нет) под именем `total` для трёх клиентов с наибольшей суммой; при равенстве — по `id` клиента. Результат — в `otvety/35`.
36. Выведите название товара и разницу между его ценой и средней ценой всех товаров, округлённую до одного знака, под именем `diff` (подзапрос со средним внутри выражения в списке столбцов). Покажите пять самых дорогих товаров (по `diff`, по убыванию). Результат — в `otvety/36`.
37. Выведите название поставщика, число его товаров (`n`) и суммарный остаток на складе (`total_stock`, 0 если товаров нет). Два подзапроса в списке столбцов; все поставщики, по возрастанию `id`. Результат — в `otvety/37`.
38. Для заказов с `id` от 1 до 10 выведите `id`, число позиций (`positions`) и сумму заказа `SUM(qty * price)` (`total`). По возрастанию `id`. Результат — в `otvety/38`.
39. Выведите название поставщика и название самого дорогого его товара (`top_item`). Подзапрос — с `ORDER BY price DESC LIMIT 1`. Все поставщики по возрастанию `id`. Результат — в `otvety/39`.
40. Выведите имя клиента, число его заказов `n` и долю его заказов в общем числе заказов магазина в процентах `pct` (до одного знака) для пяти клиентов с наибольшим `n`; при равенстве — по `id` клиента. Три подзапроса в списке столбцов: `n`, и `pct` как `ROUND(100.0 * (число заказов клиента) / (SELECT COUNT(*) FROM orders), 1)`. Результат — в `otvety/40`.

Раздел 40. Подзапрос в FROM

40.1. Теоретические сведения

Назначение. Результат запроса можно использовать как временную таблицу: `FROM (SELECT ...)`. Это нужно, когда по результату группировки надо посчитать что-то ещё, например среднее от сумм.

Синтаксис:

```
SELECT ... FROM (SELECT ... FROM таблица GROUP BY ...) [AS псевдоним];
```

Правила:

- внутренний запрос выполняется первым, внешний работает с его столбцами так, как будто это таблица;
- вычисляемым столбцам внутреннего запроса давайте имена через `AS`: по этим именам к ним обращается внешний запрос;
- псевдоним самой «таблицы» в SQLite необязателен, но с ним запрос читается легче, особенно в `JOIN`;
- нельзя вложить агрегат в агрегат в одном запросе (`AVG(SUM(x))` — ошибка): именно для этого и нужен вложенный запрос;
- условия по вычисленному столбцу (`WHERE t > 20000`) и `ORDER BY ... LIMIT` пишутся во внешнем запросе — после внутреннего `GROUP BY`; во внешнем запросе можно снова группировать (`GROUP BY n`): так считают «группы групп»;
- такую «таблицу» можно соединять с обычной: `FROM таблица1 а JOIN (SELECT ключ, агрегат AS имя FROM ... GROUP BY ключ) б ON б.ключ = а.ключ;` в `ON` и в `SELECT` пишут `б.имя`;
- если вложенность растёт, переходите к `WITH` (тема 8) — тот же приём, но читается сверху вниз.

Пример:

```
SELECT MAX(n) FROM (SELECT COUNT(*) AS n FROM orders GROUP BY customer_id);   --
наибольшее число заказов у одного клиента

SELECT p.name, p.category, p.price
FROM products p
JOIN (SELECT category, AVG(price) AS ap FROM products GROUP BY category) a ON
a.category = p.category
WHERE p.price > a.ap;                                    -- дороже средней цены
своей категории

SELECT n, COUNT(*) AS clients
FROM (SELECT customer_id, COUNT(*) AS n FROM orders GROUP BY customer_id)
GROUP BY n ORDER BY n;                                   -- сколько клиентов
сделали n заказов

SELECT ROUND(AVG(cnt), 1) AS avg_items
FROM (SELECT order_id, COUNT(*) AS cnt FROM order_items GROUP BY order_id) AS t;
```

40.2. Задания

Рабочий каталог: `~/sql-lab/07_podzaprosy`. Ответ каждого задания записывайте в файл `otvety/N`, где `N` — номер задания.

41. Вычислите средний чек: среднее по заказам значение суммы заказа (`SUM(qty * price)` по `order_items`, сгруппированной по `order_id`), округлённое до двух знаков. Используйте подзапрос в `FROM`. Запишите число в `otvety/41`.
42. Какое наибольшее число заказов у одного клиента? Сначала посчитайте заказы по клиентам во вложенном запросе, затем возьмите `MAX`. Запишите число в `otvety/42`.
43. Сколько в среднем позиций (строк `order_items`) в одном заказе? Округлите до двух знаков. Запишите число в `otvety/43`.
44. Какова наибольшая сумма одного заказа (`SUM(qty * price)` по заказу)? Запишите число в `otvety/44`.
45. Сколько клиентов сделали больше трёх заказов? Сгруппируйте заказы по клиентам во вложенном запросе (`HAVING COUNT(*) > 3`) и посчитайте строки. Запишите число в `otvety/45`.
46. Для каждого статуса заказа выведите статус и средний чек (среднюю сумму заказа), округлённый до двух знаков, под именем `avg_total`. По алфавиту статусов. Результат — в `otvety/46`.
47. Среди заказов, сумма которых больше 20000, выведите `order_id` и сумму `t` для пяти самых дешёвых. При равенстве — по `order_id`. Сумму посчитайте вложенным запросом. Результат — в `otvety/47`.
48. Выведите имя клиента и его средний чек (среднюю сумму его заказов, округлённую до двух знаков) под именем `avg_check` для трёх клиентов с наибольшим средним чеком; при равенстве — по `id` клиента. Суммы заказов посчитайте подзапросом в `JOIN`. Результат — в `otvety/48`.
49. Выведите название, категорию и цену товаров, которые дороже средней цены **своей категории**. Среднюю цену по категориям посчитайте подзапросом в `FROM` и соедините с `products`. По возрастанию `id`. Результат — в `otvety/49`.
50. Выведите, сколько клиентов сделали ровно 1 заказ, ровно 2 и т. д.: столбцы `n` (число заказов) и `clients` (число таких клиентов). Внутри `FROM (...)` — заказы по клиентам, снаружи — `GROUP BY n`. По возрастанию `n`. Результат — в `otvety/50`.

Раздел 41. Команда UNION, UNION ALL — объединение результатов

41.1. Теоретические сведения

Назначение. Объединяет результаты двух и более запросов в один список: `UNION` — все строки без повторов, `UNION ALL` — все строки с повторами.

Синтаксис:

```
SELECT ... FROM ... [WHERE ...]
UNION [ALL]
SELECT ... FROM ... [WHERE ...]
[ORDER BY ...];
```

Правила:

- оба запроса должны возвращать **одинаковое число столбцов** с совместимым смыслом; имена столбцов берутся из первого запроса;
- `ORDER BY` пишется один раз, **в самом конце** всего составного запроса, и ссылается на имена столбцов первого запроса;
- `UNION` сравнивает строки целиком; `NULL` при таком сравнении считаются равными друг другу;
- `LIMIT` внутри части запроса не пишется: оберните часть в подзапрос `SELECT * FROM (SELECT ... ORDER BY ... LIMIT ...)`;
- `UNION ALL` нужен, когда повторы важны (например, при подсчёте) или когда части заведомо не пересекаются;
- частей может быть больше двух: `SELECT ... UNION SELECT ... UNION SELECT ...`; в частях допустимы `WHERE`, `GROUP BY`, `JOIN` и подзапросы;
- чтобы пометить, откуда строка, добавьте в каждую часть константу: в первой `'клиент' AS kind`, во второй достаточно `'поставщик'`;
- в `ORDER BY` нельзя писать `c.name`: дайте столбцу имя (`SELECT c.name AS name ...`) и сортируйте по `name`;
- чтобы получить итоговую строку, добавьте часть с агрегатом и константой: `SELECT 'ИТОГО', COUNT(*) FROM orders`; итоговая строка встаёт в общий список, а порядок задаёт `ORDER BY` в конце.

Операции:

Элемент	Описание
`UNION`	объединение: строки любого из запросов, повторы убираются
`UNION ALL`	объединение с сохранением повторов; работает быстрее, так как не сравнивает строки

Пример:

```
SELECT name FROM customers WHERE city = 'Омск'
UNION
SELECT name FROM customers WHERE vip = 1
```

```
ORDER BY name;                                      -- омичи или VIP, без повторов

SELECT 'клиент' AS kind, COUNT(*) AS n FROM customers
UNION ALL
SELECT 'товар', COUNT(*) FROM products;             -- две строки с итогами из разных
таблиц

SELECT * FROM (SELECT name, price FROM products ORDER BY price LIMIT 2)
UNION ALL
SELECT * FROM (SELECT name, price FROM products ORDER BY price DESC LIMIT 2)
ORDER BY price;                                     -- два самых дешёвых и два самых
дорогих
```

41.2. Задания

Рабочий каталог: `~/sql-lab/07_podzaprosy`. Ответ каждого задания записывайте в файл `otvety/N`, где `N` — номер задания.

51. Выведите без повторов названия товаров, которые стоят дороже 9000 **или** отсутствуют на складе (`stock = 0`), объединив два запроса через `UNION`. Порядок — по названию. Результат — в `otvety/51`.
52. Выведите единый список: имена VIP-клиентов (`vip = 1`) с пометкой `клиент` и названия всех поставщиков с пометкой `поставщик`. Столбцы: `name`, `kind`. Объедините через `UNION ALL`; порядок — по `kind`, затем по `name`. Результат — в `otvety/52`.
53. Сколько различных клиентов имеют заказ со статусом `paid` **или** со статусом `shipped`? Объедините два запроса через `UNION` во вложенном запросе и посчитайте строки. Запишите число в `otvety/53`.
54. Выведите название и цену трёх самых дешёвых и трёх самых дорогих товаров одним списком, по возрастанию цены. `LIMIT` в части запроса нельзя — оберните каждую часть в `SELECT * FROM (...)`. Результат — в `otvety/54`.
55. Выведите по названию без повторов товары, которые относятся к категории `Книги`, **или** к категории `Мебель`, **или** стоят меньше 1000. Три запроса через `UNION`. Результат — в `otvety/55`.
56. Выведите число заказов по каждому статусу (`status`, `n`) и ещё одну строку `ИТОГО` с общим числом заказов (часть с константой `'ИТОГО'` и `COUNT(*)`). Объедините через `UNION ALL`; порядок — по `n` по убыванию, затем по `status`. Результат — в `otvety/56`.
57. Выведите три строки: `мин`, `макс`, `средняя` и соответствующие значения цены (`what`, `v`); среднюю цену округлите до целого. Объедините три запроса через `UNION ALL`; порядок — по `v`. Результат — в `otvety/57`.
58. Выведите по алфавиту без повторов имена клиентов, у которых есть отменённый заказ **или** заказ с датой раньше `2025-03-01`. В каждой части запроса соедините `customers` и `orders`. Результат — в `otvety/58`.
59. Выведите название и цену товаров, которые отсутствуют на складе, **или** стоят дороже 9000, без повторов. Порядок — по цене по убыванию, затем по названию. Результат — в `otvety/59`.
60. Выведите по возрастанию `id` клиентов, которые сделали либо самый ранний, либо самый поздний заказ магазина (по дате). Подзапросы `MIN` и `MAX` — внутри двух частей `UNION`. Результат — в `otvety/60`.

Раздел 42. Команда INTERSECT, EXCEPT — пересечение и разность

42.1. Теоретические сведения

Назначение. INTERSECT оставляет только строки, которые есть и в первом, и во втором результате; EXCEPT — строки первого результата, которых нет во втором.

Синтаксис:

```
SELECT ... FROM ... [WHERE ...]
{INTERSECT | EXCEPT}
SELECT ... FROM ... [WHERE ...]
[ORDER BY ...];
```

Правила:

- число и смысл столбцов в обоих запросах совпадают, как и в UNION;
- оба оператора убирают повторы и сравнивают строки целиком;
- A EXCEPT B и B EXCEPT A — разные результаты;
- операции выполняются слева направо без приоритетов; для другого порядка используйте подзапрос в FROM;
- INTERSECT и EXCEPT часто можно заменить на IN и NOT IN (или EXISTS), но запись через множества читается проще;
- части могут содержать WHERE, JOIN и подзапросы; сравниваются значения в столбцах, поэтому для «клиентов» выбирайте в обеих частях один и тот же столбец (customer_id или id);
- ORDER BY пишется один раз в конце и использует имена первого запроса (ORDER BY id); для JOIN дайте столбцу имя: SELECT c.name AS name ...;
- результат множественных операций можно сосчитать: SELECT COUNT(*) FROM (... EXCEPT ...);
- цепочка A UNION B EXCEPT C выполняется слева направо: сначала (A UNION B), затем вычитается C.

Операции:

Элемент	Описание
INTERSECT	пересечение: строки, которые есть и в первом, и во втором результате
EXCEPT	разность: строки первого результата, которых нет во втором; порядок запросов важен

Пример:

```
SELECT product_id FROM order_items WHERE qty >= 5
EXCEPT
SELECT id FROM products WHERE stock = 0;         -- заказывали помногу, но товар
на складе есть
```

```
SELECT id FROM customers WHERE vip = 1
INTERSECT
SELECT customer_id FROM orders;                  -- VIP-клиенты, у которых есть
заказы

SELECT customer_id FROM orders WHERE status = 'new'
UNION
SELECT customer_id FROM orders WHERE status = 'paid'
EXCEPT
SELECT customer_id FROM orders WHERE status = 'cancelled'
ORDER BY customer_id;                            -- new или paid, но без cancelled
```

42.2. Задания

Рабочий каталог: `~/sql-lab/07_podzaprosy`. Ответ каждого задания записывайте в файл `otvety/N`, где `N` — номер задания.

61. Выведите `id` клиентов, у которых есть заказы **и** в 2025, **и** в 2026 году, пересечением двух запросов (`INTERSECT`). Порядок — по возрастанию. Результат — в `otvety/61`.
62. Выведите `id` клиентов, которых нет среди клиентов с заказами, с помощью `EXCEPT`: из всех `id` клиентов вычтите `customer_id` из заказов. Порядок — по возрастанию. Результат — в `otvety/62`.
63. Выведите `id` клиентов, у которых есть и заказ `paid`, и заказ `cancelled`, с помощью `INTERSECT`. По возрастанию. Результат — в `otvety/63`.
64. Выведите `id` клиентов, у которых есть заказы, но нет ни одного отменённого. Вычтите (`EXCEPT`) клиентов с отменой из всех клиентов с заказами. По возрастанию. Результат — в `otvety/64`.
65. Выведите `id` оплаченных заказов (`status = 'paid'`), в которых нет ни одной позиции с `qty = 5`. Используйте `EXCEPT`. По возрастанию. Результат — в `otvety/65`.
66. Выведите по алфавиту имена клиентов, которые заказывали и товары категории `Электроника`, и товары категории `Книги`. Два запроса с `JOIN` и `INTERSECT`. Результат — в `otvety/66`.
67. Сколько клиентов не сделали ни одного заказа в 2026 году (включая тех, у кого заказов нет вовсе)? Вычтите через `EXCEPT` из всех `id` клиентов `customer_id` заказов 2026 года, а строки посчитайте внешним запросом. Запишите число в `otvety/67`.
68. Выведите по названию товары, цена которых выше средней **и** остаток больше нуля. Используйте два запроса и `INTERSECT`; в первом — подзапрос со средней ценой. Результат — в `otvety/68`.
69. Выведите по названию товары, которые заказывали и VIP-клиенты (`vip = 1`), и не-VIP клиенты (`vip = 0`). Два запроса с `JOIN` по четырём таблицам и `INTERSECT`. Результат — в `otvety/69`.
70. Выведите `id` клиентов, у которых есть заказ `new` или `paid`, но нет ни одного `cancelled`. Три запроса в одной цепочке: `new UNION paid EXCEPT cancelled` (операции выполняются слева направо). По возрастанию. Результат — в `otvety/70`.

Вопросы для размышления

1. Когда подзапрос можно заменить соединением, а когда это изменит результат?
2. Почему `NOT IN` опасен при наличии `NULL` в списке, и чем в этом случае лучше

```
NOT EXISTS?
```

Тема 8. CTE, рекурсия и оконные функции

Общие сведения

Когда запрос состоит из нескольких логических шагов, вложенные подзапросы становятся нечитаемыми. **Общее табличное выражение** (CTE, `WITH имя AS (запрос)`) даёт промежуточному результату имя, и дальше его используют как таблицу, в том числе несколько раз. Запрос читается сверху вниз: сначала «суммы заказов», затем «средний чек», затем «заказы выше среднего».

Рекурсивное выражение (`WITH RECURSIVE`) состоит из начальной части, `UNION ALL` и рекурсивной части, которая ссылается на само выражение. Так строят последовательности чисел и дат и обходят иерархии (сотрудник — руководитель, каталог — подкаталоги). Рекурсию обязательно ограничивают условием, иначе запрос не завершится.

Оконные функции вычисляют значение для каждой строки с учётом «окна» — связанных с ней строк, но не сворачивают строки в группы. Запись: `функция(...) OVER (PARTITION BY ... ORDER BY ...)`. `PARTITION BY` делит строки на независимые окна, `ORDER BY` задаёт порядок внутри окна. Часто применяют: `ROW_NUMBER()` (номер строки), `RANK()` (место с учётом равных значений), `LAG`/`LEAD` (значение из предыдущей/следующей строки), `SUM(...) OVER (ORDER BY ...)` (нарастающий итог), `SUM(...) OVER ()` (итог по всей таблице, удобно для долей). Оконную функцию нельзя поставить в `WHERE`: её сначала вычисляют во вложенном запросе или CTE, а потом отбирают, например, `rn = 1`.

Примеры.

```
WITH t AS (SELECT order_id, SUM(qty * price) AS total FROM order_items GROUP BY
order_id)
SELECT ROUND(AVG(total), 2), MAX(total) FROM t;

SELECT id, name, price, ROW_NUMBER() OVER (ORDER BY price DESC, id) AS rn FROM
products LIMIT 5;

SELECT category, name, price, RANK() OVER (PARTITION BY category ORDER BY price
DESC) AS r FROM products;

SELECT id, customer_id, order_date,
       LAG(order_date) OVER (PARTITION BY customer_id ORDER BY order_date, id) AS
       prev FROM orders;

WITH RECURSIVE n(x) AS (SELECT 1 UNION ALL SELECT x + 1 FROM n WHERE x < 100)
SELECT SUM(x) FROM n;
```

Что чаще всего идёт не так

Путают `RANK` и `ROW_NUMBER`: при равных значениях `RANK` даёт одинаковое место и «пропускает» следующие, `ROW_NUMBER` всегда нумерует подряд. Для «ровно один лидер на группу» нужен `ROW_NUMBER`.

Рабочий каталог и проверка

Каталог полигона: `~/sql-lab/08_cte_okna`. Проверка: `python3 proverka.py 8`.

Сложные отчёты быстро превращаются в вложенные друг в друга подзапросы, которые невозможно читать. Два средства делают их понятными. **Общее табличное выражение** (`WITH имя AS (...)`, CTE) — это именованный временный результат, который можно использовать ниже, как таблицу; рекурсивное `WITH RECURSIVE` умеет обходить деревья (руководитель — подчинённый) и строить последовательности. **Оконные функции** (`ROW_NUMBER`, `RANK`, `LAG`, `SUM() OVER (...)`) считают значение для каждой строки, «глядя» на соседние, и при этом не сворачивают строки в группы. Они нужны для рейтингов, нарастающих итогов и сравнения со строкой выше. В теме семь конструкций, по десять заданий на каждую. Базы: `shop.db`, `hr.db`.

Раздел 43. Команда WITH — общее табличное выражение (CTE)

43.1. Теоретические сведения

Назначение. WITH имя AS (запрос) создаёт именованный временный результат, на который можно ссылаться в основном запросе как на таблицу. Сложный отчёт разбивается на понятные шаги вместо вложенных подзапросов.

Синтаксис:

```
WITH имя1 [(столбец1, ...)] AS (SELECT ...),
     имя2 AS (SELECT ... FROM имя1 ...)
SELECT ... FROM имя2 ...;
```

Правила:

- CTE существует только в пределах одного запроса, в базе ничего не остаётся;
- несколько CTE перечисляют через запятую, слово WITH пишут один раз; последующие могут использовать предыдущие;
- на CTE можно ссылаться несколько раз: в FROM, в JOIN и в подзапросах — это удобно для сравнения «строка и среднее по всем»;
- имена столбцов CTE берутся из внутреннего SELECT (поэтому вычисляемым столбцам нужен AS) или из списка в скобках после имени;
- после CTE сразу идёт основной запрос; запятая или точка с запятой между ними — ошибка;
- CTE, дающее одно число (среднее, максимум), используют двумя способами: подзапросом WHERE total > (SELECT AVG(total) FROM имя) или соединением через запятую FROM t, m (каждая строка t получает единственную строку m: WHERE t.total > m.a);
- CTE годится и для сравнения с максимумом: WHERE n = (SELECT MAX(n) FROM имя) выберет все строки с наибольшим n, включая ничьи;
- внутри CTE допустимы JOIN, GROUP BY, DISTINCT, функции (SUBSTR(order_date, 1, 7) — месяц); ORDER BY и LIMIT — в основном запросе.

Пример:

```
WITH cheap AS (
    SELECT category, price FROM products WHERE price < 1000
)
SELECT category, COUNT(*) FROM cheap GROUP BY category;   -- сколько дешёвых
товаров в каждой категории

WITH t AS (SELECT order_id, SUM(qty * price) AS total FROM order_items GROUP BY
order_id),
     m AS (SELECT AVG(total) AS a FROM t)                 -- второе CTE использует
     первое
SELECT COUNT(*) FROM t, m WHERE t.total > m.a;            -- заказов выше средней
суммы

WITH s(cat, v) AS (SELECT category, SUM(stock) FROM products GROUP BY category)
-- имена столбцов в скобках
SELECT cat, v FROM s WHERE v = (SELECT MAX(v) FROM s);
```

43.2. Задания

Рабочий каталог: `~/sql-lab/08_cte_okna`. Ответ каждого задания записывайте в файл `otvety/N`, где `N` — номер задания.

1. С помощью CTE `t` (сумма `qty * price` по каждому заказу) выведите одной строкой средний (округлите до двух знаков) и наибольший размер заказа. Результат — в `otvety/1`.
2. Составьте CTE `rev` с выручкой по клиентам (все заказы, `SUM(qty * price)`). Выведите имена клиентов, чья выручка выше средней по `rev`, и выручку, от большей к меньшей; при равенстве — по `id` клиента. Результат — в `otvety/2`.
3. С помощью CTE `cnt` (число заказов каждого клиента) посчитайте, сколько клиентов сделали больше трёх заказов. Запишите число в `otvety/3`.
4. Дайте столбцам CTE имена в скобках: `s(cat, v)` — суммарный остаток `stock` по категориям. Выведите `cat` и `v` от большего остатка к меньшему; при равенстве — по `cat`. Результат — в `otvety/4`.
5. Составьте два CTE через запятую: `t` — сумма каждого заказа, `m` — средняя сумма заказа (`AVG(total) AS a`). Посчитайте заказы, чья сумма выше средней (соедините `FROM t, m` и отберите `t.total > m.a`). Запишите число в `otvety/5`.
6. Составьте CTE `a` со средней ценой товаров каждого поставщика. Выведите `id`, название и цену товаров, цена которых выше средней по их поставщику; по `id`. Результат — в `otvety/6`.
7. Составьте CTE `m` с числом заказов по месяцам (`substr(order_date, 1, 7)` под именем `ym`, число — `n`). Выведите месяцы с наибольшим числом заказов (`n = (SELECT MAX(n) FROM m)`) и это число; по возрастанию месяца. Результат — в `otvety/7`.
8. Составьте CTE `c` с числом отменённых заказов (`status = 'cancelled'`) каждого клиента (`k`). Выведите имена трёх клиентов с наибольшим `k` и само `k`; при равенстве — по `id` клиента. Результат — в `otvety/8`.
9. Составьте CTE `has` из различных `customer_id` таблицы `orders`. Выведите имена клиентов, которых в `has` нет, по `id`. Результат — в `otvety/9`.
10. Составьте CTE `rev` с выручкой по клиентам. Для каждого города (клиенты без города не учитываются) выведите город, среднюю выручку клиента (до двух знаков) и число клиентов в `rev`; по городу. Результат — в `otvety/10`.

Раздел 44. Команда `WITH RECURSIVE` — рекурсивные запросы

44.1. Теоретические сведения

Назначение. Рекурсивное CTE повторяет шаг, пока он даёт новые строки. Им строят последовательности чисел и дат и обходят деревья: руководитель — подчинённый, категория — подкатегория.

Синтаксис:

```
WITH RECURSIVE имя(столбцы) AS (
    SELECT ...                                          -- стартовая часть (якорь)
    UNION ALL
    SELECT ... FROM имя ... WHERE условие_остановки   -- рекурсивная часть
)
SELECT ... FROM имя;
```

Правила:

- стартовая часть выполняется один раз, рекурсивная — снова и снова над строками, полученными на предыдущем шаге;
- рекурсивная часть обращается к самому CTE по имени в своём `FROM` (в подзапросе нельзя);
- рекурсия заканчивается, когда шаг не вернул ни одной строки; поэтому нужно **условие остановки** (`WHERE x < 100`) или естественный конец дерева;
- если в данных есть цикл (А — начальник Б, Б — начальник А), а остановки нет, запрос не закончится; `UNION` вместо `UNION ALL` отбрасывает уже встречавшиеся строки и помогает от зацикливания;
- для обхода дерева вниз в стартовой части берут корень (`manager_id IS NULL`), в рекурсивной соединяют таблицу с найденными строками (`e.manager_id = имя.id`) и увеличивают счётчик уровня; для подъёма вверх условие обратное (`e.id = имя.mgr`);
- стартовая часть может быть константой (`SELECT 1`, `SELECT 3`, `SELECT '2026-01-01'`) или строкой таблицы (`WHERE id = 22`, `WHERE manager_id = 2`); число и порядок столбцов в обеих частях одинаковы, имена берутся из списка после имени CTE;
- в рекурсии можно хранить несколько столбцов, например пару соседних чисел и номер шага: `SELECT b, a + b, n + 1 FROM fib WHERE n < 15` (старое `b` становится новым `a`);
- даты строят шагом `date(x, '+1 day')` с остановкой `WHERE x < '2026-01-31'`; после рекурсии по полученной таблице можно считать как обычно (`COUNT`, `SUM`, `GROUP BY lvl`) и соединять её с исходными таблицами (`JOIN employees e ON e.id = t.id`).

Пример:

```
WITH RECURSIVE sq(n, v) AS (
    SELECT 1, 1
    UNION ALL
    SELECT n + 1, (n + 1) * (n + 1) FROM sq WHERE n < 10
)
```

```
SELECT n, v FROM sq;                                        -- квадраты чисел от 1 до
10

WITH RECURSIVE t(id, lvl, path) AS (
    SELECT id, 0, name FROM employees WHERE manager_id IS NULL
    UNION ALL
    SELECT e.id, t.lvl + 1, t.path || ' > ' || e.name
    FROM employees e JOIN t ON e.manager_id = t.id
)
SELECT lvl, path FROM t ORDER BY lvl, id;                   -- путь от директора до
каждого сотрудника

WITH RECURSIVE d(x) AS (
    SELECT '2026-03-01'
    UNION ALL
    SELECT date(x, '+1 day') FROM d WHERE x < '2026-03-07'
)
SELECT x, strftime('%w', x) FROM d;                         -- даты недели и номера
дней недели

WITH RECURSIVE up(id, mgr, depth) AS (
    SELECT id, manager_id, 0 FROM employees WHERE id = 22
    UNION ALL
    SELECT e.id, e.manager_id, up.depth + 1 FROM employees e JOIN up ON e.id =
    up.mgr
)
SELECT id, depth FROM up;                                   -- цепочка начальников
вверх до директора
```

44.2. Задания

Рабочий каталог: `~/sql-lab/08_cte_okna`. Ответ каждого задания записывайте в файл `otvety/N`, где `N` — номер задания.

11. С помощью `WITH RECURSIVE` постройте числа от 1 до 100 и вычислите их сумму. Запишите число в `otvety/11`.
12. Рекурсивно постройте таблицу `p(n, v)`: `n` от 0 до 20, `v = 2^n` (старт — `0, 1`, шаг — `n + 1, v * 2`). Выведите все строки по `n`. Результат — в `otvety/12`.
13. Вычислите рекурсией факториал числа 10 (произведение чисел от 1 до 10). Запишите число в `otvety/13`.
14. Рекурсивно вычислите 15-е число Фибоначчи, считая последовательность 0, 1, 1, 2, 3, … (первое число — 0). Храните в CTE три столбца: пару соседних чисел `a, b` и номер `n` (старт — `0, 1, 1`; шаг — `b, a + b, n + 1`, остановка — `n < 15`). Запишите число в `otvety/14`.
15. Постройте рекурсивно все даты с 2026-01-01 по 2026-01-31 (старт — `'2026-01-01'`, шаг — `date(x, '+1 day')`, остановка — `x < '2026-01-31'`) и посчитайте, сколько среди них воскресений (`strftime('%w', x) = '0'`). Запишите число в `otvety/15`.
16. В `hr.db` обойдите дерево руководителей рекурсивным CTE (корень — директор без руководителя, уровень 0) и выведите для каждого уровня число сотрудников: `уровень, число`, по возрастанию уровня. Результат — в `otvety/16`.
17. В `hr.db` найдите всех подчинённых сотрудника с `id = 2` (прямых и косвенных, сам он не входит). Выведите имя и уровень (у прямых подчинённых он 1); по уровню, затем по `id` сотрудника. Результат — в `otvety/17`.
18. В `hr.db` поднимитесь от сотрудника с `id = 22` вверх до директора: стартовая строка — сам сотрудник (глубина 0), шаг — его руководитель. Выведите `id` и

глубину по возрастанию глубины. Результат — в `otvety/18`.

19. В `hr.db` найдите рекурсивно сотрудника с `id = 3` и всех его подчинённых и вычислите их суммарную зарплату. Стартовая часть — просто `SELECT 3`; затем суммируйте `salary` у сотрудников, чей `id` есть в CTE (`id IN (SELECT id FROM t)`). Запишите число в `otvety/19`.
20. В `hr.db` обойдите дерево от директора (уровень 0) и для каждого уровня выведите число сотрудников и среднюю зарплату, округлённую до целых; по уровню. Результат — в `otvety/20`.

Раздел 45. Команда OVER, PARTITION BY — оконные агрегаты

45.1. Теоретические сведения

Назначение. Оконная функция считает значение для каждой строки, «глядя» на связанные с ней строки (окно), но строки при этом **не сворачиваются в группы**, как в `GROUP BY`. Окно задаётся предложением `OVER (...)`; обычные агрегаты (`SUM`, `AVG`, `COUNT`, `MIN`, `MAX`) с `OVER` дают итог по всему результату или по группе рядом с каждой строкой.

Синтаксис:

```
агрегат(...) OVER ()                        -- по всему результату
агрегат(...) OVER (PARTITION BY столбцы)    -- по группе
```

Правила:

- оконные функции вычисляются **после** `WHERE`, `GROUP BY` и `HAVING`, но до `ORDER BY` и `LIMIT` внешнего запроса; поэтому `WHERE` сужает окно (посчитанное `COUNT(*) OVER ()` учтёт только отобранные строки);
- в `WHERE` и `HAVING` оконную функцию писать нельзя: сначала вычислите её в CTE или подзапросе, а потом отбирайте;
- на выходе столько же строк, сколько на входе: итог повторяется в каждой строке своего окна;
- доли считают так: `100.0 * x / SUM(x) OVER ()`; число `100.0`, а не `100`, иначе целочисленное деление обнулит результат;
- если нужен итог по результату группировки, сначала посчитайте группы в CTE, затем применяйте окно к CTE;
- оконная функция — обычное выражение: её можно вычитать, делить, округлять (`price - AVG(price) OVER (PARTITION BY category)`, `ROUND(100.0 * price / SUM(price) OVER (PARTITION BY category), 1)`);
- в одном запросе допустимо несколько окон: `MAX(price) OVER (PARTITION BY category) - MIN(price) OVER (PARTITION BY category)`;
- `ORDER BY ... LIMIT` внешнего запроса не меняют значения окна: сначала окно считается по всем отобранным строкам, потом остаётся нужное число строк;
- оконные функции доступны в SQLite начиная с версии 3.25.

Части описания окна:

Элемент	Описание
`OVER ()`	пустое окно: все строки результата
`PARTITION BY a`	делит строки на независимые группы по значению `a`; функция считается внутри каждой группы

Пример:

```
SELECT name, category, price,
```

```
       AVG(price) OVER (PARTITION BY category) AS avg_in_cat,   -- среднее по
       категории рядом с каждым товаром
       COUNT(*)   OVER ()                      AS total_rows    -- всего строк в
       результате
FROM products ORDER BY category, id;
```

45.2. Задания

Рабочий каталог: `~/sql-lab/08_cte_okna`. Ответ каждого задания записывайте в файл `otvety/N`, где `N` — номер задания.

21. Для первых пяти товаров по `id` выведите `id`, цену и среднюю цену всех товаров, округлённую до двух знаков (`AVG(price) OVER ()`). Результат — в `otvety/21`.
22. Для первых восьми товаров по `id` выведите `id`, категорию и число товаров в этой категории (`COUNT(*) OVER (PARTITION BY category)`). Результат — в `otvety/22`.
23. Для первых восьми товаров по `id` выведите `id`, название и разность между ценой и средней ценой категории (`price - AVG(price) OVER (PARTITION BY category)`), округлённую до одного знака. Результат — в `otvety/23`.
24. В `hr.db` для первых десяти сотрудников по `id` из тех, у кого указан отдел, выведите `id`, зарплату и среднюю зарплату отдела (до целых). Результат — в `otvety/24`.
25. Для первых десяти товаров по `id` выведите `id`, категорию и долю цены товара в сумме цен его категории в процентах (`100.0 * price / SUM(price) OVER (PARTITION BY category)`), округлённую до одного знака. Результат — в `otvety/25`.
26. В `hr.db` отберите сотрудников с зарплатой выше 100000 и для первых пяти по `id` выведите `id`, зарплату и число таких сотрудников (`COUNT(*) OVER ()`). Условие `WHERE` пишите в том же запросе. Результат — в `otvety/26`.
27. Для каждой категории вычислите стоимость остатков `v = SUM(price * stock)` и её долю в общей стоимости в процентах, округлённую до одного знака (`ROUND(100.0 * v / SUM(v) OVER (), 1)`). Выведите категорию, `v` и долю по алфавиту категорий. Результат — в `otvety/27`.
28. Для первых десяти заказов по `id` выведите `id`, клиента и число всех заказов этого клиента (`COUNT(*) OVER (PARTITION BY customer_id)`). Результат — в `otvety/28`.
29. Для первых шести товаров по `id` выведите `id`, цену и размах цен его категории: наибольшая цена категории минус наименьшая (два окна `MAX` и `MIN` с `PARTITION BY category`). Результат — в `otvety/29`.
30. Посчитайте выручку (`SUM(qty * price)`) по городам клиентов в CTE (клиенты без города не учитываются), затем окном `SUM(v) OVER ()` вычислите долю города, и выведите город, выручку и её долю в общей выручке этих городов в процентах (до одного знака); от большей выручки к меньшей, при равенстве — по городу. Результат — в `otvety/30`.

Раздел 46. Команда ROW_NUMBER, RANK, DENSE_RANK — нумерация и ранги

46.1. Теоретические сведения

Назначение. Функции нумерации присваивают строкам номера в порядке, заданном в `OVER (ORDER BY ...)`. Нужны для рейтингов и поиска лидеров.

Синтаксис:

```
ROW_NUMBER() OVER ([PARTITION BY ...] ORDER BY ...)
RANK()       OVER ([PARTITION BY ...] ORDER BY ...)
DENSE_RANK() OVER ([PARTITION BY ...] ORDER BY ...)
```

Правила:

- у функций нет аргументов, но скобки обязательны; `ORDER BY` внутри `OVER` определяет, что считать «первым»;
- для `ROW_NUMBER` добавляйте последним ключом уникальный столбец (`id`): иначе при равных значениях номера распределятся произвольно;
- `PARTITION BY` начинает нумерацию заново в каждой группе, поэтому номер 1 — первая строка каждой группы;
- `RANK` и `DENSE_RANK` подходят, когда равные значения должны делить место;
- направление сортировки внутри `OVER` задают как обычно: `ORDER BY price DESC`; оно не связано с `ORDER BY` всего запроса;
- наибольший `DENSE_RANK` равен числу различных значений ключа: `SELECT MAX(d) FROM (SELECT DENSE_RANK() OVER (ORDER BY stock) AS d FROM products);`
- ранг можно считать над результатом CTE (например, суммы по клиентам) и тут же соединять его с другими таблицами.

Функции:

Элемент	Описание
`ROW_NUMBER()`	сквозной номер 1, 2, 3, ...; одинаковые значения всё равно получают разные номера
`RANK()`	место с «пропусками»: при ничьей одинаковые номера, следующий пропускается (1, 2, 2, 4)
`DENSE_RANK()`	место без пропусков (1, 2, 2, 3)

Пример:

```
SELECT name, city,
       ROW_NUMBER() OVER (ORDER BY city, id) AS rn,
       RANK()       OVER (ORDER BY city)     AS rk,
       DENSE_RANK() OVER (ORDER BY city)     AS dr
FROM customers;                                   -- у клиентов одного города rk и
dr совпадают, rn — нет
```

46.2. Задания

Рабочий каталог: `~/sql-lab/08_cte_okna`. Ответ каждого задания записывайте в файл `otvety/N`, где `N` — номер задания.

31. Пронумеруйте товары по убыванию цены (`ROW_NUMBER() OVER (ORDER BY price DESC, id)` под именем `rn`) и выведите `id`, название, цену и `rn` для первых пяти. Результат — в `otvety/31`.
32. Пронумеруйте клиентов от самой поздней даты регистрации к ранней (при равенстве дат — по `id`) и выведите номер, `id`, имя и дату для первых пяти. Результат — в `otvety/32`.
33. Для каждого товара вычислите место по цене внутри категории (`RANK() OVER (PARTITION BY category ORDER BY price DESC)` под именем `r`). Выведите категорию, название, цену и `r`; порядок — по категории, затем по `r`. Результат — в `otvety/33`.
34. Вычислите `RANK() OVER (ORDER BY stock DESC)` под именем `rk` и выведите `id`, остаток и `rk` для первых десяти товаров в порядке `rk`, затем `id`. Результат — в `otvety/34`.
35. Сделайте то же с `DENSE_RANK() OVER (ORDER BY stock DESC)` (имя `dr`): `id`, остаток и `dr` для первых десяти товаров в порядке `dr`, затем `id`. Результат — в `otvety/35`.
36. С помощью `DENSE_RANK() OVER (ORDER BY stock)` во вложенном запросе найдите наибольший плотный ранг — сколько различных значений остатка есть у товаров. Запишите число в `otvety/36`.
37. Выведите для первых восьми товаров в порядке `ROW_NUMBER()` по `stock DESC, id`: `id`, остаток, `ROW_NUMBER()`, `RANK()` и `DENSE_RANK()` (все три по `ORDER BY stock DESC`; у `ROW_NUMBER` добавьте `id`). Результат — в `otvety/37`.
38. В `hr.db` для сотрудников с указанным отделом пронумеруйте их по убыванию зарплаты внутри отдела (`ROW_NUMBER` с `PARTITION BY dept_id`, при равенстве — по `id`). Выведите отдел, имя, зарплату и номер для первых десяти строк в порядке отдела и номера. Результат — в `otvety/38`.
39. Составьте CTE `rev` с выручкой по клиентам и присвойте место `RANK() OVER (ORDER BY total DESC)`. Выведите место, имя и выручку первых пяти клиентов; порядок — по месту, затем по `id` клиента. Результат — в `otvety/39`.
40. Для первых десяти заказов по `id` выведите `id`, клиента, дату и порядковый номер заказа у этого клиента: `ROW_NUMBER() OVER (PARTITION BY customer_id ORDER BY order_date, id)`. Результат — в `otvety/40`.

Раздел 47. Нарастающий итог и рамка окна

47.1. Теоретические сведения

Назначение. Если в окне есть `ORDER BY`, агрегат считается нарастающим итогом: от начала окна до текущей строки. Границы суммирования уточняет **рамка** `ROWS BETWEEN ... AND ...`, например для скользящего среднего.

Синтаксис:

```
SUM(выражение) OVER (ORDER BY k)                              -- нарастающий итог
SUM(выражение) OVER (PARTITION BY a ORDER BY k)               -- нарастающий итог по
группе
AVG(выражение) OVER (ORDER BY k ROWS BETWEEN 2 PRECEDING AND CURRENT ROW)
```

Правила:

- по умолчанию, если рамка не указана, суммируются строки от начала окна до текущей **вместе со всеми строками с тем же значением ключа**;
- поэтому при неуникальном ключе сортировки у равных строк итог одинаков; сделайте ключ уникальным (добавьте `id`) или укажите `ROWS`;
- `ROWS` считает физические строки, а не значения ключа; рамка `ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW` даёт «честный» нарастающий итог;
- `CURRENT ROW AND UNBOUNDED FOLLOWING` суммирует остаток: от текущей строки до конца;
- если итог нужен по результату группировки, сначала посчитайте группы в CTE;
- ключ `ORDER BY` в окне может быть любым: `ORDER BY stock DESC`, `ORDER BY hired, id`; рамка пишется сразу после него: `OVER (ORDER BY k ROWS BETWEEN ...)`;
- скользящее среднее трёх строк: `AVG(price) OVER (ORDER BY id ROWS BETWEEN 2 PRECEDING AND CURRENT ROW)`; `PARTITION BY` ставят перед `ORDER BY`;
- чтобы найти первую строку, где нарастающий итог достиг порога, посчитайте итог в CTE (`cum`), а условие `WHERE cum * 2 >= tot` с `ORDER BY ... LIMIT 1` пишите снаружи; общий итог рядом даёт `SUM(v) OVER ()`.

Границы рамки:

Элемент	Описание
`UNBOUNDED PRECEDING`	с первой строки окна
`N PRECEDING`	N строк до текущей
`CURRENT ROW`	текущая строка
`N FOLLOWING`	N строк после текущей
`UNBOUNDED FOLLOWING`	до последней строки окна

Пример:

```
SELECT id, registered,
```

```
       COUNT(*) OVER (ORDER BY registered, id)                  AS reg_total,   --
       сколько клиентов зарегистрировано к этой дате
       ROUND(AVG(id) OVER (ORDER BY registered, id
             ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING), 1)      AS smooth       --
             среднее с соседями
FROM customers ORDER BY registered, id LIMIT 5;
```

47.2. Задания

Рабочий каталог: `~/sql-lab/08_cte_okna`. Ответ каждого задания записывайте в файл `otvety/N`, где `N` — номер задания.

41. Вычислите сумму каждого заказа (CTE `t`) и выведите для первых пяти заказов по `id` номер заказа, сумму и нарастающий итог `SUM(total) OVER (ORDER BY order_id)`. Результат — в `otvety/41`.
42. Для первых шести товаров по `id` выведите `id`, остаток и нарастающую сумму остатков в порядке `id`. Результат — в `otvety/42`.
43. В `hr.db` для первых пяти сотрудников в порядке найма (`hired`, затем `id`) выведите `id`, дату найма, зарплату и нарастающий фонд зарплат в том же порядке. Результат — в `otvety/43`.
44. В `hr.db` для сотрудников с указанным отделом посчитайте нарастающий фонд зарплат внутри отдела (по `hired`, затем `id`). Выведите отдел, `id`, зарплату и итог для первых шести строк в порядке отдела, даты, `id`. Результат — в `otvety/44`.
45. Для восьми товаров с наибольшим остатком (порядок — `stock DESC`, затем `id`) выведите `id`, остаток и `SUM(stock) OVER (ORDER BY stock DESC)` — без рамки и без `id` в окне. Обратите внимание: товары с равным остатком получают одинаковый итог. Результат — в `otvety/45`.
46. Повторите предыдущее задание с рамкой `ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW` и ключом окна `stock DESC, id`: `id`, остаток, нарастающий итог для восьми товаров с наибольшим остатком. Результат — в `otvety/46`.
47. Для первых восьми товаров по `id` выведите `id`, цену и среднюю цену трёх товаров: текущего и двух предыдущих (`ROWS BETWEEN 2 PRECEDING AND CURRENT ROW`), до одного знака. Результат — в `otvety/47`.
48. Составьте CTE `m` с выручкой по месяцам (`ym` — `substr(order_date, 1, 7)`, `v` — `SUM(qty * price)`). Выведите для первых шести месяцев `ym`, `v` и нарастающую выручку. Результат — в `otvety/48`.
49. Вычислите сумму каждого заказа (CTE `t`) и для первых пяти заказов по `id` выведите номер, сумму и сумму этого и всех следующих заказов (`ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING`). Результат — в `otvety/49`.
50. Составьте CTE `m` с выручкой по месяцам, в запросе над ним посчитайте нарастающую выручку `cum` и общую выручку `tot` (`SUM(v) OVER ()`). Выведите первый по порядку месяц, в котором `cum * 2 >= tot`, и значение `cum`: окна считайте во втором CTE `x`, а условие и `ORDER BY ym LIMIT 1` пишите в основном запросе. Результат — в `otvety/50`.

Раздел 48. Команда LAG, LEAD — соседние строки

48.1. Теоретические сведения

Назначение. LAG возвращает значение из предыдущей строки окна, LEAD — из следующей. Так сравнивают строку с соседней: дата прошлого заказа, изменение показателя, пауза между событиями.

Синтаксис:

```
LAG(выражение [, смещение [, значение_по_умолчанию]]) OVER ([PARTITION BY ...]
ORDER BY ...)
LEAD(выражение [, смещение [, значение_по_умолчанию]]) OVER ([PARTITION BY ...]
ORDER BY ...)
```

Правила:

- порядок строк задаёт ORDER BY внутри OVER; без него «предыдущая» строка не определена;
- PARTITION BY ограничивает поиск соседа одной группой: у первой строки каждой группы LAG даёт NULL;
- разность с соседом вычисляют так: x - LAG(x) OVER (...); для первой строки результат тоже NULL;
- условие по результату LAG (например, WHERE prev IS NOT NULL) пишут уже во внешнем запросе;
- добавляйте уникальный столбец последним ключом ORDER BY, чтобы при равных значениях «соседство» было однозначным;
- смещение 2 берёт строку через одну: LAG(price, 2), LEAD(price, 2); третий аргумент заменяет NULL: LAG(stock, 1, 0);
- пауза в днях между датами: CAST(julianday(order_date) - julianday(prev) AS INTEGER); прирост в процентах: 100.0 * (v - p) / p, где p — значение LAG;
- типичная схема: WITH x AS (SELECT ..., LAG(...) OVER (...) AS prev FROM ...) SELECT ... FROM x WHERE prev IS NOT NULL — окно считается в CTE, фильтр снаружи.

Аргументы:

Элемент	Описание
выражение	что взять у соседней строки: столбец или вычисление
смещение	на сколько строк отступить (по умолчанию 1)
значение_по_умолчанию	что вернуть, если соседней строки нет (по умолчанию NULL)

Пример:

```
SELECT id, registered,
       LAG(registered)  OVER (ORDER BY registered, id) AS prev_date,
       LEAD(registered) OVER (ORDER BY registered, id) AS next_date,
```

```
       LAG(id, 1, 0)    OVER (ORDER BY registered, id) AS prev_id      -- 0 вместо
       NULL у первой строки
FROM customers ORDER BY registered, id LIMIT 6;
```

48.2. Задания

Рабочий каталог: `~/sql-lab/08_cte_okna`. Ответ каждого задания записывайте в файл `otvety/N`, где `N` — номер задания.

51. Для первых десяти заказов по `id` выведите номер заказа, клиента, дату и дату предыдущего заказа **того же клиента** (`LAG(order_date) OVER (PARTITION BY customer_id ORDER BY order_date, id)`). Результат — в `otvety/51`.
52. Для первых шести товаров по `id` выведите `id`, цену и цену предыдущего товара по `id` (`LAG`). Результат — в `otvety/52`.
53. В `hr.db` для первых шести сотрудников в порядке найма (`hired`, затем `id`) выведите имя и имя сотрудника, принятого следом (`LEAD(name)`). Результат — в `otvety/53`.
54. Для первых шести товаров по `id` выведите `id`, цену и её разность с ценой предыдущего товара (`price - LAG(price) OVER (ORDER BY id)`). Результат — в `otvety/54`.
55. Для первых шести товаров по `id` выведите `id`, остаток и остаток предыдущего товара, а для первого товара — `0` (третий аргумент `LAG`). Результат — в `otvety/55`.
56. Для первых шести товаров по `id` выведите `id`, цену и цену товара, стоящего на две позиции раньше (`LAG(price, 2)`), и цену на две позиции позже (`LEAD(price, 2)`). Результат — в `otvety/56`.
57. Во внешнем запросе над CTE `x` (заказы, дата и `LAG(order_date)` по клиенту, порядок — `order_date, id`) выведите для первых восьми заказов по `id`, у которых есть предыдущий, номер заказа, клиента и паузу в днях `CAST(julianday(order_date) - julianday(prev) AS INTEGER)`. Результат — в `otvety/57`.
58. Найдите наибольшую паузу в днях между двумя соседними заказами одного клиента: `LAG(order_date)` с `PARTITION BY customer_id ORDER BY order_date, id` в CTE, `MAX` от разности `julianday` (приведите к `INTEGER`) снаружи. Запишите число в `otvety/58`.
59. В `hr.db` посчитайте сотрудников, чья зарплата выше зарплаты сотрудника, принятого непосредственно перед ним (порядок найма — `hired`, затем `id`): `LAG(salary)` в CTE, сравнение снаружи. Запишите число в `otvety/59`.
60. Составьте CTE `m` с выручкой по месяцам и добавьте `p` — выручку предыдущего месяца (`LAG`). Для первых шести месяцев, у которых есть предыдущий, выведите месяц, выручку и прирост в процентах `ROUND(100.0 * (v - p) / p, 1)`. Результат — в `otvety/60`.

Раздел 49. Отбор по результату оконной функции

49.1. Теоретические сведения

Назначение. Оконную функцию нельзя поставить в WHERE: условие проверяется раньше, чем окно вычислено. Поэтому номер или ранг считают во вложенном запросе (чаще всего в CTE), а отбор делают снаружи. Так находят лидера в каждой группе.

Синтаксис:

```
WITH r AS (
    SELECT ..., ROW_NUMBER() OVER (PARTITION BY группа ORDER BY ключ DESC) AS rn
    FROM таблица
)
SELECT ... FROM r WHERE rn <= N;
```

Правила:

- `rn = 1` — один лидер в группе; `rn <= 3` — первые три в каждой группе;
- чтобы при ничьей получить всех лидеров, а не одного, используйте `RANK()` и условие `rk = 1`;
- порядок внутри `OVER` должен быть однозначным (добавьте `id`), иначе при равных значениях лидер выбирается произвольно;
- то же правило действует для `HAVING` и `GROUP BY`: окно в них не используется;
- `rn = 2`, `rn = 3` дают вторую или третью строку каждой группы; у групп, где строк меньше, такой строки нет и группа в результат не попадает;
- второе по величине **значение** (с учётом ничьих) находят через `DENSE_RANK() OVER (PARTITION BY ... ORDER BY ... DESC)` и условие `dr = 2`;
- название из соседней таблицы (отдела, клиента) добавляют `JOIN`-ом во внешнем запросе: `... FROM r JOIN departments d ON d.id = r.dept_id WHERE r.rn = 1`;
- для поиска лидера по сумме сначала посчитайте суммы в одном CTE, затем номера в следующем: `WITH rev AS (...), r AS (SELECT ..., ROW_NUMBER() OVER (...) AS rn FROM rev ...) SELECT ... FROM r WHERE rn <= 2`.

Пример:

```
WITH r AS (
    SELECT supplier_id, name, stock,
           ROW_NUMBER() OVER (PARTITION BY supplier_id ORDER BY stock DESC, id) AS
           rn
    FROM products
)
SELECT supplier_id, name, stock FROM r
WHERE rn <= 2 ORDER BY supplier_id, rn;                -- два самых «запасённых»
товара каждого поставщика
```

49.2. Задания

Рабочий каталог: `~/sql-lab/08_cte_okna`. Ответ каждого задания записывайте в файл `otvety/N`, где `N` — номер задания.

61. С помощью CTE и `ROW_NUMBER() OVER (PARTITION BY category ORDER BY price DESC, id)` выведите для каждой категории самый дорогой товар (`rn = 1`): категорию, название, цену — по алфавиту категорий. Результат — в `otvety/61`.
62. Выведите по два самых дорогих товара каждой категории (`ROW_NUMBER() OVER (PARTITION BY category ORDER BY price DESC, id)` в CTE и `rn <= 2` снаружи): категорию, название, цену; порядок — по категории, затем по убыванию цены. Результат — в `otvety/62`.
63. Выведите самый дешёвый товар каждой категории (`ROW_NUMBER() OVER (PARTITION BY category ORDER BY price, id)` в CTE, `rn = 1` снаружи): категорию, название, цену — по алфавиту категорий. Результат — в `otvety/63`.
64. Для каждого города (клиенты без города не учитываются) выведите клиента с самой ранней датой регистрации (`ROW_NUMBER() OVER (PARTITION BY city ORDER BY registered, id)`, `rn = 1`): город, имя, дату — по городу. Результат — в `otvety/64`.
65. Для каждого клиента, у которого есть заказы, выведите его последний заказ (`ROW_NUMBER() OVER (PARTITION BY customer_id ORDER BY order_date DESC, id DESC)`, `rn = 1`): клиент, номер заказа, дата — по номеру клиента. Результат — в `otvety/65`.
66. В `hr.db` для каждого отдела выведите самого высокооплачиваемого сотрудника (`ROW_NUMBER() OVER (PARTITION BY dept_id ORDER BY salary DESC, id)`, `rn = 1`; название отдела добавьте `JOIN` с `departments`): название отдела, имя, зарплату — по `id` отдела. Сотрудники без отдела не учитываются. Результат — в `otvety/66`.
67. Выведите товары, у которых остаток наибольший **в своей категории** (при ничьей — все такие): категорию, название, остаток; по категории, затем по `id` товара. Используйте `RANK()` и `rk = 1`. Результат — в `otvety/67`.
68. Составьте CTE `rev` с выручкой по клиентам. Для каждого города (клиенты без города не учитываются) выведите двух клиентов с наибольшей выручкой (второе CTE `r` с `ROW_NUMBER() OVER (PARTITION BY c.city ...)`, `rn <= 2`): город, имя, выручку; порядок — по городу, затем по убыванию выручки (при равенстве выручки — меньший `id` клиента). Результат — в `otvety/68`.
69. Выведите третий по дате заказ каждого клиента (`ROW_NUMBER()` с `PARTITION BY customer_id ORDER BY order_date, id` и `rn = 3`; у клиентов с менее чем тремя заказами строки не будет): клиент, номер заказа, дата — по номеру клиента. Результат — в `otvety/69`.
70. В `hr.db` для каждого отдела найдите сотрудника со вторым по величине окладом (`DENSE_RANK() OVER (PARTITION BY dept_id ORDER BY salary DESC)` равен 2); отдел без второго сотрудника в результат не попадает. Выведите название отдела, имя и зарплату по `id` отдела. Результат — в `otvety/70`.

Вопросы для размышления

1. Чем оконная функция отличается от агрегата с `GROUP BY`, и в каких отчётах без неё не обойтись?
2. Почему рекурсивный запрос обязательно нужно ограничивать условием, и что произойдёт, если в данных есть цикл?
3. Чем `ROW_NUMBER`, `RANK` и `DENSE_RANK` отличаются при равных значениях, и по-

чему в `ORDER BY` окна полезен уникальный столбец?

Тема 9. Изменение данных: INSERT, UPDATE, DELETE

Общие сведения

Три команды меняют данные. `INSERT INTO таблица (столбцы) VALUES (значения)` добавляет строки (несколько строк — списком в скобках через запятую; `INSERT ... SELECT` копирует результат запроса). `UPDATE таблица SET столбец = выражение WHERE условие` изменяет существующие строки. `DELETE FROM таблица WHERE условие` удаляет строки. **Забытый `WHERE` относится ко всей таблице**: `UPDATE` без него изменит все строки, `DELETE` без него удалит всё.

Поэтому у администратора выработан порядок: *сначала* `SELECT` с тем же `WHERE` (посмотреть, что будет затронуто), *затем* изменение внутри транзакции, *потом* снова `SELECT`. Перед массовыми изменениями делают копию базы.

Среди возможностей SQLite: `INSERT ... ON CONFLICT (ключ) DO UPDATE SET ...` (*upsert*: добавить или, если такая запись уже есть, обновить; значения вставляемой строки доступны как `excluded.столбец`); `RETURNING` (вернуть значения затронутых строк, например автоматически назначенный `id`); `CREATE TABLE ... AS SELECT` (создать таблицу и заполнить результатом запроса). Если таблицы связаны внешними ключами и проверка включена (`PRAGMA foreign_keys = ON`), удалять сначала нужно дочерние строки, затем родительские; иначе база откажет: «FOREIGN KEY constraint failed».

Примеры.

```
INSERT INTO customers (id, name, city, email, registered, vip)
VALUES (1000, 'Тестов Тест', 'Тула', 'test@sever.example', '2026-10-11', 0);

SELECT id, price FROM products WHERE category = 'Книги';                  --
сначала посмотреть
UPDATE products SET price = price * 11 / 10 WHERE category = 'Книги';

INSERT INTO products (id, name, category, price, stock, supplier_id) VALUES (5,
'x', 'x', 0, 10, 1)
ON CONFLICT(id) DO UPDATE SET stock = stock + excluded.stock;

INSERT INTO suppliers (name, country) VALUES ('Новый поставщик', 'Россия')
RETURNING id;

DELETE FROM order_items WHERE order_id IN (SELECT id FROM orders WHERE status =
'cancelled');
DELETE FROM orders WHERE status = 'cancelled';
```

Что чаще всего идёт не так

Самая дорогая ошибка в практике — `UPDATE` или `DELETE` без `WHERE` или с условием, затрагивающим больше строк, чем рассчитывали. Защита проста: всегда `SELECT` вместо `UPDATE`/`DELETE` сначала, всегда транзакция, всегда копия перед массовым изменением.

Рабочий каталог и проверка

Каталог полигона: `~/sql-lab/09_izmenenie_dannyh`. Проверка: `python3 proverka.py 9`.

Читать данные недостаточно: их нужно добавлять, исправлять и удалять. За это отвечают команды `INSERT`, `UPDATE` и `DELETE`. Они меняют базу **насовсем**, поэтому у них действует правило администратора: сначала `SELECT` с тем же условием `WHERE` (посмотреть, что заденет), затем изменение, затем снова `SELECT` (убедиться). Забытый `WHERE` у `UPDATE` или `DELETE` затрагивает **все** строки таблицы. В этой теме вы работаете с копией базы `shop.db`; если что-то испортили, раздел можно пересоздать (`python3 setup.py 9`). Проверка сравнивает состояние базы с исходным. Выполняйте задания **по порядку**: некоторые из них опираются на изменения предыдущих.

Раздел 50. Изменение данных: INSERT, UPDATE, DELETE

50.1. Теоретические сведения

INSERT INTO … VALUES — добавление строк

Назначение. INSERT добавляет в таблицу новые строки. Значения перечисляют в VALUES, а столбцы, которым они соответствуют, — в скобках после имени таблицы.

Синтаксис:

```
INSERT INTO таблица (столбец1, столбец2, ...) VALUES (значение1, значение2, ...);
INSERT INTO таблица (столбец1, столбец2, ...) VALUES (...), (...), (...);
INSERT INTO таблица (столбец1, столбец2, ...) SELECT ...;
```

Правила:

- столбцы лучше перечислять явно: порядок значений должен им соответствовать, а пропущенные столбцы получат DEFAULT или NULL;
- без списка столбцов значения должны идти во **всех** столбцах и в порядке определения таблицы — такая запись ломается при изменении схемы;
- строки-значения заключают в одинарные кавычки, числа — без кавычек; пустое значение пишется NULL;
- несколько строк одной командой: после VALUES перечисляют скобки через запятую; это быстрее и атомарно (все строки или ни одной);
- столбец INTEGER PRIMARY KEY можно не указывать: SQLite назначит следующий номер сам;
- нарушение NOT NULL, UNIQUE, CHECK или внешнего ключа отменяет всю команду с ошибкой constraint failed;
- вместо VALUES можно подставить SELECT: вставятся все строки его результата.

Пример:

```
INSERT INTO suppliers (id, name, country) VALUES (50, 'Пробный', 'Польша');

INSERT INTO suppliers (name, country) VALUES ('Альфа', 'Китай'), ('Бета',
'Россия');   -- id назначится сам

INSERT INTO suppliers (name, country)
SELECT 'Копия ' || name, country FROM suppliers WHERE country = 'Китай';
-- строки из запроса
```

UPDATE … SET … WHERE — изменение строк

Назначение. UPDATE меняет значения столбцов в уже существующих строках. Какие строки изменить, определяет WHERE.

Синтаксис:

```
UPDATE таблица SET столбец1 = выражение1 [, столбец2 = выражение2 ...] [WHERE
условие];
```

Правила:

- **без WHERE изменяются все строки таблицы**; поэтому сначала выполните SELECT с тем же условием и посмотрите, что заденет команда;
- в правой части можно использовать текущие значения столбцов: `stock = stock - 1`, `price = price * 11 / 10`;
- деление целых чисел даёт целое: `price * 11 / 10` отбрасывает дробную часть, а `price * 1.1` даст вещественное число;
- в WHERE можно использовать всё, что было в SELECT: IN, LIKE, подзапросы, EXISTS, например `WHERE id IN (SELECT ...)`;
- несколько столбцов меняют одной командой через запятую: все правые части вычисляются по **старым** значениям;
- если условие не нашло ни одной строки, ошибки нет: просто ничего не изменится (число изменённых строк покажет `SELECT changes();`);
- защита от ошибки: начните `BEGIN;`, выполните UPDATE, проверьте результат и закончите `COMMIT;` или откатите `ROLLBACK;`.

Пример:

```
SELECT id, name, stock FROM products WHERE category = 'Мебель';      -- 1.
смотрим, что заденет

UPDATE products SET stock = stock + 5 WHERE category = 'Мебель';      -- 2. меняем

UPDATE customers SET vip = 1, city = 'Казань' WHERE id = 3;           -- два
столбца сразу

UPDATE products SET price = price - 100
WHERE id IN (SELECT product_id FROM order_items GROUP BY product_id HAVING
SUM(qty) > 20);   -- условие с подзапросом
```

DELETE FROM — удаление строк

Назначение. DELETE удаляет из таблицы строки, удовлетворяющие условию. Сама таблица и её структура остаются.

Синтаксис:

```
DELETE FROM таблица [WHERE условие];
```

Правила:

- **без WHERE удаляются все строки таблицы**; перед удалением выполните `SELECT * FROM таблица WHERE условие;`;
- удалённые строки не вернуть, если нет копии или открытой транзакции, которую можно откатить;
- при включённом `PRAGMA foreign_keys = ON` нельзя удалить родительскую строку, на которую ссылаются дочерние (если не задано `ON DELETE CASCADE` или `SET NULL`); при выключенном — получатся «осиротевшие» строки;
- поэтому дочерние строки удаляют **первыми**, родительские — вторыми: сначала `order_items`, затем `orders`;
- условие для дочерней таблицы часто записывают подзапросом: `WHERE order_id IN (SELECT id FROM orders WHERE ...)`; удалять нужно до того, как исчезли родительские строки, иначе подзапрос ничего не найдёт;
- для удаления самой таблицы служит `DROP TABLE` (тема 10), а не DELETE.

Пример:

```
SELECT COUNT(*) FROM orders WHERE order_date < '2025-03-01';                    --
сколько заденет

DELETE FROM order_items
WHERE order_id IN (SELECT id FROM orders WHERE order_date < '2025-03-01');    --
сначала дочерние
DELETE FROM orders WHERE order_date < '2025-03-01';                             --
затем родительские
```

CREATE TABLE … AS SELECT — таблица из запроса

Назначение. Создаёт новую таблицу и сразу заполняет её результатом запроса. Так делают архивы, копии перед рискованным изменением и промежуточные выборки.

Синтаксис:

```
CREATE TABLE новая_таблица AS SELECT ... FROM таблица [WHERE ...];
```

Правила:

- столбцы новой таблицы получают имена и типы из результата запроса; для вычисляемых столбцов задавайте AS;
- копируются только данные: ограничения (PRIMARY KEY, NOT NULL, CHECK, UNIQUE), значения по умолчанию, индексы и внешние ключи **не переносятся**;
- если таблица с таким именем уже есть, будет ошибка; безопасный вариант — CREATE TABLE IF NOT EXISTS;
- WHERE 0 (или LIMIT 0) создаёт пустую таблицу с той же структурой;
- таблица-копия — самая простая «резервная копия» перед массовым UPDATE или DELETE.

Пример:

```
CREATE TABLE customers_backup AS SELECT * FROM customers;                    --
полная копия

CREATE TABLE vip_customers AS
SELECT id, name, city FROM customers WHERE vip = 1;                           --
только часть строк и столбцов

SELECT COUNT(*) FROM customers_backup;                                        --
проверка
```

ON CONFLICT DO UPDATE — вставить или обновить

Назначение. Предложение ON CONFLICT задаёт, что делать, если вставляемая строка нарушает ограничение уникальности. С DO UPDATE команда превращается в «вставить, а если такая строка уже есть, — обновить» (upsert).

Синтаксис:

```
INSERT INTO таблица (столбцы) VALUES (значения)
ON CONFLICT (столбец_ключа) DO UPDATE SET столбец = выражение [, ...];

INSERT INTO таблица (столбцы) VALUES (значения)
ON CONFLICT (столбец_ключа) DO NOTHING;
```

Правила:

- в скобках после `ON CONFLICT` указывают столбец (или столбцы) с ограничением `PRIMARY KEY` или `UNIQUE`, по которому ищется конфликт;
- в `SET` перечисляют только те столбцы, которые нужно изменить; остальные в существующей строке остаются прежними;
- `stock = stock + excluded.stock` — увеличить имеющееся значение на пришедшее, а не заменить его;
- значения в `VALUES` для столбцов, которые не обновляются, всё равно должны быть допустимы (`NOT NULL` проверяется);
- если конфликта нет, строка просто вставляется и ветка `DO UPDATE` не выполняется;
- команда доступна в SQLite с версии 3.24.

Что можно использовать в DO UPDATE:

Элемент	Описание
`столбец`	текущее значение в существующей строке
`excluded.столбец`	значение, которое вставлялось бы в новой строке
`DO NOTHING`	вместо обновления: тихо пропустить вставку конфликтующей строки

Пример:

```
INSERT INTO suppliers (id, name, country) VALUES (1, 'x', 'x')
ON CONFLICT (id) DO UPDATE SET country = excluded.country;      -- id=1 есть:
меняем только страну

INSERT INTO suppliers (id, name, country) VALUES (1, 'x', 'x')
ON CONFLICT (id) DO NOTHING;                                     -- ничего не
меняем, ошибки нет
```

RETURNING — получить изменённые строки

Назначение. Предложение `RETURNING` в конце `INSERT`, `UPDATE` или `DELETE` возвращает значения затронутых строк так, как будто это `SELECT`. Так узнают автоматически назначенный `id`, не делая второго запроса.

Синтаксис:

```
INSERT INTO таблица (...) VALUES (...) RETURNING столбец1 [, столбец2 ...];
UPDATE таблица SET ... WHERE ... RETURNING *;
DELETE FROM таблица WHERE ... RETURNING *;
```

Правила:

- после `RETURNING` можно перечислить столбцы, выражения или * (все столбцы);
- для `INSERT` и `UPDATE` возвращаются значения **после** изменения, для `DELETE` — удалённые строки;
- если команда не затронула ни одной строки, результат пуст;
- команда с `RETURNING` всё равно изменяет базу — это не просмотр;
- результат `RETURNING` выводится как результат `SELECT`, поэтому его сохраняют так же: `.output otvety/N`, затем команда, затем `.output stdout`;
- доступно в SQLite с версии 3.35.

Пример:

```
INSERT INTO suppliers (name, country) VALUES ('Гамма', 'Польша') RETURNING id;
-- узнаём новый id

UPDATE products SET stock = stock + 1 WHERE id = 3 RETURNING id, name, stock;  --
видим результат изменения
```

50.2. Задания

Рабочий каталог: `~/sql-lab/09_izmenenie_dannyh`. Задания выполняются *по порядку*: каждое опирается на результат предыдущих. Ответ, где он нужен, записывайте в файл `otvety/N`, где `N` — номер задания.

1. Добавьте в `customers` клиента: `id` = 1000, имя `Тестов Тест`, город `Тула`, почта `test@sever.example`, дата регистрации `2026-10-11`, `vip` = 0 (`INSERT INTO ... VALUES ...`).
2. Добавьте одной командой три товара категории `Тест`: `Тест-1` (цена 100), `Тест-2` (200), `Тест-3` (300); `id` 101, 102, 103; остаток 10; поставщик 1.
3. Командой `UPDATE ... SET ... WHERE` повысьте цену всех товаров категории `Книги` на 10 % (`price = price * 11 / 10`, целочисленно). Цены других категорий не должны измениться.
4. Командой `UPDATE ... SET ... WHERE` установите остаток `stock` равным 0 у всех товаров дороже 9000 (условие `price > 9000`). Остальные товары не должны измениться.
5. Поступило 10 единиц товара с `id` = 5. Одной командой `INSERT ... ON CONFLICT(id) DO UPDATE SET stock = stock + excluded.stock` увеличьте его остаток на 10; название и остальные поля менять нельзя, новых строк в `products` появиться не должно.
6. Командой `UPDATE ... SET ... WHERE id IN (подзапрос)` присвойте статус VIP (`vip = 1`) всем клиентам, у которых не менее четырёх заказов (подзапрос: `GROUP BY customer_id` и `HAVING COUNT(*) >= 4`). Остальные клиенты не должны измениться.
7. Создайте таблицу `orders_archive` как копию заказов, оформленных до `2026-01-01` (`CREATE TABLE ... AS SELECT * FROM orders WHERE ...`). Таблицу `orders` оставьте как есть.
8. Добавьте поставщика `Новый поставщик` (страна `Россия`) командой с `RETURNING id` и запишите возвращённый `id` в `otvety/8` (`.output otvety/8`, затем команда, затем `.output stdout`).
9. У нескольких клиентов город записан с опечаткой `Твер`. Найдите их (`SELECT ... WHERE city = 'Твер'`) и командой `UPDATE ... SET city = ... WHERE ...` исправьте на `Тверь`.
10. Двумя командами `DELETE FROM ... WHERE` удалите все заказы со статусом `cancelled` **вместе с их строками** в `order_items`: сначала строки (условие с подзапросом `order_id IN (SELECT ...)`), затем заказы. Других данных удалять нельзя, «осиротевших» строк остаться не должно.

Вопросы для размышления

1. Почему перед `UPDATE` и `DELETE` полезно сначала выполнить `SELECT` с тем же `WHERE`?
2. Чем удаление строк отличается от удаления таблицы, и как защитить базу от случайной потери данных?

Тема 10. Создание таблиц и ограничения

Общие сведения

Структуру таблицы описывает команда `CREATE TABLE`: имя, столбцы, их типы и **ограничения**. Ограничения — правила, которые база проверяет при каждой записи, и они надёжнее проверок в программе, потому что действуют для любого способа доступа. Основные: `PRIMARY KEY` (однозначный идентификатор строки; `INTEGER PRIMARY KEY` в SQLite нумеруется автоматически), `NOT NULL` (значение обязательно), `UNIQUE` (значения не повторяются), `DEFAULT` (значение по умолчанию), `CHECK(условие)` (произвольное условие, например `qty >= 0`), `REFERENCES другая_таблица(столбец)` (внешний ключ). У внешнего ключа задают действие при удалении родителя: `ON DELETE RESTRICT` (запретить), `CASCADE` (удалить дочерние), `SET NULL` (обнулить ссылку).

Типы столбцов в SQLite — «склонности» (INTEGER, REAL, TEXT, BLOB, NUMERIC); даты хранят текстом. Внешние ключи проверяются, *только если* в соединении выполнено `PRAGMA foreign_keys = ON;`.

Менять существующую таблицу позволяет `ALTER TABLE`: `ADD COLUMN`, `RENAME COLUMN`, `RENAME TO`, `DROP COLUMN`. Но добавить `CHECK`, `UNIQUE` к существующему столбцу или внешний ключ к готовому столбцу нельзя: в SQLite таблицу для этого *пересоздают* по стандартному рецепту: создать новую таблицу с нужными ограничениями, перенести данные (`INSERT ... SELECT`), удалить старую, переименовать новую, восстановить индексы — и всё это в одной транзакции. Команда `DROP TABLE` необратима, поэтому перед ней делают копию.

Примеры.

```
CREATE TABLE items (id INTEGER PRIMARY KEY, name TEXT NOT NULL, qty INTEGER NOT
NULL DEFAULT 0, price REAL);
CREATE UNIQUE INDEX items_name_uq ON items(name);

CREATE TABLE places (id INTEGER PRIMARY KEY, name TEXT NOT NULL UNIQUE);
ALTER TABLE items ADD COLUMN place_id INTEGER REFERENCES places(id) ON DELETE SET
NULL;
ALTER TABLE items RENAME COLUMN price TO unit_price;

-- добавить CHECK: пересоздать таблицу
BEGIN;
CREATE TABLE items_new (id INTEGER PRIMARY KEY, name TEXT NOT NULL,
                        qty INTEGER NOT NULL DEFAULT 0 CHECK (qty >= 0), price
                        REAL);
INSERT INTO items_new SELECT id, name, qty, price FROM items;
DROP TABLE items;
ALTER TABLE items_new RENAME TO items;
CREATE UNIQUE INDEX items_name_uq ON items(name);
COMMIT;
```

Что чаще всего идёт не так

При пересоздании таблицы забывают восстановить индексы и триггеры: они исчезают вместе со старой таблицей. Второй источник неприятностей — включённые внешние ключи при `DROP TABLE`: перед пересозданием родительской таблицы их временно отключают (`PRAGMA foreign_keys = OFF`) и затем проверяют `PRAGMA foreign_key_check`.

Рабочий каталог и проверка

Каталог полигона: `~/sql-lab/10_tablitsy`. Проверка: `python3 proverka.py 10`.

Склад компании ведёт учёт в таблице Excel, и это уже мешает: названия повторяются, остатки уходят в минус, непонятно, где что лежит. Руководитель поручил перенести учёт в базу данных SQLite. Вы будете **проектировать таблицы**: выбирать типы столбцов и ограничения (`PRIMARY KEY`, `NOT NULL`, `UNIQUE`, `CHECK`, `DEFAULT`, внешние ключи `REFERENCES`), добавлять и переименовывать столбцы (`ALTER TABLE`) и удалять ненужное (`DROP TABLE`). Работа идёт в новой базе `sklad.db`, в которой пока лежит только временная таблица `tmp_import`. Задания связаны: каждое развивает схему предыдущего, поэтому выполняйте их **по порядку**. Внешние ключи в SQLite проверяются, только если в соединении включено `PRAGMA foreign_keys = ON;` (в проверке оно включено).

Раздел 51. Создание таблиц и ограничения

51.1. Теоретические сведения

CREATE TABLE — создание таблицы

Назначение. Создаёт новую пустую таблицу: задаёт имена и типы столбцов и ограничения целостности. Структура таблицы (схема) хранится в самой базе.

Синтаксис:

```
CREATE TABLE [IF NOT EXISTS] таблица (
    столбец1 ТИП [ограничения],
    столбец2 ТИП [ограничения],
    ...
    [ограничения таблицы]
);
```

Правила:

- имя таблицы и столбцов — латиница, цифры и _, без пробелов; имя не должно начинаться с `sqlite_`;
- столбцы перечисляют через запятую, после последнего запятую не ставят;
- SQLite проверяет типы мягко: в столбец `INTEGER` можно записать строку, которая не похожа на число. Строгие типы дают `CREATE TABLE ... STRICT` (с версии 3.37);
- `IF NOT EXISTS` отменяет ошибку, если таблица уже есть (но не сравнивает её структуру с описанной);
- порядок столбцов в `CREATE TABLE` определяет порядок в `SELECT *` и в `INSERT` без списка столбцов;
- после создания список таблиц показывает `.tables`, а структуру — `.schema таблица` или `PRAGMA table_info(таблица);`.

Типы данных SQLite (аффинность столбца):

Элемент	Описание
`INTEGER`	целые числа; столбец `INTEGER PRIMARY KEY` нумеруется автоматически
`REAL`	вещественные числа (с плавающей точкой)
`TEXT`	строки; даты и время хранят строкой `'2026-10-11'` или `'2026-10-11 14:30:00'`
`BLOB`	двоичные данные без преобразования
`NUMERIC`	число, если значение похоже на число, иначе строка

Пример:

```
CREATE TABLE brands (
    id      INTEGER PRIMARY KEY,
    name    TEXT NOT NULL,
    founded INTEGER,
```

```
    rating  REAL
);

PRAGMA table_info(brands);                -- имена, типы и ограничения столбцов
```

Ограничения: PRIMARY KEY, NOT NULL, UNIQUE, CHECK, DEFAULT

Назначение. Ограничения — правила, которые база проверяет при каждой вставке и изменении. Неверные данные не попадают в таблицу вообще, независимо от того, какая программа в неё пишет.

Синтаксис:

```
столбец ТИП PRIMARY KEY
столбец ТИП NOT NULL
столбец ТИП UNIQUE
столбец ТИП CHECK (условие)
столбец ТИП DEFAULT значение

-- ограничения на уровне таблицы (в конце списка столбцов):
PRIMARY KEY (столбец1, столбец2)
UNIQUE (столбец1, столбец2)
CHECK (условие)
```

Правила:

- ограничение CHECK считается нарушенным только когда условие **ложно**; если оно даёт NULL, строка проходит, поэтому к CHECK добавляют NOT NULL;
- DEFAULT срабатывает только когда столбец **не указан** в INSERT; явная запись NULL значение по умолчанию не включает;
- DEFAULT CURRENT_TIMESTAMP подставляет текущие дату и время по UTC в виде строки '2026-10-11 14:30:00';
- при нарушении INSERT или UPDATE отменяются целиком, выдаётся ошибка вида UNIQUE constraint failed: items.name или CHECK constraint failed;
- составной ключ записывают отдельным предложением в конце списка столбцов (после всех столбцов);
- в таблице может быть только один PRIMARY KEY, а UNIQUE и CHECK — сколько угодно;
- ограничения можно комбинировать: qty INTEGER NOT NULL DEFAULT 0 CHECK (qty >= 0).

Ограничения:

Элемент	Описание
PRIMARY KEY	первичный ключ: значение уникально и идентифицирует строку; INTEGER PRIMARY KEY нумеруется автоматически
PRIMARY KEY (a, b)	составной ключ: уникальна **пара** значений, повторяться могут отдельные a и b
NOT NULL	значение обязательно: NULL не допускается
UNIQUE	значения в столбце не повторяются; NULL допускается и не считается повтором
CHECK (условие)	значение должно удовлетворять условию, например qty >= 0

Элемент	Описание
`DEFAULT значение`	что записать, если при вставке значение не указано; допустимо `CURRENT_TIMESTAMP`, `CURRENT_DATE`

Пример:

```
CREATE TABLE reviews (
    id       INTEGER PRIMARY KEY,
    author   TEXT NOT NULL,
    email    TEXT UNIQUE,
    stars    INTEGER NOT NULL CHECK (stars BETWEEN 1 AND 5),
    posted   TEXT NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE likes (
    review_id INTEGER NOT NULL,
    user      TEXT NOT NULL,
    PRIMARY KEY (review_id, user)           -- один пользователь лайкает отзыв
    один раз
);

INSERT INTO reviews (author, stars) VALUES ('Анна', 5);    -- email и posted
заполнятся сами или останутся NULL
INSERT INTO reviews (author, stars) VALUES ('Борис', 9);   -- ошибка: CHECK
constraint failed
```

CREATE INDEX — индексы

Назначение. Индекс — дополнительная структура, которая ускоряет поиск по столбцу. `UNIQUE`-индекс одновременно запрещает повторяющиеся значения, поэтому его используют как способ добавить уникальность уже существующей таблице.

Синтаксис:

```
CREATE [UNIQUE] INDEX [IF NOT EXISTS] имя_индекса ON таблица (столбец1 [, столбец2
...]);
DROP INDEX [IF EXISTS] имя_индекса;
```

Правила:

- обычный индекс ускоряет `WHERE`, `JOIN` и `ORDER BY` по этому столбцу, но замедляет вставку и занимает место;
- `CREATE UNIQUE INDEX` не удастся создать, если в столбце уже есть повторы: сначала устраните их;
- индекс из нескольких столбцов уникален по сочетанию значений;
- индекс принадлежит таблице и удаляется вместе с ней: при пересоздании таблицы его нужно создать заново;
- имена индексов принято делать понятными: `items_name_uq`, `orders_customer_idx`;
- список индексов видно в `.indexes` и в `.schema`.

Пример:

```
CREATE UNIQUE INDEX brands_name_uq ON brands (name);        -- название бренда не
повторяется
```

```
CREATE INDEX reviews_stars_idx ON reviews (stars);          -- ускорение выборок
по оценке
DROP INDEX reviews_stars_idx;
```

REFERENCES — внешние ключи

Назначение. Внешний ключ связывает столбец дочерней таблицы со столбцом родительской и гарантирует, что в нём нет значений, отсутствующих в родительской таблице («висячих ссылок»).

Синтаксис:

```
столбец ТИП [NOT NULL] REFERENCES родитель (столбец) [ON DELETE действие] [ON
UPDATE действие]

-- или отдельным предложением:
FOREIGN KEY (столбец) REFERENCES родитель (столбец) [ON DELETE действие]
```

Правила:

- в SQLite внешние ключи проверяются, только если в соединении выполнено `PRAGMA foreign_keys = ON;` (его нужно включать при каждом подключении);
- ссылаться нужно на столбец с `PRIMARY KEY` или `UNIQUE` родительской таблицы; родительскую таблицу создают раньше дочерней;
- `NULL` в столбце внешнего ключа допустим: он означает «связи нет» (если нет `NOT NULL`);
- `ON DELETE CASCADE` удобен, но опасен: одно `DELETE` может стереть цепочку связанных данных;
- родительскую таблицу нельзя удалить через `DROP TABLE`, пока на неё ссылаются строки (при включённых ключах);
- список внешних ключей таблицы покажет `PRAGMA foreign_key_list(таблица);`.

Действия при удалении или изменении родительской строки:

Элемент	Описание
`NO ACTION`	по умолчанию: операция, нарушающая связь, отклоняется с ошибкой
`RESTRICT`	запрет, как `NO ACTION`, но проверка срабатывает сразу
`SET NULL`	в дочерних строках значение заменяется на `NULL`; столбец не должен быть `NOT NULL`
`SET DEFAULT`	в дочерних строках ставится значение по умолчанию
`CASCADE`	дочерние строки удаляются (или изменяются) вместе с родительской

Пример:

```
PRAGMA foreign_keys = ON;

CREATE TABLE brand_models (
    id       INTEGER PRIMARY KEY,
    brand_id INTEGER NOT NULL REFERENCES brands (id) ON DELETE CASCADE,
    title    TEXT NOT NULL
);
```

```
INSERT INTO brand_models (brand_id, title) VALUES (999, 'Модель');   -- ошибка:
FOREIGN KEY constraint failed
```

ALTER TABLE — изменение таблицы

Назначение. Изменяет структуру существующей таблицы, не затрагивая данные: добавляет столбец, переименовывает столбец или саму таблицу.

Синтаксис:

```
ALTER TABLE таблица ADD COLUMN столбец ТИП [ограничения];
ALTER TABLE таблица RENAME COLUMN старое_имя TO новое_имя;
ALTER TABLE таблица RENAME TO новое_имя;
ALTER TABLE таблица DROP COLUMN столбец;
```

Правила:

- `ADD COLUMN` добавляет столбец в конец; у существующих строк в нём будет `NULL` или значение `DEFAULT`;
- добавляемый столбец с `NOT NULL` обязан иметь `DEFAULT`, не равный `NULL`; нельзя добавить столбец с `PRIMARY KEY` или `UNIQUE`;
- столбец с внешним ключом `REFERENCES` добавить можно, если у него `DEFAULT NULL` (то есть он допускает `NULL`);
- `RENAME COLUMN` и `RENAME TO` переписывают имя и в индексах, внешних ключах и представлениях (с версии 3.25);
- `DROP COLUMN` (с версии 3.35) не удалит столбец, который входит в ключ, индекс, `UNIQUE`, `CHECK` или внешний ключ;
- командой `ALTER TABLE` нельзя добавить `CHECK` или `UNIQUE`, изменить тип столбца: таблицу придётся пересоздать (следующая запись).

Пример:

```
ALTER TABLE brands ADD COLUMN site TEXT;
ALTER TABLE brands ADD COLUMN active INTEGER NOT NULL DEFAULT 1;
ALTER TABLE brands RENAME COLUMN rating TO score;
ALTER TABLE brands RENAME TO makers;
ALTER TABLE brand_models ADD COLUMN series_id INTEGER REFERENCES series (id) ON
DELETE SET NULL;   -- ссылка, допускающая NULL
```

Пересоздание таблицы и транзакция BEGIN … COMMIT

Назначение. Когда `ALTER TABLE` не умеет нужное изменение (добавить `CHECK`, изменить тип, поменять ключ), таблицу пересоздают: создают новую с нужной схемой, переносят данные, удаляют старую и переименовывают новую. Всё это оформляют **транзакцией**: либо выполнится целиком, либо не изменится ничего.

Синтаксис:

```
BEGIN;
CREATE TABLE таблица_new (... новая схема ...);
INSERT INTO таблица_new (столбцы) SELECT столбцы FROM таблица;
DROP TABLE таблица;
ALTER TABLE таблица_new RENAME TO таблица;
-- заново создать индексы
COMMIT;
```

Правила:

- порядок важен: сначала создать и заполнить новую таблицу, и только потом удалять старую;
- индексы и триггеры старой таблицы удаляются вместе с ней: создайте их заново после переименования;
- перенос данных `INSERT ... SELECT` выполняйте с перечислением столбцов в нужном порядке;
- если на таблицу ссылаются внешние ключи других таблиц, на время пересоздания выключают `PRAGMA foreign_keys = OFF;`, а после — включают снова;
- если что-то пошло не так до `COMMIT`, выполните `ROLLBACK;` — база останется как была; в SQLite `CREATE TABLE`, `DROP TABLE` и `ALTER TABLE` откатываются вместе с остальным;
- перед такой операцией сделайте копию файла базы.

Команды управления транзакцией:

Элемент	Описание
`BEGIN`	начать транзакцию: изменения пока не окончательные
`COMMIT`	зафиксировать все изменения транзакции
`ROLLBACK`	отменить все изменения с момента `BEGIN`

Пример:

```
BEGIN;
CREATE TABLE brands_new (
    id INTEGER PRIMARY KEY,
    name TEXT NOT NULL,
    founded INTEGER CHECK (founded > 1800),
    rating REAL
);
INSERT INTO brands_new (id, name, founded, rating) SELECT id, name, founded,
rating FROM brands;
DROP TABLE brands;
ALTER TABLE brands_new RENAME TO brands;
COMMIT;
```

DROP TABLE — удаление таблицы

Назначение. Удаляет таблицу вместе со всеми её данными, индексами и триггерами. В отличие от `DELETE`, исчезает не только содержимое, но и сама структура.

Синтаксис:

```
DROP TABLE [IF EXISTS] таблица;
```

Правила:

- действие необратимо (кроме открытой транзакции): перед удалением убедитесь, что таблица никому не нужна, и сделайте копию;
- без `IF EXISTS` попытка удалить несуществующую таблицу — ошибка; с `IF EXISTS` ошибки нет;

- при включённых внешних ключах нельзя удалить таблицу, на которую ссылаются строки других таблиц; сначала удаляют дочерние таблицы;
- очистка таблицы без удаления структуры — это `DELETE FROM таблица;`;
- место в файле базы после удаления освобождается командой `VACUUM;`.

Пример:

```
DROP TABLE IF EXISTS tmp_brands;
DROP TABLE brands;
```

Служебные команды sqlite3: .schema, .tables, .output

Назначение. Команды оболочки `sqlite3` начинаются с точки, пишутся без `;` и не относятся к языку SQL. Они показывают структуру базы и направляют вывод в файл — так можно сохранить схему и воссоздать базу.

Синтаксис:

```
.tables
.schema [таблица]
.output файл
.output stdout
.read файл
```

Правила:

- точка с запятой в конце служебных команд не нужна;
- сохранение схемы: `.output schema.sql`, затем `.schema`, затем `.output stdout`; без последней команды вывод продолжит уходить в файл;
- восстановление: `sqlite3 новая.db < schema.sql` или внутри оболочки `.read schema.sql`;
- то же без оболочки: `sqlite3 sklad.db .schema > schema.sql`;
- для данных, а не только структуры, служит `.dump`: он выдаёт и `CREATE`, и `INSERT`.

Команды:

Элемент	Описание
`.tables`	список таблиц базы
`.schema`	печатает команды `CREATE`, которыми создана база (или одна таблица, если указана)
`.output файл`	направляет вывод последующих команд в файл вместо экрана
`.output stdout`	возвращает вывод на экран
`.read файл`	выполняет команды из файла

Пример:

```
.tables
.schema items
.output schema.sql
.schema
.output stdout
```

51.2. Задания

Рабочий каталог: `~/sql-lab/10_tablitsy`. Задания выполняются *по порядку*: каждое опирается на результат предыдущих. Ответ, где он нужен, записывайте в файл `otvety/N`, где `N` — номер задания.

1. Командой `CREATE TABLE` создайте в `sklad.db` таблицу `items`: `id` — `INTEGER PRIMARY KEY`; `name` — `TEXT NOT NULL`; `qty` — `INTEGER NOT NULL DEFAULT 0`; `price` — `REAL` (в этом порядке).
2. Запретите повторяющиеся названия товаров: создайте уникальный индекс по `name` (`CREATE UNIQUE INDEX ...`). Затем добавьте два товара: `Болт` (остаток 100, цена 1.5) и `Гайка` (200, 0.5).
3. Добавьте ограничение `CHECK (qty >= 0)` для `qty` (запись ограничений — в справочнике). В SQLite его нельзя добавить командой `ALTER TABLE`, поэтому таблицу **пересоздают**: создайте `items_new` с ограничением, перенесите строки (`INSERT ... SELECT`), удалите `items`, переименуйте `items_new` в `items` и **заново создайте уникальный индекс** по `name`. Товары `Болт` и `Гайка` должны сохраниться.
4. Создайте таблицу `places` (`id` — `INTEGER PRIMARY KEY`, `name` — `TEXT NOT NULL UNIQUE`). Добавьте в `items` столбец `place_id` — ссылку на `places(id)` с действием `ON DELETE SET NULL` (`ALTER TABLE ... ADD COLUMN ... REFERENCES ...`).
5. Командой `ALTER TABLE ... ADD COLUMN` добавьте в `items` текстовый столбец `note` (тип `TEXT`).
6. Переименуйте столбец `price` таблицы `items` в `unit_price` (`ALTER TABLE ... RENAME COLUMN ... TO ...`).
7. Создайте таблицу `moves`: `id` — `INTEGER PRIMARY KEY`; `item_id` — целое, обязательное, внешний ключ (`REFERENCES items(id)`); `delta` — целое, обязательное, **не равное нулю** (`CHECK (delta <> 0)`); `moved_at` — текст, обязательный, по умолчанию текущее время (`DEFAULT CURRENT_TIMESTAMP`). Столбцы — именно в этом порядке.
8. Удалите временную таблицу `tmp_import` (`DROP TABLE`).
9. Создайте таблицу `item_tags` с составным первичным ключом (`PRIMARY KEY (item_id, tag)`): `item_id` (целое, обязательное, `REFERENCES items(id)`) и `tag` (текст, обязательный); пара не должна повторяться.
10. Сохраните схему базы в файл `schema.sql` (в каталоге темы): `.output schema.sql`, затем `.schema`. Файл должен позволять воссоздать пустую базу с таблицами `items`, `places`, `moves`, `item_tags` (и без `tmp_import`).

Вопросы для размышления

1. Какие правила целостности лучше задать в самой базе, а не в программе, которая в неё пишет, и почему?
2. Что делает `ON DELETE SET NULL`, и когда вместо него нужен `ON DELETE CASCADE` или запрет удаления?

Тема 11. Проектирование схемы и нормализация

Общие сведения

Плохо спроектированная таблица приводит к **аномалиям**. Если название отдела и город записаны в каждой строке сотрудника, то при смене города надо исправить много строк (*аномалия обновления*); опечатка в одной строке создаёт противоречие; нельзя добавить отдел, пока в нём нет сотрудников (*аномалия вставки*); удаление последнего сотрудника стирает сведения об отделе (*аномалия удаления*). Лечит их **нормализация**: разложение данных по таблицам так, чтобы каждый факт хранился один раз.

Нормальные формы формулируют критерии. **1НФ**: значения атомарны (в ячейке нет списка), строки различимы. **2НФ**: каждый неключевой столбец зависит от *всего* составного ключа, а не от его части. **3НФ**: неключевые столбцы зависят только от ключа, а не друг от друга (город отдела зависит от отдела, поэтому ему место в таблице отделов). На практике достаточно привычки: «одна сущность — одна таблица», связи оформляются внешними ключами.

Виды связей: *«один ко многим»* (отдел — сотрудники): внешний ключ в таблице «многих». *«Многие ко многим»* (сотрудники — навыки): отдельная таблица связи из двух внешних ключей, составной первичный ключ которой исключает повторы. *«Один к одному»*: редко; обычно это одна таблица, разделённая по соображениям доступа. Перенос данных из старой таблицы в новые выполняют последовательно (`INSERT ... SELECT` с `JOIN`), сверяют число и содержимое строк, а старую таблицу удаляют *последней*. Для обратной совместимости создают **представление**, воспроизводящее прежний вид.

Нормализация не догма. **Денормализацию** (сознательное дублирование) применяют для ускорения отчётов и архивов, но тогда ответственность за согласованность переходит к программе или триггерам.

Примеры.

```
-- было: sheet(row_id, employee, dept, dept_city, device, serial, issued)
CREATE TABLE departments (id INTEGER PRIMARY KEY, name TEXT NOT NULL UNIQUE, city
TEXT NOT NULL);
CREATE TABLE employees (id INTEGER PRIMARY KEY, name TEXT NOT NULL UNIQUE,
                        dept_id INTEGER NOT NULL REFERENCES departments(id));
INSERT INTO employees (name, dept_id)
SELECT DISTINCT s.employee, d.id FROM sheet s JOIN departments d ON d.name =
s.dept;

-- связь «многие ко многим»
CREATE TABLE employee_skills (employee_id INTEGER NOT NULL REFERENCES
employees(id),
                              skill_id    INTEGER NOT NULL REFERENCES skills(id),
                              PRIMARY KEY (employee_id, skill_id));
```

Что чаще всего идёт не так

Нормализуя, не теряйте строки: при переносе данных сравнивайте число строк и контрольные выборки до и после, и только потом удаляйте исходную таблицу. Предварительно найдите противоречия (запрос `GROUP BY ... HAVING COUNT(DISTINCT ...) > 1`): их придётся разрешить вручную.

Рабочий каталог и проверка

Каталог полигона: `~/sql-lab/11_proektirovanie`. Проверка: `python3 proverka.py 11`.

Сотрудник отдела кадров принёс «таблицу учёта техники», которую вёл несколько лет в Excel: в одной таблице `sheet` перемешаны сотрудники, отделы, города отделов, устройства и даты выдачи. Название отдела и его город повторяются в каждой строке, опечатка в одном месте приводит к противоречию, а сотрудника без техники в таблице вообще нет. Задача темы — **нормализовать** данные: разложить их по отдельным таблицам (отделы, сотрудники, устройства, выдачи), связать внешними ключами, перенести данные запросами `INSERT ... SELECT` и убедиться, что ничего не потеряно. База — `uchet.db`. Задания выполняются **по порядку**: каждое использует таблицы предыдущих.

Раздел 52. Проектирование схемы и нормализация

52.1. Теоретические сведения

COUNT(DISTINCT), HAVING — поиск противоречий

Назначение. Прежде чем разносить данные по таблицам, нужно найти в них аномалии: один и тот же факт (например, город отдела), записанный по-разному. Для этого группируют строки и считают число **различных** значений внутри группы.

Синтаксис:

```
SELECT ключ, COUNT(DISTINCT столбец) FROM таблица
GROUP BY ключ
HAVING COUNT(DISTINCT столбец) > 1;
```

Правила:

- `COUNT(DISTINCT x)` считает разные значения `x` в группе, а `COUNT(x)` — все непустые, включая повторы;
- `NULL` в `COUNT(x)` и `COUNT(DISTINCT x)` не учитывается, а `COUNT(*)` считает строки целиком;
- условие на результат агрегата пишут в `HAVING`, а не в `WHERE`: `WHERE` отбирает строки **до** группировки;
- если группа должна иметь один «правильный» вариант, его часто выбирают по частоте: сначала считают `COUNT(*)` по паре `(ключ, значение)`, потом берут пару с наибольшим счётчиком;
- «самое частое значение» выбирают так: подзапрос в `FROM` даёт счётчик `n` для каждой пары, а условие `WHERE n = (SELECT MAX(...) ...)` оставляет пару, чей счётчик равен наибольшему среди пар того же ключа (подзапрос ссылается на внешнюю строку через `t.ключ`); образец — в примере ниже;
- подзапрос в `FROM` обязательно получает псевдоним (`... ) t`), а коррелированный подзапрос в `WHERE` может ссылаться на столбцы внешней строки.

Пример:

```
-- устройства, у которых в таблице записано больше одной даты выдачи
SELECT serial, COUNT(DISTINCT issued) FROM sheet
GROUP BY serial HAVING COUNT(DISTINCT issued) > 1;

-- сколько раз каждая модель выдавалась в каждом городе
SELECT dept_city, device, COUNT(*) AS n FROM sheet
GROUP BY dept_city, device ORDER BY dept_city, n DESC;

-- самая частая модель в каждом городе: счётчик пары равен максимуму счётчиков
своего города
SELECT dept_city, device FROM (
    SELECT dept_city, device, COUNT(*) AS n FROM sheet GROUP BY dept_city, device
) t
WHERE n = (SELECT MAX(n2) FROM (
    SELECT COUNT(*) AS n2 FROM sheet WHERE dept_city = t.dept_city GROUP BY device
));
```

CREATE TABLE — ограничения столбцов

Назначение. Создаёт новую таблицу. Для нормализованной схемы важно не только перечислить столбцы, но и описать **ограничения**: база сама откажется принять данные, нарушающие правила.

Синтаксис:

```
CREATE TABLE имя (
    столбец тип [ограничения],
    ...,
    [PRIMARY KEY (столбец1, столбец2)]
);
```

Правила:

- ограничения пишут после типа столбца, через пробел: `name TEXT NOT NULL UNIQUE`;
- `INTEGER PRIMARY KEY` делает столбец псевдонимом внутреннего номера строки: при вставке без значения он получает следующий номер сам;
- составной первичный ключ (из двух и более столбцов) задают отдельной строкой `PRIMARY KEY (a, b)` в конце списка;
- в SQLite `PRIMARY KEY` в столбцах других типов, чем `INTEGER`, **допускает `NULL`**, если не добавить `NOT NULL`;
- типы в SQLite — «рекомендации»: значение другого типа будет принято, если не добавить `CHECK`;
- повторное `CREATE TABLE` с тем же именем даёт ошибку; `CREATE TABLE IF NOT EXISTS` её подавляет;
- изменить ограничения существующей таблицы командой `ALTER TABLE` нельзя: таблицу создают заново, переносят данные и удаляют старую.

Основные ограничения:

Элемент	Описание
`PRIMARY KEY`	первичный ключ: значения уникальны и идентифицируют строку; в таблице он один (возможно составной)
`NOT NULL`	значение обязательно: вставка `NULL` или пропуск столбца без `DEFAULT` даёт ошибку
`UNIQUE`	значения не повторяются (несколько `NULL` допускаются); в отличие от первичного ключа, `UNIQUE` может быть у многих столбцов
`DEFAULT значение`	значение, подставляемое при вставке, если столбец не указан
`CHECK (условие)`	произвольное условие для значения, например `CHECK (qty > 0)`
`REFERENCES таблица(столбец)`	внешний ключ, см. следующий раздел

Пример:

```
CREATE TABLE cities (
    id   INTEGER PRIMARY KEY,
    name TEXT NOT NULL UNIQUE
);
```

```
CREATE TABLE loans (
    device_serial TEXT NOT NULL,
    taken_on      TEXT NOT NULL,
    days          INTEGER NOT NULL DEFAULT 7 CHECK (days > 0),
    PRIMARY KEY (device_serial, taken_on)
);
INSERT INTO cities (name) VALUES ('Москва');
-- INSERT INTO cities (name) VALUES ('Москва');   -- ошибка: UNIQUE constraint
failed
```

REFERENCES — внешние ключи

Назначение. Внешний ключ связывает столбец дочерней таблицы с первичным ключом родительской и не даёт появиться «висячей» ссылке: сотруднику в несуществующем отделе или выдаче несуществующего устройства.

Синтаксис:

```
столбец тип REFERENCES родитель(ключ)
-- или отдельной строкой в конце CREATE TABLE:
FOREIGN KEY (столбец) REFERENCES родитель(ключ)

PRAGMA foreign_keys = ON;
```

Правила:

- по умолчанию SQLite внешние ключи **не проверяет**: `PRAGMA foreign_keys = ON;` нужно выполнять в каждом новом соединении;
- при включённой проверке вставка строки с отсутствующим в родителе значением даёт `FOREIGN KEY constraint failed`;
- по той же причине нельзя удалить родительскую строку, пока на неё ссылаются дочерние;
- родительская таблица должна существовать раньше дочерней, поэтому таблицы создают в порядке «справочники, затем то, что на них ссылается»;
- внешний ключ можно задать только при `CREATE TABLE`; к готовой таблице его не добавить;
- значение `NULL` во внешнем ключе допускается, если столбец не `NOT NULL`: оно означает «связи нет».

Пример:

```
PRAGMA foreign_keys = ON;
CREATE TABLE cities (id INTEGER PRIMARY KEY, name TEXT NOT NULL UNIQUE);
CREATE TABLE sites (
    id      INTEGER PRIMARY KEY,
    title   TEXT NOT NULL,
    city_id INTEGER NOT NULL REFERENCES cities(id)
);
INSERT INTO cities (name) VALUES ('Тверь');
INSERT INTO sites (title, city_id) VALUES ('Офис', 1);
-- INSERT INTO sites (title, city_id) VALUES ('Склад', 99);   -- ошибка: FOREIGN
KEY constraint failed
-- DELETE FROM cities WHERE id = 1;                            -- ошибка: на город
ссылается площадка
```

INSERT ... SELECT — перенос данных

Назначение. Вставляет в таблицу не заданные вручную значения, а результат запроса. Так данные переносят из «плоской» таблицы в нормализованные:

SELECT выбирает нужное, INSERT складывает по месту.

Синтаксис:

```
INSERT INTO таблица (столбец1, столбец2, ...)
SELECT выражение1, выражение2, ... FROM ... [JOIN ...] [WHERE ...];
```

Правила:

- число и порядок столбцов SELECT должны совпадать со списком столбцов INSERT; если список не указан, берутся все столбцы таблицы в порядке создания;
- столбец, оставленный вне списка (например, id INTEGER PRIMARY KEY), получает значение по умолчанию или следующий номер;
- SELECT DISTINCT оставляет по одной строке на уникальную комбинацию и защищает от нарушения UNIQUE при повторах в источнике;
- чтобы заменить текстовый ключ числовым идентификатором, источник соединяют (JOIN) со справочником по текстовому полю;
- если хоть одна строка нарушает ограничение, ошибка отменяет **всю** команду: в таблице не останется ни одной новой строки;
- порядок строк при вставке задаёт ORDER BY в SELECT; от него зависят номера, выдаваемые INTEGER PRIMARY KEY.

Пример:

```
CREATE TABLE cities (id INTEGER PRIMARY KEY, name TEXT NOT NULL UNIQUE);
INSERT INTO cities (name)
SELECT DISTINCT dept_city FROM sheet ORDER BY dept_city;

CREATE TABLE city_dev (city_id INTEGER NOT NULL REFERENCES cities(id), serial TEXT
NOT NULL);
INSERT INTO city_dev (city_id, serial)
SELECT c.id, s.serial FROM sheet s JOIN cities c ON c.name = s.dept_city;
```

Связь «многие ко многим»

Назначение. Когда одному сотруднику соответствует много навыков, а одному навыку — много сотрудников, их связывают отдельной **таблицей-связкой** из двух внешних ключей. Её первичный ключ составной: пара ключей.

Синтаксис:

```
CREATE TABLE связка (
    a_id INTEGER NOT NULL REFERENCES a(id),
    b_id INTEGER NOT NULL REFERENCES b(id),
    PRIMARY KEY (a_id, b_id)
);
```

Правила:

- составной ключ не даёт записать одну и ту же пару дважды;
- в связке можно хранить и свойства самой связи (дату, количество, роль);
- значения для связки удобно получать вложенным SELECT: INSERT INTO связка SELECT 1, id FROM b WHERE name = '...';
- связь «один ко многим» обходится без связки: внешний ключ ставят на стороне «многих».

Пример:

```
CREATE TABLE tags (id INTEGER PRIMARY KEY, name TEXT NOT NULL UNIQUE);
CREATE TABLE sheet_tags (
    row_id INTEGER NOT NULL REFERENCES sheet(row_id),
    tag_id INTEGER NOT NULL REFERENCES tags(id),
    PRIMARY KEY (row_id, tag_id)
);
INSERT INTO tags (name) VALUES ('срочно'), ('гарантия');
INSERT INTO sheet_tags (row_id, tag_id) SELECT 1, id FROM tags WHERE name IN
('срочно', 'гарантия');
-- INSERT INTO sheet_tags VALUES (1, 1);   -- ошибка: пара уже есть
```

Нормальные формы

Назначение. Нормальные формы — ступени «чистоты» таблицы. Каждая следующая устраняет определённый вид избыточности и вытекающих из неё аномалий вставки, обновления и удаления.

Синтаксис:

```
1НФ  значения атомарны, строки различимы (есть ключ)
2НФ  1НФ + каждый неключевой столбец зависит от ключа целиком
3НФ  2НФ + неключевые столбцы не зависят друг от друга
```

Правила:

- частичная зависимость возможна только при составном ключе: при ключе из одного столбца таблица в 1НФ автоматически и во 2НФ;
- чтобы определить форму, выпишите ключ и для каждого неключевого столбца спросите: «от чего он зависит на самом деле?»;
- типичный признак нарушения 3НФ — повторяющиеся от строки к строке пары «название — его атрибут» (отдел и город отдела);
- лечение всегда одно: вынести зависимые столбцы в отдельную таблицу с собственным ключом и сослаться на неё внешним ключом;
- ради скорости иногда осознанно оставляют данные ненормализованными, но тогда сами отвечают за их согласованность.

Нормальные формы и их нарушения:

Элемент	Описание
1НФ	нарушена, если в ячейке список («SQL, Linux») или нет ключа и строки дублируются
2НФ	нарушена, если часть составного ключа определяет неключевой столбец
3НФ	нарушена, если неключевой столбец определяется другим неключевым столбцом

Пример:

```
-- таблица заказов: ключ order_id; в строке: customer, customer_city
-- customer_city зависит от customer, а не от order_id -> нарушена 3НФ
-- решение: customers(id, name, city) и orders(id, customer_id REFERENCES
customers(id))
```

CREATE VIEW, DROP TABLE — представление и удаление

Назначение. После переноса данных нормализованную схему часто закрывают **представлением**, которое выглядит как прежняя таблица, а ненужную исходную таблицу удаляют.

Синтаксис:

```
CREATE VIEW имя AS SELECT ...;
DROP VIEW имя;
DROP TABLE имя;
```

Правила:

- представление хранит не данные, а текст запроса; при каждом обращении запрос выполняется заново;
- имена столбцов представления задают псевдонимами `AS` в списке `SELECT`;
- `DROP TABLE` необратимо удаляет таблицу вместе с данными, индексами и триггерами; перед ним проверяйте, что всё перенесено (сравните количество строк);
- при включённых внешних ключах таблицу, на которую ссылаются, удалить нельзя, пока в ссылающейся есть данные;
- `DROP TABLE IF EXISTS имя` не даёт ошибки, если таблицы уже нет.

Пример:

```
CREATE VIEW v_city_devices AS
SELECT dept_city AS city, device AS model, serial FROM sheet;
SELECT city, COUNT(*) FROM v_city_devices GROUP BY city;
DROP VIEW v_city_devices;
CREATE TABLE tmp_copy AS SELECT * FROM sheet;
DROP TABLE tmp_copy;
```

52.2. Задания

Рабочий каталог: `~/sql-lab/11_proektirovanie`. Задания выполняются *по порядку*: каждое опирается на результат предыдущих. Ответ, где он нужен, записывайте в файл `otvety/N`, где `N` — номер задания.

1. Найдите отделы, для которых в таблице `sheet` записано **больше одного** города: группировка `GROUP BY dept` и условие на агрегат `HAVING COUNT(DISTINCT dept_city) > 1`. Выведите название отдела и число различных городов. Результат — в `otvety/1` (`.output otvety/1` или `> otvety/1`).
2. Создайте таблицу `departments` командой `CREATE TABLE` (`id` — `INTEGER PRIMARY KEY`, `name` — `TEXT NOT NULL UNIQUE`, `city` — `TEXT NOT NULL`) и заполните её из `sheet` командой `INSERT ... SELECT` одной строкой на отдел. Если для отдела записано несколько городов, берите тот, который встречается **чаще**: подсчитайте `COUNT(*)` по паре `(dept, dept_city)` в подзапросе и оставьте пару, у которой счётчик равен максимальному в её отделе (коррелированный подзапрос с `MAX`, образец — в справочнике, запись «COUNT(DISTINCT), HAVING»).
3. Создайте `employees` командой `CREATE TABLE` (`id` — `INTEGER PRIMARY KEY`, `name` — `TEXT NOT NULL UNIQUE`, `dept_id` — `INTEGER NOT NULL` с внешним ключом `REFERENCES departments(id)`) и заполните её из `sheet` одной строкой на сотрудника: `INSERT ... SELECT DISTINCT` с соединением `JOIN departments` по названию отдела (так текстовое название заменяется числовым `id`).

4. Создайте `devices` командой `CREATE TABLE` (`id` — `INTEGER PRIMARY KEY`, `serial` — `TEXT NOT NULL UNIQUE`, `model` — `TEXT NOT NULL`) и заполните её из `sheet` командой `INSERT ... SELECT DISTINCT`: серийный номер (`serial`) и модель (столбец `device`).
5. Создайте `assignments` командой `CREATE TABLE` со столбцами `employee_id` (`INTEGER NOT NULL REFERENCES employees(id)`), `device_id` (`INTEGER NOT NULL REFERENCES devices(id)`), `issued_on` (`TEXT NOT NULL`) и **составным первичным ключом** `PRIMARY KEY (device_id, issued_on)` отдельной строкой в конце списка. Заполните её командой `INSERT ... SELECT` с двумя соединениями (`JOIN employees` по имени, `JOIN devices` по серийному номеру) так, чтобы каждая строка `sheet` превратилась в одну выдачу.
6. Проверьте перенос: выведите по новым таблицам (соединив `JOIN` таблицы `assignments`, `employees` и `devices`) имя сотрудника, модель, серийный номер и дату выдачи пяти устройств с наименьшими серийными номерами (`ORDER BY serial LIMIT 5`). Результат — в `otvety/6` (`.output otvety/6`); он должен совпасть с данными исходной таблицы.
7. Убедитесь, что база защищает связи. Включите `PRAGMA foreign_keys = ON;` и попробуйте (1) вставить `INSERT` сотрудника с несуществующим `dept_id` (например, 999) и (2) выполнить `DELETE` отдела, на который ссылаются сотрудники: обе команды должны дать ошибку `FOREIGN KEY constraint failed`. Ничего менять не требуется: если в заданиях 3 и 5 заданы `REFERENCES`, всё уже защищено. Задание засчитывается, когда это так.
8. Свяжите сотрудников с навыками (связь «многие ко многим» через таблицу-связку): `skills` (`id INTEGER PRIMARY KEY`, `name TEXT NOT NULL UNIQUE`) и `employee_skills` (`employee_id` и `skill_id` — внешние ключи `REFERENCES employees(id)` и `REFERENCES skills(id)`, **составной** `PRIMARY KEY (employee_id, skill_id)`). Добавьте навыки `SQL`, `Linux`, `Python` (`INSERT INTO ... VALUES`) и пары (номера — это `employees.id`; `skill_id` можно получить вложенным `SELECT id FROM skills WHERE name ...`): сотрудник 1 — SQL и Linux; сотрудник 2 — SQL; сотрудник 3 — Linux и Python.
9. Создайте представление `v_assignments`, которое возвращает столбцы `employee`, `dept`, `city`, `model`, `serial`, `issued` (в таком порядке) и воспроизводит прежнюю таблицу `sheet`, но по нормализованным данным: `CREATE VIEW ... AS SELECT` с соединением `JOIN` четырёх таблиц (`assignments`, `employees`, `departments`, `devices`), имена столбцов задайте псевдонимами `AS`; число строк должно совпасть с `sheet`.
10. Все данные перенесены. Удалите таблицу `sheet` командой `DROP TABLE`. Затем запишите в `otvety/10` одно число — номер высшей нормальной формы (1, 2 или 3), которой удовлетворяла таблица `sheet` **до** нормализации (число пишут как результат запроса: `.output otvety/10`, `SELECT N;`, `.output stdout`, где `N` — ваш ответ). Правила определения формы — в справочнике, запись «Нормальные формы»: первичный ключ у `sheet` — один столбец `row_id`; спросите себя, от чего зависит `dept_city`.

Вопросы для размышления

1. Какие аномалии (вставки, обновления, удаления) порождала исходная таблица, и как каждую из них устраняет нормализация?
2. В каких случаях разумно оставить данные ненормализованными (денормализовать) и чем за это приходится платить?

Тема 12. Индексы и планы запросов

Общие сведения

Чем больше таблица, тем дороже поиск перебором. Если в запросе `WHERE ip = '192.168.1.14'` нет подходящего индекса, SQLite просматривает таблицу целиком (`SCAN`) — время растёт пропорционально размеру. **Индекс** — дополнительная упорядоченная структура (дерево) по одному или нескольким столбцам, по которой нужные строки находят быстро (`SEARCH ... USING INDEX`); время растёт лишь логарифмически. Ценой становятся место на диске и замедление записи: при каждом `INSERT`, `UPDATE`, `DELETE` обновляются и индексы.

Как выбирает путь выполнения сам SQLite, показывает `EXPLAIN QUERY PLAN запрос`. Виды индексов: *простой* (по столбцу), *составной* (`(status, ts)`: порядок столбцов важен — индекс помогает запросам, где указан *начальный* отрезок столбцов; `WHERE status = 404 ORDER BY ts` им обслуживается без отдельной сортировки), *покрывающий* (содержит все столбцы, нужные запросу, и таблицу читать не надо: `COVERING INDEX`), *частичный* (`... WHERE условие`: хранит только часть строк), по *выражению* (`ON access(substr(ts, 1, 13))`: запрос должен использовать то же выражение). Уникальный индекс (`CREATE UNIQUE INDEX`) одновременно защищает от повторов.

Команда `ANALYZE` собирает статистику (таблица `sqlite_stat1`), по которой планировщик точнее выбирает индекс. Индексы, которые никто не использует, и дубликаты (одинаковые индексы под разными именами) удаляют (`DROP INDEX`). Индекс не помогает, если значение «сломано» функцией или шаблоном с `\%` в начале (`LIKE '%.14'`), а также при выборке большой доли таблицы: тогда проще просмотреть её целиком.

Примеры.

```
EXPLAIN QUERY PLAN SELECT * FROM access WHERE ip = '192.168.1.14';
--  `--SCAN access
CREATE INDEX idx_access_ip ON access(ip);
EXPLAIN QUERY PLAN SELECT * FROM access WHERE ip = '192.168.1.14';
--  `--SEARCH access USING INDEX idx_access_ip (ip=?)

CREATE INDEX idx_access_status_ts ON access(status, ts);          -- WHERE status
= 404 ORDER BY ts
CREATE INDEX idx_access_ip_path ON access(ip, path);              -- покрывающий
для SELECT path ... WHERE ip = ?
CREATE INDEX idx_access_err ON access(ip) WHERE status >= 400;    -- частичный
CREATE INDEX idx_access_hour ON access(substr(ts, 1, 13));        -- по выражению
ANALYZE;
DROP INDEX idx_status_b;
```

Что чаще всего идёт не так

Индекс «на всякий случай» по каждому столбцу — не решение: он замедляет запись и усложняет выбор плана. Индексы создают под *конкретные* запросы и проверяют по `EXPLAIN QUERY PLAN`, что они действительно используются.

Рабочий каталог и проверка

Каталог полигона: `~/sql-lab/12_indeksy`. Проверка: `python3 proverka.py 12`.

Журнал запросов к сайту вырос, и отчёты стали «задумываться». Причина почти всегда одна: чтобы найти строки с нужным адресом, SQLite просматривает **всю таблицу** (`SCAN`). **Индекс** — отдельная упорядоченная структура, по которой нужные строки находятся быстро (`SEARCH ... USING INDEX`). Посмотреть, как SQLite собирается выполнять запрос, позволяет команда `EXPLAIN QUERY PLAN`. В этой теме вы создаёте индексы разных видов (простой, составной, покрывающий, частичный, по выражению), читаете планы запросов, собираете статистику (`ANALYZE`) и удаляете лишнее: индексы ускоряют чтение, но замедляют запись и занимают место. База — `logs.db` (таблица `access`, 600 строк). Задания выполняются **по порядку**.

Раздел 53. Индексы и планы запросов

53.1. Теоретические сведения

EXPLAIN QUERY PLAN — план запроса

Назначение. Команда `EXPLAIN QUERY PLAN`, поставленная перед запросом, не выполняет его, а показывает, как SQLite собирается его выполнять: просматривать таблицу целиком или искать по индексу.

Синтаксис:

```
EXPLAIN QUERY PLAN запрос;
```

Правила:

- в выводе нас интересует столбец `detail` (последний): именно его текст и сохраняют как «план»;
- `SCAN таблица` — просмотр **всех** строк; на больших таблицах это медленно; `SCAN таблица USING COVERING INDEX имя` — тоже полный проход, но по (более узкому) индексу, а не по таблице: поиска по условию здесь нет;
- `SEARCH таблица USING INDEX имя (условие)` — быстрый поиск по индексу; в скобках указано, по какому условию идёт поиск: `(ip=?)` — равенство, `(ts>? AND ts<?)` — диапазон;
- `SEARCH ... USING COVERING INDEX` — все нужные столбцы взяты прямо из индекса, к самой таблице обращений нет;
- `USE TEMP B-TREE FOR ORDER BY` (или `GROUP BY`) — сортировка выполняется во временной структуре, потому что подходящего индекса нет;
- формулировки плана могут слегка различаться у разных версий SQLite, поэтому ищите по ключевым словам `SCAN`, `SEARCH`, `INDEX`;
- в консоли `sqlite3` результат выводится деревом: первая строка `QUERY PLAN`, ниже строки плана, каждая с префиксом из чёрточек `--` (сам план — текст после префикса); его можно сохранить в файл, как обычный запрос (`.output otvety/N`, затем команда, затем `.output stdout`, или перенаправление `>`);
- в плане важно слово в начале строки: `SCAN` — плохо (читается всё), `SEARCH` — хорошо (читается нужная часть); `COVERING` — ещё лучше (таблица не читается); `TEMP B-TREE` — запрос тратит время на сортировку.

Что встречается в плане:

Элемент	Описание
`SCAN`	полный просмотр таблицы (или индекса) строка за строкой
`SEARCH ... USING INDEX`	поиск по индексу с отбором по условию в скобках, например `(ip=?)`
`USING COVERING INDEX`	запрос целиком обслуживается индексом
`USE TEMP B-TREE FOR ORDER BY`	сортировка во временном дереве: признак того, что индекс порядок не обеспечивает

Пример:

```
EXPLAIN QUERY PLAN SELECT * FROM access WHERE method = 'POST';
-- QUERY PLAN
-- `--SCAN access                                        <- индекса по method нет:
читается вся таблица
EXPLAIN QUERY PLAN SELECT COUNT(*) FROM access WHERE id = 100;     -- по
первичному ключу: SEARCH access USING INTEGER PRIMARY KEY (rowid=?)
EXPLAIN QUERY PLAN SELECT * FROM access ORDER BY rt DESC;
-- |--SCAN access
-- `--USE TEMP B-TREE FOR ORDER BY                      <- сортировка во временной
структуре
```

CREATE INDEX, DROP INDEX — создание и удаление индекса

Назначение. Индекс — отдельная упорядоченная структура, в которой значения столбца хранятся вместе со ссылками на строки. Поиск по индексу не просматривает всю таблицу, но каждый индекс занимает место и замедляет `INSERT`, `UPDATE`, `DELETE`.

Синтаксис:

```
CREATE [UNIQUE] INDEX [IF NOT EXISTS] имя_индекса ON таблица (столбец1 [ASC |
DESC], столбец2, ...);
DROP INDEX [IF EXISTS] имя_индекса;
PRAGMA index_list(таблица);
```

Правила:

- имя индекса уникально в пределах всей базы; принято писать `idx_таблица_столбцы`;
- первичный ключ и ограничение `UNIQUE` автоматически создают собственные индексы (`sqlite_autoindex_...`), повторно по ним индекс не нужен;
- `UNIQUE` в индексе, кроме ускорения, запрещает повторы значений;
- два одинаковых индекса бесполезны: они занимают место и замедляют запись, а пользы не добавляют;
- список индексов таблицы: `PRAGMA index_list(таблица);` или команда оболочки `.indexes таблица`; состав индекса показывает `PRAGMA index_info(индекс);`;
- `DROP INDEX` удаляет только индекс, данные таблицы не затрагиваются;
- индекс на небольшой таблице (сотни строк) почти не влияет на скорость: выигрыш виден на тысячах и миллионах строк.

Пример:

```
CREATE INDEX idx_access_method ON access(method);
PRAGMA index_list(access);          -- видно, что индекс появился
EXPLAIN QUERY PLAN SELECT * FROM access WHERE method = 'POST';
DROP INDEX idx_access_method;
```

Составной и покрывающий индекс

Назначение. Индекс может строиться по нескольким столбцам сразу. Порядок столбцов в нём важен: строки упорядочены по первому, внутри равных — по второму и так далее. Если в индексе есть все столбцы запроса, таблицу читать не нужно: такой индекс называется **покрывающим**.

Синтаксис:

```
CREATE INDEX имя ON таблица (a, b, c);
```

Правила:

- индекс `(a, b)` работает для условий по `a` и по паре `a, b`, но **не** помогает запросу только по `b` (правило «левого префикса»);
- условие на равенство ставят первыми столбцами, диапазон или столбец сортировки — следующим: тогда строки идут уже в нужном порядке и `TEMP B-TREE` не нужен;
- покрывающим индекс получается, когда все столбцы из `SELECT`, `WHERE` и `ORDER BY` перечислены в нём; в плане это `USING COVERING INDEX`;
- лишние столбцы в индексе делают его больше и замедляют запись, поэтому «покрывают» только самые частые запросы;
- `SELECT *` покрывающий индекс практически никогда не обслуживает.

Пример:

```
CREATE INDEX idx_a_method_ts ON access(method, ts);
EXPLAIN QUERY PLAN SELECT * FROM access WHERE method = 'GET' ORDER BY ts;     --
сортировки нет
CREATE INDEX idx_a_ip_bytes ON access(ip, bytes);
EXPLAIN QUERY PLAN SELECT bytes FROM access WHERE ip = '192.168.1.11';        --
COVERING INDEX
```

Частичный индекс

Назначение. Частичный индекс содержит только строки, удовлетворяющие условию `WHERE`. Он компактнее обычного и ускоряет запросы, которые интересуются именно этим подмножеством строк.

Синтаксис:

```
CREATE INDEX имя ON таблица (столбец) WHERE условие;
```

Правила:

- SQLite использует частичный индекс, только если условие запроса **гарантированно** включает условие индекса (например, индекс `WHERE status >= 400`, запрос `WHERE status = 500 AND ip = ...`);
- в условии индекса можно использовать только столбцы этой таблицы и детерминированные функции; подзапросы и `date('now')` нельзя;
- индекс обновляется только при вставке и изменении подходящих строк, поэтому запись замедляется меньше, чем с полным индексом;
- полезен, когда нужных строк мало (ошибки, неоплаченные заказы, неудалённые записи).

Пример:

```
CREATE INDEX idx_a_slow ON access(path) WHERE rt > 1;
EXPLAIN QUERY PLAN SELECT path FROM access WHERE rt > 1 AND path = '/api/report';
```

Индекс по выражению

Назначение. Индекс можно построить не по столбцу, а по значению выражения или функции. Тогда запрос с **тем же** выражением ищет по индексу, а не вычисляет функцию для каждой строки.

Синтаксис:

```
CREATE INDEX имя ON таблица (выражение);
```

Правила:

- выражение в запросе должно быть записано **точно так же**, как в индексе, иначе индекс не будет использован;
- в выражении допускаются только детерминированные функции (`substr`, `lower`, `length`), но не `random()` и не `date('now')`;
- обычный индекс по `ts` не помогает условию `substr(ts, 1, 10) = ...`: функция «прячет» столбец от индекса;
- проверить использование индекса можно командой `EXPLAIN QUERY PLAN`.

Пример:

```
CREATE INDEX idx_a_lower_path ON access(lower(path));
EXPLAIN QUERY PLAN SELECT * FROM access WHERE lower(path) = '/shop';
CREATE INDEX idx_a_day ON access(substr(ts, 1, 10));
EXPLAIN QUERY PLAN SELECT substr(ts, 1, 10), COUNT(*) FROM access GROUP BY
substr(ts, 1, 10);
```

Условия, которые использует индекс

Назначение. Не всякое условие в `WHERE` способно работать с индексом. Индекс упорядочен по значениям, поэтому он ускоряет сравнения и диапазоны, но бессилен там, где значение сначала нужно преобразовать.

Синтаксис:

```
WHERE столбец = значение
WHERE столбец >= a AND столбец < b
WHERE столбец BETWEEN a AND b
WHERE столбец GLOB 'префикс*'
```

Правила:

- индекс используют `=`, `<`, `<=`, `>`, `>=`, `BETWEEN`, `IN`; для диапазона по времени в текстовом формате `ГГГГ-ММ-ДД ЧЧ:ММ:СС` достаточно индекса по этому столбцу;
- поиск по началу строки: `GLOB 'абв*'` использует обычный индекс (в плане диапазон `path>? AND path<?`), а `LIKE 'абв%'` на столбце без `COLLATE NOCASE` — нет, потому что `LIKE` не различает регистр латиницы; `LIKE '%абв'` и `GLOB '*абв'` индекс не используют никогда;
- функция над столбцом (`WHERE substr(ts, 1, 10) = ...`, `WHERE price * 2 > 100`) отключает обычный индекс: перепишите условие или создайте индекс по выражению;
- сравнение `<>` и условия вида `NOT ...` обычно приводят к полному просмотру;
- в диапазоне верхнюю границу чаще задают строго (`< '2026-10-11 06'`), чтобы не терять строки на границе.

Пример:

```
CREATE INDEX idx_a_rt ON access(rt);
EXPLAIN QUERY PLAN SELECT * FROM access WHERE rt >= 5 AND rt < 10;       -- SEARCH
... USING INDEX
EXPLAIN QUERY PLAN SELECT * FROM access WHERE rt * 2 > 10;               -- SCAN:
столбец внутри выражения
CREATE INDEX idx_a_path ON access(path);
EXPLAIN QUERY PLAN SELECT * FROM access WHERE path GLOB '/shop*';        --
SEARCH: поиск по началу строки
```

ANALYZE — статистика для планировщика

Назначение. ANALYZE собирает сведения о распределении значений в индексах (сколько строк в среднем приходится на значение) и записывает их в служебную таблицу sqlite_stat1. По ним планировщик выбирает, какой индекс выгоднее.

Синтаксис:

```
ANALYZE;
ANALYZE таблица_или_индекс;
SELECT * FROM sqlite_stat1;
```

Правила:

- статистика не обновляется автоматически: после массовых изменений данных ANALYZE запускают заново;
- sqlite_stat1 создаётся самой командой и содержит строки только для таблиц, у которых есть индексы;
- столбец stat содержит числа: сначала число строк таблицы, затем среднее число строк на значение первого, второго... столбца индекса;
- команда PRAGMA optimize; перед закрытием соединения обновляет статистику только там, где это действительно нужно.

Пример:

```
CREATE INDEX idx_a_status ON access(status);
ANALYZE;
SELECT tbl, idx, stat FROM sqlite_stat1;
```

53.2. Задания

Рабочий каталог: ~/sql-lab/12_indeksy. Задания выполняются *по порядку*: каждое опирается на результат предыдущих. Ответ, где он нужен, записывайте в файл otvety/N, где N — номер задания.

1. Узнайте, как SQLite выполняет запрос SELECT * FROM access WHERE ip = '192.168.1.14'; (команда EXPLAIN QUERY PLAN, поставленная перед запросом, сам запрос не выполняет). Сохраните вывод в otvety/1: в консоли .output otvety/1, затем команда, затем .output stdout (или в оболочке sqlite3 logs.db 'запрос;' > otvety/1). В нём должно быть SCAN access: таблица просматривается целиком.
2. Создайте индекс по столбцу ip таблицы access командой CREATE INDEX имя ON таблица(столбец); имя выберите сами по образцу idx_таблица_столбец.

3. Снова получите EXPLAIN QUERY PLAN того же запроса и сохраните вывод в otvety/3 (как в задании 1). Теперь в нём должна быть строка вида SEARCH access USING INDEX ... (ip=?) (или USING COVERING INDEX), а слова SCAN быть не должно.
4. Запрос SELECT * FROM access WHERE status = 404 ORDER BY ts; сортирует результат во временной структуре (USE TEMP B-TREE FOR ORDER BY). Создайте **составной** индекс по двум столбцам (status, ts) (сначала столбец с условием =, затем столбец сортировки), чтобы сортировка не понадобилась. Проверьте EXPLAIN QUERY PLAN этого запроса: строки USE TEMP B-TREE FOR ORDER BY в плане быть не должно.
5. Запрос SELECT path FROM access WHERE ip = ? читает только ip и path. Создайте составной индекс по двум столбцам (ip, path): все столбцы запроса окажутся в индексе, и таблицу читать вообще не придётся. Проверьте EXPLAIN QUERY PLAN SELECT path FROM access WHERE ip = '192.168.1.14';: в плане должно быть USING COVERING INDEX.
6. Создайте **частичный** индекс по ip только для ошибочных запросов: CREATE INDEX имя ON access(ip) WHERE status >= 400; Он меньше обычного и ускоряет отчёты об ошибках (условие status >= 400 пишется после списка столбцов).
7. Часто нужна статистика по часам: GROUP BY substr(ts, 1, 13). Создайте индекс по выражению: CREATE INDEX имя ON access(substr(ts, 1, 13)) (выражение в скобках записывается точно так же, как в запросе).
8. Соберите статистику для планировщика командой ANALYZE; (появится служебная таблица sqlite_stat1).
9. Кто-то создал два одинаковых индекса по status (idx_status_a и idx_status_b). Лишний — вред: он занимает место и замедляет запись. Посмотрите индексы таблицы (PRAGMA index_list(access); или .indexes access) и командой DROP INDEX имя удалите лишний, чтобы остался ровно один индекс по одному столбцу status.
10. Создайте обычный индекс по ts (CREATE INDEX имя ON access(ts)), затем получите EXPLAIN QUERY PLAN запроса SELECT COUNT(*) FROM access WHERE ts >= '2026-10-11 05' AND ts < '2026-10-11 06'; Сохраните вывод в otvety/10 (как в задании 1); в плане должен быть SEARCH ... USING INDEX (диапазон ts>? AND ts<?), а не SCAN.

Вопросы для размышления

1. Почему лишние индексы вредны, и как понять, какой индекс на самом деле не используется?
2. Что произойдёт с индексом по ip, если в запросе написать WHERE ip LIKE '%.14', и как с этим бороться?

Тема 13. Представления и триггеры

Общие сведения

Представление (`CREATE VIEW имя AS запрос`) — сохранённый запрос, который выглядит как таблица. Оно ничего не хранит (данные берутся из базовых таблиц при каждом обращении), зато скрывает сложные соединения, даёт «публичный» вид данных и позволяет менять устройство таблиц, не ломая запросы пользователей. Представления в SQLite обычно только для чтения; запись можно разрешить триггером `INSTEAD OF`.

Триггер (`CREATE TRIGGER`) — набор команд, которые база выполняет автоматически при `INSERT`, `UPDATE` или `DELETE`: до события (`BEFORE`), после (`AFTER`) или вместо него (`INSTEAD OF`, для представлений). Внутри доступны значения `OLD` (до изменения) и `NEW` (после). Необязательное условие `WHEN` ограничивает срабатывание; `UPDATE OF столбец` — только при изменении указанного столбца. Для отмены операции вызывают `SELECT RAISE(ABORT, 'сообщение')`.

Что реализуют триггерами: журнал изменений (кто и когда изменил цену), поддержку производных значений (списание остатка при продаже), правила, которые нельзя выразить ограничениями (запрет удаления клиента с заказами, неизменность даты). Платой за удобство становится «невидимость»: поведение базы зависит от триггеров, о которых легко забыть, поэтому список представлений и триггеров (`SELECT type, name FROM sqlite_master WHERE type IN ('view', 'trigger')`) входит в документацию любой базы. Триггеры пробуют на копии базы.

Примеры.

```
CREATE VIEW v_customer_revenue AS
SELECT c.id AS customer_id, c.name AS name, SUM(i.qty * i.price) AS revenue
FROM customers c JOIN orders o ON o.customer_id = c.id JOIN order_items i ON
i.order_id = o.id
WHERE o.status <> 'cancelled' GROUP BY c.id;

CREATE TRIGGER trg_price_log AFTER UPDATE OF price ON products WHEN OLD.price <>
NEW.price
BEGIN
    INSERT INTO price_log (product_id, old_price, new_price) VALUES (OLD.id,
    OLD.price, NEW.price);
END;

CREATE TRIGGER trg_stock_nonneg BEFORE UPDATE OF stock ON products WHEN NEW.stock
< 0
BEGIN SELECT RAISE(ABORT, 'остаток не может быть отрицательным'); END;
```

Точка с запятой внутри тела триггера обязательна, а завершающая после `END` — конец всей команды. В консоли `sqlite3` многострочный триггер вводят целиком до `END;`.

Что чаще всего идёт не так

Триггеры, изменяющие таблицы, на которые срабатывают другие триггеры, образуют цепочки, которые трудно отлаживать. Держите триггеры корот-

кими, а сложную логику — в программе.

Рабочий каталог и проверка

Каталог полигона: `~/sql-lab/13_predstavleniya_triggery`. Проверка: `python3 proverka.py 13`.

Бухгалтерия каждый день просит один и тот же сложный запрос («сумма каждого заказа с именем клиента»), а разработчики хотят, чтобы база **сама** следила за правилами: записывала, кто и когда менял цену, не давала удалить клиента с заказами, списывала остаток при продаже. Для первого служат **представления** (`CREATE VIEW`): сохранённый запрос, который выглядит как таблица. Для второго — **триггеры** (`CREATE TRIGGER`): команды, которые база выполняет автоматически до или после `INSERT`, `UPDATE`, `DELETE`. Триггер может записать сообщение в журнал, изменить другую таблицу или отменить операцию (`SELECT RAISE(ABORT, 'сообщение')`). Внутри триггера доступны значения `OLD` (до изменения) и `NEW` (после). Задания выполняются в `shop.db`; проверка пробует ваши триггеры на временной копии базы.

Раздел 54. Представления и триггеры

54.1. Теоретические сведения

CREATE VIEW, DROP VIEW — представление

Назначение. Представление — сохранённый запрос SELECT, к которому обращаются как к таблице. Оно не хранит данных, а при каждом обращении выполняет запрос заново на актуальных данных.

Синтаксис:

```
CREATE VIEW [IF NOT EXISTS] имя AS SELECT ...;
DROP VIEW [IF EXISTS] имя;
```

Правила:

- имена столбцов представления берутся из списка SELECT; для вычисляемых столбцов задавайте псевдонимы через AS;
- к представлению обращаются как к таблице: SELECT ... FROM имя WHERE ... ORDER BY ..., его можно соединять с другими таблицами;
- в SQLite представление **только для чтения**: INSERT, UPDATE, DELETE по нему дают ошибку, пока нет триггера INSTEAD OF;
- сложный запрос с JOIN и GROUP BY в представлении удобен для повторного использования, но выполняется при каждом обращении;
- изменить представление нельзя: его удаляют и создают заново;
- представление не копирует данные, поэтому изменения в таблицах сразу видны в нём.

Пример:

```
CREATE VIEW v_big_items AS
SELECT order_id, product_id, qty * price AS amount FROM order_items WHERE qty *
price >= 5000;
SELECT order_id, amount FROM v_big_items ORDER BY amount DESC LIMIT 3;
DROP VIEW v_big_items;
```

CREATE TRIGGER — триггер

Назначение. Триггер — набор команд, который база выполняет автоматически до или после INSERT, UPDATE, DELETE над указанной таблицей. Так правила выносят из программ в саму базу.

Синтаксис:

```
CREATE TRIGGER [IF NOT EXISTS] имя
{BEFORE | AFTER | INSTEAD OF} {INSERT | UPDATE [OF столбец, ...] | DELETE}
ON таблица_или_представление
[FOR EACH ROW]
[WHEN условие]
BEGIN
    команда1;
    команда2;
END;
DROP TRIGGER [IF EXISTS] имя;
```

Правила:

- триггер срабатывает **для каждой изменяемой строки** (в SQLite другого режима нет);
- `BEFORE` выполняется до изменения и подходит для проверок и отказов, `AFTER` — после и подходит для журналов и каскадных изменений;
- `UPDATE OF столбец` ограничивает срабатывание изменением этого столбца; без `OF` триггер срабатывает при любом `UPDATE`;
- `WHEN условие` — дополнительная проверка: если она ложна, тело не выполняется;
- тело состоит из команд `INSERT`, `UPDATE`, `DELETE`, `SELECT`, каждая обязательно завершается `;`, после последней идёт `END;`;
- в `sqlite3` команду `CREATE TRIGGER` можно вводить в несколько строк: оболочка понимает, что `;` внутри `BEGIN ... END` ещё не конец;
- триггер принадлежит таблице: при `DROP TABLE` он удаляется вместе с ней;
- в `WHEN` и в теле можно использовать подзапрос, например `WHEN EXISTS (SELECT 1 FROM orders WHERE customer_id = OLD.id)`; в `INSERT` внутри тела столбцы перечисляют явно: `INSERT INTO журнал (a, b) VALUES (OLD.id, NEW.x)`;
- действия триггера выполняются в той же транзакции, что и вызвавшая его команда: при отказе откатывается всё.

Когда срабатывает триггер:

Элемент	Описание
`BEFORE INSERT`, `BEFORE UPDATE`, `BEFORE DELETE`	до изменения строки: проверки и запреты
`AFTER INSERT`, `AFTER UPDATE`, `AFTER DELETE`	после изменения: журнал, пересчёт, связанные таблицы
`INSTEAD OF ...`	вместо команды; допустим только на представлении
`UPDATE OF столбец`	только при изменении указанного столбца

Пример:

```
CREATE TABLE note_log (id INTEGER PRIMARY KEY, product_id INTEGER, note TEXT, at
TEXT DEFAULT CURRENT_TIMESTAMP);
CREATE TRIGGER trg_demo_cat AFTER UPDATE OF category ON products
BEGIN
    INSERT INTO note_log (product_id, note) VALUES (NEW.id, 'категория изменена');
END;
UPDATE products SET category = 'Книги' WHERE id = 1;
SELECT product_id, note FROM note_log;
DROP TRIGGER trg_demo_cat;
```

OLD и NEW — значения до и после

Назначение. Внутри триггера доступны две «строки»: `OLD` — как строка выглядела до изменения, `NEW` — как будет выглядеть после. Из них берут значения для журнала, проверок и вторичных изменений.

Синтаксис:

```
OLD.столбец
NEW.столбец
```

Правила:

- в триггере на INSERT есть только NEW, на DELETE — только OLD, на UPDATE — обе;
- обращаться к недоступной строке (например, к OLD в триггере на INSERT) бессмысленно: значения не определены, SQLite этого не заметит при создании;
- в BEFORE-триггере NEW ещё не записано в таблицу, в AFTER — уже записано;
- WHEN OLD.x <> NEW.x отсекает «пустые» обновления, при которых значение не изменилось; при NULL такое сравнение даёт NULL, поэтому используют IS NOT: OLD.x IS NOT NEW.x;
- изменить NEW напрямую нельзя; чтобы изменить вставляемую строку, после неё выполняют UPDATE в AFTER-триггере.

Пример:

```
CREATE TABLE stock_log (product_id INTEGER, was INTEGER, became INTEGER);
CREATE TRIGGER trg_stock_log AFTER UPDATE OF stock ON products WHEN OLD.stock IS
NOT NEW.stock
BEGIN
    INSERT INTO stock_log VALUES (OLD.id, OLD.stock, NEW.stock);
END;
UPDATE products SET stock = stock + 10 WHERE id = 2;
SELECT * FROM stock_log;
```

RAISE — отмена операции

Назначение. Функция RAISE внутри триггера прерывает выполнение команды и сообщает об ошибке. На ней строятся проверки данных: «этого делать нельзя».

Синтаксис:

```
SELECT RAISE(ABORT, 'сообщение');
SELECT RAISE(IGNORE);
```

Правила:

- RAISE допустим только в теле триггера, причём в команде SELECT: SELECT RAISE(ABORT, '...');;
- ABORT отменяет текущую команду и все её изменения, транзакция при этом продолжается; пользователь получает сообщение об ошибке;
- IGNORE молча пропускает текущую строку и остальные действия триггера;
- ROLLBACK и FAIL также существуют: ROLLBACK откатывает всю транзакцию, FAIL оставляет уже сделанные изменения;
- чаще всего RAISE ставят в BEFORE-триггер вместе с условием WHEN, чтобы запрет действовал только в нужных случаях;
- сообщение пишут в одинарных кавычках: именно его видит программист при ошибке.

Пример:

```
CREATE TRIGGER trg_no_free_goods BEFORE UPDATE OF price ON products WHEN NEW.price
<= 0
BEGIN
    SELECT RAISE(ABORT, 'цена должна быть положительной');
END;
UPDATE products SET price = price + 1 WHERE id = 3;     -- допустимо
-- UPDATE products SET price = 0 WHERE id = 3;           -- ошибка: цена должна
быть положительной
```

INSTEAD OF — запись через представление

Назначение. Триггер `INSTEAD OF` вешается на представление и срабатывает **вместо** попытки вставить, изменить или удалить его строки. Так «только для чтения» представление можно сделать записываемым: триггер сам выполняет действия над настоящими таблицами.

Синтаксис:

```
CREATE TRIGGER имя INSTEAD OF {INSERT | UPDATE | DELETE} ON представление
BEGIN
    команды над таблицами;
END;
```

Правила:

- `INSTEAD OF` допустим только для представлений; на обычной таблице он вызывает ошибку, а на представлении недопустимы `BEFORE` и `AFTER`;
- в теле доступны `NEW` (вставляемые или новые значения) и `OLD` (для `UPDATE` и `DELETE`);
- столбцы, которых нет в представлении, триггер заполняет сам: значением по умолчанию, `date('now')`, константой;
- без триггера любая запись в представление завершается ошибкой `cannot modify ... because it is a view`.

Пример:

```
CREATE VIEW v_names AS SELECT id, name FROM customers;
CREATE TRIGGER trg_v_names_upd INSTEAD OF UPDATE ON v_names
BEGIN
    UPDATE customers SET name = NEW.name WHERE id = OLD.id;
END;
UPDATE v_names SET name = 'Тестов Тест' WHERE id = 1;
SELECT name FROM customers WHERE id = 1;
```

CURRENT_TIMESTAMP, date('now') — дата и время

Назначение. Для автоматической отметки времени в SQLite есть значение `CURRENT_TIMESTAMP` и функции даты и времени. Они часто стоят в `DEFAULT` журнальных таблиц и в телах триггеров.

Синтаксис:

```
столбец TEXT NOT NULL DEFAULT CURRENT_TIMESTAMP
date('now')
datetime('now')
```

Правила:

- `CURRENT_TIMESTAMP` даёт текущие дату и время (UTC) строкой `ГГГГ-ММ-ДД ЧЧ:ММ:СС`;
- `date('now')` возвращает только дату `ГГГГ-ММ-ДД`, `datetime('now')` — дату и время; для местного времени добавляют модификатор: `datetime('now', 'localtime')`;
- `DEFAULT` срабатывает только когда столбец не указан в `INSERT`, поэтому в триггере-журнале его просто не перечисляют;

- в DEFAULT выражение нужно брать в скобки, если это не простое значение или CURRENT_TIMESTAMP: DEFAULT (date('now')).

Пример:

```
CREATE TABLE audit (id INTEGER PRIMARY KEY, what TEXT, at TEXT NOT NULL DEFAULT
CURRENT_TIMESTAMP, day TEXT DEFAULT (date('now')));
INSERT INTO audit (what) VALUES ('проверка');
SELECT what, length(at), length(day) FROM audit;
```

sqlite_master — каталог объектов базы

Назначение. Служебная таблица sqlite_master перечисляет все объекты базы: таблицы, индексы, представления и триггеры вместе с текстом, которым они были созданы.

Синтаксис:

```
SELECT type, name, tbl_name, sql FROM sqlite_master [WHERE ...] [ORDER BY ...];
```

Правила:

- type принимает значения table, index, view, trigger;
- name — имя объекта, tbl_name — таблица, к которой он относится, sql — текст команды создания;
- таблица доступна только для чтения: объекты создают и удаляют командами CREATE и DROP, а не правкой этих строк;
- автоматически созданные индексы (sqlite_autoindex_...) тоже попадают в список, но с пустым sql;
- для вывода в сортированном виде добавляют ORDER BY type, name;
- результат запроса из оболочки ОС сохраняют в файл перенаправлением: sqlite3 shop.db "SELECT type, name FROM sqlite_master;" > файл (по умолчанию столбцы разделяет |).

Пример:

```
SELECT type, name FROM sqlite_master WHERE type IN ('view', 'trigger') ORDER BY
type, name;
SELECT type, COUNT(*) FROM sqlite_master GROUP BY type;
SELECT sql FROM sqlite_master WHERE name = 'orders';
```

54.2. Задания

Рабочий каталог: ~/sql-lab/13_predstavleniya_triggery. Задания выполняются *по порядку*: каждое опирается на результат предыдущих. Ответ, где он нужен, записывайте в файл otvety/N, где N — номер задания.

1. Создайте представление v_order_totals со столбцами order_id, customer (имя клиента), order_date, status, total (сумма qty * price по строкам заказа). Заказы без строк в представлении не нужны. Конструкция: CREATE VIEW ... AS SELECT ... (соединение JOIN, GROUP BY o.id, псевдонимы столбцов через AS).

2. Создайте представление `v_customer_revenue` со столбцами `customer_id`, `name`, `revenue` — выручка по каждому клиенту без учёта отменённых заказов (клиенты без заказов не нужны). Конструкция: `CREATE VIEW ... AS SELECT ...`, отменённые заказы исключите условием `WHERE o.status <> 'cancelled'` внутри представления.
3. Выведите имена и выручку трёх клиентов с наибольшей выручкой (по `v_customer_revenue`), при равенстве — по `customer_id`. Обращайтесь к представлению как к таблице: `SELECT ... FROM v_customer_revenue ORDER BY ... LIMIT 3`. Результат запишите в `otvety/3` перенаправлением `>`.
4. Заведите таблицу `price_log` (`id` — первичный ключ, `product_id`, `old_price`, `new_price`, `changed_at` — текст со значением по умолчанию `CURRENT_TIMESTAMP`; в таком порядке) и триггер `AFTER UPDATE OF price ON products`, который при **изменении** цены записывает старое и новое значения. Если цена не изменилась или менялось другое поле, запись добавляться не должна. Нужны: `CREATE TABLE`, условие `WHEN OLD.price <> NEW.price` и в теле триггера `INSERT INTO price_log (...) VALUES (OLD.id, OLD.price, NEW.price)` между `BEGIN` и `END;`.
5. Создайте триггер `BEFORE DELETE ON customers`, который отменяет удаление клиента, если у него есть заказы: `SELECT RAISE(ABORT, 'у клиента есть заказы')`. Клиенты без заказов удаляются свободно. Условие срабатывания — `WHEN EXISTS (SELECT 1 FROM orders WHERE customer_id = OLD.id)` (подзапрос с `OLD.id`, см. справочник `CREATE TRIGGER`).
6. Создайте триггер `AFTER INSERT ON order_items`, который уменьшает остаток товара (`products.stock`) на проданное количество (`NEW.qty`). В теле триггера выполните `UPDATE products SET stock = ... WHERE id = NEW.product_id;`.
7. Создайте триггер `BEFORE UPDATE OF stock ON products`, который отменяет изменение, если новый остаток отрицательный (`WHEN NEW.stock < 0`), с помощью `SELECT RAISE(ABORT, 'сообщение');` в теле.
8. Создайте представление `v_customers_simple` (`id`, `name`, `city`) и триггер `INSTEAD OF INSERT` на нём, который добавляет клиента в `customers` с сегодняшней датой регистрации (`date('now')`) и `vip = 0` (в теле: `INSERT INTO customers (name, city, registered, vip) VALUES (NEW.name, NEW.city, ...)`). Тогда `INSERT INTO v_customers_simple (name, city) VALUES (...)` будет работать.
9. Создайте триггер, запрещающий изменять `registered` у клиентов (`BEFORE UPDATE OF registered ON customers, RAISE(ABORT, ...)`). Изменение других полей должно оставаться возможным.
10. Запишите в `otvety/10` список созданных вами представлений и триггеров: тип и имя, по одной паре в строке, по типу и имени (запрос `SELECT type, name FROM sqlite_master WHERE type IN ('view', 'trigger') ORDER BY type, name;`). Из оболочки: `sqlite3 shop.db "запрос" > otvety/10`.

Вопросы для размышления

1. Чем представление отличается от временной таблицы, и когда оно может оказаться медленным?
2. Какие правила стоит реализовать триггерами, а какие — в прикладной программе, и чем опасны «невидимые» триггеры?

Тема 14. Транзакции, целостность и PRAGMA

Общие сведения

Транзакция — группа команд, которая выполняется как одно целое: либо все изменения применяются (`COMMIT`), либо ни одно (`ROLLBACK`). Классический пример — перевод денег: списание и зачисление нельзя разделять. Свойства транзакций описывает аббревиатура **ACID**: *атомарность* (всё или ничего), *согласованность* (после транзакции выполнены все ограничения), *изолированность* (параллельные транзакции не видят чужих промежуточных состояний), *долговечность* (зафиксированное переживает сбой). SQLite соблюдает ACID с помощью журнала: перед изменением файла записывается журнал отката (по умолчанию) или журнал упреждающей записи (`WAL`).

Вне явной транзакции каждая команда фиксируется сама. Явную начинают `BEGIN` (в сценариях часто `BEGIN IMMEDIATE`, если сразу известно, что будет запись). **Точки сохранения** (`SAVEPOINT имя`, `ROLLBACK TO имя`, `RELEASE имя`) позволяют отменять часть транзакции. Нарушение ограничения (`CHECK`, `UNIQUE`, внешнего ключа) вызывает ошибку и отменяет команду; что делать с остальной транзакцией, решает вызывающий (чаще всего — откатить).

`PRAGMA` — служебные команды SQLite для чтения и изменения настроек. Полезные: `foreign_keys` (проверка внешних ключей; по умолчанию *выключена*, включают в каждом соединении), `journal_mode` (`DELETE` по умолчанию, `WAL` позволяет читать во время записи и хранится в файле базы), `user_version` (число, которое приложение использует как версию схемы), `integrity_check` (проверка целостности; у исправной базы результат `ok`), `page_size`, `page_count` (размер файла равен произведению), `busy_timeout` (сколько ждать, если база занята).

Примеры.

```
BEGIN;
UPDATE accounts SET balance = balance - 500 WHERE id = 1;
UPDATE accounts SET balance = balance + 500 WHERE id = 2;
COMMIT;                          -- или ROLLBACK; чтобы отменить обе операции

BEGIN;
INSERT INTO accounts (owner, balance) VALUES ('Тест A', 100);
SAVEPOINT s1;
INSERT INTO accounts (owner, balance) VALUES ('Тест B', 200);
ROLLBACK TO s1;                  -- отменено только добавление Тест B
COMMIT;

PRAGMA foreign_keys;             -- 0 в новом соединении
PRAGMA foreign_keys = ON;
PRAGMA journal_mode = WAL;
PRAGMA user_version = 3;
PRAGMA integrity_check;          -- ok
```

Что чаще всего идёт не так

Длинная транзакция блокирует других писателей. Оставленный открытым `BEGIN` без `COMMIT` — частая причина ошибки «database is locked» у соседней программы: завершайте транзакции быстро.

Рабочий каталог и проверка

Каталог полигона: `~/sql-lab/14_tranzaktsii`. Проверка: `python3 proverka.py 14`.

Касса «Северного ветра» хранит остатки на внутренних счетах в базе `bank.db`. Перевод между счетами — это **две** операции: списать у одного и зачислить другому. Если база «упадёт» между ними, деньги исчезнут. Чтобы такого не было, связанные изменения объединяют в **транзакцию**: либо выполняются все (`COMMIT`), либо ни одной (`ROLLBACK`). Транзакции — основа **ACID**: атомарность, согласованность, изолированность, долговечность. В этой теме вы выполняете переводы, откатываете изменения, вызываете ошибки ограничений, читаете и меняете параметры базы командой `PRAGMA` (внешние ключи, режим журнала `WAL`, версия схемы, проверка целостности) и пользуетесь точками сохранения (`SAVEPOINT`). Ответы, полученные в `sqlite3`, записывайте в файлы командой `.output` или перенаправлением.

Раздел 55. Транзакции, целостность и PRAGMA

55.1. Теоретические сведения

BEGIN, COMMIT, ROLLBACK — транзакция

Назначение. Транзакция объединяет несколько команд в одну неделимую операцию: либо применяются все, либо ни одна. Именно так делают переводы денег, где списание и зачисление нельзя разделять.

Синтаксис:

```
BEGIN [DEFERRED | IMMEDIATE | EXCLUSIVE];
команды;
COMMIT;       -- зафиксировать
ROLLBACK;     -- отменить
```

Правила:

- вне явной транзакции каждая команда выполняется как отдельная, сразу сохраняется и откатить её уже нельзя;
- `COMMIT` (синоним `END`) делает изменения постоянными и видимыми другим соединениям; до него видны только вашему соединению;
- `ROLLBACK` возвращает базу к состоянию на момент `BEGIN`;
- если команда внутри транзакции завершилась ошибкой ограничения, транзакция остаётся открытой: решайте сами, продолжать или `ROLLBACK`;
- незавершённая транзакция при закрытии соединения откатывается;
- вложенных `BEGIN` нет: повторный `BEGIN` внутри транзакции — ошибка; для вложенности служат точки сохранения;
- транзакции поддерживают и команды изменения структуры (`CREATE TABLE`, `DROP TABLE`): в SQLite их тоже можно откатить.

Виды BEGIN:

Элемент	Описание
`BEGIN` или `BEGIN DEFERRED`	блокировки берутся лениво: при первом чтении или записи (по умолчанию)
`BEGIN IMMEDIATE`	сразу занимает блокировку записи; другие соединения могут только читать
`BEGIN EXCLUSIVE`	сразу блокирует базу целиком (в режиме `WAL` равносильна `IMMEDIATE`)

Пример:

```
BEGIN;
UPDATE accounts SET balance = balance - 100 WHERE id = 1;
UPDATE accounts SET balance = balance + 100 WHERE id = 2;
COMMIT;

BEGIN;
```

```
DELETE FROM accounts WHERE id = 4;
SELECT COUNT(*) FROM accounts;      -- 3
ROLLBACK;
SELECT COUNT(*) FROM accounts;      -- снова 4
```

SAVEPOINT, RELEASE, ROLLBACK TO — точки сохранения

Назначение. Точка сохранения — именованная отметка внутри транзакции. К ней можно вернуться, не отменяя всю транзакцию, и тем самым откатывать только часть сделанного.

Синтаксис:

```
SAVEPOINT имя;
ROLLBACK [TRANSACTION] TO [SAVEPOINT] имя;
RELEASE [SAVEPOINT] имя;
```

Правила:

- `ROLLBACK TO имя` отменяет изменения после точки, но сама точка и транзакция остаются: можно продолжать работу;
- `RELEASE имя` убирает точку (и все вложенные) и **сохраняет** сделанное; окончательно оно фиксируется при `COMMIT`;
- точки можно вкладывать друг в друга; откат к внешней точке отменяет и внутренние;
- если `SAVEPOINT` выполнен вне транзакции, он сам начинает транзакцию; её завершают `RELEASE` внешней точки или `COMMIT`;
- имена точек можно повторять: действует самая недавняя точка с таким именем.

Пример:

```
BEGIN;
INSERT INTO accounts (owner, balance) VALUES ('Демо 1', 10);
SAVEPOINT a;
INSERT INTO accounts (owner, balance) VALUES ('Демо 2', 20);
ROLLBACK TO a;                          -- 'Демо 2' отменена, 'Демо 1' осталась
RELEASE a;
COMMIT;
SELECT owner FROM accounts WHERE id > 4;
```

CHECK и ошибки ограничений

Назначение. Ограничение `CHECK` проверяет каждое новое значение столбца. Нарушающая его команда отвергается целиком, а база остаётся в прежнем состоянии: это один из механизмов **согласованности** из набора ACID.

Синтаксис:

```
CREATE TABLE имя (столбец тип CHECK (условие), ...);
```

Правила:

- сообщение об ошибке имеет вид `CHECK constraint failed: условие` и появляется при нарушении как при `INSERT`, так и при `UPDATE`;

- ошибка отменяет только саму команду; внутри открытой транзакции остальные изменения сохраняются;
- сообщения об ошибках `sqlite3` печатает в поток **ошибок**: чтобы сохранить их в файл, перенаправляют `2>`, а не `>`;
- аналогично себя ведут `NOT NULL constraint failed`, `UNIQUE constraint failed`, `FOREIGN KEY constraint failed`;
- `NULL` проходит `CHECK`: условие с `NULL` не считается нарушенным; требуется непустое значение — добавляйте `NOT NULL`.

Пример:

```
CREATE TABLE wallets (id INTEGER PRIMARY KEY, amount INTEGER NOT NULL CHECK
(amount >= 0));
INSERT INTO wallets (amount) VALUES (100);
UPDATE wallets SET amount = amount - 30 WHERE id = 1;       -- допустимо
-- UPDATE wallets SET amount = amount - 500 WHERE id = 1;   -- Runtime error:
CHECK constraint failed: amount >= 0
```

PRAGMA — чтение и изменение параметров

Назначение. `PRAGMA` — служебная команда SQLite: читает и меняет параметры базы и соединения, а также выдаёт служебные сведения о схеме. Это не стандарт SQL, а особенность SQLite.

Синтаксис:

```
PRAGMA параметр;
PRAGMA параметр = значение;
PRAGMA параметр(аргумент);
```

Правила:

- без `=` команда читает значение, с `=` — устанавливает;
- одни параметры действуют только на текущее **соединение** (например, `foreign_keys`), другие записываются в **файл базы** (`journal_mode = WAL`, `user_version`);
- название параметра регистр не различает; несуществующий параметр молча игнорируется, поэтому опечатка в названии не даёт ошибки;
- результат читающего `PRAGMA` — обычная таблица, её можно вывести в файл как результат запроса;
- список всех индексов, столбцов и ключей таблицы выдают `PRAGMA index_list(т)`, `PRAGMA table_info(т)`, `PRAGMA foreign_key_list(т)`.

Параметры, используемые в модуле:

Элемент	Описание
`foreign_keys`	проверка внешних ключей: 0 — выключена (по умолчанию), 1 — включена
`journal_mode`	режим журнала: `DELETE` (по умолчанию), `WAL` и другие
`user_version`	целое число для номера версии схемы, хранится в заголовке файла
`integrity_check`	проверка целостности файла базы
`page_size`, `page_count`	размер страницы в байтах и число страниц

Элемент	Описание
`table_info(таблица)`	столбцы таблицы: имя, тип, `NOT NULL`, значение по умолчанию, участие в первичном ключе

Пример:

```
PRAGMA table_info(accounts);
PRAGMA foreign_key_list(accounts);
PRAGMA encoding;
```

PRAGMA foreign_keys — проверка внешних ключей

Назначение. Включает или выключает проверку внешних ключей (REFERENCES). Без неё SQLite принимает ссылки на несуществующие строки.

Синтаксис:

```
PRAGMA foreign_keys;          -- 0 или 1
PRAGMA foreign_keys = ON;     -- включить
PRAGMA foreign_keys = OFF;
```

Правила:

- по умолчанию проверка **выключена**, и результат чтения равен `0`;
- параметр действует только на текущее соединение и сбрасывается при его закрытии: включать нужно в каждом новом сеансе;
- внутри открытой транзакции параметр менять нельзя: команда молча ничего не делает;
- включённая проверка не проверяет задним числом уже записанные данные; найти нарушения помогает `PRAGMA foreign_key_check;`.

Пример:

```
PRAGMA foreign_keys;              -- 0
PRAGMA foreign_keys = ON;
PRAGMA foreign_keys;              -- 1
```

PRAGMA journal_mode — режим журнала

Назначение. Журнал хранит прежнее или новое содержимое страниц на время транзакции, чтобы при сбое можно было восстановить базу. Режим журнала определяет способ такой защиты и поведение при одновременной работе.

Синтаксис:

```
PRAGMA journal_mode;
PRAGMA journal_mode = DELETE | WAL | TRUNCATE | PERSIST | MEMORY | OFF;
```

Правила:

- команда возвращает **новый** режим строкой, например `wal`; по этому ответу видно, принял ли SQLite запрос;

- DELETE — режим по умолчанию: журнал-отката создаётся на время транзакции и удаляется;
- WAL (журнал упреждающей записи) позволяет читать во время записи; режим сохраняется в файле базы и сохраняется между соединениями;
- в режиме WAL рядом с базой появляются файлы имя-wal и имя-shm; при копировании базы их надо переносить вместе с ней;
- для переключения в WAL не должно быть открытых транзакций, а базе нужна файловая система с общей памятью (не сетевая папка);
- MEMORY и OFF ускоряют работу, но при сбое база может быть повреждена.

Пример:

```
PRAGMA journal_mode;            -- delete
PRAGMA journal_mode = WAL;      -- возвращает wal
PRAGMA journal_mode;            -- wal
```

PRAGMA user_version, integrity_check, page_size, page_count — сведения о базе

Назначение. Группа параметров, которые хранят номер версии схемы и помогают проверить состояние файла базы и оценить его размер.

Синтаксис:

```
PRAGMA user_version;
PRAGMA user_version = n;
PRAGMA integrity_check;
PRAGMA page_size;
PRAGMA page_count;
```

Правила:

- user_version — целое число, которое SQLite само никогда не меняет; приложения хранят в нём номер версии схемы для миграций;
- integrity_check обходит всю базу; для исправной базы возвращает одну строку ok, иначе — перечень найденных проблем;
- более быстрая проверка без сверки индексов — PRAGMA quick_check;;
- размер файла равен page_size * page_count;
- page_size можно изменить только до создания первых таблиц (или командой VACUUM), поэтому для существующей базы параметр служит для чтения.

Параметры:

Элемент	Описание
user_version	пользовательский номер версии схемы; читается без =, пишется с = n
integrity_check	проверка структуры базы и индексов; ожидаемый ответ ok
page_size	размер страницы в байтах; обычно 4096
page_count	общее число страниц в файле базы

Пример:

```
PRAGMA user_version = 2;
```

```
PRAGMA user_version;                    -- 2
PRAGMA integrity_check;                 -- ok
SELECT p.page_size * c.page_count AS bytes FROM pragma_page_size p,
pragma_page_count c;   -- размер файла
```

Сохранение результатов: .output, .once, перенаправление

Назначение. Задания модуля требуют записывать результаты запросов и сообщения об ошибках в файлы. Для этого служат команды оболочки `sqlite3` и перенаправление потоков операционной системы.

Синтаксис:

```
.output файл          -- весь дальнейший вывод в файл
.output stdout        -- вернуть вывод на экран
.once файл            -- только вывод следующей команды
sqlite3 база "запрос" > файл     -- вывод результата в файл
sqlite3 база "запрос" 2> файл    -- только сообщения об ошибках
```

Правила:

- точечные команды (`.output`, `.once`, `.headers`, `.mode`) не относятся к SQL, не заканчиваются `;` и работают только в оболочке `sqlite3`;
- результат запроса идёт в стандартный вывод, сообщение об ошибке — в стандартный поток ошибок, поэтому `>` и `2>` сохраняют разное;
- после `.output файл` нужно вернуть вывод на экран командой `.output stdout`, иначе файл останется открытым;
- несколько команд в одном вызове `sqlite3 база "BEGIN; ...; COMMIT;"` выполняются в одном соединении, поэтому транзакция и `PRAGMA foreign_keys = ON;` действуют на все следующие в этом вызове команды; следующий вызов `sqlite3` начинает с чистого соединения;
- результат `PRAGMA` (например, `PRAGMA integrity_check;` даёт `ok`) сохраняется так же, как результат запроса: `sqlite3 bank.db "PRAGMA integrity_check;" > файл`;
- в PowerShell перенаправление `>` создаёт файл в кодировке UTF-16; для чистого текста используйте `| Out-File -Encoding utf8` или `.output`;
- для вывода без оформления используйте режим по умолчанию (`.mode list`), заголовки включает `.headers on`.

Пример:

```
-- внутри sqlite3:
.once result.txt
SELECT COUNT(*) FROM accounts;
-- из оболочки ОС:
-- sqlite3 bank.db "SELECT SUM(balance) FROM accounts;" > total.txt
```

55.2. Задания

Рабочий каталог: `~/sql-lab/14_tranzaktsii`. Задания выполняются *по порядку*: каждое опирается на результат предыдущих. Ответ, где он нужен, записывайте в файл `otvety/N`, где `N` — номер задания.

1. Переведите 500 со счёта 1 на счёт 2 **одной транзакцией**: `BEGIN;`, два `UPDATE`, `COMMIT;`. После перевода у счёта 1 должно быть 500, у счёта 2 — 2500,

общая сумма остаться равной 5000. Команды можно передать одним вызовом: `sqlite3 bank.db "BEGIN; UPDATE ...; UPDATE ...; COMMIT;"` (формы `UPDATE accounts SET balance = balance - N WHERE id = K`).

2. В транзакции спишите 300 со счёта 3 (`UPDATE ... WHERE id = 3`), убедитесь запросом, что остаток стал 1200, а затем отмените изменение командой `ROLLBACK;`. Запишите остаток счёта 3 **после** отката в `otvety/2`. Транзакция живёт только в одном сеансе, поэтому откат выполняйте в интерактивном `sqlite3 bank.db` (или одним вызовом `BEGIN; UPDATE ...; ROLLBACK;`), а затем отдельно: `sqlite3 bank.db "SELECT balance FROM accounts WHERE id = 3;" > otvety/2`.
3. Попытайтесь списать со счёта 4 сумму 9999 (остаток станет отрицательным). Таблица защищена `CHECK (balance >= 0)`, поэтому SQLite откажет. Сохраните **текст ошибки** в `otvety/3` (в оболочке: `sqlite3 bank.db "UPDATE ..." 2> otvety/3`). Остаток счёта 4 должен остаться 500. Команда: `UPDATE accounts SET balance = balance - 9999 WHERE id = 4;`.
4. В свежем сеансе `sqlite3` выполните `PRAGMA foreign_keys;` и запишите результат в `otvety/4a`. Затем включите проверку (`PRAGMA foreign_keys = ON;`), повторите запрос и запишите результат в `otvety/4b`. Из оболочки: `sqlite3 bank.db "PRAGMA foreign_keys;" > otvety/4a`, а для 4b включение и чтение нужны в одном вызове (параметр живёт только в соединении): `sqlite3 bank.db "PRAGMA foreign_keys = ON; PRAGMA foreign_keys;" > otvety/4b`.
5. Переключите базу `bank.db` в режим журнала WAL (`PRAGMA journal_mode = WAL;`). Режим сохраняется в файле базы: новое соединение должно показывать `wal` (проверка: `PRAGMA journal_mode;`).
6. Запишите в базу номер версии схемы 3: `PRAGMA user_version = 3;` (так принято отслеживать миграции); прочитать значение обратно — `PRAGMA user_version;`.
7. Проверьте целостность базы (`PRAGMA integrity_check;`) и запишите результат в `otvety/7`. У исправной базы единственная строка ответа — `ok`, иначе выводится перечень найденных проблем. Из оболочки: `sqlite3 bank.db "PRAGMA integrity_check;" > otvety/7`.
8. Узнайте размер страницы (`PRAGMA page_size;`) и число страниц (`PRAGMA page_count;`) базы `bank.db`; запишите первое значение в `otvety/8a`, второе — в `otvety/8b` (так же, как в заданиях 4 и 7: `sqlite3 bank.db "PRAGMA page_size;" > otvety/8a`).
9. В одной транзакции добавьте счета: `Тест A` (100), затем `SAVEPOINT s1`, счёт `Тест B` (200), `SAVEPOINT s2`, счёт `Тест C` (300), затем `ROLLBACK TO s2;`, `RELEASE s1;` и `COMMIT;`. В итоге в таблице должны остаться `Тест A` и `Тест B`, но не `Тест C`. Вводите всё в одном сеансе (интерактивный `sqlite3 bank.db` или один вызов со всеми командами); счета добавляйте как `INSERT INTO accounts (owner, balance) VALUES ('Тест A', 100);`.
10. Подсчитайте общую сумму остатков на всех счетах после всех операций (`SELECT SUM(balance) FROM accounts;`) и запишите в `otvety/10`: `sqlite3 bank.db "SELECT SUM(balance) FROM accounts;" > otvety/10`.

Вопросы для размышления

1. Что произошло бы при переводе денег без транзакции, если питание выключится между двумя `UPDATE`?
2. Почему режим `WAL` удобнее для приложений с одновременным чтением и записью, и какие у него ограничения?

Тема 15. Резервные копии, обмен данными и обслуживание базы

Общие сведения

База SQLite — обычный файл, но *копировать его работающим нельзя*: если в момент копирования программа пишет в базу, копия окажется внутренне противоречивой. Для консистентной копии есть команда `.backup файл` (то же через программный интерфейс), а текстовую копию, пригодную для переноса между системами, делает `.dump`: результат — SQL-скрипт из команд `CREATE` и `INSERT`. Копия бесполезна, пока не проверено её **восстановление**: скрипт выполняют в новую базу (`sqlite3 restored.db < dump.sql`, `.read`) и сравнивают число строк, а целостность копии проверяет `PRAGMA integrity_check`. Копию делают регулярно, называют с датой (`backup-2026-10-11.db`), хранят отдельно от рабочей и удаляют по правилу хранения.

Обмен данными — обычно файлы CSV. Экспорт: `.headers on`, `.mode csv`, `.output файл.csv`, `SELECT ...`. Импорт: `.import --csv файл.csv таблица` (если таблицы нет, она создаётся по заголовку, все столбцы текстовые; после импорта типы приводят запросами). Перед массовым импортом данные загружают во *временную* таблицу, проверяют и лишь потом переносят в рабочую.

Обслуживание. Удаление строк и таблиц не уменьшает файл: освободившиеся страницы остаются внутри. Команда `VACUUM` перестраивает файл без пустот (`VACUUM INTO 'файл'` записывает компактную копию отдельно); ей нужно свободное место, примерно равное размеру базы. **Миграции** схемы оформляют нумерованными SQL-файлами (`migrate_001.sql`, `migrate_002.sql`), каждый из которых заканчивается установкой `PRAGMA user_version`; применять их нужно по порядку и один раз, а перед применением — сделать копию.

Примеры.

```
sqlite3 shop.db ".backup backup-2026-10-11.db"
sqlite3 shop.db .dump > dump.sql
sqlite3 restored.db < dump.sql                        # восстановление «в другое
место»
sqlite3 backup-2026-10-11.db "PRAGMA integrity_check;"

sqlite3 shop.db
sqlite> .headers on
sqlite> .mode csv
sqlite> .output customers.csv
sqlite> SELECT * FROM customers;
sqlite> .output stdout
sqlite> .import --csv new_products.csv products_new
sqlite> DROP TABLE junk;
sqlite> VACUUM;
sqlite> VACUUM INTO 'compact.db';
sqlite> .read migrate_001.sql
```

Что чаще всего идёт не так

Классическая ошибка — копировать `shop.db` командой `cp` во время работы приложения; другая — не проверять копию. Третья — забыть про сопутствующие файлы режима WAL (`shop.db-wal`, `shop.db-shm`): они должны копироваться вместе с базой, поэтому для копий используют `.backup`.

Рабочий каталог и проверка

Каталог полигона: `~/sql-lab/15_kopii_obsluzhivanie`. Проверка: `python3 proverka.py 15`.

База данных — это файл, а значит, с ней случаются те же беды, что и с любым файлом: его удаляют, портят, теряют вместе с диском. Администратор отвечает за четыре вещи: **резервные копии** (`.backup`, `.dump`), **восстановление** (проверка копии — это умение развернуть её в новое место), **обмен данными** (экспорт и импорт CSV) и **обслуживание** (сжатие файла `VACUUM`, версии схемы и миграции). Работа идёт с базой магазина `shop.db`, к которой добавлена таблица `junk` (временные данные, раздувшие файл до нескольких мегабайт), файл `new_products.csv` с новыми товарами и сценарий миграции `migrate_001.sql`. Задания выполняются **по порядку**: сначала копии (пока база не изменилась), затем обслуживание.

Раздел 56. Резервные копии, обмен данными и обслуживание базы

56.1. Теоретические сведения

.backup — горячая копия базы

Назначение. Команда консоли `sqlite3` делает согласованную копию всей базы в другой файл средствами самого SQLite. Копия получается целой, даже если в этот момент с базой работает другая программа.

Синтаксис:

```
sqlite3 база.db ".backup файл-копии.db"
sqlite> .backup ?БАЗА? ФАЙЛ
```

Правила:

- обычное копирование файла (`cp`, `copy`) такой гарантии не даёт: в момент копирования программа может записывать страницы, а в режиме WAL часть данных лежит ещё в файле `база.db-wal`;
- необязательное `БАЗА` — имя подключённой базы, по умолчанию `main`;
- если файл копии уже существует, он **перезаписывается без вопросов**; поэтому дату принято включать в имя файла: `work-2026-01-05.db`;
- команда начинается с точки и не заканчивается `;` — это команда оболочки `sqlite3`, а не SQL; в командной строке её заключают в кавычки целиком;
- копию нужно проверить: `PRAGMA integrity_check;` должен вывести `ok`, а число строк в таблицах должно совпасть с оригиналом.

Пример:

```
# копия рабочей базы с датой в имени
sqlite3 work.db ".backup work-2026-01-05.db"
# проверка целостности копии: ожидаем строку ok
sqlite3 work-2026-01-05.db "PRAGMA integrity_check;"

sqlite3 work.db
-- то же из консоли, база main указана явно
sqlite> .backup main extra.db
```

.dump — текстовая копия и восстановление

Назначение. Команда `.dump` печатает базу в виде SQL-скрипта: команды `CREATE` для схемы и `INSERT` для каждой строки. Скрипт читается человеком и подходит к любой версии SQLite. Восстановить базу — значит выполнить этот скрипт в **новой** пустой базе.

Синтаксис:

```
sqlite3 база.db .dump > файл.sql
sqlite> .dump ?ТАБЛИЦА?
sqlite3 новая.db < файл.sql
sqlite> .read файл.sql
```

Правила:

- без аргумента выгружаются схема и данные всех таблиц, с аргументом — только указанная таблица;
- скрипт обёрнут в `BEGIN TRANSACTION; ... COMMIT;`, поэтому восстановление идёт одной транзакцией;
- восстанавливать нужно в новый файл: если таблицы уже есть, будут ошибки `table ... already exists`;
- перенаправление `>` в Windows PowerShell 5.1 пишет файл в UTF-16, и скрипт нельзя будет прочитать как UTF-8; надёжнее выводить через `.output`; оператор `<` в PowerShell не поддерживается — используйте `.read`;
- дамп медленнее и больше, чем `.backup`, зато переносится между версиями и системами и годится для хранения в репозитории;
- единственный способ убедиться, что копия пригодна, — развернуть её в другом месте и сверить число строк.

Пример:

```
# Linux, macOS, cmd.exe
sqlite3 work.db .dump > work.sql
# без перенаправления (подходит и для PowerShell)
sqlite3 work.db ".output work.sql" ".dump"
# восстановление в новый файл и быстрая проверка
sqlite3 check.db ".read work.sql"
sqlite3 check.db "SELECT COUNT(*) FROM products;"
# только одна таблица
sqlite3 work.db ".dump suppliers" > suppliers.sql
```

Экспорт в CSV — .headers, .mode, .output

Назначение. Результат любого запроса можно записать в файл формата CSV (значения через запятую). Для этого настраивают режим вывода и направляют вывод в файл.

Синтаксис:

```
sqlite> .headers on
sqlite> .mode csv
sqlite> .output файл.csv
sqlite> SELECT ...;
sqlite> .output stdout
```

Правила:

- `.mode csv` экранирует запятые, кавычки и переводы строк в значениях по правилам CSV; `NULL` выводится как пустое поле;
- настройки действуют до конца сеанса (или до новой команды), поэтому после экспорта верните `.output stdout`, иначе результаты следующих запросов не появятся на экране;
- без `.headers on` первая строка файла будет данными, а не именами столбцов;
- файл записывается в UTF-8; русский Excel ожидает разделитель `;`, поэтому его открывают через «Данные → Из текста» с кодировкой UTF-8 либо задают `.separator ;`;
- команды с точкой пишутся без `;` и без комментария в конце строки: всё до конца строки считается аргументами;
- в PowerShell 5.1 перенаправление `>` даёт UTF-16 — для файлов с кириллицей используйте `.output`.

- в Windows (cmd.exe, Git Bash) команда `sqlite3 -csv ... > файл.csv` записывает перевод строки дважды (`\r\r\n`), и `.import` такой файл не прочитает (ошибка `unescaped " character`); экспортируйте CSV командами `.once` или `.output`, они пишут корректный файл;

Команды настройки вывода:

Элемент	Описание
`.headers on` или `.headers off`	печатать (или нет) строку с именами столбцов
`.mode csv`	режим вывода CSV; другие режимы: `list`, `column`, `table`, `json`, `insert`
`.separator СИМВОЛ`	разделитель полей, например `.separator ;`
`.output ФАЙЛ`	направить весь дальнейший вывод в файл (старое содержимое заменяется)
`.once ФАЙЛ`	направить в файл вывод только следующего запроса
`.output stdout`	вернуть вывод на экран
`-header -csv`	то же в командной строке: `sqlite3 -header -csv база.db "SELECT ...;" > файл.csv`

Пример:

```
sqlite3 work.db
sqlite> .headers on
sqlite> .mode csv
sqlite> .output products.csv
sqlite> SELECT id, name, price FROM products WHERE stock > 0;
sqlite> .output stdout

# одной командой, вывод в файл через .once
sqlite3 work.db ".headers on" ".mode csv" ".once suppliers.csv" "SELECT * FROM
suppliers;"
```

.import — загрузка из CSV

Назначение. Команда `.import` читает CSV-файл и добавляет его строки в таблицу. Если таблицы ещё нет, она создаётся по первой строке файла.

Синтаксис:

```
sqlite> .import --csv ФАЙЛ ТАБЛИЦА
sqlite> .import --csv --skip 1 ФАЙЛ ТАБЛИЦА
sqlite> .mode csv
sqlite> .import ФАЙЛ ТАБЛИЦА
```

Правила:

- таблицы нет — она создаётся, имена столбцов берутся из первой строки файла, а типы у всех столбцов отсутствуют (значения хранятся как текст);
- таблица уже есть — **все** строки файла, включая строку заголовка, добавляются как данные; чтобы пропустить заголовок, укажите `--skip 1`;
- ключ `--csv` появился в SQLite 3.32; в старых версиях сначала включают `.mode csv`, потом вызывают `.import` без ключа;

- число полей в строке должно совпадать с числом столбцов таблицы, иначе строка пропускается или недостающие поля заполняются NULL (консоль сообщает об этом);
- путь к файлу считается от текущего каталога, откуда запущен sqlite3, а не от каталога базы; кодировка файла — UTF-8;
- числа, загруженные в таблицу без типов, сортируются как текст (100 раньше 20): создайте таблицу заранее с типами или преобразуйте CAST(x AS INTEGER);
- файл CSV из Excel может начинаться с невидимой метки BOM (кодировка utf-8-sig): тогда первое имя столбца получится с лишним символом; сохраняйте файл как «CSV UTF-8 без BOM» или в простом редакторе;
- перед импортом посмотрите файл: cat файл.csv (Linux) или type файл.csv (Windows);
- .import — команда консоли; из Python или другой программы файл загружают через модуль csv и executemany.

Варианты записи:

Элемент	Описание
--csv	разбирать файл как CSV (запятые, кавычки)
--skip N	пропустить первые N строк файла, обычно 1 — заголовок
.mode csv перед командой	то же, что --csv, для старых версий; режим остаётся включённым и влияет на вывод запросов

Пример:

```
-- таблицы нет: она будет создана по заголовку, все столбцы без типов
sqlite> .import --csv extra_prices.csv prices_tmp
-- таблица готовая, с типами: заголовок пропускаем
sqlite> CREATE TABLE prices (id INTEGER PRIMARY KEY, price INTEGER NOT NULL);
sqlite> .import --csv --skip 1 extra_prices.csv prices
sqlite> SELECT COUNT(*) FROM prices;
```

VACUUM и VACUUM INTO — сжатие файла

Назначение. Удаление данных не уменьшает файл базы: освободившиеся страницы остаются внутри него для повторного использования. VACUUM перестраивает файл без пустот, а VACUUM INTO записывает такую компактную копию в новый файл.

Синтаксис:

```
VACUUM;
VACUUM INTO 'файл.db';
```

Правила:

- DELETE и DROP TABLE лишь помечают страницы свободными; размер файла уменьшается только после VACUUM;
- VACUUM создаёт временную копию, поэтому на диске нужно свободное место примерно на размер базы, а во время работы база недоступна для записи другим программам;

- внутри транзакции `VACUUM` не выполняется (ошибка `cannot VACUUM from within a transaction`): сначала `COMMIT`;
- `VACUUM INTO` (SQLite 3.27+) не меняет исходную базу; файл назначения не должен существовать или должен быть пустым, иначе ошибка `output file already exists`;
- сжатую копию можно использовать и как резервную, но для регулярных копий надёжнее `.backup`;
- сколько места можно вернуть, видно по счётчикам страниц: `PRAGMA freelist_count;` (свободные) и `PRAGMA page_count;` (все).

Счётчики размера:

Элемент	Описание
`PRAGMA page_count;`	общее число страниц в файле
`PRAGMA freelist_count;`	число свободных страниц (их вернёт `VACUUM`)
`PRAGMA page_size;`	размер страницы в байтах; размер файла равен `page_count * page_size`

Пример:

```
DELETE FROM log_archive;            -- файл не стал меньше
PRAGMA freelist_count;              -- много свободных страниц
VACUUM;                             -- файл перестроен, пустот нет
VACUUM INTO 'light-copy.db';        -- компактная копия, рабочая база не тронута
```

Миграции — .read и PRAGMA user_version

Назначение. Миграция — это SQL-файл с изменениями схемы, который применяют к базе по порядку номеров. Номер последней применённой миграции хранят в самой базе в числе `user_version`, чтобы знать, какие файлы уже выполнены.

Синтаксис:

```
PRAGMA user_version;
PRAGMA user_version = N;
sqlite> .read файл.sql
sqlite3 база.db < файл.sql
```

Правила:

- `user_version` — целое число в заголовке файла базы; SQLite его не использует, это место для вашего номера версии схемы; у новой базы он равен `0`;
- значение в `PRAGMA user_version = N` пишут числом, подставить параметр `?` нельзя;
- хороший порядок: файл `migrate_NNN.sql` меняет схему и последней строкой ставит новый номер версии; перед применением проверяют, что текущая версия на единицу меньше;
- `.read` выполняет файл как введённые команды; по умолчанию ошибка в середине не прерывает остальное, поэтому файл оборачивают в `BEGIN; ... COMMIT;` (схема в SQLite меняется транзакционно) или запускают консоль с ключом `-bail`;

- миграции не правят задним числом: уже применённый файл не изменяют, а добавляют следующий;
- перед применением миграции к рабочей базе делают копию (`.backup`);
- текст файла миграции можно посмотреть: `cat файл.sql` (Linux) или `type файл.sql` (Windows); новый файл создают в любом текстовом редакторе (кодировка UTF-8);
- проверка результата: `PRAGMA user_version;` и `PRAGMA table_info(таблица);`.

Что обычно делают в миграции:

Элемент	Описание
`ALTER TABLE ... ADD COLUMN ...`	добавить столбец; `NOT NULL` допустим только вместе с `DEFAULT`
`ALTER TABLE ... DROP COLUMN ...`	удалить столбец (SQLite 3.35+), если он не участвует в индексе, ключе и ограничениях
`ALTER TABLE ... RENAME TO ...`	переименовать таблицу
`CREATE TABLE`, `CREATE INDEX`	добавить таблицу или индекс; `IF NOT EXISTS` делает повторный запуск безопасным
ограничения столбца в `CREATE TABLE`	`NOT NULL`, `REFERENCES таблица(столбец)` (ссылка на другую таблицу), `DEFAULT значение` (в том числе `DEFAULT CURRENT_TIMESTAMP` — текущие дата и время)

Пример:

```
-- upgrade_demo.sql
BEGIN;
ALTER TABLE products ADD COLUMN barcode TEXT;
CREATE INDEX idx_products_barcode ON products(barcode);
-- новая таблица с ограничениями: ссылка на suppliers, обязательное поле, значение
по умолчанию
CREATE TABLE contacts (
    id INTEGER PRIMARY KEY,
    supplier_id INTEGER NOT NULL REFERENCES suppliers(id),
    phone TEXT NOT NULL,
    added TEXT NOT NULL DEFAULT CURRENT_TIMESTAMP
);
PRAGMA user_version = 3;
COMMIT;

sqlite3 work.db
sqlite> PRAGMA user_version;
sqlite> .read upgrade_demo.sql
sqlite> PRAGMA user_version;
```

Запрос из командной строки и UNION ALL

Назначение. Запрос можно выполнить, не входя в консоль: `sqlite3 файл.db "запрос"`. Так удобно сверять копию с оригиналом. Оператор `UNION ALL` склеивает результаты нескольких запросов в одну таблицу, что позволяет получить сводку по нескольким таблицам одним запросом.

Синтаксис:

```
sqlite3 файл.db "SELECT ...;" > результат.txt
SELECT 'метка1', COUNT(*) FROM таблица1
```

```
UNION ALL
SELECT 'метка2', COUNT(*) FROM таблица2;
```

Правила:

- в `UNION ALL` у всех частей одинаковое число столбцов; имена столбцов берутся из первой части;
- `UNION ALL` оставляет все строки, а `UNION` убирает повторы (и может изменить порядок);
- порядок строк результата гарантирует только `ORDER BY`, который пишется один раз в конце и относится ко всему объединению; текстовая метка в первом столбце делает строки узнаваемыми;
- запрос в командной строке заключают в двойные кавычки, а строковые литералы внутри него — в одинарные;
- сверять копию надо тем же запросом на обоих файлах: числа должны совпасть;
- результат в файл: в Linux и cmd.exe `> файл`; в PowerShell 5.1 надёжнее `.once файл` перед запросом: `sqlite3 база.db ".once файл" "SELECT ...;"`;
- в PowerShell 5.1 перенаправление `>` пишет UTF-16; для кириллицы используйте `.output`.

Пример:

```
# число строк в двух таблицах одним запросом
sqlite3 work.db "SELECT 'suppliers', COUNT(*) FROM suppliers UNION ALL SELECT
'products', COUNT(*) FROM products ORDER BY 1;"
# тот же запрос на копии: результат должен совпасть
sqlite3 work-2026-01-05.db "SELECT 'suppliers', COUNT(*) FROM suppliers UNION ALL
SELECT 'products', COUNT(*) FROM products ORDER BY 1;"
```

56.2. Задания

Рабочий каталог: `~/sql-lab/15_kopii_obsluzhivanie`. Задания выполняются *по порядку*: каждое опирается на результат предыдущих. Ответ, где он нужен, записывайте в файл `otvety/N`, где `N` — номер задания.

1. Сделайте копию базы командой `.backup` в файл `backup-2026-10-11.db` (в каталоге темы): `sqlite3 shop.db ".backup backup-2026-10-11.db"`. Это команда консоли (не SQL): в командной строке её берут в кавычки целиком. Копия должна быть целой (`PRAGMA integrity_check;` выводит `ok`) и содержать те же таблицы с тем же числом строк.
2. Сохраните базу в виде SQL-скрипта `dump.sql` командой `.dump` (`sqlite3 shop.db .dump > dump.sql`). В Linux и cmd.exe перенаправьте вывод знаком `>`; в PowerShell 5.1 он даёт UTF-16, поэтому там используйте `sqlite3 shop.db ".output dump.sql" ".dump"`. Скрипт должен позволять воссоздать все таблицы с данными.
3. Проверьте, что копия читается: разверните `dump.sql` в **новую** базу `restored.db`: запустите `sqlite3 restored.db` и выполните в консоли `.read dump.sql` (в Linux и cmd.exe годится и `sqlite3 restored.db < dump.sql`, в PowerShell оператор `<` не работает). В `restored.db` должны оказаться те же таблицы с тем же числом строк.
4. Выгрузите таблицу `customers` в файл `customers.csv` (UTF-8) с заголовками столбцов. В консоли `sqlite3 shop.db` включите заголовки (`.headers on`),

режим CSV (`.mode csv`), направьте вывод в файл (`.output customers.csv`), выполните `SELECT * FROM customers;` и верните вывод на экран (`.output stdout`).
5. Загрузите файл `new_products.csv` (пять новых товаров) в новую таблицу `products_new` командой консоли `.import --csv new_products.csv products_new` (запускайте `sqlite3` из каталога, где лежит файл; таблицы нет, поэтому она создаётся по заголовку файла). Проверьте `SELECT COUNT(*) FROM products_new;`. В таблице должно оказаться пять строк.
6. Таблица `junk` занимает почти весь файл `shop.db` (размер: `ls -l shop.db` в Linux, `dir shop.db` в Windows). Удалите её (`DROP TABLE junk;`) и выполните `VACUUM;`: без `VACUUM` файл не уменьшится, потому что освободившиеся страницы остаются в нём. После этого файл должен стать меньше половины прежнего размера.
7. Создайте компактную копию рабочей базы в файл `compact.db` командой `VACUUM INTO 'compact.db';` (SQL-оператор, выполняется в консоли `sqlite3 shop.db`; файла `compact.db` ещё не должно быть).
8. Примените к `shop.db` сценарий `migrate_001.sql` командой консоли `.read migrate_001.sql` (запустите `sqlite3 shop.db`; содержимое файла можно посмотреть: `cat migrate_001.sql` в Linux, `type migrate_001.sql` в Windows): он добавляет столбец `phone` и ставит версию схемы 1.
9. Напишите свой сценарий `migrate_002.sql`: он создаёт таблицу `notes` (`id` — первичный ключ, `customer_id` — обязательная ссылка на `customers(id)`, `text` — обязательный текст, `created_at` — по умолчанию `CURRENT_TIMESTAMP`) и ставит `PRAGMA user_version = 2;`. Создайте файл в текстовом редакторе (синтаксис ограничений столбца — в справочнике «Миграции»), затем выполните его в `shop.db` командой `.read migrate_002.sql`. Проверка: `PRAGMA user_version;` вернёт 2.
10. Сверьте копию с оригиналом: выведите из файла `backup-2026-10-11.db` число строк в таблицах `customers`, `order_items`, `orders`, `products` в виде «таблица, число», по алфавиту имён таблиц. Выполните запрос из командной строки (`sqlite3 backup-2026-10-11.db "запрос"`): склейте четыре запроса по одному на таблицу через `UNION ALL`, первый столбец каждого — текстовая метка с именем таблицы (`SELECT 'customers', COUNT(*) FROM customers`). Запишите результат в `otvety/10`: в Linux и cmd.exe знаком `>`, в PowerShell 5.1 — командой `.once otvety/10` перед запросом (`sqlite3 backup-2026-10-11.db ".once otvety/10" "запрос"`).

Вопросы для размышления

1. Чем `.backup` отличается от простого копирования файла `shop.db`, когда с базой работает приложение?
2. Как организовать регулярное резервное копирование базы SQLite и проверку копий, чтобы восстановление не оказалось сюрпризом?

Тема 16. SQL в сценариях, безопасность и анализ журналов

Общие сведения

В реальных системах SQL выполняют программы. Самое важное правило при этом — **никогда не собирать запрос склейкой строк из чужого ввода**. Если программа пишет `"SELECT secret FROM users WHERE name = '" + ввод + "'"`, то ввод `x' OR '1'='1` превращает условие в «всегда истинно» и возвращает все записи. Это **SQL-инъекция**: атакующий дописывает собственные команды к вашему запросу. Защита одна и надёжная: **параметризованные запросы**. Текст запроса содержит заполнители (`?`), а значения передаются отдельно; база разбирает запрос *до* подстановки, и значение никогда не станет частью команды. Экранирование кавычек вручную защитой не считается: всегда найдётся случай, который забыли.

Вторая привычка — выполнять связанные изменения в транзакции. В Python соединение `с` используют как менеджер контекста: `with c: ...` зафиксирует изменения при успешном выходе и откатит при исключении. Ошибку ограничения перехватывают (`sqlite3.IntegrityError`), сообщают пользователю и завершают сценарий ненулевым кодом, чтобы вызвавшая его система узнала о сбое.

Анализ журналов в базе данных (загрузка журнала веб-сервера в таблицу) хорош там, где вопросы сложнее одной фильтрации: «адреса с множеством неудачных входов», «трафик по часам», «самый медленный путь». Те же вопросы решают `grep` и `awk` (модуль «Журналы и расследование»), но SQL удобнее при группировке, соединении с другими таблицами и повторных запросах к одним данным. Сценарии для администратора пишут на Bash, PowerShell или Python; программа `sqlite3` удобна и в конвейерах: `sqlite3 -csv logs.db "SELECT ..." > отчёт.csv`.

Примеры.

```
# УЯЗВИМО: ввод становится частью команды
c.execute("SELECT secret FROM users WHERE name = '" + name + "'")

# ПРАВИЛЬНО: параметры
c.execute("SELECT secret FROM users WHERE name = ?", (name,))

# всё или ничего
try:
    with c:                                     # COMMIT при успехе, ROLLBACK при
    исключении
        for r in rows:
            c.execute("INSERT INTO users (name, secret) VALUES (?, ?)",
            (r["name"], r["secret"]))
except sqlite3.IntegrityError as e:
    print("Ошибка, всё отменено:", e)
    sys.exit(1)
```

```
sqlite3 -csv logs.db "SELECT ip, COUNT(*) FROM access GROUP BY ip HAVING COUNT(*)
> 20" > report.csv
```

Что чаще всего идёт не так

Параметрами можно передавать только *значения*, но не имена таблиц и столбцов. Если имя приходится подставлять, его сверяют со списком допустимых значений, а не «очищают».

Рабочий каталог и проверка

Каталог полигона: `~/sql-lab/16_stsenarii_bezopasnost`. Проверка: `python3 proverka.py 16`.

Запросы редко выполняют вручную: их вызывают программы и сценарии. Здесь два главных правила администратора. Первое: **значения в запрос передают параметрами** (?), а не склеивают в строку, иначе пользователь сможет подменить запрос — это **SQL-инъекция**, одна из самых частых причин утечек данных. Второе: изменения, связанные друг с другом, выполняют в транзакции, чтобы сбой на середине не оставил базу наполовину обновлённой. В теме вы пишете небольшие сценарии на Python (модуль `sqlite3` входит в стандартную поставку), сначала ломаете «наивную» программу инъекцией, затем чините её, а потом разбираете журнал веб-сервера, загруженный в базу `logs.db` (те же вопросы, что в теме «Журналы и расследование», но на языке SQL). Сценарии лежат в каталоге темы; запуск: `python3 сценарий.py` (в Windows `python сценарий.py`). Базы: `logs.db` (таблица `access`: `id`, `ts` — время, `ip`, `method`, `path`, `status` — код ответа, `bytes`, `rt` — время ответа), `security.db` (таблица `users`: `id`, `name`, `secret`). Задания 7–10 — обычные запросы: их выполняют из командной строки `sqlite3 logs.db "запрос"` и записывают в `otvety/N` (в Linux и cmd.exe знаком `>`, в PowerShell 5.1 — командой `.once otvety/N` перед запросом).

Раздел 57. SQL в сценариях, безопасность и анализ журналов

57.1. Теоретические сведения

sqlite3 в Python — соединение и запрос

Назначение. Модуль `sqlite3` входит в стандартную поставку Python. Он открывает файл базы, выполняет запросы и отдаёт строки результата в виде кортежей.

Синтаксис:

```
import sqlite3
c = sqlite3.connect('файл.db')
for строка in c.execute('SELECT ...'):
    ...
одна = c.execute('SELECT ...').fetchone()
все = c.execute('SELECT ...').fetchall()
c.close()
```

Правила:

- сценарий — обычный текстовый файл `имя.py` (UTF-8), который создают в редакторе; запуск из каталога с файлами: `python3 сценарий.py` в Linux и `python сценарий.py` в Windows;
- сценарий печатает результат командой `print(...)`; значения через пробел: `print(ip, n)`; чтобы записать вывод в файл, используют `>` или открывают файл в самом сценарии (см. `csv`);
- `connect` с несуществующим именем **молча создаёт пустую базу**: опечатка в имени файла проявится позже ошибкой `no such table`;
- путь считается от текущего каталога, откуда запущен сценарий, а не от каталога самого сценария;
- `execute` возвращает курсор: его можно перебирать циклом `for`; каждая строка — кортеж, поэтому для запроса из одного столбца пишут `for (x,) in ...` или `row[0]`;
- `fetchone()` возвращает один кортеж или `None`, если строк нет; `fetchall()` — список всех кортежей;
- в `execute` допускается ровно один SQL-оператор; несколько операторов подряд выполняет `executescript`;
- вывод русских слов в Windows может дать ошибку кодировки; помогает переменная окружения `PYTHONIOENCODING=utf-8`.

Основные вызовы:

Элемент	Описание
`sqlite3.connect(путь)`	открыть базу (создать файл, если его нет), вернуть соединение
`c.execute(запрос, значения)`	выполнить один запрос; значения подставляются в места `?`

Элемент	Описание
`c.executemany(запрос, список)`	выполнить один запрос для каждого набора значений из списка
`курсор.fetchone()`, `fetchall()`	взять одну строку или все строки результата
`c.commit()`, `c.rollback()`	зафиксировать или отменить изменения
`c.close()`	закрыть соединение; несохранённые изменения пропадают

Пример:

```
import sqlite3

c = sqlite3.connect('work.db')
for name, price in c.execute('SELECT name, price FROM products ORDER BY price DESC
LIMIT 3'):
    print(name, price)

total = c.execute('SELECT COUNT(*) FROM suppliers').fetchone()[0]   # одно число
print('suppliers:', total)
c.close()
```

Параметры ? и SQL-инъекция

Назначение. Значения, которые приходят извне (из аргументов, файла, формы), нельзя подставлять в текст запроса склеиванием строк: это открывает путь **SQL-инъекции**. Правильный способ — параметры ?: запрос и данные передаются раздельно.

Синтаксис:

```
c.execute('SELECT ... WHERE столбец = ?', (значение,))
c.execute('INSERT INTO т (а, б) VALUES (?, ?)', (знач1, знач2))
c.execute('SELECT ... WHERE столбец = :имя', {'имя': значение})
```

Правила:

- при склеивании `"... WHERE name = '" + ввод + "'"` символы ввода становятся частью **текста запроса**: кавычка в значении закрывает строковый литерал, и остаток ввода SQLite читает как команды;
- даже безобидное имя с апострофом (`O'Brien`) ломает склеенный запрос синтаксической ошибкой — защита «вручную» всегда неполна;
- с параметром значение попадает в запрос уже как готовое значение: кавычки, `OR`, `--` в нём теряют особый смысл;
- параметры подставляют только на место **значений**; имена таблиц и столбцов параметром сделать нельзя;
- значения передают кортежем; для одного значения нужна запятая: `(x,)`, иначе это просто скобки;
- число значений должно совпадать с числом знаков ?, иначе сценарий завершится ошибкой `ProgrammingError`;
- то, что `execute` не принимает несколько операторов, не защита: подмена условия `WHERE` работает и в одном операторе;

- как работает подмена: запрос `"... WHERE name = '" + ввод + "'"` при вводе `q' OR 1=1 --` превращается в `... WHERE name = 'q' OR 1=1 --'`; кавычка после `q` закрыла литерал, `OR 1=1` сделало условие истинным для всех строк, а `--` превращает остаток строки в комментарий (SQLite его игнорирует); можно обойтись и без `--`, оставив пару для последней кавычки;

Пример:

```
# запуск: python3 tool.py anna
import sqlite3, sys
c = sqlite3.connect('work.db')
who = sys.argv[1]

# плохо: ввод становится частью запроса
#   c.execute("SELECT id FROM customers WHERE name = '" + who + "'")
# хорошо: ввод передан отдельно
row = c.execute('SELECT id FROM customers WHERE name = ?', (who,)).fetchone()
print(row)

c.execute('UPDATE products SET stock = ? WHERE id = ?', (10, 1))
c.commit()
```

Транзакция в Python — with, IntegrityError

Назначение. Связанные изменения выполняют как одно целое: либо применяются все, либо ни одно. В Python соединение работает как менеджер контекста: блок `with c:` фиксирует изменения при успехе и откатывает их при исключении.

Синтаксис:

```
try:
    with c:
        c.execute(...)
        c.execute(...)
except sqlite3.IntegrityError as e:
    print('Ошибка:', e)
    sys.exit(1)
```

Правила:

- `with c:` **не закрывает** соединение: он делает `COMMIT` при нормальном выходе из блока и `ROLLBACK`, если внутри возникло исключение;
- после отката исключение продолжает подниматься — его нужно перехватить `try ... except`, иначе сценарий завершится с трассировкой;
- `IntegrityError` возникает при нарушении ограничений: `UNIQUE`, `NOT NULL`, `CHECK`, `PRIMARY KEY` (внешние ключи — только если включено `PRAGMA foreign_keys = ON`);
- изменения (`INSERT`, `UPDATE`, `DELETE`) без `commit()` не сохраняются: при закрытии программы они теряются;
- всё, что может упасть не из-за базы (чтение файла, разбор данных), выполняют **до** блока `with`, чтобы не открывать транзакцию зря;
- код завершения `sys.exit(1)` сообщает вызывающей программе об ошибке; успех — код 0.

Исключения модуля sqlite3:

Элемент	Описание
`sqlite3.Error`	общий предок всех ошибок модуля
`sqlite3.IntegrityError`	нарушено ограничение целостности (`UNIQUE`, `NOT NULL`, `CHECK`, внешний ключ)
`sqlite3.OperationalError`	ошибка выполнения: нет таблицы или столбца, синтаксическая ошибка, база заблокирована
`sqlite3.ProgrammingError`	ошибка в программе: неверное число параметров, закрытое соединение

Пример:

```
import sqlite3, sys
c = sqlite3.connect('work.db')
new_rows = [('Tver', 1), ('Tula', 2)]
try:
    with c:                       # COMMIT при успехе, ROLLBACK при ошибке
        for city, n in new_rows:
            c.execute('INSERT INTO cities (name, zone) VALUES (?, ?)', (city, n))
except sqlite3.IntegrityError as e:
    print('rolled back:', e)
    sys.exit(1)
```

Модуль csv — чтение и запись CSV

Назначение. Модуль `csv` разбирает и записывает файлы CSV, сам обрабатывая запятые, кавычки и переводы строк внутри значений. С его помощью результаты запросов сохраняют в файл, а данные из файла загружают в базу.

Синтаксис:

```
import csv
with open('файл.csv', 'w', newline='', encoding='utf-8') as f:
    w = csv.writer(f)
    w.writerow([...])
    w.writerows(строки)
with open('файл.csv', newline='', encoding='utf-8') as f:
    for r in csv.DictReader(f):
        r['столбец']
```

Правила:

- файл открывают с `newline=''`; без него в Windows между строками появятся пустые строки;
- кодировку указывают явно: `encoding='utf-8'`, иначе Windows использует системную (обычно cp1251);
- `csv.writer` пишет строки из списков и кортежей; `writerows` принимает сразу весь результат `fetchall()`; `None` записывается пустым полем;
- `csv.DictReader` берёт имена из первой строки и отдаёт каждую следующую как словарь `{имя: значение}`;
- читатель возвращает **только строки**: числа преобразуют сами (`int(r['qty'])`), иначе в базу попадёт текст;
- русский Excel лучше открывает файл с разделителем `;` и кодировкой `utf-8-sig`: `csv.writer(f, delimiter=';')`.

Пример:

```
import csv, sqlite3
c = sqlite3.connect('work.db')
rows = c.execute('SELECT id, name FROM suppliers ORDER BY id').fetchall()
with open('suppliers_out.csv', 'w', newline='', encoding='utf-8') as f:
    w = csv.writer(f)
    w.writerow(['id', 'name'])      # заголовок
    w.writerows(rows)

with open('suppliers_out.csv', newline='', encoding='utf-8') as f:
    for r in csv.DictReader(f):
        print(int(r['id']), r['name'])
```

sys.argv и sys.exit — параметры сценария

Назначение. Сценарий получает значения из командной строки через список `sys.argv` и сообщает результат своей работы кодом завершения.

Синтаксис:

```
import sys
arg1 = sys.argv[1]
sys.exit(1)
```

Правила:

- `sys.argv[0]` — имя сценария, `sys.argv[1]` — первый аргумент, `sys.argv[2]` — второй и так далее; все значения — строки;
- если аргумент не передан, обращение к `sys.argv[1]` даёт `IndexError`; число аргументов проверяют через `len(sys.argv)`;
- аргумент с пробелами или кавычками в оболочке берут в кавычки: `python find.py "a b"`; одиночная кавычка внутри значения безопасна в двойных кавычках: `python find.py "x' OR '1'='1"`;
- код `0` означает успех, ненулевой — ошибку; его читают оболочки: `$?` (bash), `$LASTEXITCODE` (PowerShell), `%ERRORLEVEL%` (cmd);
- сообщения об ошибках лучше печатать в поток ошибок: `print('...', file=sys.stderr)`.

Пример:

```
import sys
if len(sys.argv) < 2:
    print('usage: python tool.py NAME', file=sys.stderr)
    sys.exit(2)
print('name =', sys.argv[1])

# проверка кода завершения
#   python tool.py            ->  $? равен 2
#   python tool.py anna       ->  $? равен 0
```

Сводки по журналу — GROUP BY, HAVING, SUBSTR

Назначение. Журнал веб-сервера, загруженный в таблицу, анализируют теми же запросами, что и любые данные: группируют строки по адресу, коду ответа или часу и считают, сколько их в каждой группе.

Синтаксис:

```
SELECT ключ, COUNT(*) AS n FROM журнал
WHERE условие_для_строк
GROUP BY ключ
HAVING условие_для_групп
ORDER BY n DESC, ключ
LIMIT k;
SUBSTR(строка, начало, длина)
```

Правила:

- `WHERE` отбирает **строки** до группировки, `HAVING` отбирает **группы** после неё и может использовать `COUNT(*)`, `SUM(...)` и псевдонимы;
- `SUBSTR(ts, 1, 13)` берёт 13 символов с первого: так из метки времени получают часть, по которой группируют (дату, час); нумерация символов начинается с 1;
- группировать можно по выражению или по его псевдониму: `GROUP BY h`;
- чтобы порядок строк был однозначным, добавляйте в `ORDER BY` последним ключом сам столбец группировки;
- `SUM` и `AVG` пропускают `NULL`; на пустой группе `SUM` возвращает `NULL`;
- самую большую (малую) строку находят сортировкой с `LIMIT 1`, а не `MAX` — тогда доступны остальные столбцы этой строки;
- запрос к базе из командной строки: `sqlite3 logs.db "SELECT ...;"`; ответ записывают в файл знаком `>` (Linux, cmd.exe) или `.once файл` перед запросом (PowerShell 5.1: `sqlite3 logs.db ".once otvety/7" "SELECT ...;"`);
- условие по нескольким столбцам строят через `AND`: `WHERE method = 'POST' AND path = '/login' AND status = 401`; в журнале `status` — число, `method` и `path` — текст в одинарных кавычках;
- из командной строки результат сразу пишут в CSV: `sqlite3 -header -csv logs.db "SELECT ...;" > отчёт.csv`.

Пример:

```
-- адреса, чаще всего получавшие ошибку сервера
SELECT ip, COUNT(*) AS n FROM access WHERE status >= 500
GROUP BY ip HAVING n >= 3 ORDER BY n DESC, ip;

-- запросов по дням (первые 10 символов метки времени)
SELECT SUBSTR(ts, 1, 10) AS d, COUNT(*) FROM access GROUP BY d ORDER BY d;

-- самый тяжёлый ответ
SELECT path, bytes FROM access ORDER BY bytes DESC, id LIMIT 1;
```

57.2. Задания

Рабочий каталог: `~/sql-lab/16_stsenarii_bezopasnost`. Задания выполняются *по порядку*: каждое опирается на результат предыдущих. Ответ, где он нужен, записывайте в файл `otvety/N`, где `N` — номер задания.

1. Напишите сценарий `top_ip.py` (`import sqlite3`, `sqlite3.connect('logs.db')`, цикл `for ... in c.execute(...)`, `print`), который подключается к `logs.db` и печатает пять адресов с наибольшим числом запросов в таблице `access`: по строке `адрес число`, от большего числа к меньшему; при равенстве — по адресу (`GROUP BY ip`, `ORDER BY ... DESC`, `LIMIT 5`). Запустите: `python3 top_ip.py` (в Windows `python top_ip.py`).

2. Напишите `find_ip.py АДРЕС`: адрес приходит аргументом командной строки (`sys.argv[1]`), сценарий печатает число запросов с этого адреса (`fetchone()[0]`). Запуск: `python3 find_ip.py 192.168.1.14`. Значение подставляйте **параметром** `?` (`c.execute('... WHERE ip = ?', (адрес,))`), а не склеиванием строк.
3. «Наивная» программа проверяет пользователя запросом, склеенным из ввода: `SELECT name, secret FROM users WHERE name = '` + ввод + `'`. Подберите **ввод** (одну строку), который заставит этот запрос вернуть **все три** записи таблицы `users`, и запишите его в файл `injection.txt` (текстовым редактором или командой `echo`). Ввод не должен быть настоящим именем. Приём: ввод должен закрыть открытую кавычку, добавить условие через `OR`, которое истинно для любой строки, и сделать так, чтобы оставшаяся в запросе кавычка не вызвала ошибку (оставить её парной или закомментировать хвост через `--`).
4. Напишите `safe_login.py ИМЯ`: печатает секрет пользователя с этим именем, **не** склеивая запрос. Значение берите из `sys.argv[1]` и передавайте параметром `?`. Обычное имя (`python3 safe_login.py anna`) должно давать её секрет, а ваш ввод из `injection.txt` (передайте его аргументом в кавычках: `python3 safe_login.py "x' OR ..."`) — ничего.
5. Напишите `report.py`, который создаёт файл `report.csv` (первая строка `ip,hits`) с адресами, у которых в `access` больше 20 запросов, и числом запросов: от большего к меньшему, при равенстве — по адресу. Используйте модуль `csv`: `csv.writer`, `writerow` для заголовка, `writerows` для данных; файл открывайте как `open('report.csv', 'w', newline='', encoding='utf-8')`. Условие «больше 20» отбирайте в `HAVING`.
6. Напишите `import_users.py ФАЙЛ.csv`, который добавляет пользователей из CSV (столбцы `name`, `secret`) в `security.db` **одной транзакцией**. Если хотя бы одна запись нарушает ограничение (например, имя уже есть), не должно добавиться ничего, сценарий должен напечатать ошибку и завершиться с кодом 1. Файл читайте через `csv.DictReader` (читайте до блока `with`), добавляйте записи `INSERT ... VALUES (?, ?)` внутри `with c:`, ошибку ловите как `sqlite3.IntegrityError`, выход с кодом — `sys.exit(1)`. Проверьте на `users_new.csv` (в нём повтор `anna`) и `users_ok.csv`: `python3 import_users.py users_ok.csv`; код завершения — `echo $?` (bash) или `$LASTEXITCODE` (PowerShell). Повторный запуск на `users_ok.csv` даст ошибку: имена уже есть.
7. Найдите адреса, с которых выполнено **не менее 20** запросов `POST /login` с кодом 401. Выведите адрес и число таких запросов (`WHERE` отбирает строки, `HAVING COUNT(*) >= 20` — группы). Команда: `sqlite3 logs.db "SELECT ...;"`. Результат — в `otvety/7`.
8. Найдите адреса, получившие **больше 50** ответов с кодом 404. Выведите адрес и число ответов (`GROUP BY ip` и `HAVING`). Результат — в `otvety/8`.
9. Каков путь (`path`) самого медленного запроса (наибольшее `rt`)? Найдите её сортировкой `ORDER BY rt DESC` с `LIMIT 1`. Запишите в `otvety/9`.
10. Для каждого часа (первые 13 символов `ts`, например `2026-10-11 05`) выведите час и суммарное число отданных байт (`SUM(bytes)`), по возрастанию часа (`SUBSTR(ts, 1, 13)`, `GROUP BY` по этому выражению, `ORDER BY` по нему же). Результат — в `otvety/10`.

Вопросы для размышления

1. Почему экранирование кавычек вручную — плохая защита от инъекций, и чем параметры лучше?
2. Какие данные из журнала веб-сервера лучше анализировать в базе данных, а какие проще обработать утилитами `grep` и `awk`?

Тема 17. Итоговый проект: заявки службы поддержки

Общие сведения

Итоговый проект собирает приёмы курса в одну работу, похожую на рабочую: база заявок службы поддержки, к которой руководитель предъявляет требования. Хороший порядок такой работы одинаков для любой задачи с базой данных. **1. Изучить данные:** таблицы, связи, объёмы, пустые значения. **2. Ответить на вопросы запросами:** сводки по приоритетам, серверам, инженерам. **3. Вынести повторяющееся в справочники:** сроки реагирования хранятся в одной таблице, а не вшиты в запросы. **4. Оформить сложное представлением:** просроченные заявки — сохранённый запрос, которым пользуются все. **5. Ускорить частое:** частичный индекс по открытым заявкам. **6. Защитить правила:** триггер запрещает закрывать заявку без исполнителя. **7. Менять данные осторожно:** массовое назначение выполняется одной транзакцией с точным условием. **8. Сделать копию и проверить её. 9. Оформить отчёт:** файл запросов и итоговое число.

Проект проверяет не память на синтаксис, а умение выбирать подходящее средство: что здесь должно быть ограничением, что триггером, что представлением, а что запросом. Если вы справились с ним без подсказок, вы готовы поддерживать небольшие базы SQLite и общаться с разработчиками на их языке.

Примеры.

```
sqlite3 helpdesk.db
sqlite> SELECT priority, COUNT(*) FROM tickets WHERE closed IS NULL GROUP BY
priority;
sqlite> CREATE TABLE priorities (code TEXT PRIMARY KEY, sla_hours INTEGER NOT
NULL);
sqlite> CREATE VIEW v_overdue AS SELECT ... ;
sqlite> CREATE INDEX idx_tickets_open ON tickets(priority) WHERE closed IS NULL;
sqlite> CREATE TRIGGER trg_close_needs_engineer BEFORE UPDATE OF closed ON tickets
   ...>   WHEN NEW.closed IS NOT NULL AND NEW.engineer_id IS NULL
   ...>   BEGIN SELECT RAISE(ABORT, 'заявку без инженера закрыть нельзя'); END;
sqlite> .backup helpdesk-backup.db
```

Что чаще всего идёт не так

Типичная ошибка итоговой работы — пропустить проверку: создали представление и не сравнили его с запросом, создали триггер и не попробовали сработать. Каждое средство проверяют так же, как это делает программа `proverka`: наблюдением результата.

Рабочий каталог и проверка

Каталог полигона: `~/sql-lab/17_itogovyj_proekt`. Проверка: `python3 proverka.py 17`.

Итоговый проект. Служба поддержки «Северного ветра» ведёт заявки в базе

`helpdesk.db`: серверы (`servers`), инженеры (`engineers`) и заявки (`tickets`; открыта, пока `closed` равно `NULL`). Руководитель хочет видеть просроченные заявки, не терять данные и быстро получать сводки. Вам предстоит применить всё, чему вы научились: запросы с группировкой и соединениями, новая таблица справочника, представление, индекс, триггер, транзакция, резервная копия и итоговый отчёт. «Сейчас» для расчётов — фиксированный момент `2026-10-11 12:00`. Приоритеты заявок: `P1` (срок 4 часа), `P2` (24 часа), `P3` (72 часа). Задания выполняются **по порядку**.

Раздел 58. Итоговый проект: заявки службы поддержки

58.1. Теоретические сведения

План работы над проектом

Назначение. Итоговый проект — не набор новых команд, а порядок работы: сначала изучить базу, потом по шагам менять её и после каждого шага проверять результат. Так ошибка находится сразу, а не в конце.

Синтаксис:

```
1. Изучить схему:      .tables    .schema таблица    PRAGMA table_info(таблица);
2. Посмотреть данные:  SELECT * FROM таблица LIMIT 5;
3. Написать запрос или изменение
4. Проверить результат до записи (SELECT с тем же WHERE)
5. Записать изменение (в транзакции)
6. Проверить после записи и сделать копию
```

Правила:

- задания идут по порядку: следующие используют таблицы, представления и индексы, созданные в предыдущих;
- перед `UPDATE` и `DELETE` выполните `SELECT` с тем же условием `WHERE` и посмотрите, сколько строк затронет изменение;
- прочитайте сообщение об ошибке целиком: в нём названы таблица, столбец или ограничение, которые вызвали сбой;
- перед рискованным изменением данных сделайте копию (`.backup`), а итоговую копию — после всех изменений;
- результат запроса в файл ответа записывают перенаправлением: `sqlite3 helpdesk.db "SELECT ...;" > otvety/1` (или `.output otvety/1` в консоли, см. SQL-14); при ошибке в запросе файл останется пустым;
- справочник сроков создают как обычную таблицу (`CREATE TABLE`, тема 10), триггер и представление — как в SQL-13, частичный индекс — как в SQL-12, `BEGIN/COMMIT` — как в SQL-14, `.backup` — как в SQL-15;
- в проекте «сейчас» фиксировано (`2026-10-11 12:00`) — не используйте `datetime('now')`, иначе ответ будет зависеть от дня проверки.

Что проверять на каждом шаге:

Элемент	Описание
после запроса	сколько строк получилось и совпадает ли это с ожиданием; нет ли `NULL` в неожиданных местах
после создания таблицы, представления, индекса	`.schema имя` или запрос к `sqlite_master`
после изменения данных	`SELECT COUNT(*)` с теми же условиями до и после

Элемент	Описание
после всего	`PRAGMA integrity_check;` должен вывести `ok`

Пример:

```
-- что есть в базе и как устроена таблица
SELECT name, type FROM sqlite_master WHERE type IN ('table', 'view', 'index',
'trigger') ORDER BY type, name;
PRAGMA table_info(servers);
-- сколько строк затронет будущее изменение
SELECT COUNT(*) FROM servers WHERE city = 'Tula';
```

Сводки — JOIN, GROUP BY, HAVING и даты

Назначение. Большинство отчётов проекта — это соединение нескольких таблиц, группировка и подсчёт. Время между двумя моментами вычисляют функцией `julianday`, которая превращает дату-текст в число суток.

Синтаксис:

```
SELECT группа, COUNT(*) AS n
FROM таблица1 t JOIN таблица2 s ON s.id = t.ссылка
WHERE условие
GROUP BY группа
HAVING n >= 5
ORDER BY n DESC, группа;
(julianday(позже) - julianday(раньше)) * 24
```

Правила:

- незавершённость записи в проекте обозначена `NULL`: условие пишут `closed IS NULL` или `closed IS NOT NULL`, а не `= NULL` (оно никогда не истинно);
- `JOIN ... ON` связывает строки по ключу; группировать удобно по `id` объекта, а выводить его название — так одинаковые названия не сольются;
- `WHERE` отбирает строки до группировки, `HAVING` — группы после неё (`COUNT(*)`, `AVG`, `SUM`);
- `julianday('2026-10-11 12:00')` понимает текст `ГГГГ-ММ-ДД ЧЧ:ММ`; разность двух значений — в сутках, умножение на 24 даёт часы;
- `AVG` пропускает `NULL`; `ROUND(x, 1)` округляет до одного знака после запятой;
- когда в результате нужен определённый порядок, всегда пишите `ORDER BY` и добавляйте последним ключом уникальный столбец.

Пример:

```
-- серверы, у которых есть хотя бы две незакрытые записи
SELECT s.name, COUNT(*) AS n
FROM jobs j JOIN servers s ON s.id = j.server_id
WHERE j.finished IS NULL
GROUP BY s.id
HAVING n >= 2
ORDER BY n DESC, s.name;

-- сколько часов прошло от начала до заданного момента
SELECT id, ROUND((julianday('2026-10-11 12:00') - julianday(started)) * 24, 1) AS
hours FROM jobs ORDER BY id;
```

Справочник, представление, частичный индекс

Назначение. Справочник хранит общие для многих строк значения (например, срок для каждого приоритета) в одном месте. Представление — сохранённый запрос, который выглядит как таблица. Частичный индекс ускоряет выборку, охватывая только часть строк.

Синтаксис:

```
CREATE TABLE справочник (код TEXT PRIMARY KEY, значение INTEGER NOT NULL);
INSERT INTO справочник VALUES ('A', 1), ('B', 2);
CREATE VIEW имя AS SELECT ... ;
CREATE INDEX имя ON таблица(столбец) WHERE условие;
```

Правила:

- первичный ключ типа TEXT допустим: `код TEXT PRIMARY KEY` не даст добавить два одинаковых кода;
- представление не хранит данные: каждый раз выполняется его SELECT, поэтому оно всегда показывает актуальное состояние таблиц; с представлением работают как с таблицей: `SELECT * FROM имя`;
- имена столбцов представления задают псевдонимами AS; обращаться к ним можно только по этим именам;
- представление может соединять несколько таблиц, в том числе справочник; условия отбора помещают в его WHERE;
- частичный индекс содержит только строки, для которых истинно условие WHERE; он меньше обычного, но SQLite использует его лишь для запросов, условие которых включает условие индекса (`closed IS NULL`);
- индекс не меняет результат запроса, только скорость; увидеть, применён ли он, можно командой `EXPLAIN QUERY PLAN SELECT ...`;
- объект с занятым именем нельзя создать повторно; убрать ненужный — `DROP VIEW имя`, `DROP INDEX имя`.

Пример:

```
CREATE TABLE levels (code TEXT PRIMARY KEY, limit_hours INTEGER NOT NULL);
INSERT INTO levels VALUES ('L1', 2), ('L2', 12);

CREATE VIEW v_long_jobs AS
SELECT j.id AS job_id, s.name AS server, j.level AS level
FROM jobs j JOIN servers s ON s.id = j.server_id JOIN levels l ON l.code = j.level
WHERE j.finished IS NULL AND (julianday('2026-10-11 12:00') -
julianday(j.started)) * 24 > l.limit_hours;

SELECT * FROM v_long_jobs ORDER BY job_id;

CREATE INDEX idx_jobs_open ON jobs(level) WHERE finished IS NULL;
EXPLAIN QUERY PLAN SELECT level, COUNT(*) FROM jobs WHERE finished IS NULL GROUP
BY level;
```

Триггер и транзакция — защита данных

Назначение. Триггер — это код, который база выполняет сама при изменении таблицы; с его помощью проверяют правила, которые не выразить ограничениями столбца. Транзакция объединяет несколько изменений в одно целое: фиксируются все или ни одно.

Синтаксис:

```
CREATE TRIGGER имя BEFORE UPDATE OF столбец ON таблица
WHEN условие
BEGIN
    SELECT RAISE(ABORT, 'сообщение');
END;

BEGIN;
UPDATE ...;
COMMIT;
```

Правила:

- `BEFORE` выполняет триггер до изменения, поэтому `RAISE(ABORT, ...)` отменяет сам оператор; `UPDATE OF столбец` срабатывает, только если столбец указан в `SET`;
- в условии `WHEN` и теле триггера строка видна под именами `OLD` (до изменения) и `NEW` (после); для `NULL` пишут `IS NULL` и `IS NOT NULL`;
- тело триггера состоит из операторов между `BEGIN` и `END;`; в консоли `sqlite3` весь `CREATE TRIGGER` вводят целиком — до `END;` приглашение остаётся `...>`;
- сообщение из `RAISE` увидит тот, чей запрос отклонён; проверить триггер можно, выполнив запрещённое изменение и убедившись, что оно отклонено, а данные не изменились;
- транзакция: `BEGIN;`, затем изменения, затем `COMMIT;` (закрепить) или `ROLLBACK;` (отменить); без `BEGIN` каждый оператор фиксируется сам;
- условие `WHERE` у `UPDATE` ограничивает круг затронутых строк — проверьте его `SELECT`-ом заранее;
- если оператор внутри транзакции завершился ошибкой, выполните `ROLLBACK;`, иначе транзакция останется открытой.

Пример:

```
-- запретить завершать работу без исполнителя
CREATE TRIGGER trg_need_worker BEFORE UPDATE OF finished ON jobs
WHEN NEW.finished IS NOT NULL AND NEW.worker IS NULL
BEGIN
    SELECT RAISE(ABORT, 'no worker');
END;

-- изменение группой: либо все строки, либо ни одной
BEGIN;
UPDATE jobs SET worker = 2 WHERE level = 'L2' AND finished IS NULL;
COMMIT;
```

Копия и отчёт — .backup и файл запросов

Назначение. Завершая работу, делают резервную копию базы и оформляют сводки в виде файла SQL, который можно выполнить целиком в любой момент.

Синтаксис:

```
sqlite3 база.db ".backup копия.db"
sqlite3 копия.db "PRAGMA integrity_check;"
sqlite3 база.db < файл.sql
sqlite> .read файл.sql
```

Правила:

- копию делают **после** всех изменений, иначе в ней не будет последних таблиц, представлений и триггеров;
- проверка копии: `PRAGMA integrity_check;` должен вывести `ok`, а `.tables` и `.schema` в копии показывают те же объекты;
- файл со сводками — обычный текст: запросы разделены `;`, пояснения — комментарии `--`; выполнить его можно через `.read` или перенаправлением `<` (в PowerShell оператор `<` не работает — используйте `.read`);
- для читаемого вывода перед запросами включают `.headers on` и `.mode column`;
- значение, которое надо записать в файл ответа, получают отдельным запросом (например, `SELECT COUNT(*) FROM имя_представления;`) и переписывают в нужный файл.

Пример:

```
# копия после всех изменений и её проверка
sqlite3 work.db ".backup work-final.db"
sqlite3 work-final.db "PRAGMA integrity_check;"
# выполнить файл со сводками с красивым выводом
sqlite3 -header -column work.db ".read summary.sql"
```

58.2. Задания

Рабочий каталог: `~/sql-lab/17_itogovyj_proekt`. Задания выполняются *по порядку*: каждое опирается на результат предыдущих. Ответ, где он нужен, записывайте в файл `otvety/N`, где `N` — номер задания.

1. Сколько заявок открыто (`closed IS NULL`) для каждого приоритета? Выведите приоритет и число, по приоритету. Используйте `COUNT(*)`, `WHERE ... IS NULL`, `GROUP BY` (см. тема 5). Результат — в `otvety/1` (вывод запроса перенаправьте в файл: `sqlite3 helpdesk.db "..." > otvety/1`, см. тема 14).
2. Выведите названия серверов (`servers.name`), по которым было **не менее пяти** заявок, и число заявок; сначала с наибольшим числом, при равенстве — по названию. Соедините `tickets` с `servers` через `JOIN ... ON` (тема 6), сгруппируйте по серверу `GROUP BY` и отберите группы условием `HAVING` (тема 5). Результат — в `otvety/2`.
3. Для каждого инженера, закрывшего хотя бы одну заявку, вычислите среднее время решения в часах: `ROUND(AVG((julianday(closed) - julianday(opened)) * 24), 1)`. Выведите имя инженера и среднее по возрастанию `id` инженера. Нужны `JOIN engineers` по `engineer_id`, условие `closed IS NOT NULL` и `GROUP BY` по инженеру; `julianday(...)` — разность дат в сутках (описано в этой теме, запись «Сводки»). Результат — в `otvety/3`.
4. Создайте таблицу `priorities`: `code` — текстовый первичный ключ, `sla_hours` — целое обязательное. Заполните: `P1` — 4, `P2` — 24, `P3` — 72. Нужны `CREATE TABLE` с ограничениями `PRIMARY KEY` и `NOT NULL` (тема 10) и `INSERT INTO ... VALUES` (тема 9).
5. Создайте представление `v_overdue` со столбцами `ticket_id`, `title`, `server` (название сервера), `priority`, `hours_open` (часы с открытия до `2026-10-11 12:00`, округлённые до одного знака) для **открытых** заявок (`closed IS NULL`), у которых время с открытия больше срока `sla_hours` их приоритета. Используйте `CREATE VIEW ... AS SELECT` (тема 13) с `JOIN` трёх таблиц: `tickets`, `servers` (по `server_id`), `priorities` (по `priority = code`); часы — `(julianday('2026-10-11 12:00') - julianday(opened)) * 24`, округ-

ление — `ROUND(..., 1)`; столбцы назовите через `AS`.
6. Запросы по открытым заявкам выбирают приоритет и `closed IS NULL`. Создайте **частичный** индекс `idx_tickets_open` по `tickets(priority)` с условием `WHERE closed IS NULL` (`CREATE INDEX ... ON ... WHERE ...`, см. тема 12, «Частичный индекс»).
7. Создайте триггер `BEFORE UPDATE OF closed ON tickets`, который отменяет закрытие заявки (`NEW.closed IS NOT NULL`), если у неё не назначен инженер (`NEW.engineer_id IS NULL`). Условие пишется в `WHEN`, отмена — `SELECT RAISE(ABORT, 'сообщение');` в теле между `BEGIN` и `END;` (тема 13).
8. Назначьте инженера 1 на **все открытые заявки** приоритета `P1` одной транзакцией (`BEGIN; UPDATE ...; COMMIT;`). Остальные заявки менять нельзя. Нужны `UPDATE ... SET ... WHERE` (тема 9) и транзакция (тема 14).
9. Сделайте копию базы в файл `helpdesk-backup.db` командой `sqlite3 helpdesk.db ".backup helpdesk-backup.db"` (тема 15) **после** всех изменений. Копия должна быть целой и содержать справочник и представление.
10. Составьте файл `queries.sql` с не менее чем тремя запросами `SELECT` (хотя бы один с `GROUP BY`) — вашими «сводками» для руководителя. Пример запроса с `GROUP BY` смотрите в заданиях 1–2; файл можно создать в любом текстовом редакторе. В `otvety/10` запишите число просроченных заявок: получите его запросом `SELECT COUNT(*) FROM v_overdue;` и перенаправьте вывод в файл (`> otvety/10`).

Вопросы для размышления

1. Какие из применённых средств (ограничения, представления, индексы, триггеры, транзакции, копии) защищают данные, а какие ускоряют работу?
2. Что бы вы добавили в базу, если бы заявками пользовались сто человек одновременно?

Тема 18. JSON, полнотекстовый поиск и подключение баз

Общие сведения

Этот дополнительный модуль следует за итоговым проектом и показывает приёмы, без которых трудно работать с данными современных приложений. **JSON** — текстовый формат вложенных данных: объекты в фигурных скобках, массивы в квадратных. SQLite хранит JSON обычным текстом, но умеет его разбирать: `json_extract(столбец, '\$.поле')` достаёт значение по пути, `json_each` разворачивает массив в таблицу, `json_set` возвращает JSON с изменённым полем. Платить за гибкость приходится скоростью и проверкой данных: база не знает, что в поле `ms` должно быть число.

Чтобы вернуть скорость, значение выносят в **вычисляемый столбец** (`GENERATED ALWAYS AS (…)`) и по нему строят обычный индекс. Для поиска по тексту служит **FTS5**: специальная виртуальная таблица, которая разбивает текст на слова и ищет по ним быстро, с операторами `AND`, `OR`, `NOT`. Наконец, `ATTACH` подключает к соединению второй файл базы, и таблицы двух файлов можно соединять в одном запросе.

Примеры.

```
sqlite3 events.db
sqlite> SELECT json_extract(payload, '$.host'), COUNT(*) FROM events GROUP BY 1;
sqlite> SELECT e.id FROM events e, json_each(e.payload, '$.tags') t WHERE t.value
= 'db';
sqlite> ALTER TABLE events ADD COLUMN level TEXT GENERATED ALWAYS AS
(json_extract(payload, '$.level')) VIRTUAL;
sqlite> CREATE INDEX idx_events_level ON events(level);
sqlite> CREATE VIRTUAL TABLE docs_fts USING fts5(title, body);
sqlite> SELECT title FROM docs_fts WHERE docs_fts MATCH 'backup AND restore';
sqlite> ATTACH 'shop.db' AS shop;
```

Что чаще всего идёт не так

Путь в JSON начинается со знака доллара и чувствителен к регистру. Таблицу FTS5 не обновляет сама исходная таблица: после изменения `docs` индекс нужно перезаполнить. Имя файла в `ATTACH` — строка в кавычках.

Рабочий каталог и проверка

Каталог полигона: `~/sql-lab/18_json_fts`. Проверка: `python3 proverka.py 18`.

Современные приложения пишут события в журнал в формате **JSON** — гибком тексте вида `{"level": "error", "host": "srv01", "ms": 120}`. Складывать такие записи в отдельный столбец (`payload`) удобно, но спрашивать «сколько ошибок на `srv02`» приходится функциями JSON: `json_extract`, `json_each`, `json_set`. Чтобы запросы были быстрыми, значение из JSON выносят в **вычисляемый столбец** (`GENERATED ALWAYS AS (...)`) и строят по нему индекс. Для поиска по

тексту документов (инструкций, заявок) обычного LIKE мало: нужен **полнотекстовый поиск** FTS5. А чтобы объединить данные из двух файлов баз, их подключают командой ATTACH. Задания выполняются в events.db (события и документы); shop.db пригодится для задания 9. Задания 5–7 и 10 изменяют базу.

Раздел 59. JSON, полнотекстовый поиск и подключение баз

59.1. Теоретические сведения

json_extract — поле из JSON

Назначение. Функция `json_extract` достаёт значение из текста в формате JSON по **пути**. Результат — обычное значение SQL, поэтому его можно сравнивать, группировать, сортировать и усреднять.

Синтаксис:

```
json_extract(json_текст, '$.поле')
json_extract(json_текст, '$.поле.вложенное')
json_extract(json_текст, '$.массив[0]')
столбец ->> '$.поле'
```

Правила:

- путь пишут в одинарных кавычках и начинают со знака `$` — это весь документ; поле добавляют через точку, элемент массива — в квадратных скобках (нумерация с нуля, `[#-1]` — последний);
- строка JSON возвращается как текст, число — как число, `true` и `false` — как `1` и `0`, `null` и отсутствующее поле — как `NULL`; массив или объект возвращается как текст JSON;
- несуществующее поле не вызывает ошибки, а даёт `NULL`: опечатка в имени поля проявится пустым результатом, а не сообщением;
- имена полей в JSON чувствительны к регистру: `$.Host` и `$.host` — разные поля;
- если в столбце окажется не JSON, функция завершится ошибкой `malformed JSON`; проверить текст можно через `json_valid(текст)`;
- оператор `->>` (SQLite 3.38+) делает то же, что `json_extract`, и возвращает значение SQL; оператор `->` возвращает JSON-текст;
- функцию допустимо использовать в `WHERE`, `GROUP BY`, `ORDER BY`, внутри `AVG` и других; псевдоним `AS` помогает не повторять длинное выражение.

Примеры путей:

Элемент	Описание
`'$.level'`	поле `level` корневого объекта
`'$.user.name'`	поле `name` вложенного объекта `user`
`'$.tags[0]'`	первый элемент массива `tags`
`'$.tags[#-1]'`	последний элемент массива
`'$'`	весь документ

Пример:

```
-- таблица: logs (id, body), в body лежит JSON
SELECT id, json_extract(body, '$.status') FROM logs ORDER BY id;
```

```
SELECT json_extract(body, '$.client.city') AS city, COUNT(*) FROM logs GROUP BY
city ORDER BY city;
SELECT id FROM logs WHERE json_extract(body, '$.bytes') > 500 ORDER BY id;
SELECT body ->> '$.status', json_valid(body) FROM logs ORDER BY id;
```

json_each — массив JSON как таблица

Назначение. Функция `json_each` разворачивает массив (или объект) JSON в таблицу: одна строка на элемент. Так можно искать по элементам массива и считать их.

Синтаксис:

```
SELECT ... FROM таблица, json_each(таблица.столбец, '$.массив') AS алиас
WHERE алиас.value = 'значение';
```

Правила:

- `json_each` стоит в списке `FROM` через запятую (или `JOIN`) и сразу «видит» столбцы таблицы слева: для каждой строки таблицы разворачивается её собственный массив;
- значение элемента — в столбце `value`, позиция элемента — в столбце `key` (с нуля);
- если массива в строке нет или он пуст, строка исходной таблицы в результат не попадает;
- один документ с несколькими подходящими элементами даёт несколько строк; чтобы посчитать документы, а не элементы, используйте `COUNT(DISTINCT ...)` или `DISTINCT`;
- для простого вопроса «сколько элементов в массиве» есть `json_array_length(json, путь)`.

Столбцы, которые возвращает json_each:

Элемент	Описание
`key`	индекс элемента массива или имя поля объекта
`value`	значение элемента как значение SQL
`type`	тип элемента: `text`, `integer`, `real`, `true`, `false`, `null`, `array`, `object`
`fullkey`	полный путь к элементу, например `$.tags[0]`

Пример:

```
-- у записи в logs: {"status": 200, "tags": ["web", "cache"]}
SELECT l.id, t.value FROM logs l, json_each(l.body, '$.tags') t ORDER BY l.id,
t.key;
SELECT DISTINCT l.id FROM logs l, json_each(l.body, '$.tags') t WHERE t.value =
'cache' ORDER BY l.id;
SELECT t.value, COUNT(*) FROM logs l, json_each(l.body, '$.tags') t GROUP BY
t.value ORDER BY t.value;
SELECT id, json_array_length(body, '$.tags') FROM logs ORDER BY id;
```

json_set — изменение JSON

Назначение. Функции правки не меняют исходное значение, а **возвращают новый JSON-текст** с изменением. Чтобы изменить данные в базе, результат записывают обратно в столбец командой UPDATE.

Синтаксис:

```
UPDATE таблица SET столбец = json_set(столбец, '$.поле', значение)
WHERE условие;
```

Правила:

- json_set создаёт поле, если его нет, и заменяет значение, если оно есть;
- остальные поля документа остаются без изменений — именно поэтому правку делают функцией, а не заменой всего текста;
- чтобы изменить только нужные строки, обязательно ставьте WHERE; без него изменятся все документы;
- число и текст записываются соответственно как число и строка JSON; чтобы записать вложенный JSON, оберните его в json('...'), иначе он попадёт в документ как обычная строка;
- перед массовой правкой проверьте условие запросом SELECT COUNT(*) ... WHERE ..., а результат — запросом json_extract.

Семейство функций правки и построения:

Элемент	Описание
json_set(j, путь, знач)	записать значение: создать или заменить
json_insert(j, путь, знач)	добавить поле только если его ещё нет
json_replace(j, путь, знач)	заменить значение только если поле уже есть
json_remove(j, путь)	удалить поле или элемент массива
json_object('a', 1, '6', 2)	построить объект JSON из пар имя-значение
json_array(1, 2, 3)	построить массив JSON

Пример:

```
UPDATE logs SET body = json_set(body, '$.checked', 1) WHERE json_extract(body,
'$.status') >= 500;
SELECT json_insert('{"a":1}', '$.a', 9, '$.b', 2);   -- {"a":1,"b":2}
SELECT json_replace('{"a":1}', '$.b', 2);            -- {"a":1}: поля b не было
SELECT json_remove('{"a":1,"b":2}', '$.a');          -- {"b":2}
SELECT json_object('host', 'srv01', 'ms', 120);
```

GENERATED ALWAYS AS — вычисляемый столбец и индекс

Назначение. Вычисляемый (генерируемый) столбец хранит не данные, а формулу, выражающую его через другие столбцы той же строки. Если формула достаёт поле из JSON, поле читается как обычный столбец, а по нему можно построить индекс.

Синтаксис:

```
ALTER TABLE таблица ADD COLUMN имя ТИП GENERATED ALWAYS AS (выражение) VIRTUAL;
CREATE TABLE т (..., имя ТИП GENERATED ALWAYS AS (выражение) STORED);
CREATE INDEX имя_индекса ON таблица(имя);
```

Правила:

- слова `GENERATED ALWAYS` можно опустить: `имя ТИП AS (выражение)`; вычисляемые столбцы появились в SQLite 3.31;
- `VIRTUAL` — значение вычисляется при каждом чтении и места не занимает; `STORED` — вычисляется при записи и хранится в файле;
- командой `ALTER TABLE ... ADD COLUMN` можно добавить только `VIRTUAL` столбец; `STORED` задают при создании таблицы;
- вставить или изменить значение такого столбца нельзя (`INSERT`, `UPDATE` вернут ошибку): оно всегда равно формуле;
- выражение должно быть детерминированным: функции вроде `random()` или `datetime('now')` запрещены;
- индекс по вычисляемому столбцу хранит готовые значения, поэтому поиск `WHERE level = 'error'` не разбирает JSON в каждой строке;
- `SELECT *` показывает вычисляемые столбцы, а `PRAGMA table_info` их **не** перечисляет — полный список даёт `PRAGMA table_xinfo(таблица);`.

Пример:

```
ALTER TABLE logs ADD COLUMN status INTEGER GENERATED ALWAYS AS (json_extract(body,
'$.status')) VIRTUAL;
SELECT id, status FROM logs WHERE status >= 500 ORDER BY id;
CREATE INDEX idx_logs_status ON logs(status);
EXPLAIN QUERY PLAN SELECT id FROM logs WHERE status = 404;
```

FTS5 — полнотекстовый поиск

Назначение. FTS5 — встроенный модуль полнотекстового поиска. Он строит индекс слов и находит документы, в которых слова встречаются, быстрее и гибче, чем `LIKE '%слово%'`.

Синтаксис:

```
CREATE VIRTUAL TABLE имя USING fts5(столбец1, столбец2);
INSERT INTO имя (столбец1, столбец2) SELECT ... FROM исходная;
SELECT ... FROM имя WHERE имя MATCH 'запрос';
```

Правила:

- модуль входит в стандартную сборку SQLite и в модуль `sqlite3` Python; проверить: `SELECT sqlite_compileoption_used('ENABLE_FTS5');`;
- виртуальная таблица — отдельная копия текста: она **не следит** за исходной таблицей, после изменений в `docs` данные нужно загрузить заново (или связать таблицы триггерами);
- слева от `MATCH` ставят имя виртуальной таблицы (а не столбца); поиск не различает регистр букв, в слове сравниваются целые слова: `cop`y не найдёт `copies`;
- операторы `AND`, `OR`, `NOT` пишутся **только заглавными**; слова без оператора означают `AND`; строчное `and` ищется как обычное слово;
- фразу, а также слова с дефисом, точкой и другими знаками берут в двойные кавычки внутри одинарных: `'"full disk"'`;

- морфологии для русского языка нет: `копия` и `копии` — разные слова; помогает префикс `коп*`;
- в столбцах FTS-таблицы типов нет; значения хранятся как текст.

Синтаксис запроса MATCH:

Элемент	Описание
`слово1 слово2`	в документе есть оба слова (равно `AND`)
`слово1 OR слово2`	есть хотя бы одно из слов
`слово1 NOT слово2`	первое слово есть, второго нет
`"два слова"`	слова идут подряд именно в таком порядке
`преф*`	слово начинается с `преф`
`title:слово`	искать только в столбце `title`
`NEAR(слово1 слово2, 5)`	слова расположены не дальше чем в пяти словах друг от друга

Пример:

```
CREATE VIRTUAL TABLE notes_fts USING fts5(title, body);
INSERT INTO notes_fts (title, body) SELECT title, body FROM notes;

SELECT title FROM notes_fts WHERE notes_fts MATCH 'disk AND alert' ORDER BY title;
SELECT title FROM notes_fts WHERE notes_fts MATCH 'disk NOT swap' ORDER BY title;
SELECT title FROM notes_fts WHERE notes_fts MATCH '"full disk"';
SELECT title FROM notes_fts WHERE notes_fts MATCH 'title:disk OR mon*' ORDER BY
title;
SELECT title FROM notes_fts WHERE notes_fts MATCH 'disk' ORDER BY rank;   -- самые
подходящие первыми
```

ATTACH — подключение второй базы

Назначение. Команда `ATTACH` подключает к текущему соединению ещё один файл базы под коротким именем. Таблицы подключённой базы доступны в запросах как `имя.таблица`, поэтому данные из двух файлов можно соединять одним запросом.

Синтаксис:

```
ATTACH 'файл.db' AS имя;
SELECT ... FROM таблица1 JOIN имя.таблица2 ON ...;
DETACH имя;
```

Правила:

- основная база называется `main`, временная — `temp`; имя подключённой базы задаёте вы после `AS`;
- имя файла — строка в одинарных кавычках; в Windows путь можно писать с прямыми косыми чертами: `'C:/data/shop.db'`; относительный путь считается от текущего каталога;
- подключение действует только в текущем сеансе: в новом запуске `sqlite3` команду `ATTACH` нужно ввести заново; в командной строке её пишут в том же вызове перед запросом;
- `ATTACH` несуществующего файла **создаёт** новую пустую базу — проверьте имя файла, иначе запрос увидит пустые таблицы или выдаст `no such table`;

- таблицу из подключённой базы без префикса SQLite ищет после `main`; чтобы не путаться, всегда пишите `имя.таблица`;
- внутри открытой транзакции `ATTACH` не выполняется;
- в Python имя файла передают параметром: `c.execute('ATTACH ? AS shop', (путь,))`; список подключённых баз выводит `PRAGMA database_list;`.

Пример:

```
ATTACH 'archive.db' AS arch;
PRAGMA database_list;
SELECT p.name, a.reason
FROM products p JOIN arch.removed a ON a.product_id = p.id
ORDER BY p.name;
DETACH arch;

# одной командой в оболочке
sqlite3 work.db "ATTACH 'archive.db' AS arch; SELECT COUNT(*) FROM arch.removed;"
```

59.2. Задания

Рабочий каталог: `~/sql-lab/18_json_fts`. Задания выполняются *по порядку*: каждое опирается на результат предыдущих. Ответ, где он нужен, записывайте в файл `otvety/N`, где `N` — номер задания.

1. Сколько событий в таблице `events` имеют уровень `error` (`$.level` в столбце `payload`)? Используйте `json_extract(payload, '$.level')` в `WHERE` и `COUNT(*)`. Запишите число в `otvety/1`.
2. Выведите различные имена серверов (`$.host`) по возрастанию (`SELECT DISTINCT json_extract(payload, '$.host') ... ORDER BY`). Результат — в `otvety/2`.
3. Для каждого сервера выведите его имя и среднее значение `$.ms`, округлённое до одного знака (`ROUND(AVG(...), 1)`), по имени сервера: `json_extract(payload, '$.host')` с псевдонимом в `GROUP BY` и `AVG(json_extract(payload, '$.ms'))`. Результат — в `otvety/3`.
4. Поле `$.tags` — массив строк. Выведите номера (`id`) событий, у которых среди тегов есть `db`, по возрастанию. Разверните массив функцией `json_each(e.payload, '$.tags')` (в `FROM` через запятую) и отберите строки `value = 'db'`. Результат — в `otvety/4`.
5. Добавьте в `events` командой `ALTER TABLE ... ADD COLUMN` вычисляемый столбец `level` (`TEXT GENERATED ALWAYS AS (json_extract(payload, '$.level')) VIRTUAL`), чтобы уровень можно было читать как обычное поле.
6. Создайте индекс `idx_events_level` по столбцу `level` таблицы `events` (`CREATE INDEX имя ON таблица(столбец);`).
7. Создайте виртуальную таблицу `docs_fts` по модулю `fts5` со столбцами `title` и `body` (`CREATE VIRTUAL TABLE ... USING fts5(...)`) и заполните её всеми строками таблицы `docs` запросом `INSERT INTO docs_fts (title, body) SELECT title, body FROM docs`.
8. Найдите названия документов (`title`), в тексте которых (`title` или `body`) есть **оба** слова `backup` и `restore`. Используйте `docs_fts MATCH 'backup AND restore'`. Названия по возрастанию — в `otvety/8`.
9. Подключите `shop.db` (`ATTACH 'shop.db' AS shop;`) и выведите без повторов имена клиентов (`shop.customers`), чей номер указан в `$.user` событий с уровнем `error`; имена по возрастанию. Таблица `shop.customers` имеет столбцы `id` и `name`; соедините `JOIN shop.customers cu ON cu.id = json_`

`extract(e.payload, '$.user')`. Результат — в `otvety/9`.
10. Всем событиям уровня `error` добавьте в JSON поле `reviewed` со значением `1` (`json_set(payload, '$.reviewed', 1)`). Остальные события и остальные поля менять нельзя: нужен `UPDATE events SET payload = json_set(...) WHERE ...` с условием по `json_extract(payload, '$.level')`.

Вопросы для размышления

1. Когда хранить данные в JSON-столбце разумно, а когда лучше завести обычные столбцы и таблицы?
2. Чем полнотекстовый поиск лучше `LIKE '%слово%'` и что он не умеет?

КНИГА II

Сетевое администрирование

Как устроена сеть и как её построить, проверить и починить: адреса, коммутация, маршрутизация, службы, фильтрация, беспроводные сети и наблюдение. Лабораторные работы в Cisco Packet Tracer и проекты на стыке с системами.

О книге II: сетевое администрирование

Что делает сетевой администратор

Сетевой администратор отвечает за то, чтобы устройства организации могли обмениваться данными, а лишнего обмена не было. Он планирует адреса, подключает и настраивает коммутаторы и маршрутизаторы, делит сеть на изолированные группы, обеспечивает резервные пути, выдаёт адреса и имена, ограничивает доступ, публикует службы, настраивает беспроводные сети и следит за состоянием сети. Главное умение — *по симптому найти причину*: «не открывается сайт» может означать разные вещи на разных уровнях.

Лабораторная работа: как она устроена

Книга II состоит из 48 лабораторных работ (01–48) и пяти проектов на стыке с системами (I01–I05). Каждая работа решает *одну проблему*, которая названа в самом начале, и устроена по одному плану.

Блок работы	Зачем он нужен
Какую проблему решает работа	Реальная ситуация, в которой без этого умения администратор не справится, и что случится, если его нет
Что вы научитесь делать	Конкретный измеримый результат работы
Стенд и адреса	Схема, устройства, таблица адресов и подключений: единственный источник, по которому всё настраивается
Порядок работы	Пошаговые действия с объяснением: что нажать или ввести, что ожидать увидеть и почему
Проверка результата	Как убедиться, что работа выполнена, и как это доказать наблюдением
Управляемая неисправность	Одна намеренная поломка, чтобы научиться связывать симптом с причиной
Что сдать и объяснить	Минимальный отчёт; ответы на вопросы «почему так»
Самостоятельное усложнение	Необязательное задание, закрепляющее умение

Части книги II

Часть	Работы	Главная проблема
Первые шаги и устройство сети	01–08	Собрать первую сеть, понять адреса, настроить устройства и связать две сети

Маршруты и коммутация	09–18	Связать много сетей, разделить их на VLAN, защитить от петель, объединить кабели
Сетевые службы и политики доступа	19–26	Выдавать адреса и имена, публиковать службы, ограничивать доступ, экономить адреса
Динамические маршруты, IPv6 и беспроводные сети	27–35	Масштабировать сеть, внедрить IPv6, построить беспроводную сеть
Обслуживание, диагностика и проекты	36–40	Наблюдать, восстанавливать, искать неисправности, проектировать целиком
Системы в сети	I01–I05	Увидеть, как настоящие системы пользуются сетью, и передать инфраструктуру

Что нужно знать до начала

Опыт работы с сетью не требуется. Полезно, но не обязательно, уметь работать в терминале (первая часть книги I). Работы выполняются в Packet Tracer на обычном компьютере; для проектов I01–I05 нужны настоящие виртуальные машины. Работы идут по нарастающей сложности: первые подробно объясняют каждое действие, последние (37–40) требуют самостоятельности и дают только цель и критерии.

Часть I

Первые шаги и устройство сети

Глава 22

Что такое сеть и зачем нужен Packet Tracer

22.1 Начнём с обычной ситуации

Представьте учебный кабинет. За одним компьютером работает Анна, за другим – Борис. Анна подготовила документ и хочет передать его Борису. Можно перенести файл на флешке. Но если компьютеры соединены и умеют обмениваться данными, документ можно передать по этому соединению. Та же возможность позволяет нескольким людям пользоваться общим принтером или обращаться к одному учебному сайту.

Компьютерная сеть – это устройства, соединённые для обмена данными. Данные могут представлять текст, изображение, звук или запрос к другой программе. Соединение бывает проводным, через кабель, и беспроводным, например через Wi-Fi. В начале курса мы будем использовать кабели: на рисунке легко увидеть, что с чем соединено.

Небольшую сеть в кабинете или квартире называют **локальной сетью**. Интернет объединяет множество сетей. Поэтому локальная сеть может работать и без подключения к Интернету: компьютерам в одном кабинете для обмена между собой не обязательно обращаться к внешнему провайдеру.

Чтобы обмен состоялся, недостаточно только соединить устройства. Им нужны правила: как обозначить получателя, как оформить сообщение и как ответить. Такой согласованный набор правил называют **протоколом**. Пока достаточно самой идеи: участники должны понимать сообщения друг друга. Названия конкретных протоколов появятся тогда, когда мы будем наблюдать их работу.

22.2 Какие устройства участвуют в сети

22.2.1 Компьютер: отправитель и получатель

Компьютер пользователя создаёт и принимает данные. Когда Анна отправляет документ, её компьютер является отправителем; компьютер Бориса – получателем. Когда Борис отвечает, роли меняются. Устройство, на котором начинается или заканчивается обмен, называют **конечным устройством**. Обычный компьютер – один из примеров; к конечным устройствам также относятся сетевые принтеры и телефоны.

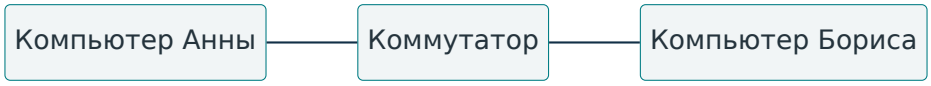
В нашем первом опыте будет два компьютера. Они представляют два разных рабочих места. Это важно: мы будем изучать обмен между устройствами, а не передачу файла из одной папки в другую на одном компьютере.

22.2.2 Коммутатор: место подключения нескольких устройств

У компьютера обычно есть разъём для сетевого кабеля. Если соединить напрямую два подходящих устройства, получится одна линия между ними. Но в кабинете могут быть десять компьютеров и принтер. Для их подключения используют **коммутатор**: устройство с несколькими сетевыми разъёмами, которое передаёт полученные данные через нужные соединения внутри локальной

сети.

Каждый компьютер соединяется отдельным кабелем с коммутатором. Когда данные идут от Анны к Борису, они сначала попадают в коммутатор, затем передаются дальше. Как коммутатор выбирает выход, мы разберём во второй главе. Сейчас важно понять его место в схеме: он связывает рабочие места, а не выполняет за них пользовательские программы.



Две линии на рисунке – два отдельных кабеля.

22.2.3 Что появится позже

Маршрутизатор передаёт данные между разными IP-сетями. Он понадобится, например, чтобы соединить сеть кабинета с другой сетью. **Сервер** предоставляет другим устройствам определённую службу: хранит общий сайт или принимает запросы к файлам. Слово «сервер» описывает роль; эту роль выполняет компьютер с соответствующей программой.

В первой схеме нам достаточно компьютеров и коммутатора. Мы сначала научимся видеть и собирать эти соединения. Затем настроим адреса и проверим обмен; после этого добавим другие устройства. Так каждое новое устройство будет решать уже понятную задачу.

22.3 Что именно делает Packet Tracer

Packet Tracer – программа Cisco для обучения работе компьютерных сетей. Вместо покупки двух компьютеров и коммутатора мы размещаем их **модели** в окне программы. Вместо настоящего кабеля выбираем соединение мышью. Программа рассчитывает, как учебные устройства реагируют на настройки и передают сообщения.

Такую программу называют **симулятором**: она воспроизводит выбранные свойства системы. Значок компьютера в Packet Tracer – учебная модель компьютера с некоторыми сетевыми настройками и приложениями. Мы не устанавливаем на него полноценную Windows. Изменения внутри нашей схемы относятся к этой модели, а не к сетевым настройкам компьютера, на котором открыта книга.

Это позволяет ставить небольшие опыты: соединить два устройства, изменить одно условие и посмотреть, что произошло. Позже мы сможем замедлить обмен и рассмотреть отдельные сообщения. Модель воспроизводит часть возможностей настоящего оборудования, поэтому её результаты нельзя использовать как измерение реальной скорости сети или проверку всех функций Cisco [2].

Что будет результатом первого занятия

Вы получите сохранённую схему из двух компьютеров и коммутатора и сможете объяснить назначение каждого элемента. Проверять передачу сообщений мы начнём во втором занятии, после объяснения сетевых адресов. Сначала – устройство схемы и управление программой, затем – настройка обмена.

22.4 Первый взгляд на окно программы

Если Packet Tracer ещё не установлен, используйте официальный ресурс Cisco Networking Academy: `https://www.netacad.com/resources/lab-downloads`. Требования к компьютеру и порядок установки зависят от выпуска; следуйте указаниям Cisco для скачиваемой версии [1, 3]. Если программа уже открыта, можно продолжить чтение с её окном перед глазами.

Центральная большая область – место для нашей схемы. В программе она называется *Logical Workspace*, то есть логическая рабочая область: здесь видно, какие устройства соединены. Сам рисунок устройств и связей называется **топологией**. Для чтения первого рисунка достаточно понимать: значок обозначает устройство, линия – кабель.

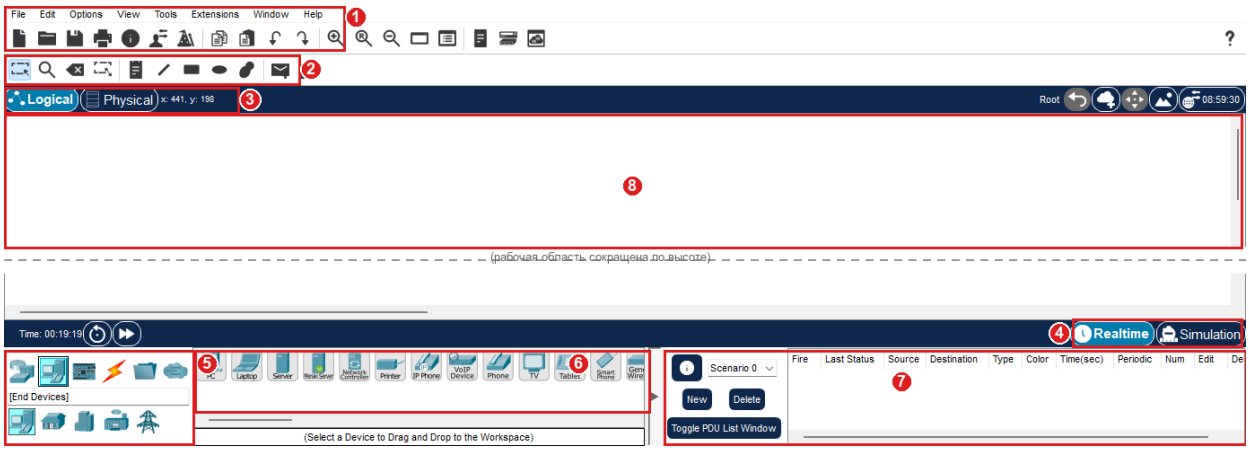
В нижней части окна находится область выбора устройств. Сначала выбирается категория, затем модель в этой категории. Для первых действий понадобятся всего три категории:

- *End Devices* – конечные устройства; здесь находится учебный компьютер.
- *Network Devices / Switches* – сетевые устройства, затем коммутаторы.
- *Connections* – соединения; здесь выбирается кабель.

Чтобы узнать название значка, задержите над ним указатель мыши и прочитайте подсказку. Не нужно запоминать положение каждой картинки: ищите название категории и модели. В разных версиях оформление панели может немного отличаться.

Щелчок по уже размещённому устройству открывает **окно этого устройства**. Оно отличается от общей рабочей области: на схеме видны все устройства, а в таком окне – параметры одного выбранного компьютера или коммутатора. Вкладка *Config* означает «Настройка». В первом занятии через неё мы изменим только подпись устройства.

Внизу справа есть режим *Realtime* – работа модели с автоматическим продвижением времени. Первую схему мы соберём в нём. Другой режим, *Simulation*, даёт управление шагами обмена; его назначение и кнопки будут разобраны перед работой 03. Вкладка рабочей области *Physical* показывает размещение оборудования; к ней вернёмся, когда будем обсуждать помещения и стойки. Ниже показано окно программы целиком. Красные цифры на рисунке соответствуют списку под ним; рисунок снят в Packet Tracer 8.2.1 с английским интерфейсом (других языков в этой версии нет).



Окно Packet Tracer. Рабочая область по высоте сокращена, чтобы рисунок уместился на странице.

1. **Меню и главная панель инструментов.** Меню *File, Edit, Options, View, Tools, Extensions, Window, Help*. Под ним значки «Новый проект», «Открыть», «Сохранить», «Печать», масштаб.
2. **Вторая панель инструментов.** Слева направо: *Select* (выбор, стрелка), *Inspect* (лупа), *Delete* (ластик), *Resize Shape*, *Place Note* (заметка), инструменты рисования, *Add Simple PDU* (закрытый конверт) и *Add Complex PDU* (открытый конверт).
3. **Переключатель Logical / Physical.** Мы работаем во вкладке *Logical*.
4. **Переключатель Realtime / Simulation.** Подсвеченный режим включён. Сейчас включён *Realtime*.
5. **Палитра устройств.** Верхний ряд значков – классы: *Network Devices* (сетевое оборудование), *End Devices* (конечные устройства), *Components*, *Connections* (кабели), *Miscellaneous*, *Multiuser*. После щелчка по классу во втором ряду появляются подклассы (например, у Network Devices – Routers, Switches, Hubs, Wireless Devices). Название выбранного класса написано под верхним рядом, например *[End Devices]*.
6. **Список моделей.** Здесь находятся значки конкретных моделей выбранного подкласса. Подпись внизу этой области показывает название модели под курсором.
7. **Список PDU.** Таблица заданий Add Simple PDU; она нужна в работе 03 и позже.
8. **Рабочая область** (Logical Workspace) – здесь собирается схема.

Рядом со счётчиком *Time* внизу слева есть две круглые кнопки. Правая, с двойной стрелкой, называется *Fast Forward Time*: она ускоряет модельное время. Это пригодится, потому что линии на коммутаторе становятся зелёными только после ожидания; три нажатия заменяют ожидание в полминуты.

Настройка программы, которую стоит сделать один раз

Откройте *Options / Preferences* (Ctrl+R), вкладку *Interface* и поставьте галочку *Always Show Port Labels in Logical Workspace*. После этого на схеме у каждого конца кабеля будет видно имя порта (Fa0, Fa0/1, Gig0/0). Так проще проверять, не перепутали ли вы порт. Галочка *Show Device Model Labels* отвечает за подпись модели (PC-PT, 2960-24TT) под значком; в схемах следующих глав она выключена, чтобы подписи не налезали друг на друга, а в первых трёх работах оставлена включённой.

22.5 PC-PT, 2960 и PC-A: три разных вида названий

PC-PT – название учебной модели персонального компьютера в каталоге Packet Tracer. *PC* означает personal computer, персональный компьютер; *PT* относится к Packet Tracer. Когда в задании встречается PC-PT, речь идёт о том, какую модель выбрать в панели.

2960 – обозначение модели коммутатора Cisco Catalyst. Оно помогает выбрать в каталоге нужный коммутатор. Это не номер компьютера, не количество устройств и не сетевой адрес. В наших начальных опытах у этой модели достаточно разъёмов, и стандартные настройки позволяют соединить компьютеры в одну локальную сеть.

PC-A, **PC-B** и **S1** – имена, которые мы сами дадим размещённым устройствам. Буквы A и B различают два компьютера; S происходит от английского *switch*, коммутатор, а 1 означает первый коммутатор в схеме. Имя помогает читать схе-

му, но само по себе не настраивает сетевой обмен. Первоначально программа может назвать устройства PC0, PC1 и Switch0; мы заменим эти подписи.

Надпись	Что она обозначает	Как ею пользоваться
PC-PT	Модель компьютера	Выбрать в каталоге конечных устройств.
2960	Модель коммутатора	Выбрать в каталоге коммутаторов.
PC-A	Имя нашего первого ПК	Присвоить устройству после размещения.
S1	Имя нашего коммутатора	Присвоить устройству после размещения.

22.6 Порт и кабель: как появляются соединения

Сетевой порт в этой главе – разъём для подключения кабеля. Два конца кабеля подключаются к двум портам. Выбрать только устройства недостаточно: у коммутатора много разъёмов, поэтому программа спрашивает, какой именно использовать. Слово «интерфейс» в описании соединения означает сетевое подключение устройства; в нашем опыте оно соответствует выбранному порту.

У модели PC-PT сетевой порт называется `FastEthernet0`. *Fast Ethernet* – вариант проводной технологии Ethernet со скоростью 100 Мбит/с, а 0 различает интерфейсы модели. У коммутатора нужные порты называются `FastEthernet0/1` и `FastEthernet0/2`. Последнее число различает два разъёма. В кратких подписях `Fa0/1` означает тот же порт, что `FastEthernet0/1`.

Copper Straight-Through переводится как «медный прямой кабель». *Copper* обозначает медную среду передачи; *straight-through* – прямую схему соединения проводников. Для подключения компьютера к коммутатору в этой работе выбираем именно этот тип [4]. В программе он обычно показан сплошной линией. Перед выбором проверьте название в подсказке, а не только цвет значка.

После подключения возле концов линии появляются индикаторы. Зелёный означает, что соответствующая линия активна. На коммутаторе индикатор может сначала быть оранжевым, а затем стать зелёным: устройство ещё подготавливает передачу. Подождите около минуты моделируемого времени. Причину этой задержки мы подробно изучим в главе о предотвращении петель.

Активная линия подтверждает готовность соединения на этом участке. Для передачи сообщения по адресу ещё понадобятся настройки, которые объясняются в следующей главе. Полезно разделять два вопроса: «есть ли соединение?» и «может ли получатель ответить на сообщение?».

22.7 Разобранный пример: одно рабочее место

Теперь понятен смысл будущих действий. Разберём подключение компьютера Анны к коммутатору по шагам. Выполняйте их сразу: этот пример – первая половина лабораторной 01. На рисунках красные цифры указывают, куда нажимать.

Шаг 1. Создать пустой проект

Выберите в меню *File / New* (Ctrl+N). Если программа спросит, сохранять ли текущую схему, ответьте по смыслу (на чистом запуске вопроса не будет). Проверьте, что включён режим *Realtime*. Результат – пустая белая рабочая область.

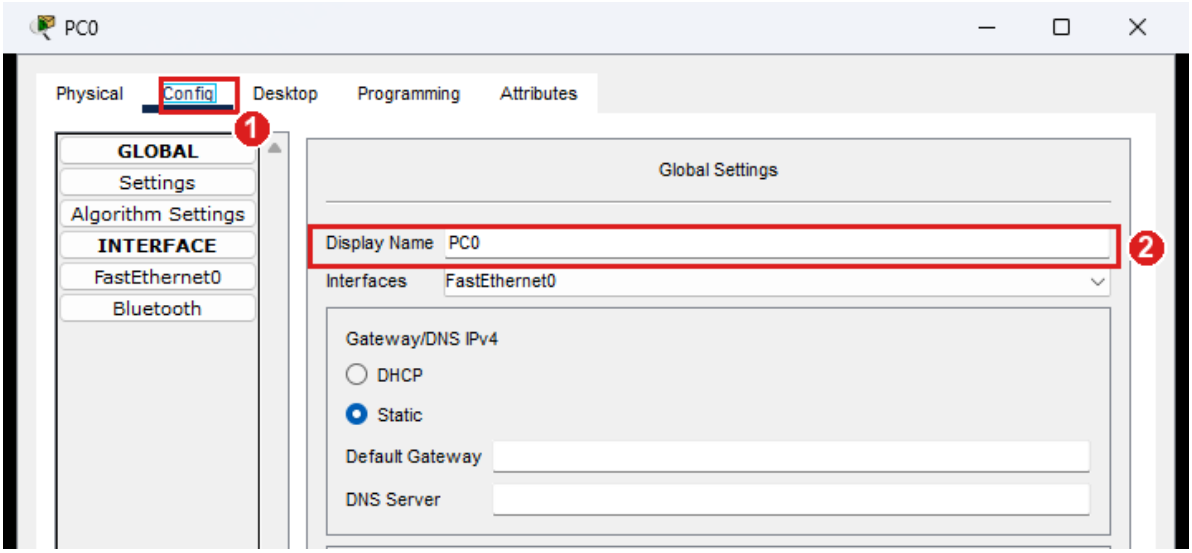
Шаг 2. Добавить компьютер и дать ему имя

1. В палитре устройств (левый нижний угол) щёлкните второй значок верхнего ряда – *End Devices*. Под рядом появится подпись *[End Devices]*, а справа – модели: *PC, Laptop, Server, Printer* и другие.
2. Нажмите на значок *PC*, не отпуская кнопку мыши перетащите его в левую часть рабочей области и отпустите. На схеме появится компьютер с двумя подписями: *PC-PT* (модель) и *PC0* (имя, которое дала программа).



Палитра: 1 – класс End Devices, 2 – модель PC.

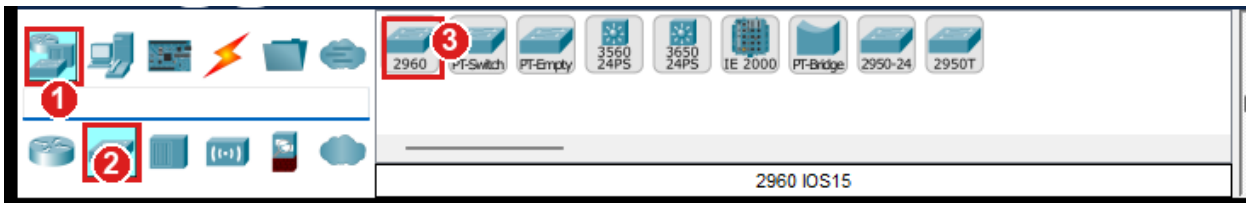
3. Щёлкните по значку компьютера один раз. Откроется отдельное окно устройства с вкладками *Physical, Config, Desktop, Programming, Attributes*.
4. Откройте вкладку *Config*. В разделе *GLOBAL / Settings* находится поле *Display Name*. Дважды щёлкните по тексту `PC0` в этом поле, чтобы выделить его, введите `PC-A` и нажмите Enter.
5. Закройте окно устройства крестиком в правом верхнем углу окна. Подпись под значком станет *PC-A*; модель *PC-PT* не изменилась.



Окно компьютера: 1 – вкладка Config, 2 – поле Display Name.

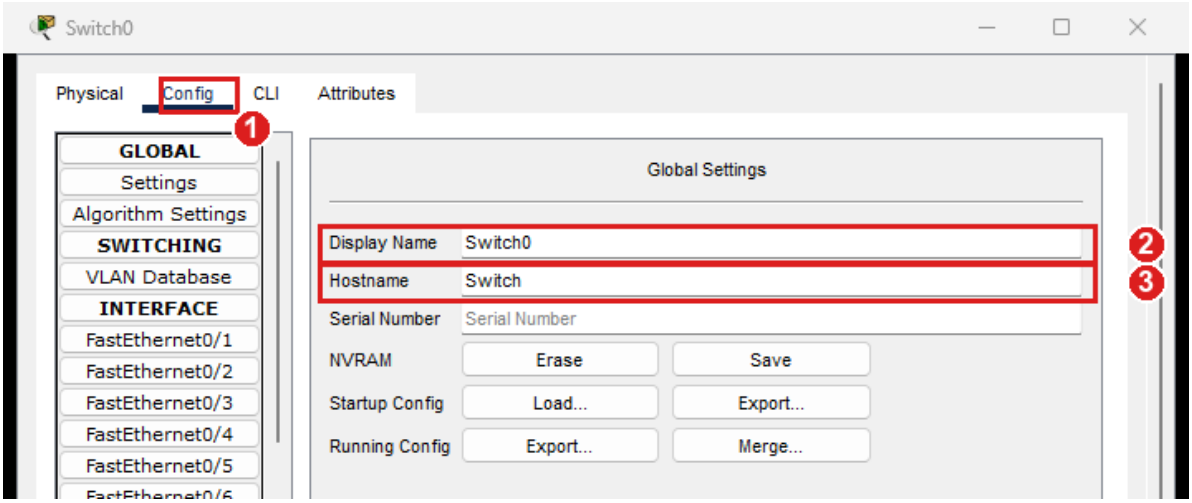
Шаг 3. Добавить коммутатор

1. В палитре щёлкните первый значок верхнего ряда – *Network Devices*. Во втором ряду выберите второй значок – *Switches* (плоская коробка).
2. В списке моделей перетащите на рабочую область значок *2960* и разместите его правее PC-A. На схеме появится устройство с подписью *2960-24TT* (полное имя модели) и именем *Switch0*.



Палитра: 1 – Network Devices, 2 – подкласс Switches, 3 – модель 2960.

3. Щёлкните по коммутатору, откройте вкладку *Config*. У коммутатора два разных поля: *Display Name* и *Hostname*. Измените *Display Name* на `S1` и нажмите Enter. Это подпись на схеме. Поле *Hostname* – имя в командной строке (приглашение `Switch>`); оно не меняется вместе с подписью. Его мы изменим командой `hostname` в главе о CLI, а сейчас оставляем.



Окно коммутатора: 1 – вкладка Config, 2 – Display Name (подпись на схеме), 3 – Hostname (имя в CLI).

Шаг 4. Соединить два конкретных порта

1. В палитре щёлкните четвёртый значок верхнего ряда – *Connections* (молния). Справа появятся виды кабелей. Наведите курсор на чёрную сплошную линию: внизу появится подпись *Copper Straight-Through*. Щёлкните по ней.

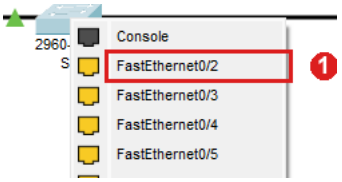


Палитра кабелей: 1 – класс Connections, 2 – Copper Straight-Through (чёрная сплошная линия).

2. Щёлкните по значку PC-A. Появится маленький список портов компьютера. Выберите `FastEthernet0`. Курсор превратится в начало провода.
3. Щёлкните по значку S1. Появится длинный список портов коммутатора. В нём показаны только свободные порты. Выберите `FastEthernet0/1`.



Список портов компьютера: выберите FastEthernet0.



Список портов коммутатора: первым в списке стоит первый свободный порт.

Ожидаемый результат – линия между PC-A и S1. Её смысл можно записать одной строкой:

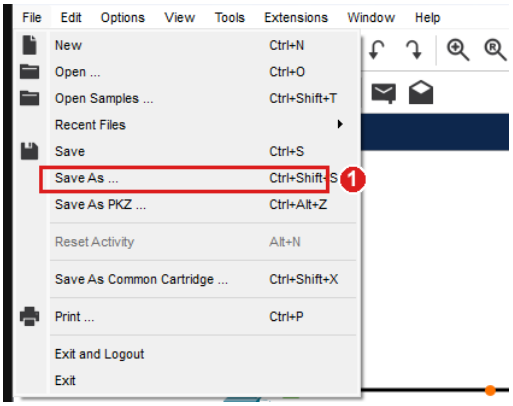
`PC-A: FastEthernet0` ⟷ `S1: FastEthernet0/1`.

Это запись двух концов одного кабеля. У концов линии появятся индикаторы: зелёный треугольник на компьютере и оранжевый кружок на коммутаторе. Оранжевый означает, что порт коммутатора ещё готовится передавать данные; через полминуты модельного времени он станет зелёным (или нажмите *Fast Forward Time* два-три раза).

Две особенности версии 8.2.1, которые легко принять за ошибку. Первая: после каждого проведённого кабеля инструмент *Connections* сбрасывается, и для следующего кабеля его нужно выбрать заново (иначе щелчок по устройству просто откроет его окно). Вторая: если линия не появилась, прежде всего проверьте, что вы выбрали порты обоих устройств; ошибочный кабель можно удалить инструментом *Delete* (ластик во второй панели инструментов).

Шаг 5. Сохранить промежуточный результат

Выберите *File / Save As* (Ctrl+Shift+S). В окне сохранения выберите папку, введите имя `L01-working` и убедитесь, что тип файла – проект Packet Tracer (расширение `.pkt`). Нажмите «Сохранить». Дальше обычное *File / Save* (Ctrl+S) обновляет этот же файл. В заголовке окна программы появится путь к файлу.



Меню File: 1 – Save As.

Короткая остановка перед практикой

Попробуйте своими словами закончить фразы: «PC-PT – это …», «PC-A – это …», «S1 нужен, чтобы …», «кабель соединяет …».
Сверка: PC-PT – модель компьютера; PC-A – выбранное нами имя одного устройства; S1 связывает подключённые устройства локальной сети; кабель соединяет конкретные порты. Если различия пока неясны, перечитайте соответствующий раздел: скорость выполнения здесь не является целью.

Лабораторная 01. Первая схема: повторяем пример и добавляем второй компьютер

Время: 45–60 минут. **Подготовка:** объяснения и разобранный пример этой главы.

Какую проблему решает работа

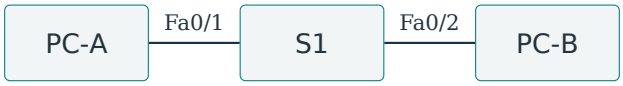
Чтобы изучать сеть, её нужно сначала собрать и увидеть. На настоящем оборудовании ошибка в подключении стоит времени и денег, а новичок не знает, какие признаки считать нормой: горит ли индикатор, подключён ли кабель, виден ли порт. Эта работа решает проблему «с чего начать»: вы повторяете готовый пример, добавляете второй компьютер и учитесь читать состояние схемы. Без этого навыка каждая следующая работа превращалась бы в борьбу с интерфейсом программы, а не с сетью.

Что вы научитесь делать

Различать модель и имя устройства, подключать кабель к заданным портам и сохранять схему так, чтобы её можно было открыть снова.

Что мы собираем и почему

Мы изображаем два рабочих места одного кабинета. Компьютер Анны называется PC-A, компьютер Бориса – PC-B. Оба подключены отдельными кабелями к S1. На схеме ниже подписи Fa0/1 и Fa0/2 обозначают порты коммутатора; у каждого компьютера используется его FastEthernet0.



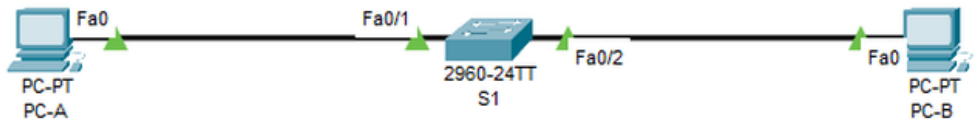
Пока мы собираем соединения. Настройка адресов будет отдельным занятием: сначала нужно разобраться с их назначением. До того момента поля сетевых адресов оставляем без изменений.

Порядок работы

1. Если вы ещё не выполняли разобранный пример, выполните его пять шагов. Проверьте, что на схеме есть PC-A, S1 и один кабель между указанными портами. Если пример уже выполнен, продолжайте с полученной схемой.
2. По образцу шага 2 добавьте ещё один PC-PT, разместите его справа от S1 и задайте Display Name `PC-B`. Это второе рабочее место; имя нужно, чтобы отличать его от первого.
3. По образцу шага 4 подключите PC-B: выберите инструмент Copper Straight-Through заново, щёлкните PC-B и выберите `FastEthernet0`, затем щёлкните

S1. В списке портов коммутатора первым будет `FastEthernet0/2`: занятый `FastEthernet0/1` в списке не показывается. Выберите `FastEthernet0/2`.

4. Дождитесь готовности линий. Сверьте свой рисунок с образцом: должно быть три устройства и два кабеля, каждый от компьютера к S1. Если линии остаются красными, проверьте тип кабеля, выбранные порты и питание устройств; не меняйте сразу несколько параметров.
5. Сохраните результат через File / Save. Затем откройте этот же `L01-working.pkt` через File / Open («Файл / Открыть»). Проверьте, сохранились ли устройства, их имена и соединения.



Итог работы 01: два компьютера на портах Fa0/1 и Fa0/2 коммутатора, все индикаторы зелёные.

Проверка результата

Результат первой работы – схема, которую можно повторно открыть: PC-A и PC-B подключены к S1, оба соединения становятся активными. Запись соединений должна соответствовать таблице ниже. Здесь проверяются сборка и сохранение, а не обмен сообщениями.

Устройство	Его порт	Устройство	Его порт	Кабель
PC-A	FastEthernet0	S1	FastEthernet0/1	Прямой медный
PC-B	FastEthernet0	S1	FastEthernet0/2	Прямой медный

Управляемая неисправность

Это небольшой опыт по уже изученному действию. Через Save As сохраните копию как `L01-fault.pkt`. Инструмент *Delete* («Удалить») удаляет выбранный элемент схемы. Выберите его и щёлкните только по кабелю PC-B–S1, затем нажмите Esc. Останутся три устройства, но одно рабочее место будет без соединения. По образцу восстановите кабель в прежние порты. Объяснение наблюдения: исчез путь подключения PC-B; сами устройства при этом не удалялись.

Что сдать и объяснить

При самостоятельном обучении достаточно сохранить рабочий проект и проверить себя по таблице. Если работа выполняется с преподавателем, приложите снимок схемы и запишите ответы на три вопроса: что обозначает PC-PT, зачем в схеме S1 и какие два порта соединяет кабель PC-B. Все ответы следуют из объяснений этой главы.

Самостоятельное усложнение. Когда основной пример получился, попробуйте без пошаговой подсказки добавить третий PC-PT. Назовите его PC-C и подключите к свободному FastEthernet0/3 на S1. Сохраните расширенную схему под

именем `L01-three-pcs.pkt`. Если потребуется помощь, используйте те же действия добавления, переименования и подключения; новых сетевых настроек в этом задании нет.

Глава 23

Пакеты, адреса и путь сообщения

В первой работе мы соединили два компьютера с коммутатором и сохранили рисунок. Теперь хотим получить наблюдаемый результат: PC-A отправляет проверочное сообщение PC-B, а PC-B отвечает. Прежде чем вводить числа и команды, разберём, как компьютер обозначает адресата и что будет считаться ответом.

23.1 Зачем компьютеру адрес

Надпись PC-B помогает человеку найти значок на схеме. Но при отправке данных компьютер не читает эту подпись как адрес доставки. Для обмена по протоколу IP интерфейсу назначают **IP-адрес**. В первой сети у каждого ПК один используемый сетевой интерфейс, поэтому можно говорить «адрес компьютера», помня, что настройка относится именно к интерфейсу.

Мы начинаем с IPv4 – версии IP, в которой привычная запись адреса состоит из четырёх чисел, разделённых точками. Пример: `192.168.10.10`. Каждое число находится между 0 и 255. Точки разделяют части одного адреса; это не четыре разных параметра. Подробное устройство двоичной записи изучим в следующей главе.

Два устройства в одной локальной IP-сети должны иметь разные адреса. В нашем примере PC-A получит `192.168.10.10`, а PC-B – `192.168.10.20`. Эти адреса относятся к частному диапазону, подходящему для внутренней учебной сети. Присваивать произвольный адрес настоящего внешнего сайта для такого опыта не нужно.

23.2 Маска: как отличить соседей от другой сети

Компьютеру недостаточно знать собственный адрес: ему нужно определить, относится ли получатель к той же IP-сети. Для этого вместе с адресом задаётся **маска подсети**. Маска указывает, какая часть адреса относится к сети, а какая различает её устройства.

Для первого опыта используется маска `255.255.255.0`. Только для этой маски действует простое правило: сравниваются первые три числа адресов. У `192.168.10.10` и `192.168.10.20` они одинаковые – `192.168.10`. Поэтому компьютеры считают друг друга соседями в одной IP-сети. У `192.168.20.20` третье число другое: с этой маской это уже другая сеть.

Запись `192.168.10.10/24` – сокращение «адрес 192.168.10.10 с маской 255.255.255.0». Число 24 называется длиной префикса; его происхождение разберём в следующей главе. В поле IPv4 Address вводится только адрес без `/24`, а в поле Subnet Mask – полная маска.

Разобранное решение

PC-A: 192.168.10.10, маска 255.255.255.0. Адресат: 192.168.10.20. Сравниваем 192.168.10 с 192.168.10: совпадают, адресат локальный. Для адресата 192.168.20.20 сравниваем 192.168.10 с 192.168.20: не совпадают, требуется

путь в другую сеть. При другой маске способ сравнения может быть другим; правило трёх чисел не универсально.

23.3 Что означают остальные поля настройки

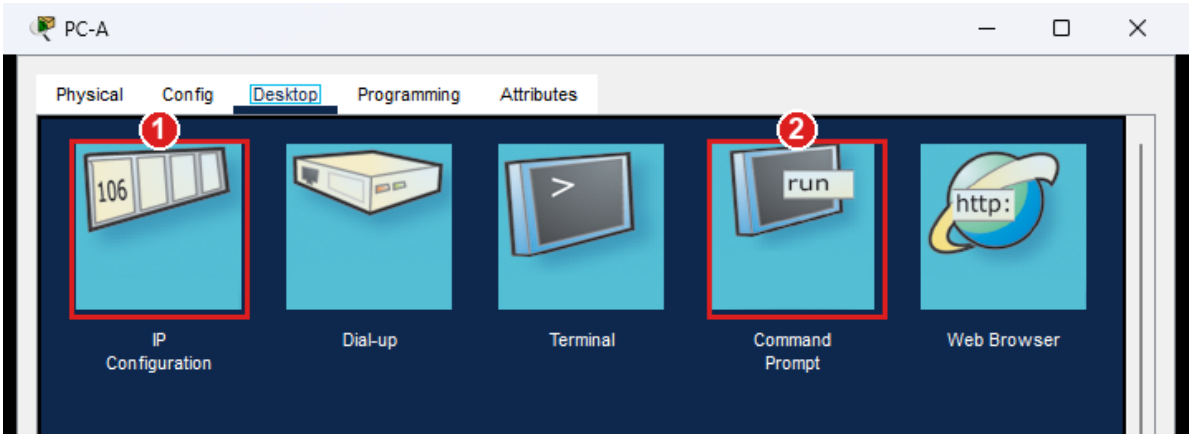
Default Gateway, или основной шлюз, – адрес интерфейса маршрутизатора в нашей сети, которому компьютер передаёт данные для других IP-сетей. Шлюз не требуется, чтобы два ПК в нашей общей подсети обменивались друг с другом. Маршрутизатора в первом стенде нет, поэтому поле шлюза оставляем пустым. Коммутатор S1 не становится шлюзом только потому, что к нему подключены кабели.

DNS Server – адрес службы, которая помогает найти IP-адрес по имени, например по имени сайта. В первом опыте мы обращаемся прямо к числовому адресу PC-B и не используем имена сайтов. Поэтому поле DNS тоже оставляем пустым. Пустое поле здесь следует из задачи, а не является общей рекомендацией для любой сети.

Во вкладке *Desktop* («Рабочий стол») учебного ПК есть приложение *IP Configuration* («Настройка IP»). В нём *Static* означает ручное назначение параметров. Вариант *DHCP* получает параметры автоматически от специальной службы; к ней вернёмся позже. Для первого опыта выбираем Static и заполняем только адрес и маску.

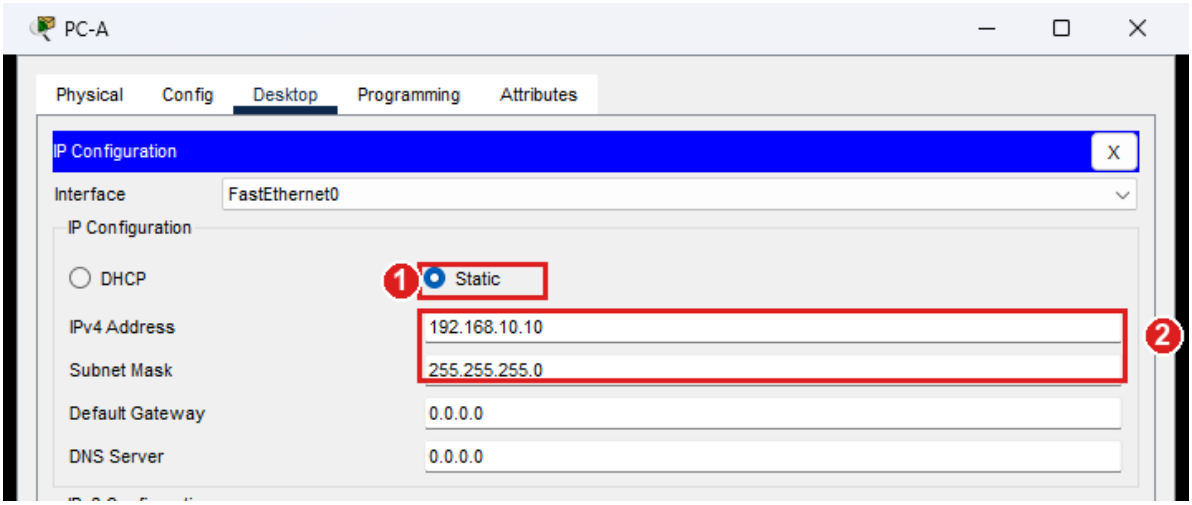
Как выглядит окно IP Configuration

Щёлкните по ПК, откройте вкладку *Desktop*: там находятся значки встроенных программ (IP Configuration, Command Prompt, Web Browser, Email, PC Wireless и другие). Нам нужны два из них.



Вкладка Desktop: 1 – IP Configuration, 2 – Command Prompt.

Откроется окно настройки. В версии 8.2.1 по умолчанию выбран вариант *Static*, поле *IPv4 Address* пусто, а поля *Default Gateway* и *DNS Server* показывают `0.0.0.0`. Значение `0.0.0.0` означает «не задано»; в этой работе его не трогаем. Когда вы введёте адрес и щёлкнете в поле *Subnet Mask*, программа сама подставит `255.255.255.0` (маска по умолчанию для адресов вида 192.168.x.x). Всё равно прочитайте её глазами: автоматическая подстановка не всегда совпадает с планом. Изменения применяются сразу, кнопки «Сохранить» нет. Окно закрывается кнопкой **X** на синей полосе *IP Configuration*.



IP Configuration: 1 – режим Static, 2 – адрес и маска.

В нижней половине того же окна находятся поля IPv6: переключатели называются *Automatic* и *Static*. В работе 31 мы будем использовать именно их.

23.4 Проверочное сообщение: что делает ping

Команда `ping` отправляет по заданному IP-адресу проверочные запросы и ждёт ответов. Их смысл похож на «ты меня слышишь?» и «да, слышу». Это отдельные служебные сообщения, а не отправка нашего документа. Обменом запросами и ответами занимается протокол **ICMP**; названия этих сообщений – *echo request* и *echo reply*.

Для ввода команды откройте на PC-A *Desktop / Command Prompt*. Command Prompt означает «Командная строка». В этом окне действие задаётся текстом, после чего нажимается Enter. Перед строкой ввода может стоять `C:\>` – приглашение программы; его переписывать не нужно. Команда состоит из слова ping, пробела и адреса *получателя*:

```
ping 192.168.10.20
```

Если PC-B отвечает, появляются строки, начинающиеся с `Reply from 192.168.10.20`. Если за время ожидания ответа нет, может появиться `Request timed out`. Первое означает «получен ответ», второе – «время ожидания истекло». Сообщение о неудаче показывает симптом; его причину нужно исследовать, а не угадывать по одной строке.

Команда `ipconfig` показывает текущие параметры этого ПК. Её полезно выполнить перед ping: так проверяется, что введённые в форму адрес и маска действительно применились. Команды вводят по одной; это две разные проверки – собственных настроек и ответа соседнего устройства.

Что доказывает ответ

Полученный ответ показывает, что проверочный запрос дошёл до указанного устройства и ответ вернулся. Он не доказывает, что на устройстве открыт сайт или работает передача файлов: для каждого приложения позже понадобится отдельная проверка. Зелёные индикаторы кабеля и ответы ping отвечают на разные вопросы.

Лабораторная 02. Первая локальная сеть и ping

Время: 60 минут. **Подготовка:** 01.

Какую проблему решает работа

Два компьютера стоят рядом и соединены кабелем, но обмениваться данными не могут. Это самая частая ситуация при подключении нового рабочего места. Причина почти всегда в адресах: каждый узел сети должен иметь свой уникальный адрес в общей сети, иначе сообщения некуда доставлять. Работа учит назначать адреса и проверять связь командой ping в обе стороны, то есть отвечать на вопрос «видят ли компьютеры друг друга».

Что вы научитесь делать

Назначить разные IPv4-адреса двум ПК, проверить собственные настройки и получить ответ на проверочное сообщение в обе стороны.

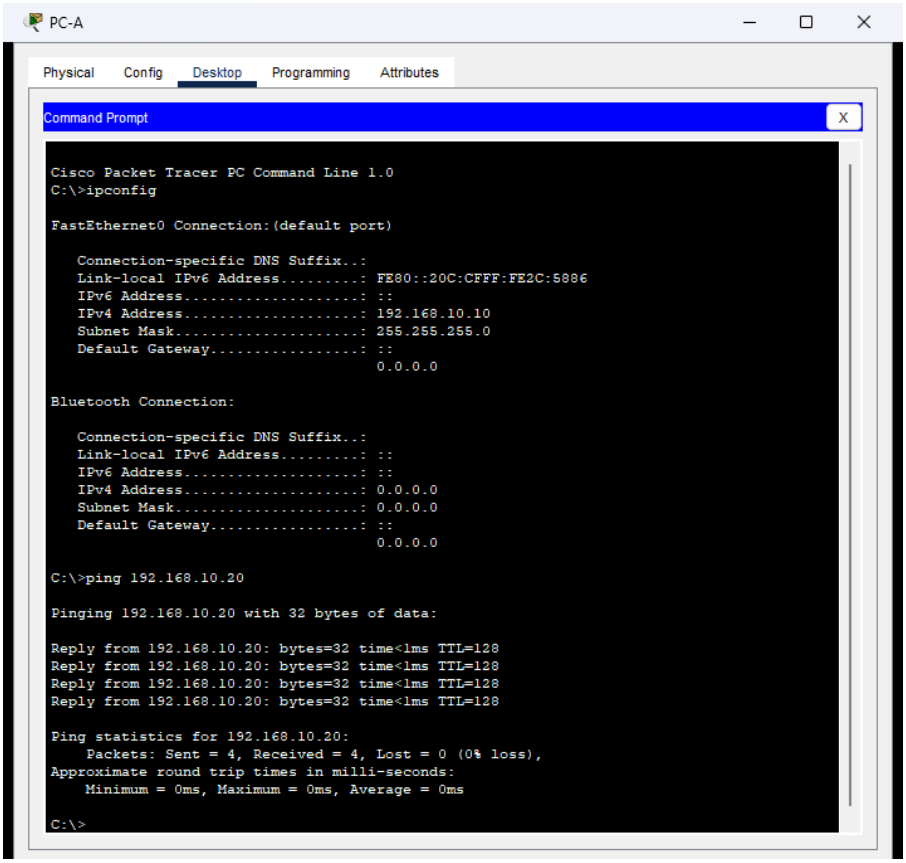
Стенд и адреса

Откройте исправную схему из работы 01 и через Save As сохраните её как `L02-working.pkt`. Так первая схема останется отдельным этапом обучения. Устройства и кабели не меняются. Новое условие – адреса в одной подсети:

Устройство	IPv4 Address	Subnet Mask	Gateway / DNS
PC-A	192.168.10.10	255.255.255.0	Не менять (0.0.0.0)
PC-B	192.168.10.20	255.255.255.0	Не менять (0.0.0.0)

Порядок работы

1. Щёлкните по PC-A, откройте вкладку *Desktop* и значок *IP Configuration*. Убедитесь, что выбран *Static*. Щёлкните в поле *IPv4 Address* и введите `192.168.10.10`. Щёлкните в поле *Subnet Mask*: появится `255.255.255.0`; проверьте, что маска именно такая. Поля *Default Gateway* и *DNS Server* оставьте как есть (`0.0.0.0`). Закройте окно настройки кнопкой **X** на синей полосе.
2. На PC-B повторите действие, используя его строку таблицы (`192.168.10.20`). Сравните адреса: последнее число отличается, маски одинаковы.
3. На PC-A откройте *Desktop / Command Prompt*. Щёлкните внутри чёрного окна, введите `ipconfig` и нажмите Enter. Найдите в первом разделе (*FastEthernet0 Connection*) строки *IPv4 Address* и *Subnet Mask*. Раздел *Bluetooth Connection* с адресами `0.0.0.0` относится к неиспользуемому модулю Bluetooth, его пропустите. Затем введите `ping 192.168.10.20` и прочитайте результат: есть ли строки *Reply from* с адресом PC-B?



Command Prompt на PC-A: ipconfig и ping. В конце видно «Received = 4, Lost = 0».

4. Если сразу после сборки схемы в строке стоит *Request timed out*, не меняйте настройки: коммутатор ещё готовит порты (оранжевые индикаторы), а ARP только выясняет адрес. Нажмите *Fast Forward Time* два-три раза или подождите около минуты и повторите ping. Если повторная серия постоянно не проходит, сверьте оба адреса, обе маски и соединения с таблицами работ 01 и 02.
5. На PC-B выполните `ipconfig`, затем `ping 192.168.10.10`. Здесь получатель – PC-A, поэтому адрес после слова ping меняется. Запишите количество ответов для каждого направления. Сохраните проект (Ctrl+S).

Успешная серия выглядит так (значения времени у вас могут отличаться):

```
C:\>ping 192.168.10.20

Pinging 192.168.10.20 with 32 bytes of data:

Reply from 192.168.10.20: bytes=32 time<1ms TTL=128
Reply from 192.168.10.20: bytes=32 time<1ms TTL=128
Reply from 192.168.10.20: bytes=32 time<1ms TTL=128
Reply from 192.168.10.20: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.10.20:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
```

Число TTL=128 – начальное значение для компьютеров PC-PT; у маршрутизаторов и коммутаторов Cisco оно равно 255, а каждый пройденный маршрутизатор уменьшает его на единицу (в работе 08 вы увидите TTL=127).

Проверка результата

PC-A получает ответы от 192.168.10.20, PC-B – от 192.168.10.10. Адреса различаются, маски совпадают, линии активны. Отсутствие шлюза не мешает этому обмену: согласно разобранному правилу оба адресата находятся в общей подсети.

Управляемая неисправность

Сохраните копию как `L02-fault.pkt`. Измените только адрес PC-B на `192.168.20.20`, оставив маску 255.255.255.0. Перед тестом сравните первые три числа адресов: теперь они различаются. На PC-A выполните `ping 192.168.20.20`, то есть обращайтесь к новому адресу PC-B. Ожидается отсутствие успешных ответов: требуется путь в другую IP-сеть, а маршрутизатора в стенде нет. Кабели при этом могут оставаться активными. Восстановите адрес PC-B и повторите исходную проверку. Добавление случайного числа в поле Gateway не создаёт маршрутизатор.

Что сдать и объяснить

Запишите адресную таблицу и результат проверки в двух направлениях. Объясните, зачем двум ПК разные адреса, что делает маска 255.255.255.0 в нашем примере и почему поле шлюза пустое. Для опыта с неисправностью укажите прогноз, наблюдение и единственное изменение, после которого обмен восстановился.

Самостоятельное усложнение. По знакомому образцу подключите PC-C к свободному порту S1. Назначьте адрес 192.168.10.30 и маску 255.255.255.0. Проверьте обмен между каждой парой: A–B, A–C, B–C, в обоих направлениях. Запишите результаты в таблицу с колонками «отправитель / получатель / есть ли ответ». Сохраните расширенную схему отдельно.

23.5 Что находится внутри обмена

Теперь у нас есть наблюдение: команда вызвала запрос, а затем появился ответ. Разберём, как устройства доставили эти сообщения. Эти объяснения нужны для следующей работы с пошаговым просмотром; в предыдущей работе было достаточно понимать адрес, маску и результат ping.

23.5.1 IP-пакет и Ethernet-кадр

IP-пакет – порция данных с информацией для доставки по IP, в том числе IP-адресами отправителя и получателя. Проверочный запрос ICMP передаётся внутри такого пакета. Чтобы отправить пакет по нашей проводной линии Ethernet, компьютер помещает его внутрь **кадра Ethernet**. У кадра есть свои адреса для обмена на этом участке – MAC-адреса. Получается вложение: кадр содержит IP-пакет, а тот содержит сообщение ICMP.

MAC-адрес обозначает интерфейс при обмене кадрами в локальной сети. Пример его записи – `00D0.BA12.3456`; ваша модель может получить другое значение. MAC и IP выполняют разные задачи. IP задаём согласно адресному плану, а MAC в первом опыте оставляем назначенным моделью. Сетевой порт с кабелем также не является ни MAC-, ни IP-адресом: это точка подключения.

23.5.2 ARP: поиск MAC по известному IP

PC-A уже знает IP получателя из команды ping. Но для кадра Ethernet ему нужен MAC этого получателя. **ARP**, Address Resolution Protocol, позволяет узнать

соответствие между IPv4-адресом и MAC в локальной сети.

Когда соответствие ещё неизвестно, PC-A отправляет запрос ARP: «У кого адрес 192.168.10.20? Сообщите свой MAC». Запрос адресован всем участникам текущей локальной сети; такую передачу называют **широковещательной**. В поле назначения Ethernet будет `FFFF.FFFF.FFFF`. Коммутатор передаёт такой кадр на остальные подходящие порты той же локальной сети.

PC-B узнаёт собственный IP и отправляет ответ ARP с указанием своего MAC. PC-A сохраняет соответствие в **ARP-таблице**, временной памяти известных соседей. Теперь проверочный запрос ICMP можно передать в кадре с конкретным MAC PC-B. Поэтому при повторном ping нового ARP-запроса может не быть: сведения ещё сохранены. Подготовка обмена также может объяснять потерю первого запроса; постоянная потеря требует проверки других причин.

23.5.3 Как коммутатор узнаёт, куда передавать кадры

При поступлении кадра коммутатор запоминает MAC отправителя и порт, из которого кадр пришёл. Эта информация собирается в **MAC-таблице**. Когда MAC получателя уже известен, кадр направляется в соответствующий порт. Если MAC назначения пока неизвестен, кадр распространяется через остальные подходящие порты той же локальной сети. Позже мы уточним её границы при изучении VLAN.

При локальном обмене S1 не заменяет IP отправителя и получателя своими адресами. Он передаёт кадр между портами. Именно поэтому важно различать адреса устройств внутри сообщения и физические соединения на рисунке.

Разобранный путь первого запроса

При пустой ARP-таблице ожидается такая последовательность: PC-A спрашивает MAC владельца 192.168.10.20; PC-B отвечает; PC-A отправляет запрос ICMP; PC-B возвращает ответ ICMP. Сообщения проходят через S1. В программе каждое сообщение может породить несколько строк событий – отправку, обработку коммутатором, приём. Поэтому четыре типа сообщений не обязательно означают ровно четыре строки в списке.

23.6 Как читать пошаговую симуляцию

Режим *Simulation* позволяет остановиться между событиями и посмотреть содержимое сообщения. *Event List* – список событий. *Edit Filters* – выбор протоколов, события которых будут видны. Фильтр скрывает лишние записи, но не запрещает соответствующие протоколы в самой сети.

Кнопка *Capture / Forward* продвигает просмотр на следующий шаг. Щелчок по значку сообщения или полю Info в строке события открывает сведения о нём. *Inbound PDU Details* показывает данные при приёме, *Outbound PDU Details* – при отправке. Сокращение **PDU** означает единицу данных протокола; в этом окне программа показывает поля сообщения на разных уровнях. Для нашего опыта ищите блоки Ethernet, IP и ICMP, а не пытайтесь сразу объяснить все поля [22].

Add Simple PDU – инструмент создания простого проверочного обмена: выбираются отправитель и получатель. Для этого занятия мы используем уже знакомую команду ping, чтобы легче связать текст команды с событиями. В режиме Simulation сообщение может ждать вашего нажатия Capture / Forward: это управление просмотром, а не обязательно неисправность сети.

Лабораторная 03. Пошаговое наблюдение ARP и ICMP

Время: 75 минут. **Подготовка:** 02.

Какую проблему решает работа

Команда ping отвечает только «да» или «нет», а администратору нужно знать, где именно рвётся путь и почему первый ответ приходит с задержкой или теряется. Работа решает проблему «чёрного ящика»: в режиме Simulation вы наблюдаете, как компьютер сначала выясняет физический адрес соседа (ARP), затем отправляет сообщение (ICMP), и учитесь объяснять увиденное, а не угадывать.

Что вы научитесь делать

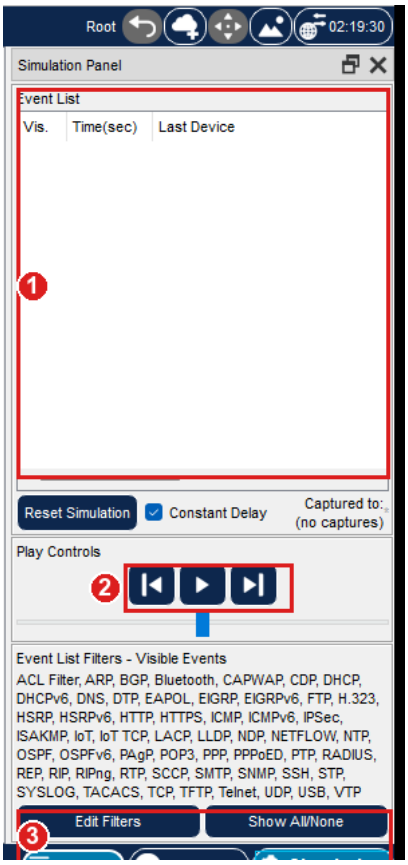
Отделять кадр от IP-пакета и связывать сетевой результат с событиями симуляции.

Стенд и адреса

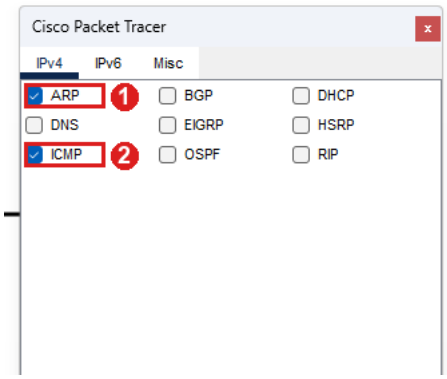
Используйте исправный стенд 02 и сохраните копию как `L03-working.pkt`. Если обмен уже выполнялся, в ARP-таблице PC-A может остаться запись для адреса PC-B. Чтобы вызвать новый поиск, в IP Configuration измените только адрес PC-B на свободный `192.168.10.21`, сохранив маску 255.255.255.0. Теперь в проверке используется именно .21. Этот приём достаточен для первого просмотра; сам сброс списка событий не гарантирует очистку ARP-таблиц.

Порядок работы

1. Нажмите кнопку *Simulation* (правый нижний угол). Справа откроется *Simulation Panel*: список событий (*Event List*), кнопки управления шагами (*Play Controls*) и фильтры. Нажмите *Show All/None*, чтобы убрать все протоколы из просмотра, затем *Edit Filters*: во вкладке *IPv4* поставьте галочки *ARP* и *ICMP* и закройте маленькое окно красным крестиком. Под списком фильтров должна появиться строка «ARP, ICMP».

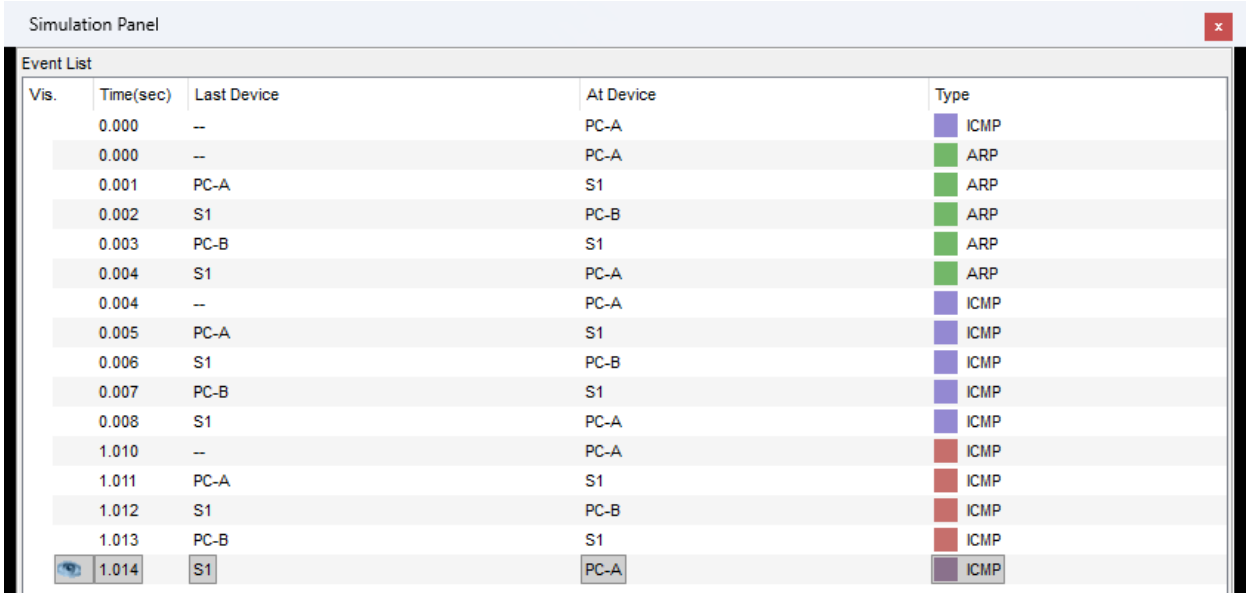


Simulation Panel: 1 – список событий, 2 – кнопки Play Controls (правая – Capture / Forward), 3 – фильтры.



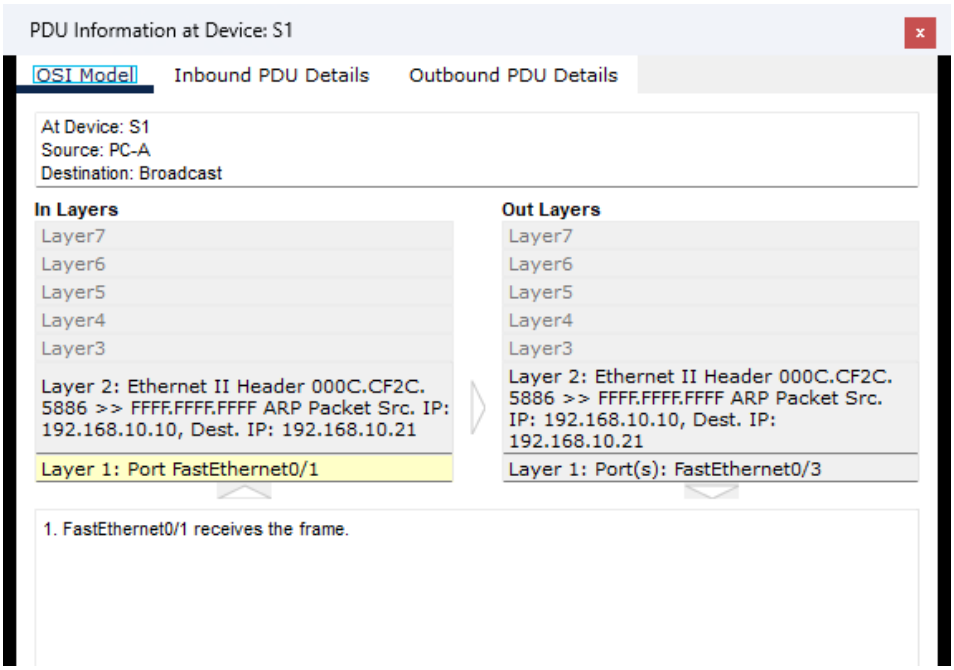
Окно Edit Filters: отмечены только ARP и ICMP.

2. На PC-A откройте *Desktop / Command Prompt*, щёлкните в чёрном окне и введите `ping 192.168.10.21`. В режиме Simulation команда не выполнится сразу: сообщения ждут ваших шагов. Закройте окно командной строки крестиком окна.
3. Нажимайте кнопку *Capture / Forward* (правая из трёх кнопок в *Play Controls*, значок со стрелкой и чертой) и следите за схемой: по линиям поедут конверты, а в *Event List* будут появляться строки. Панель узкая; чтобы увидеть все столбцы, нажмите значок «открепить» (два наложенных окна) в правом верхнем углу панели и растяните появившееся окно.

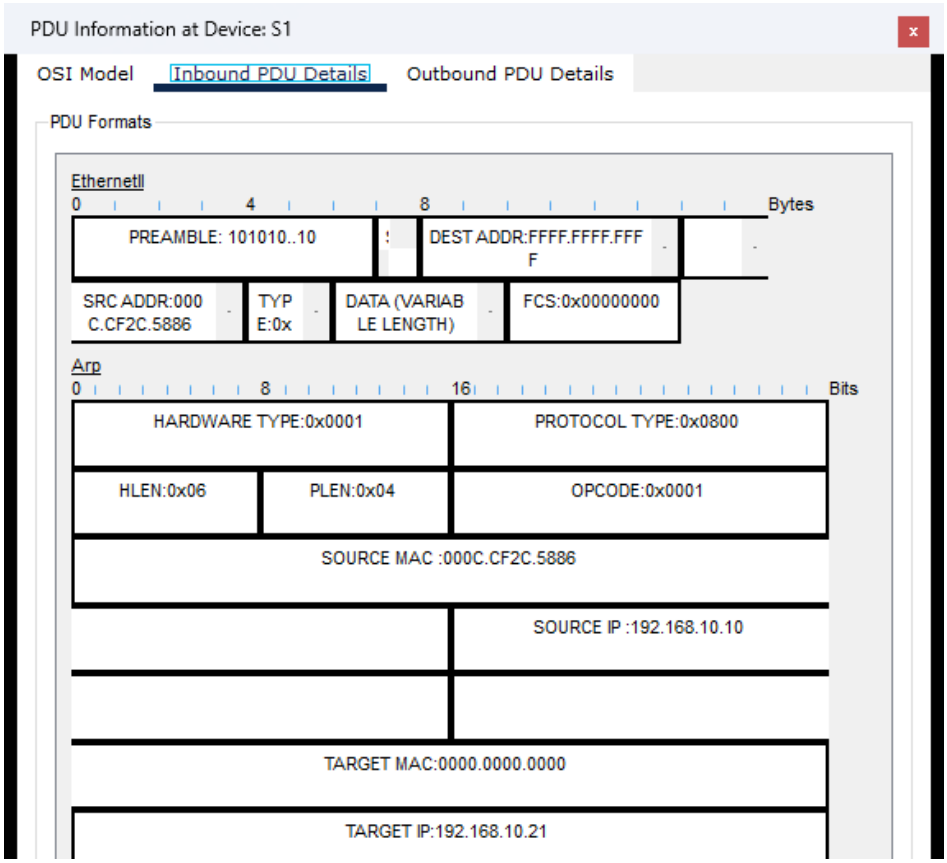


Список событий после полной серии: сначала ARP (зелёные строки), потом ICMP (фиолетовые – первый запрос и ответ, красноватые – второй запрос).

4. Найдите первый ARP-запрос. Щёлкните по цветному квадрату в столбце *Type* (или по значку конверта на схеме) – откроется окно *PDU Information*. На вкладке *OSI Model* в блоке *Layer 2* прочитайте заголовок Ethernet: адрес назначения `FFFF.FFFF.FFFF` – признак широковещательной передачи. На вкладке *Inbound PDU Details* те же поля показаны таблицей: *DEST ADDR*, *SRC ADDR*, *SOURCE IP*, *TARGET IP*; в запросе *TARGET MAC* равен `0000.0000.0000`, потому что искомый MAC неизвестен.

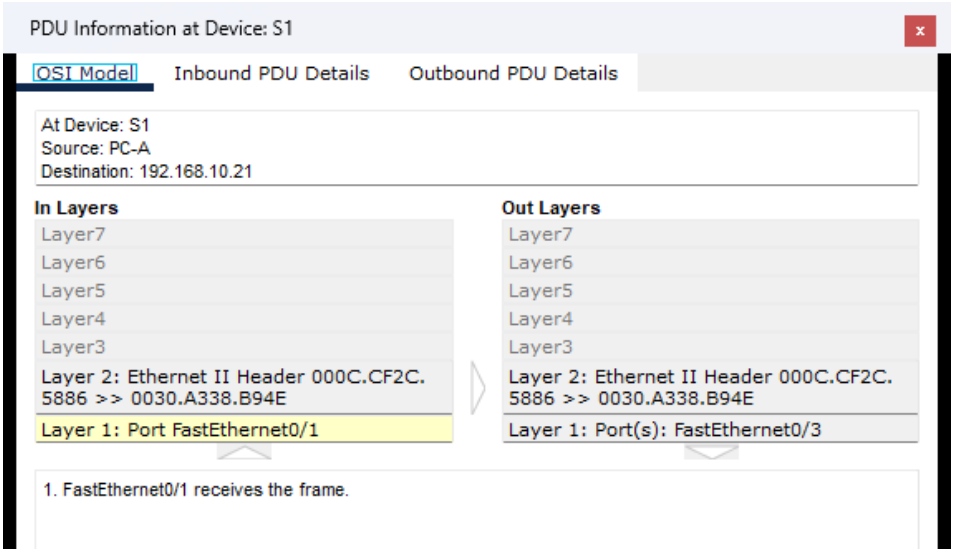


PDU Information, вкладка OSI Model: ARP-запрос на входе в S1, адрес назначения FFFF.FFFF.FFFF.



Вкладка Inbound PDU Details: поля Ethernet и ARP.

5. Продолжите шаги. Откройте событие ICMP на S1 (запрос от PC-A). У кадра теперь конкретный MAC назначения (на рисунке это `0030.A338.B94E`), а не FFFF.FFFF.FFFF; S1 видит только заголовок второго уровня и передаёт кадр в нужный порт. Ваши MAC-адреса будут другими: они назначаются при создании устройств.



ICMP-запрос на S1: MAC назначения уже известен.

6. Запишите путь PC-A – S1 – PC-B и обратный путь ответа. Затем нажмите Capture / Forward до конца серии, вернитесь в *Realtime* и повторите `ping 192.168.10.21` из командной строки. Новый ARP-запрос может отсутствовать:

соответствие уже хранится в ARP-таблице. Сравните два наблюдения и объясните различие.

Проверка результата

В первом обмене видны ARP-запрос, ARP-ответ и сообщения ICMP. У ARP-запроса широковещательный Ethernet-адрес назначения; у кадра с запросом ICMP после поиска – MAC конкретного ПК. S1 передаёт сообщение, сохраняя исходный и конечный IP. Если ARP не появился, проверьте, что новый адрес ещё не использовался, и повторите подготовку с другим свободным адресом этой подсети.

Управляемая неисправность

Сохраните копию проекта. Переключитесь в Realtime, удалите только кабель PC-B–S1 и подключите PC-B к свободному Fa0/3 на S1 тем же типом кабеля. Дождитесь активной линии и повторите ping. Это смена точки подключения, а не изменение IP или MAC компьютера. После обновления сведений коммутатора обмен должен восстановиться. Просмотр его таблицы командами будет изучен после введения в управление устройством.

Что сдать и объяснить

Приложите снимки четырёх типов сообщений и таблицу «тип сообщения / кто отправляет / кому / зачем». Объясните, почему ARP-запрос получает вся локальная сеть, почему ответ ICMP адресован конкретному ПК и почему во второй проверке ARP может отсутствовать. Не требуется объяснять остальные поля окна PDU.

Самостоятельное усложнение. Добавьте третий ПК. Покажите, какой кадр он получает при ARP, и почему получение широковещания не делает его адресатом ICMP.

Глава 24

IPv4: от маски до адресного плана

До сих пор мы использовали одну маску и сравнивали первые три числа адреса. Теперь нужно понять, откуда берётся эта маска и как выделить небольшую сеть внутри большего диапазона. Сначала разберём запись и один расчёт полностью; затем выполним аналогичные расчёты самостоятельно.

24.1 Бит, октет и двоичная запись

Бит – разряд, который принимает значение 0 или 1. Восемь бит образуют **октет**. IPv4-адрес содержит четыре октета, то есть $4 \times 8 = 32$ бита. В привычной записи каждый октет показывается одним десятичным числом. Это объясняет и четыре числа адреса, и предел 255.

В двоичной записи разряды октета имеют веса 128, 64, 32, 16, 8, 4, 2 и 1 слева направо. Чтобы получить число, складывают веса разрядов, в которых стоит единица. Например, `11000000` означает $128 + 64 = 192$, а `11111111` означает сумму всех весов, 255. Число 77 раскладывается как $64 + 8 + 4 + 1$ и записывается `01001101`.

Маска состоит из последовательности единиц, затем нулей. Единицы отмечают сетевую часть адреса, нули – часть для различения устройств, которую также называют узловой. В маске /24 первые 24 бита равны единице: три октета по восемь единиц и последний октет из нулей. Поэтому десятичная запись – 255.255.255.0. В /26 единиц на две больше: последний октет `11000000`, то есть 192. Получаем маску 255.255.255.192.

24.2 Префикс и диапазон

Адрес сети обозначает весь блок адресов; в нём узловые биты равны нулю. **Широковещательный адрес**, или *broadcast*, обозначает всех участников этой IP-подсети; в нём узловые биты равны единице. Эти два адреса обычным ПК в рассматриваемых подсетях не назначаются. Адреса между ними пригодны для интерфейсов устройств, включая интерфейс шлюза.

Если длина префикса равна p, для устройств остаётся $32 - p$ бит. Каждый бит имеет два варианта, поэтому всего в блоке 2^{32-p} адресов. После исключения адреса сети и broadcast остаётся $2^{32-p} - 2$ пригодных адреса. Например, у /26 остаётся шесть бит: $2^6 = 64$ адреса всего и 62 пригодных. Особые случаи /31 и /32 к первым расчётам не относятся.

Полностью разобранный пример: 192.168.10.77/26

1. По длине префикса находим маску: 255.255.255.192. Сетевая часть включает первые три октета и два старших бита последнего.
2. Размер блока – $2^{32-26} = 64$. В последнем октете блоки начинаются с 0, 64, 128 и 192. Число 77 лежит между 64 и 127, поэтому адрес сети – 192.168.10.64.
3. Последнее число блока равно $64 + 64 - 1 = 127$. Значит, broadcast – 192.168.10.127.
4. Между границами лежат адреса 192.168.10.65–192.168.10.126. Любые два разных адреса этого диапазона с маской /26 относятся к одной подсети.

Тот же результат можно получить по битам. Последний октет адреса – `01001101`, маски – `11000000`. Операция **AND**, логическое «И», оставляет единицу только там, где оба сравниваемых бита равны единице. Результат – `01000000`, то есть 64. Мы сохранили сетевые биты и обнулили узловые. Если вместо этого сделать все шесть узловых битов единицами, получится `01111111`, то есть 127 – последний адрес блока.

Для префиксов /24–/30 можно применять удобный способ расчёта по размеру блока в последнем октете, показанный выше. Следующая таблица помогает проверить маску и вместимость; она не заменяет понимания границ блока.

Префикс	Маска	Размер блока	Пригодных адресов
/24	255.255.255.0	256	254
/25	255.255.255.128	128	126
/26	255.255.255.192	64	62
/27	255.255.255.224	32	30
/28	255.255.255.240	16	14
/29	255.255.255.248	8	6
/30	255.255.255.252	4	2

24.3 Какие адреса использовать в учебной сети

Для внутренней IPv4-адресации доступны частные диапазоны 10.0.0.0/8, 172.16.0.0/12 и 192.168.0.0/16. Не весь диапазон 172.0.0.0/8 является частным. Внешние учебные сети далее используют 192.0.2.0/24, 198.51.100.0/24 и 203.0.113.0/24 – диапазоны для документации. «Провайдер» в книге является частью симуляции, а не выходом вашего ПК в настоящий Интернет.

24.4 VLSM и резерв роста

Адресный план – таблица, в которой заранее распределены сети и адреса устройств. **VLSM**, Variable Length Subnet Mask, означает использование масок разной длины для подсетей внутри общего диапазона. Это позволяет выделить большому отделу большой блок, а маленькому – меньший. **Пул** – исходный диапазон, из которого выделяются блоки. Сокращение **LAN** далее обозначает уже знакомую локальную сеть.

Допустим, отделу нужны адреса для 20 ПК и одного интерфейса шлюза: всего 21. У /28 только 14 пригодных адресов – мало; у /27 их 30 – достаточно. Если нужен запас ещё для десяти ПК, требование станет 31 и понадобится /26 с 62 пригодными адресами. Так префикс выбирается по потребности, а не по названию отдела.

Затем блоки размещают от больших к малым. В пуле 10.20.0.0/24 первым можно выделить 10.20.0.0/26: он занимает .0–.63. Следующий блок /27 начинается с .64 и заканчивается .95. Следующий свободный адрес – .96. Адрес сети должен лежать на границе блока: /26 нельзя начать с .70, поскольку его допустимые границы в этом октете – 0, 64, 128, 192. Проверка плана состоит из трёх вопросов: хватает ли адресов, не пересекаются ли блоки и помещаются ли они в пул.

В следующих заданиях первый пригодный адрес обычно отводится шлюзу. Это соглашение нашего плана, а не требование IP. Если шлюза в стенде пока нет, адрес можно зарезервировать и отметить это в таблице. Для дополнительного задания с /23: маска 255.255.254.0 объединяет два соседних блока /24, начало пары в третьем октете чётное; 172.16.8.0/23 занимает 172.16.8.0–172.16.9.255.

Лабораторная 04. Расчёт подсетей и проверка маски

Время: 75 минут. **Подготовка:** 02–03.

Какую проблему решает работа

Администратору выдают адрес вроде 192.168.10.37/26 и спрашивают: в какой сети находится узел, кто его соседи, какой адрес считается широковещательным и можно ли дать этот адрес принтеру. Ошибка в маске — одна из самых частых причин «странно работающей» сети: часть узлов видна, часть нет. Работа учит считать подсети вручную и проверять расчёт на схеме, чтобы не доверять догадке.

Что вы научитесь делать

Вычислять сеть, broadcast и диапазон узлов; проверять расчёт экспериментом.

Стенд и адреса

Два ПК через S1. Начальный план: PC-A 192.168.50.65/26, PC-B 192.168.50.126/26; шлюза нет.

Порядок работы

1. По четырём шагам разобранного примера рассчитайте маску, адрес сети, broadcast и пригодный диапазон для 192.168.50.65/26. Затем повторите расчёт для 192.168.50.130/27 и 10.0.0.6/30. Таблицу масок можно использовать как справочник.
2. Назначьте начальные адреса и подтвердите двусторонний ping. Определите, сколько ещё ПК можно разместить в этой /26, если маршрутизатора пока нет.
3. Измените PC-B на 192.168.50.129/26. Объясните ожидаемый результат до теста, затем проверьте его.
4. Верните .126. На обоих ПК измените маску на /27, то есть введите 255.255.255.224. Теперь .65 и .126 находятся в разных подсетях; сравните наблюдение с расчётом.
5. Восстановите исходные маски, сохраните исправный стенд.

Проверка результата

Для .65/26 сеть .64, broadcast .127, пригодные .65–.126. Для .130/27 сеть .128, broadcast .159. Для 10.0.0.6/30 сеть 10.0.0.4, broadcast .7, узлы .5 и .6. При смене маски решение о локальности может измениться без смены адреса.

Управляемая неисправность

Оставьте /26 на одном ПК и /27 на другом. Сравните решения каждого о локальности. Ошибка может давать асимметричные симптомы; одинаковый кабель не гарантирует одинакового IP-представления сети.

Что сдать и объяснить

Сдайте расчёты вручную и таблицу четырёх экспериментов: адреса, маски, прогноз, результат, объяснение.

Самостоятельное усложнение. Выберите два адреса из одной /28 внутри 192.168.50.0/24 и повторите проверку. Укажите адреса, которые использовать нельзя.

Лабораторная 05. VLSM для небольшой организации

Время: 90 минут. **Подготовка:** 04.

Какую проблему решает работа

Организации выдан один блок адресов, а отделов несколько, и размеры у них разные: бухгалтерии нужно десять адресов, офису — сто. Деление блока на равные части либо расходует адреса впустую, либо их не хватает, а исправлять адресный план после запуска сети очень дорого. Работа учит делить блок способом VLSM по реальным потребностям и оставлять резерв на рост.

Что вы научитесь делать

Составить непересекающийся план с подсетями разного размера и обосновать запас.

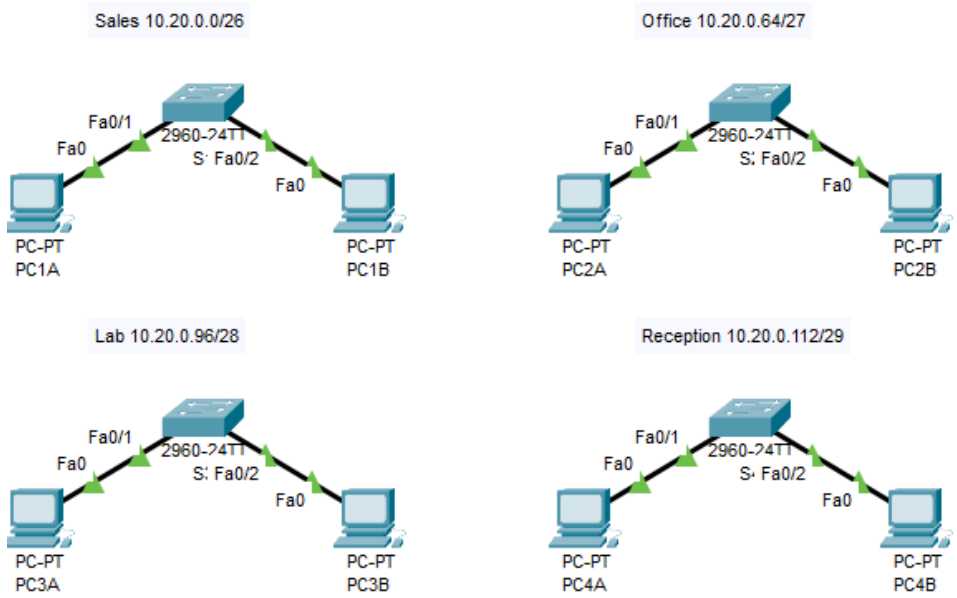
Стенд и адреса

Исходный пул 10.20.0.0/24. Требуется разместить 50, 25, 12 и 5 конечных устройств в четырёх LAN; каждой LAN нужен ещё один адрес шлюза. Два канала между маршрутизаторами требуют по два адреса. Физическую сборку можно выполнить позже, в работе 09.

Порядок работы

1. Посчитайте требование с учётом шлюза: 51, 26, 13 и 6 адресов. Выберите /26, /27, /28 и /29; для каналов – /30.
2. Разместите блоки от начала пула. Для каждой строки укажите сеть, маску, первый и последний адрес, broadcast и роль шлюза.
3. Сверьте расчёты с контрольным вариантом ниже, затем составьте второй корректный вариант, разместив блоки иначе на допустимых границах.
4. В Packet Tracer создайте четыре группы устройств. В каждой поставьте два представительных ПК, соедините их с отдельным коммутатором и назначьте адреса и маски по плану. Для подписи группы используйте инструмент Place Note («Разместить заметку»): выберите его, щёлкните возле группы и введите название отдела и подсеть. Не размещайте все 92 ПК: для проверки адресации достаточно представителей. Между разными группами обмен пока не проверяется – связывать сети маршрутизаторами будем в последующих работах.

Назначение	Сеть	Узлы	Broadcast
50 ПК + шлюз	10.20.0.0/26	.1–.62	10.20.0.63
25 ПК + шлюз	10.20.0.64/27	.65–.94	10.20.0.95
12 ПК + шлюз	10.20.0.96/28	.97–.110	10.20.0.111
5 ПК + шлюз	10.20.0.112/29	.113–.118	10.20.0.119
Канал 1	10.20.0.120/30	.121–.122	10.20.0.123
Канал 2	10.20.0.124/30	.125–.126	10.20.0.127



Возможный результат шага 4: четыре группы, у каждой коммутатор, два представительных ПК и заметка с названием отдела и подсети. Заметка создаётся инструментом Place Note (значок листка во второй панели инструментов): выберите инструмент, щёлкните по рабочей области и введите текст.

Проверка результата

Занято 128 адресов пула. Остаток 10.20.0.128/25 свободен целиком. Ни один диапазон не пересекается с другим. Подсеть на 12 ПК содержит 14 пригодных адресов: после шлюза остаётся один адрес запаса, поэтому существенный рост потребует более крупного блока.

Управляемая неисправность

Предположите, что отдел из 25 ПК вырос до 35. Объясните, почему /27 уже недостаточно и почему простое изменение маски может перекрыть соседнюю подсеть.

Что сдать и объяснить

Сдайте два адресных плана и подписи зон в проекте. Покажите расчёт одного блока в двоичном виде и предложите план расширения без пересечений.

Самостоятельное усложнение. Спроектируйте распределение внутри 172.16.8.0/23 для 120, 60, 28 и 10 ПК, четырёх шлюзов и трёх каналов /30. Контрольный ответ – в приложении с разборами.

Глава 25

CLI и базовая конфигурация IOS

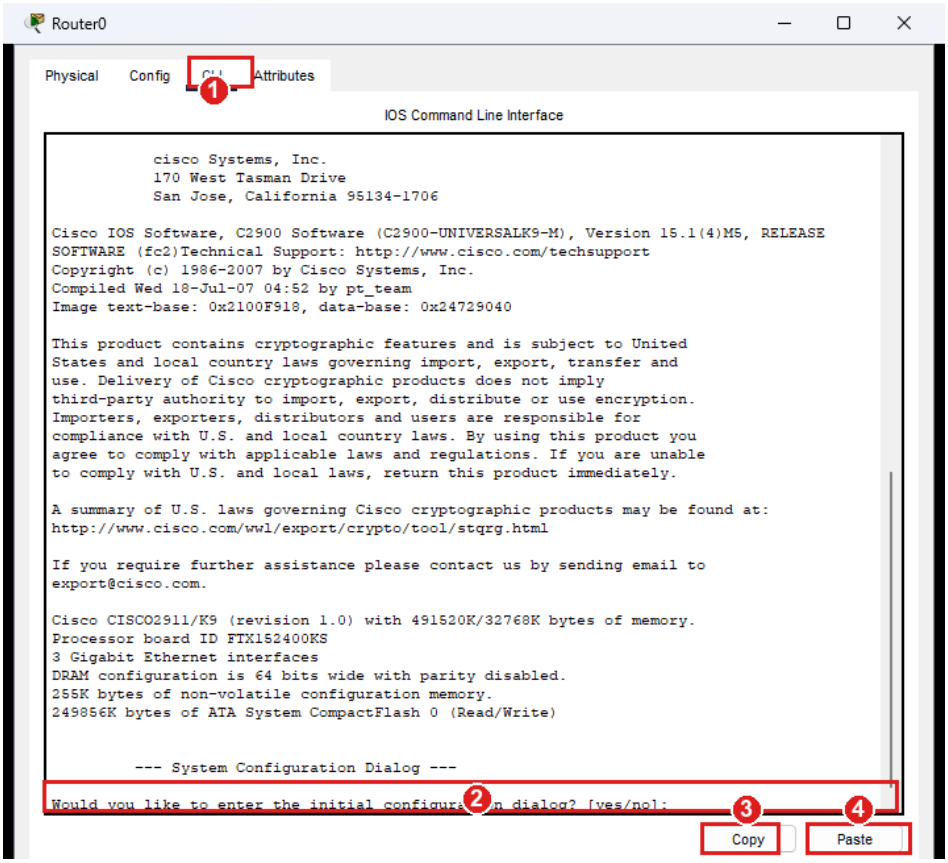
До этого мы меняли параметры компьютеров через формы. У коммутаторов и маршрутизаторов Cisco есть другой способ управления – **CLI**, Command Line Interface, интерфейс командной строки. Это окно управления конкретным устройством: введённая команда обращается к нему, а не к ПК, на котором запущен Packet Tracer. **IOS** – название программной системы Cisco, которая управляет устройством; Packet Tracer воспроизводит часть её поведения.

Во вкладке CLI устройства после нажатия Enter появляется приглашение – строка, показывающая имя устройства и режим работы. Режим определяет, какие команды разрешены. Чтобы сначала научиться читать приглашение, разберём небольшой пример, не меняющий сетевые адреса: открываем CLI коммутатора, видим `Switch>`, вводим `enable`, получаем `Switch#`, вводим `show mac address-table` и читаем сведения о MAC-адресах и портах, изученные в работе 03. Если обмена недавно не было, таблица может быть пустой. Слово `show` означает просмотр, а не изменение настройки. Само приглашение при вводе команды не копируется.

Теперь можно перейти к режимам изменения настроек. В примерах ниже каждая строка – одна команда; вводите её и нажимайте Enter, затем смотрите ответ устройства. Если предлагается первоначальный диалог настройки, ответ `no` позволяет перейти к ручному управлению, используемому в курсе.

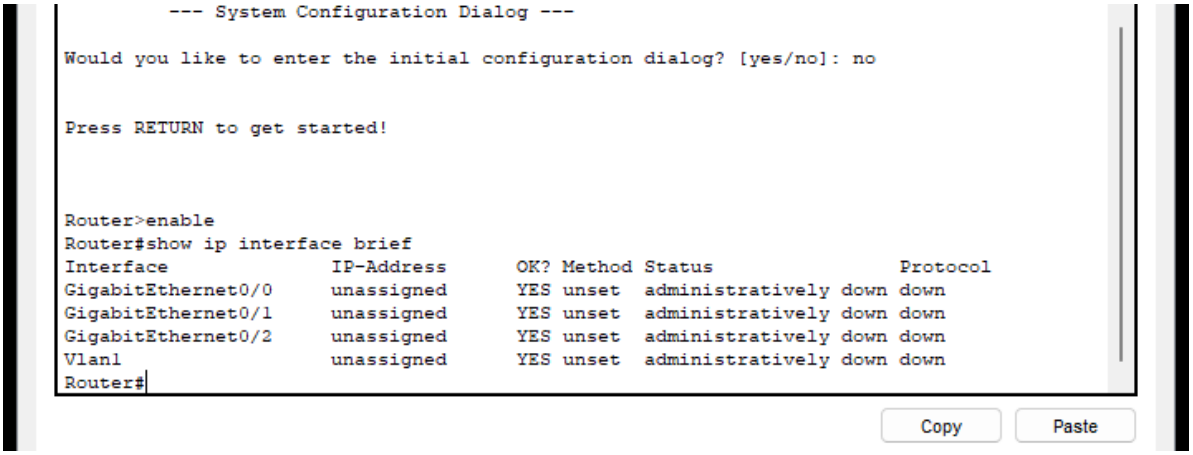
Как открыть CLI и ввести команды

1. Щёлкните по маршрутизатору или коммутатору на схеме. В окне устройства откройте вкладку *CLI* (у маршрутизатора и коммутатора она третья после Physical и Config). Окно *IOS Command Line Interface* – это консоль устройства.
2. Щёлкните внутри чёрно-белого текстового поля, чтобы дать ему фокус, и печатайте команды. Каждая команда заканчивается Enter. Клавиша Tab дополняет слово, знак ? показывает подсказку.
3. При первом открытии 2911 и 3560 (но не 2960) выводится вопрос `Would you like to enter the initial configuration dialog? [yes/no]:`. Введите `no`, Enter, дождитесь строки `Press RETURN to get started!` и нажмите Enter ещё раз. Появится приглашение `Router>`.



Окно CLI маршрутизатора 2911: 1 – вкладка CLI, 2 – вопрос о начальном диалоге, 3 и 4 – кнопки Copy и Paste.

Кнопка *Paste* под окном вставляет в консоль текст из буфера обмена. Пользоваться ею удобно для длинных блоков команд из этой книги: скопируйте блок, нажмите Paste и дождитесь, пока устройство обработает все строки. Следите за интерактивными вопросами: если в блоке есть команда, после которой устройство задаёт вопрос (например, `copy running-config startup-config` спрашивает `Destination filename [startup-config]?`), следующая строка блока будет воспринята как ответ. Поэтому такие команды вводите отдельно и отвечайте Enter. Длинные выводы разбиваются на страницы; строка `--More--` внизу означает, что для продолжения нужно нажать пробел.



После no, Enter и enable команда show ip interface brief показывает три интерфейса, выключенных административно.

Подпись на схеме и имя в командной строке – разные вещи

Команда `hostname R1` меняет приглашение (`R1>`), но подпись под значком на схеме остаётся прежней (например, *Router0*). Подпись меняется только в *Config / Settings / Display Name*. Чтобы схема совпадала с таблицами книги, в каждой работе задавайте оба имени: Display Name – в окне устройства, hostname – в CLI. На маршрутизаторах и коммутаторах эти поля находятся рядом в одном окне *Config / GLOBAL / Settings*.

25.1 Режимы и помощь

Приглашение `Router>` означает пользовательский EXEC. Команда `enable` переводит в привилегированный EXEC, где приглашение оканчивается `##`. Команда `configure terminal` открывает глобальную конфигурацию с `(config)##`; выбор интерфейса даёт `(config-if)##`. `exit` возвращает на уровень выше, `end` – в привилегированный EXEC. Команда `show` обычно выполняется из EXEC; в конфигурации иногда работает префикс `do`, но для единообразия в книге перед проверками используется `end`.

Знак вопроса показывает доступные команды или следующие аргументы. Tab дополняет однозначный ввод. Ошибка Invalid input и символ каретки часто означают неверный режим, синтаксис или неподдерживаемую функцию. Сокращения удобны после понимания полной команды; в конфигурационных примерах оставлены полные формы.

25.2 Состояние и сохранение

Running-config – действующая конфигурация; startup-config – конфигурация для следующего запуска. Команда `copy running-config startup-config` сохраняет изменения в устройстве. Копирование проекта `.pkt` сохраняет состояние всего учебного стенда. Не перезагружайте устройство только ради проверки каждой команды: сначала используйте вывод состояния.

Далее базовая модель маршрутизатора – 2911, а коммутатора – 2960; другие модели называются отдельно. У 2911 интерфейсы обычно обозначены GigabitEthernet0/0, 0/1 и 0/2. У другой модели имя может иметь дополнительное число, например GigabitEthernet0/0/0. Поэтому перед настройкой сверяйте названия через `show ip interface brief`, а на коммутаторе – через `show interfaces status`. Команда просмотра помогает найти нужное подключение, а не требует менять модель под буквальную подпись в примере.

Команда	Что проверить
`show running-config`	Текущие настройки; пароль может быть скрыт или представлен хешем.
`show startup-config`	Настройки, сохранённые для запуска.
`show ip interface brief`	Имя, адрес и краткое состояние L3-интерфейсов.
`show interfaces`	Подробности интерфейса, ошибки и параметры линии.
`show ip route`	Таблица маршрутов IPv4.
`show version`	Модель и сведения о моделируемой системе.

Лабораторная 06. Первичная настройка и сохранение

Время: 60–75 минут. **Подготовка:** 01, 03.

Какую проблему решает работа

Новое устройство Cisco приходит «пустым»: без имени, паролей и защиты, а после перезагрузки теряет всё, что не сохранено. Неподписанный и незащищённый коммутатор в шкафу — источник будущих аварий и взломов. Работа решает задачу первичного ввода в эксплуатацию: имя, защищённый доступ, предупреждающий баннер, сохранение конфигурации — то, что делается с каждым новым устройством.

Что вы научитесь делать

Освоить режимы CLI, назначить имя, ограничить доступ к консоли и сохранить конфигурацию.

Стенд и адреса

Новый 2911 с именем R1. Кабели для этой работы необязательны. Если при первом открытии CLI предлагается initial configuration dialog, ответьте `no`, затем Enter.

Порядок работы

Введите конфигурацию. Пароли ниже – учебные и публичные; в реальной сети их использовать нельзя. Измените их в собственном проекте и запишите в закрытом личном журнале, если работа требует повторного входа.

```
enable
configure terminal
hostname R1
no ip domain-lookup
enable secret LabEnable42
service password-encryption
banner motd #Authorized lab access only#
line console 0
 password LabConsole42
 login
 logging synchronous
 exec-timeout 10 0
exit
end
copy running-config startup-config
show running-config
show startup-config
```

При запросе имени файла назначения при копировании нажмите Enter. `no ip domain-lookup` предотвращает попытки трактовать ошибочно введённое слово как имя узла. `service password-encryption` скрывает ряд открытых паролей слабым обратимым механизмом и не делает их криптографически надёжными. `enable secret` предпочтительнее `enable password`. После `hostname R1` задайте также *Display Name* `R1` в окне устройства (*Config / Settings*): hostname меняет приглашение в консоли, а подпись на схеме остаётся прежней (Router0).

Проверка результата

Приглашение содержит R1; startup-config содержит hostname и настройки консоли. Введите `disable`, затем `enable`: устройство запрашивает secret. При но-

вой консольной сессии проверяется пароль линии. Не закрывайте CLI до сохранения и фиксации контрольных выводов. **Эталонный вывод** (Packet Tracer 8.2.1; хэши паролей и размер конфигурации у вас будут другими). Строки `...` пропущены:

```
R1#copy running-config startup-config
Destination filename [startup-config]?
Building configuration...
[OK]
R1#show running-config
...
service password-encryption
!
hostname R1
!
enable secret 5 $1$mERr$GKnnS.PHHBQ624lJgYYhJ.
...
no ip domain-lookup
...
banner motd ^CAuthorized lab access only^C
...
line con 0
 password 7 080D4D4C2A160B041D07095078
 logging synchronous
 login
```

Обратите внимание: пароль консоли показан как `password 7 ...` (результат service password-encryption), а enable secret – как `secret 5 ...` (хэш MD5). Строки `exec-timeout 10 0` в выводе нет: 10 минут – значение по умолчанию, и оно не выводится. Фильтр `| include hostname` в Packet Tracer 8.2.1 работает для `show running-config`, но для `show startup-config` выдаёт «Invalid input»: сравнивайте startup-config, просматривая его целиком.

Управляемая неисправность

Измените hostname на R1-TEMP и не сохраняйте. Сравните running-config и startup-config. Восстановите имя либо сохраните осознанно; объясните, какое имя загрузилось бы после перезапуска.

Что сдать и объяснить

Покажите цепочку режимов CLI, результаты проверок и отличие двух конфигураций. Почему шифрование пароля в отображаемой конфигурации не равно защите сетевой сессии?

Самостоятельное усложнение. Добавьте описания интерфейсов командой `description`, пока не назначая адреса. Используйте роли вроде `TO-LAN-A` и `TO-R2`.

Лабораторная 07. Управление коммутатором через SSH

Время: 90 минут. **Подготовка:** 02, 06.

Какую проблему решает работа

Администратор не может идти с кабелем к каждому коммутатору в шкафу: управлять устройствами нужно удалённо. Но протокол Telnet передаёт пароль открытым текстом, и любой, кто «слушает» сеть, его увидит. Работа решает проблему безопасного удалённого управления: вы настраиваете SSH, создаёте пользователя, ограничиваете доступ и проверяете вход с компьютера.

Что вы научитесь делать

Назначить адресу управления SVI и открыть аутентифицированную SSH-сессию с ПК.

Стенд и адреса

Новый S1 2960 и PC-A на Fa0/1. VLAN 99: S1 192.168.99.2/24, PC-A 192.168.99.10/24. Для локального теста шлюз не нужен. VLAN 99 здесь содержит и тестовый ПК администратора, и SVI управления.

Порядок работы

```
enable
configure terminal
hostname S1
enable secret LabEnable42
vlan 99
 name MANAGEMENT
exit
interface fastEthernet 0/1
 switchport mode access
 switchport access vlan 99
exit
interface vlan 99
 ip address 192.168.99.2 255.255.255.0
 no shutdown
exit
ip domain-name lab.test
username admin secret LabSsh42
crypto key generate rsa
```

При запросе длины ключа задайте 2048, если модель принимает это значение. Если выбранная модель ограничивает длину, задокументируйте её ограничение; не переносите это ограничение в рекомендации для реальных сетей. Далее:

```
ip ssh version 2
line vty 0 4
 login local
 transport input ssh
 exec-timeout 10 0
exit
end
copy running-config startup-config
show ip interface brief
show ip ssh
```

Если модель содержит VTY 5–15, примените к ним те же параметры, используя `line vty 5 15`. На PC-A задайте адрес и выполните:

```
ping 192.168.99.2
ssh -l admin 192.168.99.2
```

Проверка результата

Vlan99 находится up/up, ping проходит, SSH запрашивает пароль локального пользователя. Для подъёма SVI требуется созданная VLAN и активный порт этой VLAN в состоянии, допускающем передачу; `no shutdown` сам по себе недостаточен. IP на SVI не нужен для обычной коммутации пользовательских кадров. **Эталонный вывод.** Генерация ключа и проверка (Packet Tracer 8.2.1 принимает 2048 бит):

```
S1(config)#crypto key generate rsa
The name for the keys will be: S1.lab.test
Choose the size of the key modulus in the range of 360 to 4096 for your
  General Purpose Keys. Choosing a key modulus greater than 512 may take
  a few minutes.
How many bits in the modulus [512]: 2048
% Generating 2048 bit RSA keys, keys will be non-exportable...[OK]
S1(config)#ip ssh version 2
S1#show ip interface brief
...
Vlan99                 192.168.99.2    YES manual up                    up
S1#show ip ssh
SSH Enabled - version 2.0
```

С PC-A (Desktop / Command Prompt):

```
C:\>ssh -l admin 192.168.99.2
Password:
S1>enable
Password:
S1#show vlan brief
...
99   MANAGEMENT                       active    Fa0/1
```

Пароль при вводе на экране не отображается. Первый `ping` к новому адресу обычно теряет один запрос (ARP), остальные проходят.

Управляемая неисправность

Переведите Fa0/1 в VLAN 1 и наблюдайте потерю управления. Верните VLAN 99. Докажите, что причина находится на L2, а не в пароле SSH.

Что сдать и объяснить

Сдайте настройки SVI, VTY и снимок успешной SSH-сессии. Объясните различие MAC-таблицы и IP-адреса управления. Для управления из другой подсети L2-коммутатору потребуется `ip default-gateway` на достижимый адрес маршрутизатора этой VLAN.

Самостоятельное усложнение. Подключите второй административный ПК через Fa0/2 в VLAN 99. Повторите SSH и сравните поведение с Telnet, который на настроенных VTY не разрешён.

Лабораторная 08. Маршрутизатор между двумя LAN

Время: 90 минут. **Подготовка:** 04, 06.

Какую проблему решает работа

Две локальные сети — например, офис и склад — должны обмениваться данными, но компьютеры из разных сетей напрямую друг друга не видят: между ними нужен посредник. Без маршрутизатора связь невозможна, а с неправильно настроенным шлюзом она работает «в одну сторону». Работа решает проблему межсетевой связи: адреса интерфейсов, шлюз по умолчанию, проверка в обе стороны.

Что вы научитесь делать

Настроить шлюзы двух подсетей и объяснить изменение канальных адресов при маршрутизации.

Стенд и адреса

R1 2911, два 2960 и два ПК. PC-A–S1–R1 G0/0: 192.168.10.0/24; R1 G0/1–S2–PC-B: 192.168.20.0/24. На ПК используйте .10 и шлюз .1 своей подсети.

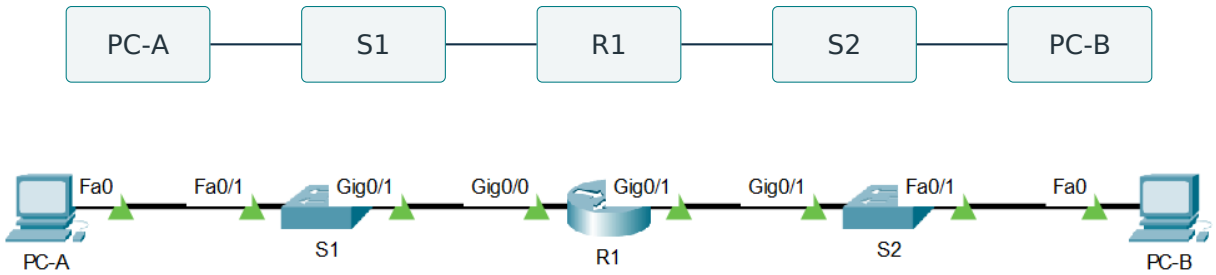


Схема работы 08: PC-A – S1 – R1 – S2 – PC-B.

Таблица соединений (проверена построением схемы в Packet Tracer 8.2.1). Порты записаны полными именами, как в списке портов программы.

Устройство	Порт	Устройство	Порт	Кабель
PC-A	FastEthernet0	S1	FastEthernet0/1	прямой
S1	GigabitEthernet0/1	R1	GigabitEthernet0/0	прямой
R1	GigabitEthernet0/1	S2	GigabitEthernet0/1	прямой
S2	FastEthernet0/1	PC-B	FastEthernet0	прямой

Порядок работы

Соедините маршрутизатор с коммутаторами прямыми медными кабелями; на S1 и S2 используйте G0/1 для uplink и Fa0/1 для ПК. Настройте R1:

```
enable
configure terminal
hostname R1
interface gigabitEthernet 0/0
 description LAN-A
 ip address 192.168.10.1 255.255.255.0
 no shutdown
exit
interface gigabitEthernet 0/1
 description LAN-B
 ip address 192.168.20.1 255.255.255.0
```

```
 no shutdown
exit
end
show ip interface brief
show ip route
copy running-config startup-config
```

С ПК сначала проверьте собственный шлюз, затем удалённый ПК. В Simulation найдите MAC назначения кадра на PC-A и новый кадр на выходе R1 к S2.

Проверка результата

Оба интерфейса up/up. В таблице маршрутов присутствуют непосредственно подключённые сети C; модель также может показывать локальные /32 с кодом L. Дополнительные статические маршруты между двумя непосредственно подключёнными LAN не нужны. При обычной передаче IP-адреса ПК сохраняются, а MAC-адреса меняются между сегментами. **Эталонный вывод.**

```
R1#show ip route
...
     192.168.10.0/24 is variably subnetted, 2 subnets, 2 masks
C       192.168.10.0/24 is directly connected, GigabitEthernet0/0
L       192.168.10.1/32 is directly connected, GigabitEthernet0/0
     192.168.20.0/24 is variably subnetted, 2 subnets, 2 masks
C       192.168.20.0/24 is directly connected, GigabitEthernet0/1
L       192.168.20.1/32 is directly connected, GigabitEthernet0/1

C:\>tracert 192.168.20.10
Tracing route to 192.168.20.10 over a maximum of 30 hops:
  1   0 ms      0 ms      0 ms      192.168.10.1
  2   0 ms      0 ms      0 ms      192.168.20.10
Trace complete.
```

Ответы от PC-B приходят с TTL=127: 128 минус один маршрутизатор. Последняя команда листинга `copy running-config startup-config` спрашивает имя файла (`Destination filename [startup-config]?`) – нажмите Enter.

Управляемая неисправность

Удалите только шлюз на PC-B. Проверьте ping PC-A–PC-B и ping PC-B–R1 G0/1. Локальный шлюз достижим, а ответ удалённому узлу не имеет корректного пути. Восстановите .1.

Что сдать и объяснить

Покажите таблицу адресов и два канальных заголовка одного маршрутизируемого обмена. Почему на PC-A MAC назначения принадлежит R1, а IP назначения – PC-B?

Самостоятельное усложнение. Подключите третью LAN 192.168.30.0/24 через R1 G0/2. Настройте .1 на маршрутизаторе и .10 на ПК. Проверьте все три LAN.

Часть II

Маршруты и коммутация

Глава 26

Статическая маршрутизация

26.1 Таблица маршрутов и обратный путь

Маршрутизатор выбирает наиболее длинный подходящий префикс назначения. Маршрут /24 специфичнее /16 и маршрута по умолчанию /0. Для кандидатов к одному и тому же префиксу учитывается административная дистанция источника, а внутри протокола – его метрика. Не сравнивайте расстояние до соседнего маршрутизатора с длиной префикса: это разные этапы решения.

Статический маршрут содержит сеть назначения, маску и следующий узел. Следующий узел должен быть достижим; для Ethernet в учебных примерах используем его IP-адрес. Маршрут по умолчанию применяется, если более специфичного совпадения нет. Ping требует маршрута туда и маршрута ответа. Если R1 знает сеть R3, это ещё не означает, что R3 знает сеть R1.

Лабораторная 09. Три маршрутизатора и статические маршруты

Время: 120 минут. **Подготовка:** 05, 08.

Какую проблему решает работа

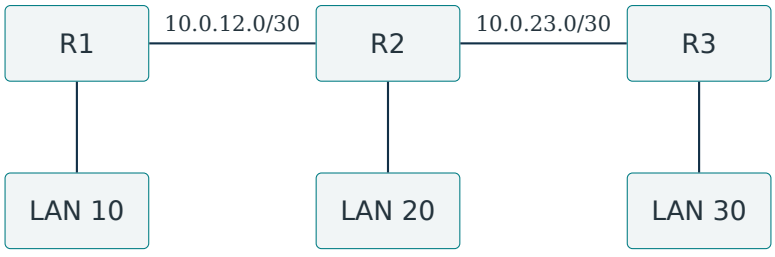
Когда сетей много, маршрутизатор знает только те, что подключены к нему напрямую, а о дальних ничего не знает и отбрасывает пакеты. Типичная жалоба: «туда пакет доходит, а обратно нет». Работа учит вручную сообщать маршрутизаторам о дальних сетях (статические маршруты) и находить ошибки в обратном пути.

Что вы научитесь делать

Собрать цепочку маршрутизаторов, настроить пути в обе стороны и прочитать таблицу маршрутов.

Стенд и адреса

Три 2911, три 2960, три PC. Каждый ПК подключён через свой коммутатор к G0/0 соответствующего маршрутизатора. Межмаршрутизаторные Ethernet-линии соединяйте crossover или автоматическим выбором кабеля.



Устройство	Порт	Адрес/префикс	Роль
R1	G0/0	192.168.10.1/24	LAN 10

R1	G0/1	10.0.12.1/30	К R2 G0/1
R2	G0/0	192.168.20.1/24	LAN 20
R2	G0/1	10.0.12.2/30	К R1
R2	G0/2	10.0.23.1/30	К R3 G0/1
R3	G0/0	192.168.30.1/24	LAN 30
R3	G0/1	10.0.23.2/30	К R2
PC-A/B/C	Fa0	.10 в своей LAN	Шлюз .1

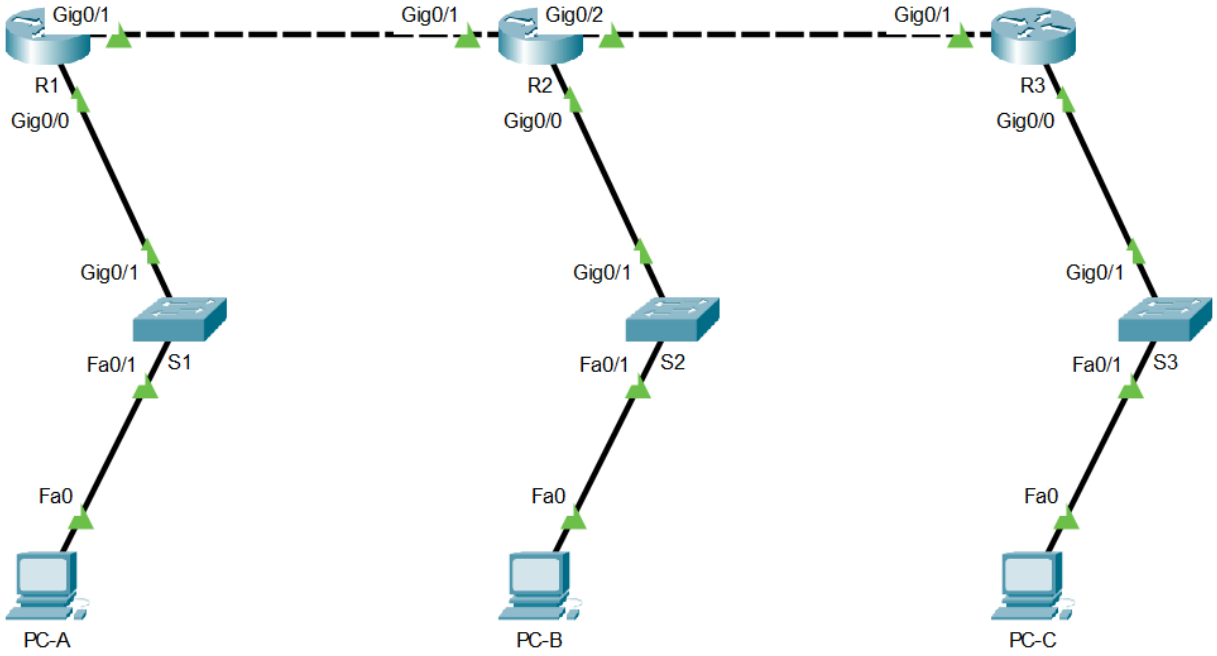


Схема работы 09. Линии между маршрутизаторами пунктирные: так Packet Tracer рисует кабель crossover.

Таблица соединений (проверена построением схемы в Packet Tracer 8.2.1). Порты записаны полными именами, как в списке портов программы.

Устройство	Порт	Устройство	Порт	Кабель
PC-A	`FastEthernet0`	S1	`FastEthernet0/1`	прямой
S1	`GigabitEthernet0/1`	R1	`GigabitEthernet0/0`	прямой
PC-B	`FastEthernet0`	S2	`FastEthernet0/1`	прямой
S2	`GigabitEthernet0/1`	R2	`GigabitEthernet0/0`	прямой
PC-C	`FastEthernet0`	S3	`FastEthernet0/1`	прямой
S3	`GigabitEthernet0/1`	R3	`GigabitEthernet0/0`	прямой
R1	`GigabitEthernet0/1`	R2	`GigabitEthernet0/1`	перекрёстный (crossover)
R2	`GigabitEthernet0/2`	R3	`GigabitEthernet0/1`	перекрёстный (crossover)

Порядок работы

1. Назначьте все адреса и выполните `no shutdown`, используя шаблон работы 08. Проверьте каждый соседний IP и шлюзы ПК до добавления маршрутов.
2. На R1 в глобальном режиме добавьте маршруты ниже, затем `end`. Аналогично настройте R2 и R3. Каждую группу вводите только на указанном устройстве.

R1:

```
ip route 192.168.20.0 255.255.255.0 10.0.12.2
ip route 192.168.30.0 255.255.255.0 10.0.12.2
```

R2:

```
ip route 192.168.10.0 255.255.255.0 10.0.12.1
ip route 192.168.30.0 255.255.255.0 10.0.23.2
```

R3:

```
ip route 192.168.10.0 255.255.255.0 10.0.23.1
ip route 192.168.20.0 255.255.255.0 10.0.23.1
```

На всех маршрутизаторах проверьте `show ip route`. Между ПК проверьте все пары в обе стороны. На ПК для просмотра пути используйте `tracert 192.168.30.10`, на IOS – `traceroute`. Время ответов симулятора не является измерением производительности реальной сети.

Проверка результата

Удалённые LAN представлены маршрутами S, непосредственно подключённые – C. Все ПК обмениваются ICMP. Адреса транзитных интерфейсов из удалённых LAN могут быть недостижимы без дополнительных маршрутов к транзитным подсетям; основной критерий этой работы – связность LAN. Если нужно управлять каждым транзитным IP, добавьте явно маршруты к этим подсетям и опишите их отдельно. **Эталонный вывод.** На R2 (две подключённые транзитные сети, две статические):

```
R2#show ip route
C       10.0.12.0/30 is directly connected, GigabitEthernet0/1
C       10.0.23.0/30 is directly connected, GigabitEthernet0/2
S    192.168.10.0/24 [1/0] via 10.0.12.1
C       192.168.20.0/24 is directly connected, GigabitEthernet0/0
S    192.168.30.0/24 [1/0] via 10.0.23.2

C:\>tracert 192.168.30.10
  1   0 ms      0 ms      0 ms      192.168.10.1
  2   0 ms      0 ms      0 ms      10.0.12.2
  3   0 ms      0 ms      1 ms      10.0.23.2
  4   1 ms      0 ms      0 ms      192.168.30.10
Trace complete.
```

(L-строки с адресами /32 пропущены.) Сразу после настройки первая серия ping между удалёнными ПК может целиком состоять из `Request timed out`: на пути три маршрутизатора, и на каждом нужно выяснить MAC следующего узла. Повторите команду через несколько секунд.

Управляемая неисправность

На R3 удалите маршрут к LAN 10 командой `no ip route 192.168.10.0 255.255.255.0 10.0.23.1`. Объясните, где теряется ответ PC-C на запрос PC-A. Верните маршрут.

Что сдать и объяснить

Сдайте адресную таблицу, матрицу связности и таблицы маршрутов. Нарисуйте прямой и обратный пути PC-A–PC-C; отличайте адрес следующего узла от сети назначения.

Самостоятельное усложнение. На R1 и R3 замените два маршрута к удалённым LAN одним default route к R2. На R2 оставьте специфичные маршруты. Объясните, почему default route на обоих краях не должен создавать петлю для несуществующего адреса при отсутствии default route на R2.

Лабораторная 10. Резервный путь и floating static route

Время: 90 минут. **Подготовка:** 09.

Какую проблему решает работа

Единственная линия связи между офисами — единственная точка отказа: обрыв кабеля, и филиал отрезан от центра. Работа решает проблему отказоустойчивости: резервный маршрут с большей административной дистанцией лежит «в запасе» и включается сам, когда основной пропадает, а при восстановлении трафик возвращается.

Что вы научитесь делать

Проверить переключение на резервный статический маршрут при падении непосредственно подключённого основного канала.

Стенд и адреса

Копия 09. Добавьте R1 G0/2–R3 G0/2: 10.0.13.0/30, R1 .1, R3 .2. Основной путь между LAN 10 и LAN 30 остаётся через R2.

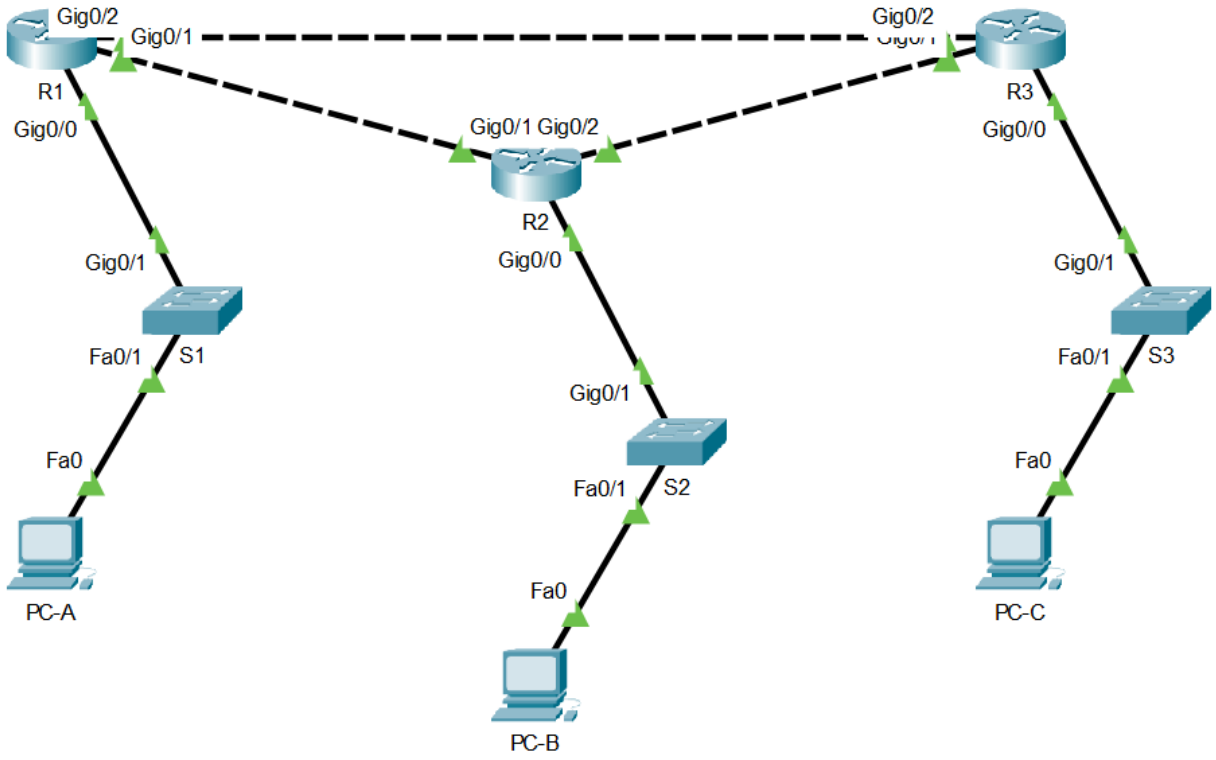


Схема работы 10: к цепочке работы 09 добавлена прямая линия R1 Gig0/2 – R3 Gig0/2.

Таблица соединений (проверена построением схемы в Packet Tracer 8.2.1). Порты записаны полными именами, как в списке портов программы.

Устройство	Порт	Устройство	Порт	Кабель
PC-A	FastEthernet0	S1	FastEthernet0/1	прямой
S1	GigabitEthernet0/1	R1	GigabitEthernet0/0	прямой
PC-B	FastEthernet0	S2	FastEthernet0/1	прямой
S2	GigabitEthernet0/1	R2	GigabitEthernet0/0	прямой
PC-C	FastEthernet0	S3	FastEthernet0/1	прямой
S3	GigabitEthernet0/1	R3	GigabitEthernet0/0	прямой
R1	GigabitEthernet0/1	R2	GigabitEthernet0/1	перекрёстный (crossover)
R2	GigabitEthernet0/2	R3	GigabitEthernet0/1	перекрёстный (crossover)
R1	GigabitEthernet0/2	R3	GigabitEthernet0/2	перекрёстный (crossover)

Порядок работы

Настройте и включите новые интерфейсы. Сохраните исходные статические маршруты и добавьте резервные:

```
! R1, global configuration
ip route 192.168.30.0 255.255.255.0 10.0.13.2 10
! R3, global configuration
ip route 192.168.10.0 255.255.255.0 10.0.13.1 10
```

Число 10 – административная дистанция резервного маршрута; основной статический маршрут имеет дистанцию 1. Строки, начинающиеся с !, здесь только подписывают устройство. Не вводите группы R1 и R3 на одном маршрутизаторе.

1. Пока все линии включены, выполните ping и tracert между PC-A и PC-C. Зафиксируйте путь через R2.
2. На R1 выключите G0/1. На R1 исчезает непосредственная достижимость 10.0.12.2; R2 также видит падение этого канала. Проверьте таблицы R1 и R3 и новый путь.
3. Для ответа R3 может продолжать выбирать основной путь через R2, который теперь не доставляет в LAN 10. В этой базовой конфигурации без отслеживания удалённого отказа резерв на R3 автоматически не обязан включиться. Выключите R3 G0/1, чтобы обе крайние стороны потеряли свои основные next hop, и повторите тест.
4. Верните оба интерфейса через `no shutdown`. Проверьте возврат основных маршрутов и объясните отличие одиночного и симметричного экспериментов.

Проверка результата

При обоих выключенных основных линках обмен LAN 10–LAN 30 проходит напрямую по 10.0.13.0/30. Активный S-маршрут имеет дистанцию 10. При восстановлении обеих линий снова выбирается путь с дистанцией 1. **Эталонный вывод.** После `shutdown` на R1 G0/1 резервный маршрут становится действующим, а на R3 основной остаётся прежним:

```
R1#show ip route
C       10.0.13.0/30 is directly connected, GigabitEthernet0/2
C       192.168.10.0/24 is directly connected, GigabitEthernet0/0
S    192.168.30.0/24 [10/0] via 10.0.13.2

R3#show ip route
S    192.168.10.0/24 [1/0] via 10.0.23.1
```

```
S    192.168.20.0/24 [1/0] via 10.0.23.1
C       192.168.30.0/24 is directly connected, GigabitEthernet0/0
```

В таком состоянии ping PC-A – PC-C завершается тайм-аутами: запрос идёт по резервной линии, а ответ R3 отправляет через R2. После `shutdown` ещё и на R3 G0/1 на R3 появляется `S 192.168.10.0/24 [10/0] via 10.0.13.1`, и ping проходит с TTL=126 (два маршрутизатора по пути).

Управляемая неисправность

Выключите только R2 G0/2, оставив R1–R2 активным. R1 всё ещё может считать свой next hop достижимым и не выбрать резерв. Это демонстрирует границу floating static route: он сам не следит за всем удалённым путём. В работе 28 вы исследуете динамическую сходимость.

Что сдать и объяснить

Три снимка таблиц маршрутов: всё исправно, один удалённый отказ, обе основные стороны отключены. Объясните, почему наличие резервной строки в конфигурации не означает обнаружение любого отказа.

Самостоятельное усложнение. Предложите, какие наблюдения понадобились бы механизму отслеживания доступности, чтобы распознать удалённый отказ. Реализация IP SLA не является обязательной: её наличие зависит от модели Packet Tracer.

Глава 27

VLAN, trunk и маршрутизация между отделами

27.1 Границы широковещания

VLAN создаёт отдельный канальный домен широковещания внутри коммутируемой инфраструктуры. Порт access обслуживает одну пользовательскую VLAN. Trunk переносит несколько VLAN, различая их тегом IEEE 802.1Q; native VLAN в обычной конфигурации связана с нетегированными кадрами и должна согласовываться на концах. Набор allowed VLAN определяет, какие VLAN разрешены на trunk.

Номер VLAN и номер IP-подсети – независимые значения. В книге VLAN 10 часто соответствует 192.168.10.0/24 ради удобства чтения. Создание VLAN само по себе не создаёт шлюз. Для связи между VLAN нужен маршрутизатор с подинтерфейсами либо L3-коммутатор с SVI и включённой маршрутизацией. На 2960 не вводите `switchport trunk encapsulation dot1q`: у этой учебной модели инкапсуляция обычно фиксирована.

Лабораторная 11. Два отдела на одном коммутаторе

Время: 75 минут. **Подготовка:** 02, 06.

Какую проблему решает работа

В одном помещении работают бухгалтерия и посетители, но все подключены к одному коммутатору: бухгалтерский трафик виден гостям, а широковещание засоряет всю сеть. Покупать отдельный коммутатор каждой группе дорого. Работа решает проблему разделения: VLAN делят один физический коммутатор на изолированные логические сети.

Что вы научитесь делать

Разделить четыре компьютера на две VLAN и подтвердить изоляцию на канальном уровне.

Стенд и адреса

S1 2960. PC-A на Fa0/1 и PC-B на Fa0/2: VLAN 10, адреса 192.168.10.10 и .11/24. PC-C на Fa0/3 и PC-D на Fa0/4: VLAN 20, адреса 192.168.20.10 и .11/24. Шлюзы пустые.

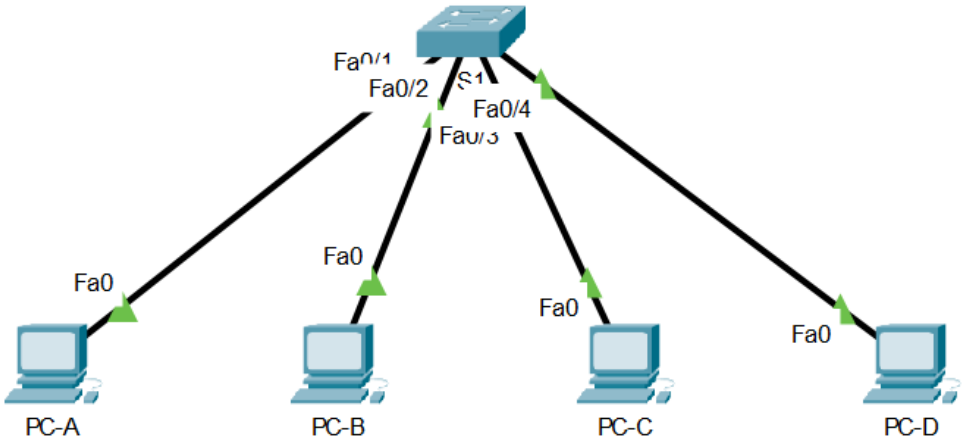


Схема работы 11: четыре компьютера на портах Fa0/1–Fa0/4 коммутатора S1.

Таблица соединений (проверена построением схемы в Packet Tracer 8.2.1). Порты записаны полными именами, как в списке портов программы.

Устройство	Порт	Устройство	Порт	Кабель
PC-A	FastEthernet0	S1	FastEthernet0/1	прямой
PC-B	FastEthernet0	S1	FastEthernet0/2	прямой
PC-C	FastEthernet0	S1	FastEthernet0/3	прямой
PC-D	FastEthernet0	S1	FastEthernet0/4	прямой

Порядок работы

```
enable
configure terminal
vlan 10
 name STAFF
exit
vlan 20
 name STUDENTS
exit
interface range fastEthernet 0/1 - 2
 switchport mode access
 switchport access vlan 10
exit
interface range fastEthernet 0/3 - 4
 switchport mode access
 switchport access vlan 20
exit
end
show vlan brief
show mac address-table
```

Назначьте адреса ПК, вызовите трафик и проверьте пары A–B и C–D. В Simulation создайте ARP-запрос от A и проследите, попадает ли он к C.

Проверка результата

Обе внутренние пары работают, межотдельская связь отсутствует. Отсутствие связи между VLAN с разными подсетями само по себе не отделяет влияние IP от VLAN; поэтому дополнительно назначьте C временный адрес 192.168.10.12/24, сохраняя VLAN 20. Он всё равно не должен общаться с A: канальные домены разделены. После теста верните C в 192.168.20.0/24. **Эталонный вывод.**

```
S1#show vlan brief

VLAN Name                              Status    Ports
---- --------------------------------- --------- -------------------------------
1    default                           active    Fa0/5, Fa0/6, Fa0/7, Fa0/8
                                                 ...
10   STAFF                             active    Fa0/1, Fa0/2
20   STUDENTS                          active    Fa0/3, Fa0/4
1002 fddi-default                      active
1003 token-ring-default                active
...
```

Ping между PC-A и PC-B проходит, а ping PC-A – PC-C заканчивается `Request timed out`: между VLAN нет маршрутизатора.

Управляемая неисправность

Переведите Fa0/2 в VLAN 20 и повторите ping A–B. В `show vlan brief` найдите неправильное назначение порта, затем исправьте.

Что сдать и объяснить

Сдайте список VLAN, матрицу связности и результат дополнительного теста с одинаковой IP-подсетью в разных VLAN. Почему VLAN – не замена маске?

Самостоятельное усложнение. Добавьте VLAN 30 и пару гостей. Сохраните изоляцию всех трёх групп без маршрутизатора.

Лабораторная 12. Trunk между коммутаторами

Время: 90 минут. **Подготовка:** 11.

Какую проблему решает работа

Отделы сидят на разных этажах, у каждого свой коммутатор, а VLAN должны быть общими. Тянуть отдельный кабель на каждый VLAN невозможно. Работа настраивает trunk: один кабель несёт несколько VLAN с метками, и учит находить типичные ошибки (не те разрешённые VLAN, несовпадающий native VLAN).

Что вы научитесь делать

Перенести VLAN через общую линию и диагностировать ошибку allowed VLAN.

Стенд и адреса

Два 2960: S1 G0/1–S2 G0/1. На каждом Fa0/1 – ПК VLAN 10, Fa0/2 – ПК VLAN 20. На S1 адреса .10, на S2 .11 соответствующих сетей 192.168.10.0/24 и 192.168.20.0/24. Шлюзы пустые.

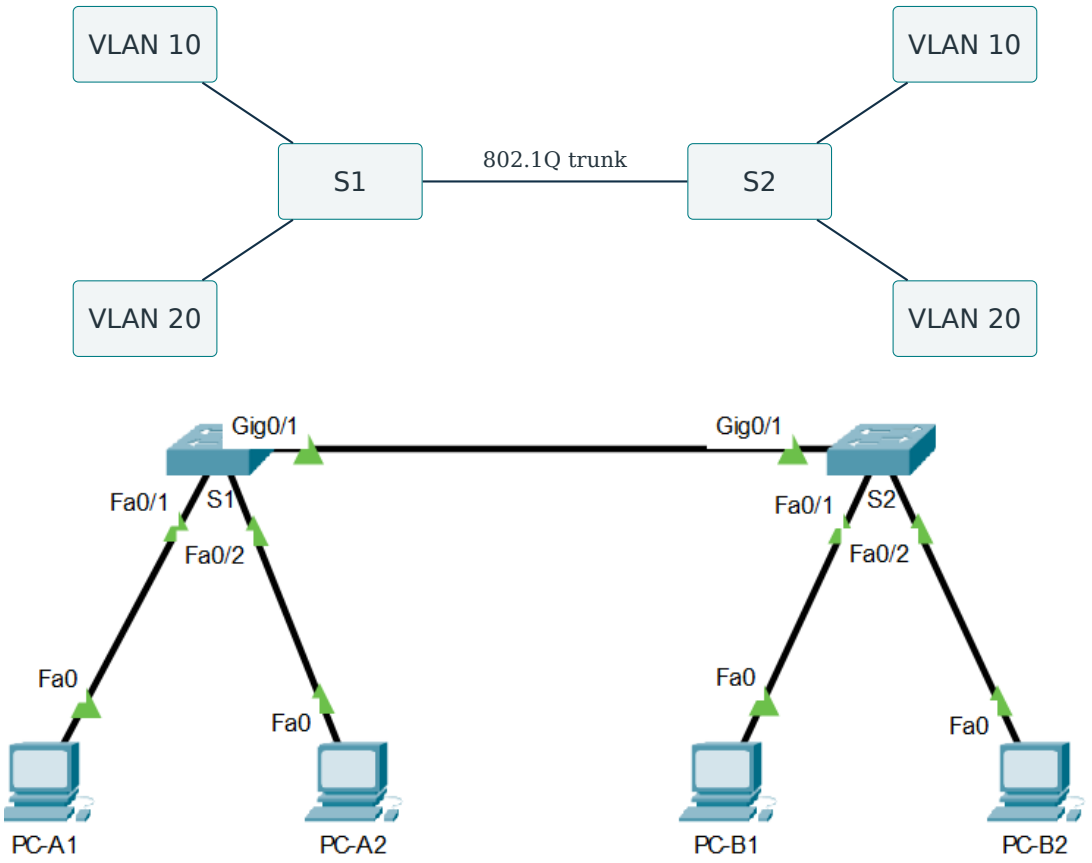


Схема работы 12: два коммутатора, trunk по линии Gig0/1 – Gig0/1.

Таблица соединений (проверена построением схемы в Packet Tracer 8.2.1). Порты записаны полными именами, как в списке портов программы.

Устройство	Порт	Устройство	Порт	Кабель
PC-A1	FastEthernet0	S1	FastEthernet0/1	прямой
PC-A2	FastEthernet0	S1	FastEthernet0/2	прямой
PC-B1	FastEthernet0	S2	FastEthernet0/1	прямой
PC-B2	FastEthernet0	S2	FastEthernet0/2	прямой
S1	GigabitEthernet0/1	S2	GigabitEthernet0/1	прямой

Порядок работы

На **обоих** коммутаторах создайте VLAN 10, 20, 99 и 999. Назначьте access-порты по образцу 11, затем настройте trunk:

```
interface gigabitEthernet 0/1
 switchport mode trunk
 switchport trunk native vlan 999
 switchport trunk allowed vlan 10,20,99,999
exit
end
show interfaces trunk
show vlan brief
```

VLAN 999 используется как согласованная native VLAN без пользовательских ПК. VLAN 99 зарезервирована для будущего управления. Не назначайте native VLAN пользовательскому порту только ради совпадения номеров.

Проверка результата

ПК одной VLAN через S1 и S2 обмениваются ICMP. ПК разных VLAN изолированы. `show interfaces trunk` показывает активный trunk, native 999 и разрешённые VLAN. В списке VLAN in spanning tree forwarding state нужные VLAN должны присутствовать после сходимости. **Эталонный вывод.** Через 30–60 секунд после настройки обоих концов:

```
S1#show interfaces trunk
Port        Mode         Encapsulation  Status        Native vlan
Gig0/1      on           802.1q         trunking      999

Port        Vlans allowed on trunk
Gig0/1      10,20,99,999

Port        Vlans allowed and active in management domain
Gig0/1      10,20,99,999

Port        Vlans in spanning tree forwarding state and not pruned
Gig0/1      10,20,99,999
```

Сразу после ввода команд последняя строка может показывать `none`: порт ещё проходит стадии STP. Это не ошибка; подождите или нажмите Fast Forward Time.

Управляемая неисправность

На S2 замените allowed list на `10,99,999`. ПК VLAN 10 продолжают работать, VLAN 20 – теряет межкоммутаторную связь. Восстановите список. Команда без `add` заменяет список полностью, а не добавляет очередной элемент.

Что сдать и объяснить

Покажите результаты до и после ограничения trunk. Объясните, почему создание VLAN 20 на обоих коммутаторах не гарантирует её передачу по линии.

Самостоятельное усложнение. Добавьте SVI управления 99 на S1 (.2) и S2 (.3), подключите административный ПК (.10) к VLAN 99 и проверьте ping. Полная сеть управления: 192.168.99.0/24.

Лабораторная 13. Router-on-a-stick

Время: 120 минут. **Подготовка:** 08, 12.

Какую проблему решает работа

Компьютеры разных VLAN изолированы друг от друга, но бухгалтерия должна обращаться к серверу в другой сети: нужна маршрутизация между VLAN. Работа решает её на одном физическом интерфейсе маршрутизатора с подинтерфейсами (router-on-a-stick) и показывает, где у такого решения предел.

Что вы научитесь делать

Создать шлюзы для двух VLAN на подинтерфейсах одного физического порта маршрутизатора.

Стенд и адреса

Возьмите 12. Добавьте R1 2911 G0/0–S1 G0/2. VLAN 10: шлюз 192.168.10.1; VLAN 20: шлюз 192.168.20.1. На S1 G0/2 используйте trunk, native 999, allowed 10,20,99,999 как в 12.

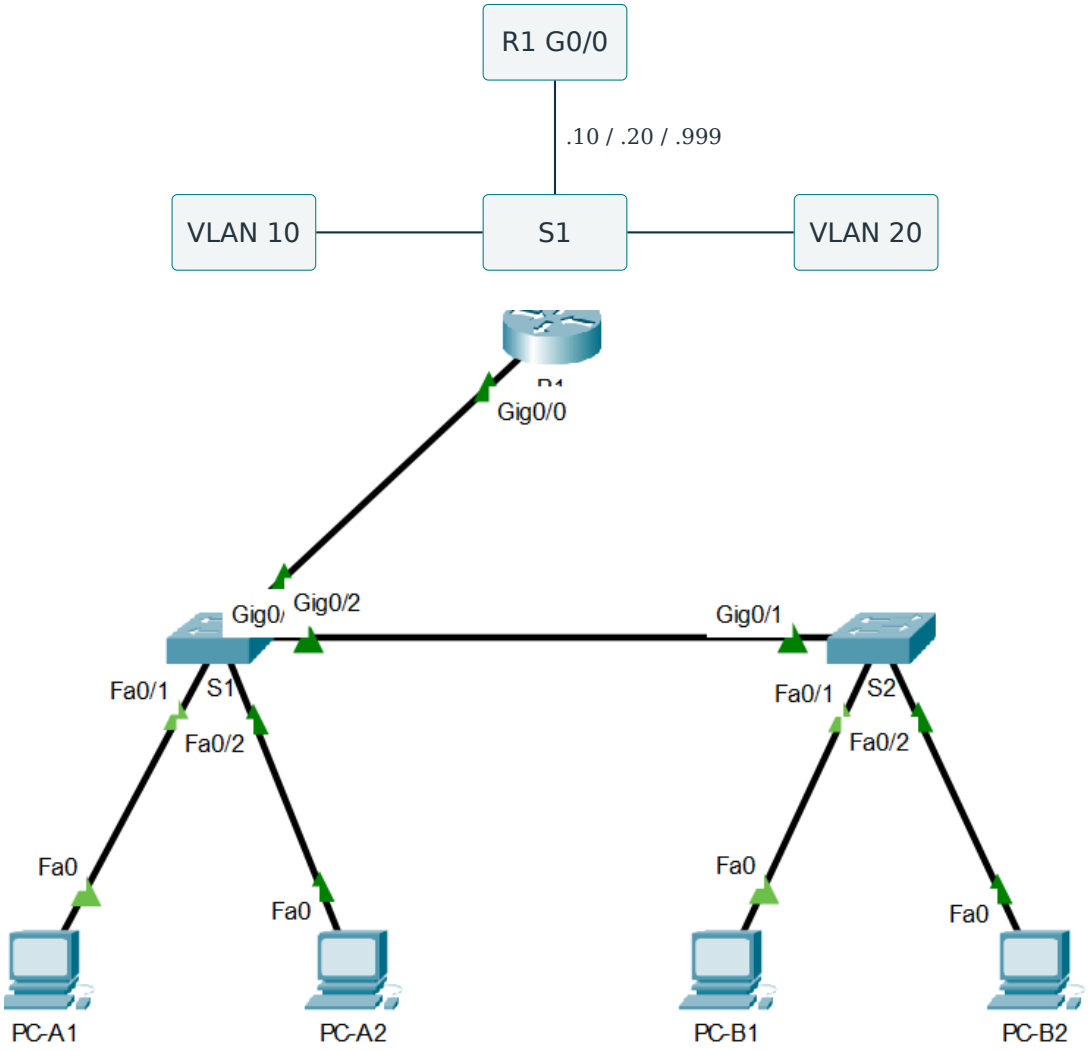


Схема работы 13: маршрутизатор R1 подключён к порту Gig0/2 коммутатора S1.

Таблица соединений (проверена построением схемы в Packet Tracer 8.2.1). Порты записаны полными именами, как в списке портов программы.

Устройство	Порт	Устройство	Порт	Кабель
PC-A1	FastEthernet0	S1	FastEthernet0/1	прямой
PC-A2	FastEthernet0	S1	FastEthernet0/2	прямой
PC-B1	FastEthernet0	S2	FastEthernet0/1	прямой
PC-B2	FastEthernet0	S2	FastEthernet0/2	прямой
S1	GigabitEthernet0/1	S2	GigabitEthernet0/1	прямой
R1	GigabitEthernet0/0	S1	GigabitEthernet0/2	прямой

Порядок работы

На R1 настройте подинтерфейсы:

```
enable
configure terminal
interface gigabitEthernet 0/0
 no ip address
 no shutdown
exit
interface gigabitEthernet 0/0.10
 encapsulation dot1Q 10
```

```
 ip address 192.168.10.1 255.255.255.0
exit
interface gigabitEthernet 0/0.20
 encapsulation dot1Q 20
 ip address 192.168.20.1 255.255.255.0
exit
interface gigabitEthernet 0/0.999
 encapsulation dot1Q 999 native
 no ip address
exit
end
show ip interface brief
show ip route
```

На ПК назначьте шлюзы своей VLAN. VLAN 99 в этой работе может оставаться без маршрутизируемого шлюза: её передача по trunk не обязывает создавать SVI или подинтерфейс с адресом.

Проверка результата

ПК достигают собственного шлюза и компьютеров другой VLAN через R1. В таблице R1 видны обе подключённые сети. В Simulation проследите обмен, который сначала проходит к R1 и возвращается по тому же физическому trunk с другим тегом VLAN. **Эталонный вывод.**

```
R1#show ip route
C       192.168.10.0/24 is directly connected, GigabitEthernet0/0.10
L       192.168.10.1/32 is directly connected, GigabitEthernet0/0.10
C       192.168.20.0/24 is directly connected, GigabitEthernet0/0.20
L       192.168.20.1/32 is directly connected, GigabitEthernet0/0.20
```

Ping PC-A1 – PC-B2 (разные VLAN) возвращает `TTL=127`: путь проходит через один маршрутизатор.

Управляемая неисправность

На G0/0.20 замените encapsulation на `dot1Q 200`. VLAN 20 теряет доступ к своему шлюзу, хотя физический порт активен. Верните `dot1Q 20`; не исправляйте ошибку изменением IP-адресов ПК.

Что сдать и объяснить

Нужны схемы портов, адресный план, вывод trunk и подинтерфейсов, матрица связности. Почему суффикс .20 не определяет VLAN автоматически, если в encapsulation указан другой номер?

Самостоятельное усложнение. Добавьте VLAN 30 с сетью 192.168.30.0/24, подинтерфейсом и парой ПК. Не забудьте создание VLAN, access-порты, allowed list на каждом trunk и шлюзы.

Лабораторная 14. Маршрутизация на L3-коммутаторе

Время: 90 минут. **Подготовка:** 13.

Какую проблему решает работа

Решение с одним кабелем к маршрутизатору (работа 13) становится узким местом, когда трафик между отделами велик: все пакеты идут туда и обратно по одному проводу. Работа решает проблему производительности: маршрутизацию между VLAN выполняет сам L3-коммутатор на виртуальных интерфейсах (SVI).

Что вы научитесь делать

Заменить внешний маршрутизатор шлюзами SVI и включить маршрутизацию на multilayer switch.

Стенд и адреса

Отдельная копия 12: удалите R1, если используете копию 13. Добавьте MLS1 3560 Fa0/1–S1 G0/2. Создайте VLAN 10,20,99,999. На S1 G0/2 настройте trunk как в 12. Шлюзы ПК остаются .1.

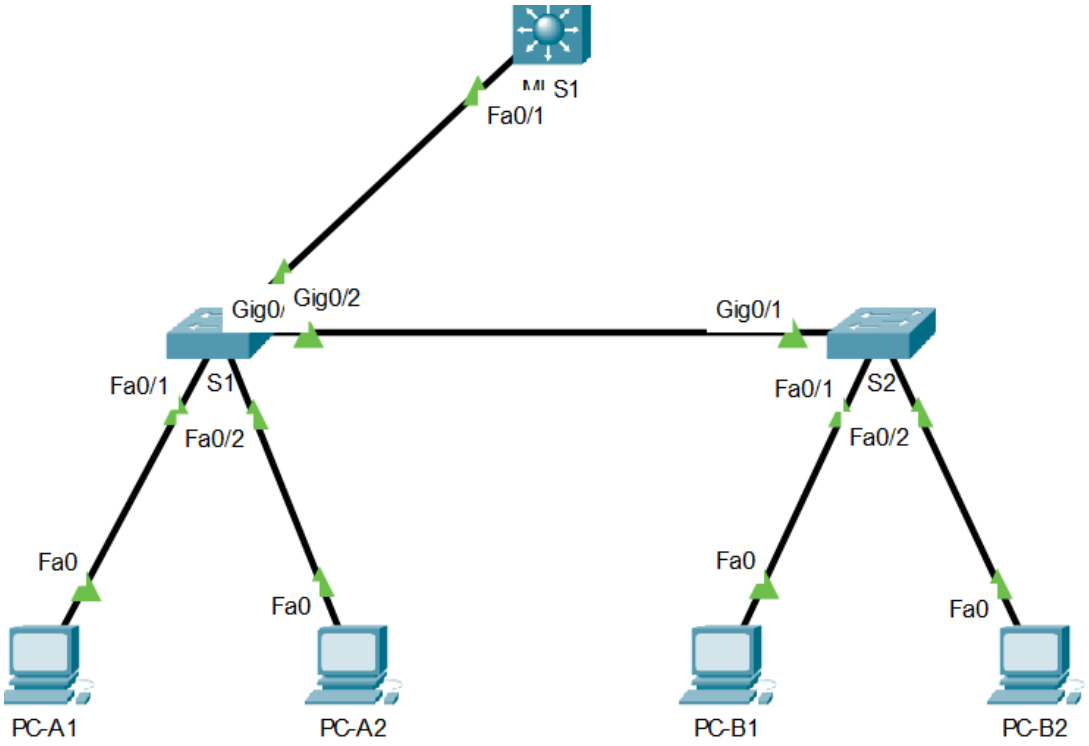


Схема работы 14: вместо R1 – многоуровневый коммутатор MLS1 (на схеме подпись MLS1).

Таблица соединений (проверена построением схемы в Packet Tracer 8.2.1). Порты записаны полными именами, как в списке портов программы.

Устройство	Порт	Устройство	Порт	Кабель
PC-A1	FastEthernet0	S1	FastEthernet0/1	прямой
PC-A2	FastEthernet0	S1	FastEthernet0/2	прямой
PC-B1	FastEthernet0	S2	FastEthernet0/1	прямой
PC-B2	FastEthernet0	S2	FastEthernet0/2	прямой
S1	GigabitEthernet0/1	S2	GigabitEthernet0/1	прямой
MLS1	FastEthernet0/1	S1	GigabitEthernet0/2	прямой

Порядок работы

На MLS1:

```
enable
configure terminal
hostname MLS1
vlan 10
exit
vlan 20
exit
vlan 99
exit
vlan 999
exit
interface fastEthernet 0/1
 switchport trunk encapsulation dot1q
 switchport mode trunk
 switchport trunk native vlan 999
 switchport trunk allowed vlan 10,20,99,999
exit
interface vlan 10
 ip address 192.168.10.1 255.255.255.0
 no shutdown
exit
interface vlan 20
 ip address 192.168.20.1 255.255.255.0
 no shutdown
exit
ip routing
end
show ip interface brief
show ip route
```

Команда выбора encapsulation относится к 3560, который её поддерживает; на модели с фиксированным 802.1Q она может отсутствовать. Не оставляйте старый R1 с такими же адресами шлюзов в подключённой сети.

Проверка результата

Vlan10 и Vlan20 up/up. Межотдельский ping проходит. Устройство одновременно коммутирует кадры внутри VLAN и маршрутизирует между SVI. Для внешней сети нужен отдельный L3-путь: порт с `no switchport` и IP-адресом либо транзитная VLAN. **Эталонный вывод.**

```
MLS1#show ip interface brief
...
Vlan10                 192.168.10.1    YES manual up                    up
Vlan20                 192.168.20.1    YES manual up                    up
MLS1#show ip route
C    192.168.10.0/24 is directly connected, Vlan10
C    192.168.20.0/24 is directly connected, Vlan20
```

В таблице маршрутов MLS1 нет строк `L` с адресами /32: в выводе многоуровневого коммутатора 3560 в Packet Tracer они не показываются. Ping между VLAN снова даёт `TTL=127`. Перед настройкой на `MLS1` (3560) так же, как на маршрутизаторе, может появиться вопрос `Would you like to enter the initial configuration dialog?`: ответьте `no`.

Управляемая неисправность

Выключите маршрутизацию командой `no ip routing`. Проверьте локальную связь внутри VLAN, достижимость SVI и межотдельскую передачу. Верните `ip`

`routing`; не делайте вывод о маршрутизации только по ping своего шлюза.

Что сдать и объяснить

Сравните путь пакета в 13 и 14. Объясните условия up/up SVI и почему `ip default-gateway` на L2-коммутаторе не заменяет таблицу маршрутов MLS1.

Самостоятельное усложнение. Добавьте R-EDGE через routed-port MLS1 Fa0/24: MLS1 10.0.1.1/30, R-EDGE 10.0.1.2/30. На MLS1 задайте default route к .2, а на R-EDGE – обратные маршруты к обеим VLAN через .1.

Глава 28

Петли, резервирование и защита доступа

28.1 Почему два кабеля не всегда лучше одного

В Ethernet широковещательный кадр не имеет IP TTL, который ограничивал бы его жизнь в L2-петле. При циклических соединениях возможны шторм и нестабильное обучение MAC. STP строит дерево: выбирает root bridge и оставляет часть избыточных путей без передачи пользовательских кадров. Блокировка одного порта при исправной резервной схеме является ожидаемым результатом.

Root выбирается по Bridge ID, в котором важен приоритет и MAC-компонент. Управляемое назначение root лучше случайного. PortFast предназначен для конечного узла, а не для линии между коммутаторами. BPDU Guard может отключить edge-порт при появлении BPDU. Проверять эти механизмы надо в маленьком сохранённом стенде: бесконтрольная петля может перегрузить сам симулятор.

EtherChannel объединяет совместимые физические линии в один логический канал. LACP active инициирует согласование; passive отвечает. Пара passive/passive не инициирует его. Распределение потоков обычно основано на хеше; один поток не обязан использовать сумму скоростей всех линий. Packet Tracer здесь помогает проверить структуру, но не служит точным тестером пропускной способности.

Лабораторная 15. STP: дерево и переключение после отказа

Время: 90 минут. **Подготовка:** 12.

Какую проблему решает работа

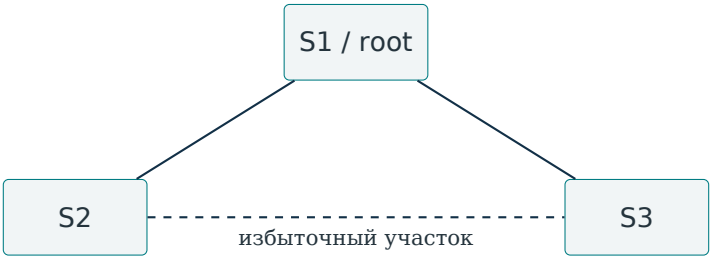
Для надёжности коммутаторы соединяют несколькими кабелями, но кольцо в сети Ethernet приводит к «шторму»: кадры ходят по кругу, и сеть парализована за секунды. Работа показывает, как протокол STP сам отключает лишнюю связь, чтобы петли не было, и включает её, когда основная линия выходит из строя.

Что вы научитесь делать

Назначить корневой коммутатор и объяснить блокировку резервного порта.

Стенд и адреса

Три 2960 в треугольнике. S1 G0/1–S2 G0/1; S2 G0/2–S3 G0/1; S3 G0/2–S1 G0/2. Используйте VLAN 1 и прямую явную настройку trunk на этих линиях. PC-A на S1 Fa0/1: 192.168.10.10/24; PC-B на S3 Fa0/1: .20/24. Шлюза нет.



На рисунке – пример дерева с root S1. Реальный блокируемый порт устанавливается по выводу STP, а не назначается пунктиром на схеме.

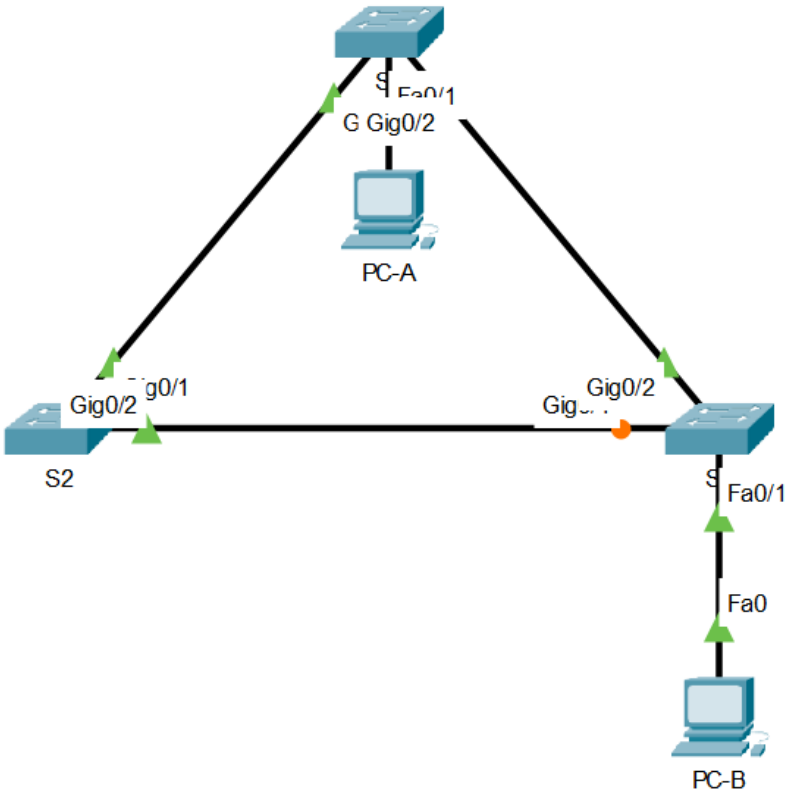


Схема работы 15: три коммутатора в треугольнике. Оранжевая точка на линии S2 – S3: этот порт заблокирован протоколом STP и поэтому не зелёный.

Таблица соединений (проверена построением схемы в Packet Tracer 8.2.1). Порты записаны полными именами, как в списке портов программы.

Устройство	Порт	Устройство	Порт	Кабель
PC-A	FastEthernet0	S1	FastEthernet0/1	прямой
PC-B	FastEthernet0	S3	FastEthernet0/1	прямой
S1	GigabitEthernet0/1	S2	GigabitEthernet0/1	прямой
S2	GigabitEthernet0/2	S3	GigabitEthernet0/1	прямой
S3	GigabitEthernet0/2	S1	GigabitEthernet0/2	прямой

Порядок работы

1. До замыкания третьей линии убедитесь, что STP включён: `show spanning-tree vlan 1`. На всех межкоммутаторных портах задайте `switchport mode trunk`. Не включайте на них PortFast.

2. На S1 в конфигурации выполните `spanning-tree vlan 1 root primary`; на S2 – `spanning-tree vlan 1 root secondary`. Замкните треугольник и дождитесь сходимости.
3. На каждом устройстве прочитайте `show spanning-tree vlan 1`: Root ID, Bridge ID, роль и состояние портов. Не пытайтесь заранее угадать блокируемый порт только по расположению схемы.
4. Запустите ping между ПК. Выключите одну forwarding-линию на пути к root и наблюдайте переход альтернативного пути к передаче.
5. Восстановите линию, дождитесь стабильного состояния, сохраните проект.

Проверка результата

S1 сообщает, что он root; на остальных Root ID совпадает с S1. В треугольнике один участок дерева не передаёт пользовательские кадры. После отказа действующей линии связность восстанавливается с задержкой, зависящей от моделируемого режима STP. Запишите наблюдаемое время, не выдавая его за универсальный норматив. **Эталонный вывод** (фрагменты `show spanning-tree vlan 1`; MAC-адреса и выбор блокируемого порта у вас могут отличаться):

```
S1#show spanning-tree vlan 1
  Root ID    Priority    24577
             Address     0001.9667.63B1
             This bridge is the root
Interface        Role Sts Cost      Prio.Nbr Type
Fa0/1            Desg FWD 19        128.1    P2p
Gi0/1            Desg FWD 4         128.25   P2p
Gi0/2            Desg FWD 4         128.26   P2p

S3#show spanning-tree vlan 1
Interface        Role Sts Cost      Prio.Nbr Type
Fa0/1            Desg FWD 19        128.1    P2p
Gi0/1            Altn BLK 4         128.25   P2p
Gi0/2            Root FWD 4         128.26   P2p
```

Приоритет root-коммутатора равен 24577 (24576 плюс номер VLAN 1): команда `root primary` сама подбирает значение ниже текущего минимального. Роль `Altn` и состояние `BLK` у порта S3 Gi0/1 – заблокированный резервный путь.

Управляемая неисправность

На S3 задайте `spanning-tree vlan 1 priority 0`. Проверьте изменение root и топологии. Восстановите управляемый выбор: на S3 верните приоритет 32768, затем проверьте S1. Приоритет задаётся допустимыми шагами, обычно 4096.

Что сдать и объяснить

Сдайте рисунок дерева до и после отказа и таблицу ролей портов. Почему зелёные кабели схемы не означают, что все порты обязательно forwarding?

Самостоятельное усложнение. Повторите опыт с `spanning-tree mode rapid-pvst`, если команда поддерживается. Сравните переходы, сохранив ту же топологию и выбранный root.

Лабораторная 16. EtherChannel с LACP

Время: 90 минут. **Подготовка:** 12, 15.

Какую проблему решает работа

Один кабель между коммутаторами ограничивает скорость и является единственной точкой отказа, а два кабеля протокол STP заблокирует. Работа решает проблему объединением нескольких кабелей в один логический канал (EtherChannel, LACP): скорость канала складывается, а отказ одного кабеля не разрывает связь.

Что вы научитесь делать

Создать логический trunk из двух линий и проверить сохранение связи при отказе одного участника.

Стенд и адреса

S1 и S2 2960. Параллельные линии: Fa0/23–Fa0/23 и Fa0/24–Fa0/24. ПК VLAN 10 на Fa0/1: S1 192.168.10.10/24, S2 .20/24. Создайте VLAN 10,20,999 на обоих коммутаторах.

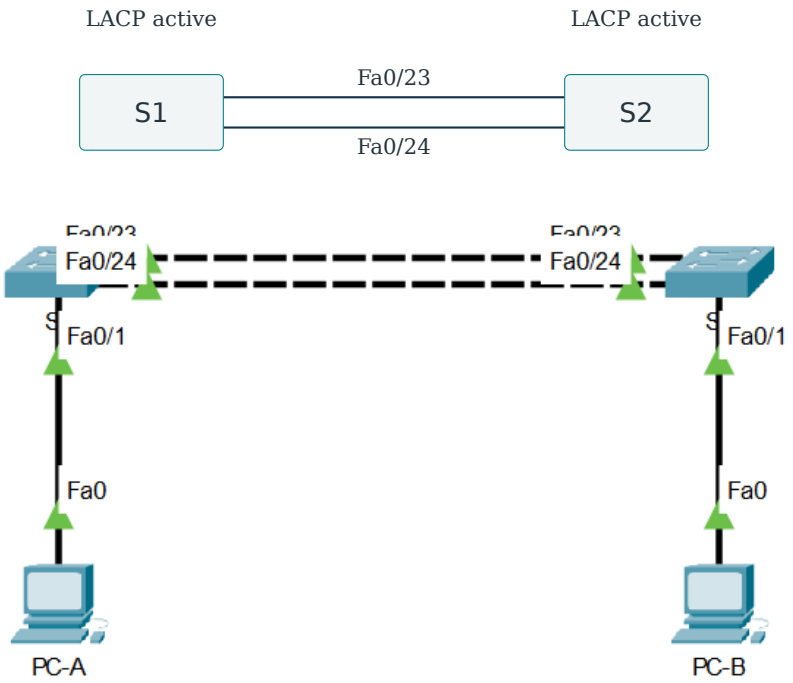


Схема работы 16: две параллельные линии Fa0/23 и Fa0/24 между S1 и S2 (пунктир – crossover).

Таблица соединений (проверена построением схемы в Packet Tracer 8.2.1). Порты записаны полными именами, как в списке портов программы.

Устройство	Порт	Устройство	Порт	Кабель
PC-A	`FastEthernet0`	S1	`FastEthernet0/1`	прямой
PC-B	`FastEthernet0`	S2	`FastEthernet0/1`	прямой
S1	`FastEthernet0/23`	S2	`FastEthernet0/23`	перекрёстный (crossover)
S1	`FastEthernet0/24`	S2	`FastEthernet0/24`	перекрёстный (crossover)

Порядок работы

Вводите одну и ту же группу на **обоих** устройствах. Для предотвращения промежуточных ошибок сначала настройте порты в shutdown.

```
enable
configure terminal
interface range fastEthernet 0/23 - 24
 shutdown
 switchport mode trunk
 switchport trunk native vlan 999
 switchport trunk allowed vlan 10,20,999
 channel-group 1 mode active
 no shutdown
exit
interface port-channel 1
 switchport mode trunk
 switchport trunk native vlan 999
 switchport trunk allowed vlan 10,20,999
exit
interface fastEthernet 0/1
 switchport mode access
 switchport access vlan 10
exit
end
show etherchannel summary
show interfaces trunk
show spanning-tree vlan 10
```

Проверка результата

В summary логический канал используется как L2-канал; участники находятся в bundle. Часто это обозначается `Po1(SU)` и `(P)` у портов; ориентируйтесь на легенду фактического вывода. STP воспринимает Port-channel как логическую линию. При shutdown Fa0/23 ping после переходного периода сохраняется через Fa0/24. **Эталонный вывод.**

```
S1#show etherchannel summary
Flags:  D - down        P - in port-channel
...
Number of channel-groups in use: 1
Number of aggregators:           1

Group  Port-channel  Protocol    Ports
------+-------------+-----------+-----------------------------------------------
1      Po1(SU)           LACP   Fa0/23(P) Fa0/24(P)
```

`SU` – Layer 2, в работе; `(P)` – порт входит в пакет. В `show interfaces trunk` теперь указан порт `Po1` вместо двух физических.

Управляемая неисправность

На обоих концах замените active на passive, предварительно выключив участников и удалив `channel-group 1`. Проверьте отсутствие инициатора и исправьте один конец на active. Если команда изменения параметров отвергается, разберите bundle в сохранённой копии и создайте заново согласованно.

Что сдать и объяснить

Сдайте summary, список trunk и результат отказа одного кабеля. Объясните, почему один ping не измеряет суммарную пропускную способность канала.

Самостоятельное усложнение. Добавьте ПК VLAN 20. Проверьте её связь через тот же Port-channel и повторите опыт с allowed list, удалив только VLAN 20 из логического trunk согласованно на концах.

Лабораторная 17. Port Security, PortFast и BPDU Guard

Время: 90 минут. **Подготовка:** 11, 15.

Какую проблему решает работа

В сетевую розетку офиса может подключиться кто угодно: чужой ноутбук, мини-коммутатор из дома, злоумышленник. Работа решает проблему защиты порта доступа: ограничение числа разрешённых MAC-адресов, быстрый запуск порта для рабочих мест и защита от посторонних служебных кадров.

Что вы научитесь делать

Ограничить число MAC на пользовательском порту и отличать нарушение политики от физического отказа.

Стенд и адреса

S1 2960, PC-A на Fa0/1, PC-B на Fa0/2. Оба VLAN 10, адреса 192.168.10.10 и .20/24. Создайте VLAN 10. Защиту настраивайте только на Fa0/1; Fa0/2 обычный access.

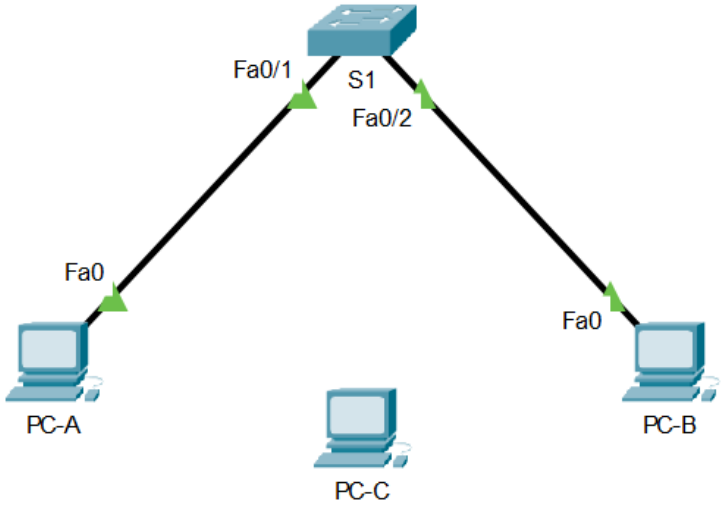


Схема работы 17: PC-A и PC-B на S1; PC-C отдельно, он понадобится для опыта с нарушением.

Таблица соединений (проверена построением схемы в Packet Tracer 8.2.1). Порты записаны полными именами, как в списке портов программы.

Устройство	Порт	Устройство	Порт	Кабель
PC-A	FastEthernet0	S1	FastEthernet0/1	прямой
PC-B	FastEthernet0	S1	FastEthernet0/2	прямой

Порядок работы

```
enable
configure terminal
```

```
interface fastEthernet 0/1
 switchport mode access
 switchport access vlan 10
 spanning-tree portfast
 spanning-tree bpduguard enable
 switchport port-security
 switchport port-security maximum 1
 switchport port-security mac-address sticky
 switchport port-security violation shutdown
exit
end
show port-security interface fastEthernet 0/1
show port-security address
```

Назначьте IP, выполните ping A–B, затем сохраните конфигурацию: sticky-запись появляется после обучения MAC. Если сохранить раньше первого трафика, нужного адреса ещё может не быть в конфигурации.

Проверка результата

Вывод показывает защищённый порт и один изученный MAC. Отключите PC-A и подключите новый PC-C с другим MAC, но тем же учебным IP, избегая одновременного дубликата адреса. Вызовите трафик: режим shutdown переводит порт в состояние нарушения. **Эталонный вывод.** До первого трафика и после него:

```
S1#show port-security interface fastEthernet 0/1
Port Security              : Enabled
Port Status                : Secure-up
Violation Mode             : Shutdown
Maximum MAC Addresses      : 1
Total MAC Addresses        : 0
Sticky MAC Addresses       : 0
Security Violation Count   : 0
   ... после ping ...
Total MAC Addresses        : 1
Sticky MAC Addresses       : 1
Last Source Address:Vlan   : 00D0.582B.6539:10

S1#show port-security address
  10    00D0.582B.6539    SecureSticky                  Fa0/1        -
```

При подключении другого ПК к Fa0/1 и первом же кадре порт отключается:

```
Port Status                : Secure-shutdown
Security Violation Count   : 1
S1#show interfaces status
Fa0/1                       err-disabled 10         auto    auto  10/100BaseTX
```

Состояние `err-disabled` – отключение по политике. После `shutdown` и `no shutdown` порт снова `Secure-up`, но старая sticky-запись остаётся, и новый ПК нарушит правило снова, пока вы не удалите запись.

Управляемая неисправность

Верните PC-A, но порт не обязан сам восстановиться. На Fa0/1 выполните `shutdown`, затем `no shutdown`; проверьте состояние и счётчик. Если вы намеренно заменяете доверенный ПК, сначала удалите конкретную старую sticky-строку из running-config, затем разрешите обучение новой. Не снимайте всю защиту ради ремонта.

Что сдать и объяснить

Сдайте вывод до нарушения, при нарушении и после восстановления. Чем violation shutdown отличается от отсутствующего кабеля? Почему MAC-ограничение не заменяет аутентификацию человека?

Самостоятельное усложнение. В отдельной копии подключите к защищённому edge-порту другой коммутатор. Проследите срабатывание BPDU Guard; учитывайте, что одновременно может мешать Port Security. Для чистого опыта временно уберите Port Security только в этой копии.

Лабораторная 18. Инвентаризация через CDP и описание портов

Время: 60–75 минут. **Подготовка:** 06, 12.

Какую проблему решает работа

Через год никто не помнит, что куда подключено: кабели без подписей, схема устарела, а чтобы найти причину сбоя, приходится идти к шкафу. Работа решает проблему инвентаризации: описания портов и протокол CDP позволяют узнать соседей устройства и восстановить схему по самим устройствам.

Что вы научитесь делать

Восстановить реальные соседства и найти ошибку в документации.

Стенд и адреса

R1 G0/0–S1 G0/2; S1 G0/1–S2 G0/1. Для CDP IP-адреса не обязательны, но интерфейс R1 G0/0 необходимо включить командой `no shutdown` (интерфейсы маршрутизатора выключены по умолчанию: пока он выключен, линия не поднимется и CDP-соседа не будет). Если используете старый стенд, заранее запишите его действующую схему портов.



Схема работы 18: R1 – S1 – S2.

Таблица соединений (проверена построением схемы в Packet Tracer 8.2.1). Порты записаны полными именами, как в списке портов программы.

Устройство	Порт	Устройство	Порт	Кабель
R1	`GigabitEthernet0/0`	S1	`GigabitEthernet0/2`	прямой
S1	`GigabitEthernet0/1`	S2	`GigabitEthernet0/1`	прямой

Порядок работы

1. На каждом IOS-узле выполните `show cdp neighbors` и `show cdp neighbors detail`. Дождитесь очередного объявления, если таблица ещё пуста.
2. Сопоставьте local interface и port ID соседа; составьте таблицу «узел / локальный порт / сосед / его порт / модель».
3. Добавьте на порты `description TO-S2-G0/1` и аналогичные подписи. Проверьте `show interfaces description`, если команда доступна; иначе прочитайте описания через running-config.

4. На одном соединении переставьте кабель в свободный порт, дождитесь обновления и исправьте документацию.

Проверка результата

Таблица соседств согласуется на обоих концах и с кабелями схемы. Отсутствие CDP-записи само по себе не доказывает отсутствия линии: протокол может быть выключен или сосед может его не поддерживать. **Эталонный вывод.** Интерфейс R1 G0/0 должен быть включён (`no shutdown`); пока он выключен, S1 не видит R1:

```
S1#show cdp neighbors
Device ID    Local Intrfce   Holdtme    Capability   Platform    Port ID
S2           Gig 0/1          150            S       2960        Gig 0/1
R1           Gig 0/2          120            R       C2900       Gig 0/0
```

Команда `show interfaces description` на коммутаторе 2960 в Packet Tracer 8.2.1 не поддерживается (ответ «Invalid input»); описания читайте в `show running-config`. Команды `lldp run` и `show lldp neighbors` на 2960 принимаются.

Управляемая неисправность

На R1 G0/0 выполните `no cdp enable`. Связь IP при настроенных адресах может продолжаться, а объявления перестанут поступать и устаревшая запись исчезнет после holdtime. Верните `cdp enable`.

Что сдать и объяснить

Сдайте инвентарную таблицу и пример расхождения подписи с фактическим портом. Почему информация discovery-протокола полезна для обслуживания и одновременно раскрывает сведения о сети?

Самостоятельное усложнение. На модели, поддерживающей LLDP, проверьте `lldp run` и `show lldp neighbors`. Если команда отсутствует, зафиксируйте модель и результат `?`; не считайте это отказом Ethernet.

Часть III

Сетевые службы и политики доступа

Глава 29

Автоматическая настройка и приложения

29.1 DHCP и relay

DHCP передаёт клиенту адрес, маску, шлюз и другие параметры. В обычном первоначальном обмене IPv4 видны Discover, Offer, Request, ACK – DORA. Сервер выделяет адрес на срок аренды; статически назначенные адреса инфраструктуры следует исключить из пула. Адрес DNS и адрес шлюза решают разные задачи.

Широковещательный Discover не проходит через маршрутизатор как обычный пользовательский broadcast. Relay на клиентском интерфейсе принимает запрос и направляет его DHCP-серверу. Адрес relay в поле giaddr помогает серверу выбрать пул подсети клиента. Команду `ip helper-address` ставят со стороны клиента, а не произвольно на ближайшем к серверу интерфейсе [9].

29.2 DNS, HTTP и транспорт

DNS возвращает записи для имени. В типичном учебном запросе используются UDP и порт 53; DNS может работать и через TCP. HTTP использует TCP, в наших простых стендах – порт 80. Успешное открытие по IP и неуспешное по имени указывают на проблему разрешения имени, но диагноз нужно подтвердить настройками и событиями DNS.

TCP устанавливает соединение: SYN, SYN-ACK, ACK. Затем передаётся прикладной запрос и ответ. Порты клиента обычно динамические, порт сервера связан со службой. UDP не выполняет такой handshake. На экране Simulation различайте установление TCP, передачу HTTP и разрешение DNS: всё это может быть частью одного действия «открыть страницу».

Лабораторная 19. DHCP на маршрутизаторе

Время: 90 минут. **Подготовка:** 08.

Какую проблему решает работа

Вручную задавать адрес каждому из сотни компьютеров долго и ведёт к ошибкам: два устройства с одним адресом, забытый шлюз, опечатка в маске. Работа настраивает DHCP на маршрутизаторе: адреса, шлюз и DNS выдаются автоматически, а адреса серверов и принтеров исключаются из выдаваемого пула.

Что вы научитесь делать

Организовать выдачу параметров ПК и проверить аренды и исключения.

Стенд и адреса

R1 G0/0 192.168.10.1/24–S1–три ПК. R1 настроен и включён как в 08. Адреса .1–.49 оставляем инфраструктуре; DHCP должен выдавать остальные адреса из /24. DNS 192.168.10.53 пока обозначает будущий сервер, его работа не является

критерием этой лабораторной.

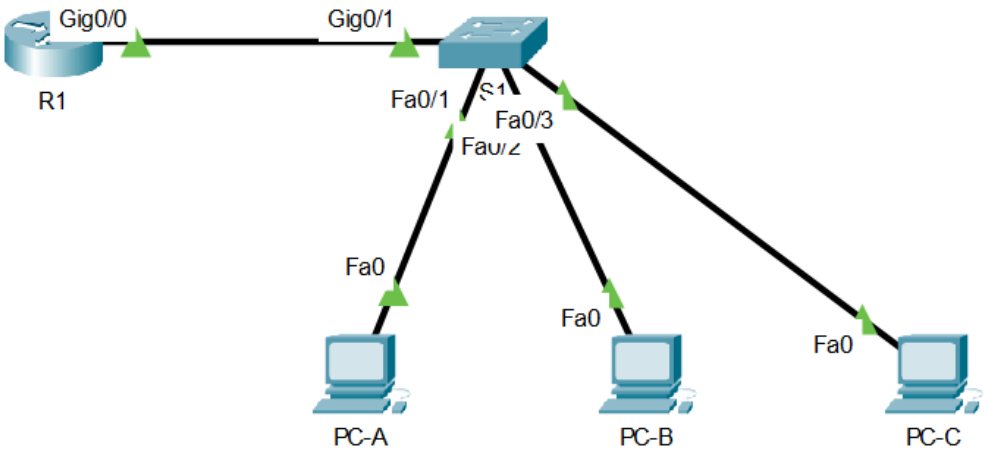


Схема работы 19: маршрутизатор R1, коммутатор S1 и три компьютера, получающих адреса по DHCP.

Таблица соединений (проверена построением схемы в Packet Tracer 8.2.1). Порты записаны полными именами, как в списке портов программы.

Устройство	Порт	Устройство	Порт	Кабель
R1	`GigabitEthernet0/0`	S1	`GigabitEthernet0/1`	прямой
PC-A	`FastEthernet0`	S1	`FastEthernet0/1`	прямой
PC-B	`FastEthernet0`	S1	`FastEthernet0/2`	прямой
PC-C	`FastEthernet0`	S1	`FastEthernet0/3`	прямой

Порядок работы

```
enable
configure terminal
ip dhcp excluded-address 192.168.10.1 192.168.10.49
ip dhcp pool LAN10
 network 192.168.10.0 255.255.255.0
 default-router 192.168.10.1
 dns-server 192.168.10.53
exit
end
show ip dhcp pool
show ip dhcp binding
```

1. На ПК в Desktop / IP Configuration выберите DHCP. Дождитесь адреса. Проверьте маску и шлюз, а не только первую строку.
2. Выполните ping шлюза и между клиентами. После выдачи выведите binding повторно и сопоставьте клиентов с адресами.
3. В отдельном повторе включите Simulation с DHCP. Переключите одного клиента в Static, затем обратно в DHCP, и найдите DORA.
4. Исключите .53 дополнительно командой `ip dhcp excluded-address 192.168.10.53`, чтобы будущий DNS не конфликтовал с пулом. В рабочем варианте исключение должно быть сделано до выдачи этого адреса клиенту.

Проверка результата

Каждый клиент получил уникальный адрес, не относящийся к исключениям, маску /24 и шлюз .1. `show ip dhcp binding` содержит действующие выдачи. Запись `dns-server` не запускает DNS-службу на маршрутизаторе и не подтверждает её доступность. **Эталонный вывод.**

```
R1#show ip dhcp binding
IP address       Client-ID/              Lease expiration        Type
                 Hardware address
192.168.10.52    000B.BEAE.9B44           --                     Automatic
192.168.10.51    0001.97EB.A692           --                     Automatic
192.168.10.50    0090.0C40.4C36           --                     Automatic
```

Клиенты получают адреса начиная с `.50`, то есть сразу после исключённого диапазона. Если в окне *IP Configuration* клиента сразу после переключения на DHCP появилось «DHCP failed. APIPA is being used» (адрес 169.254.x.x), не меняйте настройки: переключите ПК на Static и снова на DHCP – первая попытка нередко делается до того, как порт коммутатора начал передачу.

Управляемая неисправность

В pool замените default-router на 192.168.10.254. Получите параметры заново на одном ПК и сравните локальный ping с доступом в удалённую сеть, если она добавлена. Верните .1 и обновите аренду: исправление пула не обязательно немедленно изменяет параметры уже настроенного клиента.

Что сдать и объяснить

Сдайте DORA, таблицу аренд и фактические параметры трёх клиентов. Объясните, почему инфраструктурные адреса исключают из динамической выдачи.

Самостоятельное усложнение. Добавьте к R1 G0/1 LAN 20 и второй pool. Убедитесь, что клиенты каждой LAN получают параметры именно своей сети.

Лабораторная 20. DHCP-сервер в другой подсети

Время: 105 минут. **Подготовка:** 19, 08.

Какую проблему решает работа

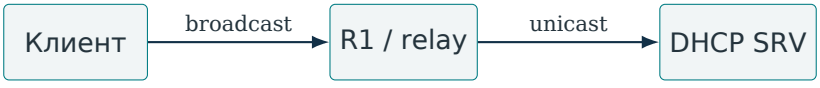
DHCP-сервер один, а сетей несколько, и широковещательный запрос клиента через маршрутизатор не проходит. Поэтому клиенты в «чужих» сетях остаются без адреса. Работа решает проблему настройкой ретрансляции (ip helper-address): один центральный сервер обслуживает разные сети.

Что вы научитесь делать

Настроить relay и отличить маршрут к серверу от механизма передачи DHCP.

Стенд и адреса

R1 G0/0–S1–PC-A: 192.168.10.0/24, R1 .1. R1 G0/1–S2–SRV: 192.168.50.0/24, R1 .1, SRV .10 со шлюзом .1. DHCP на R1 в этом новом стенде не настраивается.



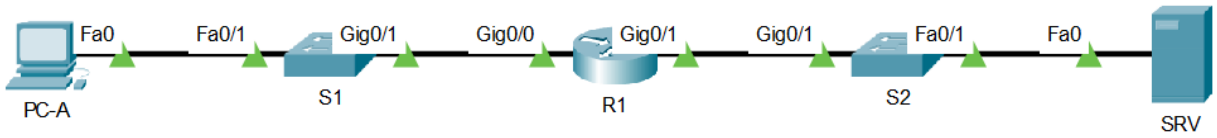


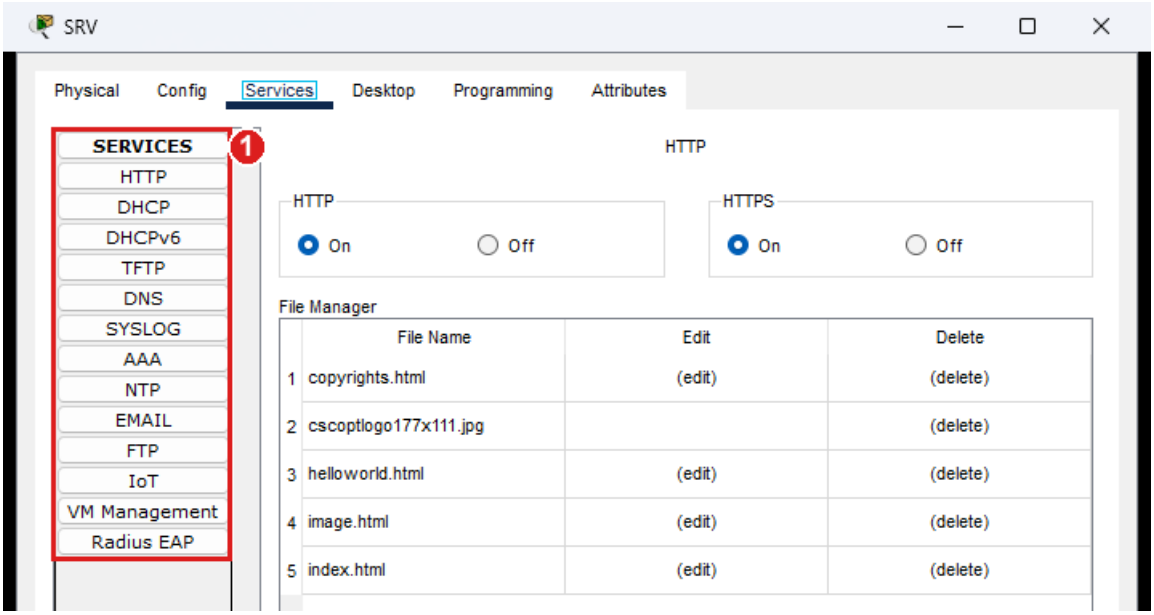
Схема работы 20: клиент в LAN 192.168.10.0/24, сервер SRV в LAN 192.168.50.0/24.

Таблица соединений (проверена построением схемы в Packet Tracer 8.2.1). Порты записаны полными именами, как в списке портов программы.

Устройство	Порт	Устройство	Порт	Кабель
PC-A	`FastEthernet0`	S1	`FastEthernet0/1`	прямой
S1	`GigabitEthernet0/1`	R1	`GigabitEthernet0/0`	прямой
R1	`GigabitEthernet0/1`	S2	`GigabitEthernet0/1`	прямой
S2	`FastEthernet0/1`	SRV	`FastEthernet0`	прямой

Порядок работы

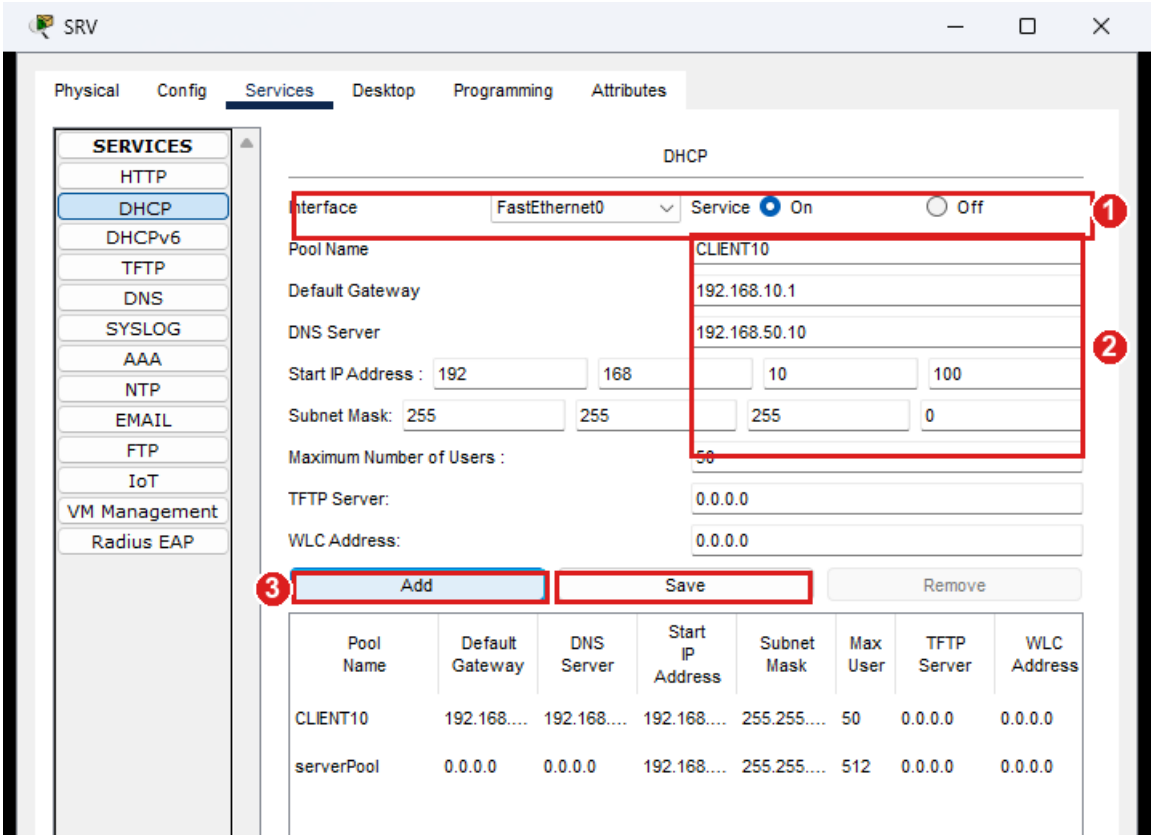
1. Настройте оба интерфейса R1 по образцу работы 08: G0/0 – 192.168.10.1/24, G0/1 – 192.168.50.1/24, на обоих `no shutdown`. На SRV откройте *Desktop / IP Configuration*, оставьте *Static* и введите адрес 192.168.50.10, маску 255.255.255.0 и шлюз 192.168.50.1.
2. На SRV откройте вкладку *Services*. Слева находится список служб: *HTTP, DHCP, DHCPv6, TFTP, DNS, SYSLOG, AAA, NTP, EMAIL, FTP, IoT* и другие. Выберите *DHCP*.



Вкладка Services сервера: 1 – список служб.

3. В окне DHCP переключатель *Service* по умолчанию стоит в положении *Off*: включите *On*. В таблице внизу уже есть пул `serverPool` (сеть 192.168.50.0, 512 адресов); он относится к серверной сети и нашему пулу не мешает, оставьте его. Заполните поля нового пула: *Pool Name* – `CLIENT10`; *Default Gateway* – `192.168.10.1`; *DNS Server* – `192.168.50.10`; *Start IP Address* – четыре отдельных окошка: в первых двух уже стоит 192 и 168, в третье впишите `10`, в четвёртое `100` (щёлкните в окошко, выделите содержимое Ctrl+A и введите число);

Subnet Mask – `255.255.255.0` (стоит по умолчанию); *Maximum Number of Users* – `50`. Нажмите кнопку **Add**: строка CLIENT10 появится в таблице. Кнопка *Save* нужна только для правки уже созданного пула (выберите его строку, измените поля, нажмите Save).

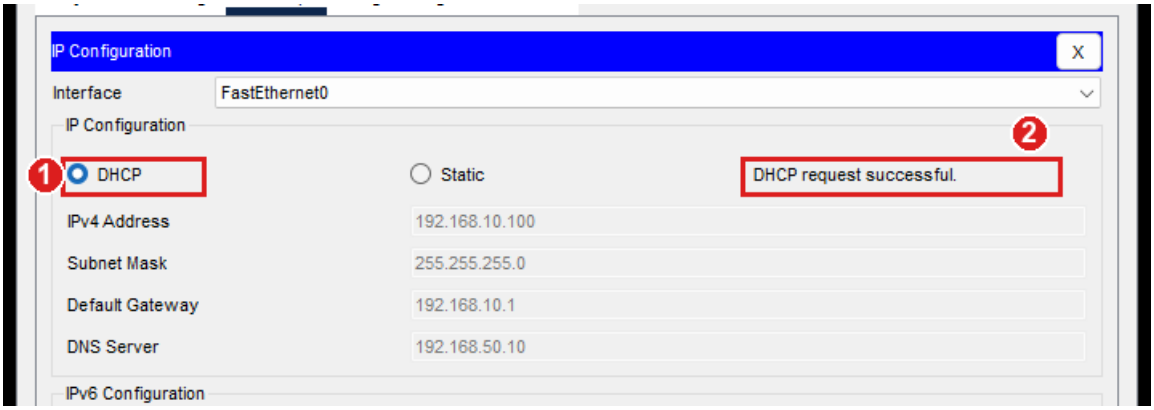


Окно DHCP на сервере: 1 – служба включена, 2 – поля нового пула, 3 – Add и Save.

4. Если в таблице осталась строка с уже созданным пулом, который перекрывает сеть 192.168.10.0/24, исправьте или удалите её (кнопка *Remove*); не оставляйте два противоречивых пула. На R1 настройте relay на клиентском интерфейсе:

```
enable
configure terminal
interface gigabitEthernet 0/0
 ip helper-address 192.168.50.10
exit
end
show running-config
```

Переведите PC-A в DHCP. На втором ПК из той же LAN повторите выдачу. В Simulation оставьте DHCP и проследите участок клиент–relay и relay–сервер. На PC-A откройте *Desktop / IP Configuration* и выберите *DHCP*. В правой части окна появится строка с результатом: «DHCP request successful.» и полученные адрес, маска, шлюз и DNS.



Клиент после переключения на DHCP: 1 – выбран DHCP, 2 – сообщение об успехе. Адрес .100 – первый адрес пула.

Первая попытка сразу после сборки схемы нередко заканчивается сообщением «DHCP failed. APIPA is being used» и адресом 169.254.x.x: запрос ушёл раньше, чем порты коммутаторов перешли в рабочее состояние. Это не ошибка настройки. Подождите (или нажмите Fast Forward Time), переключите ПК на *Static* и снова на *DHCP*.

Проверка результата

PC-A получает .100–.149 сети 192.168.10.0/24, а не адрес серверной LAN. Сначала убедитесь, что SRV достигает R1 и что клиент при временном статическом адресе достигает SRV. Затем проверяйте DHCP. Включённый relay не исправляет выключенную службу или ошибочный pool.

Управляемая неисправность

Удалите helper только с G0/0. Новый клиент не получает корректную аренду, хотя статический клиент продолжает пинговать сервер. Восстановите helper и выполните повторный запрос.

Что сдать и объяснить

Сдайте настройки пула, relay и параметры клиентов. Объясните, зачем DHCP-серверу нужен обратный путь к сети клиентов и как он выбирает нужный пул.

Самостоятельное усложнение. Добавьте LAN 20 через R1 G0/2: 192.168.20.1/24, второй helper и отдельный пул .100–.149. Проверьте отсутствие смешивания адресных пространств.

Лабораторная 21. DNS, веб-сервер и TCP в Simulation

Время: 105 минут. **Подготовка:** 20.

Какую проблему решает работа

Пользователь вводит имя сайта, а сеть работает с адресами. Без DNS имя не превращается в адрес, а без понимания TCP непонятно, почему страница «не открывается»: не найден адрес, не установилось соединение или сервер не ответил. Работа показывает путь запроса по шагам: имя, адрес, TCP-соединение, HTTP-ответ.

Что вы научитесь делать

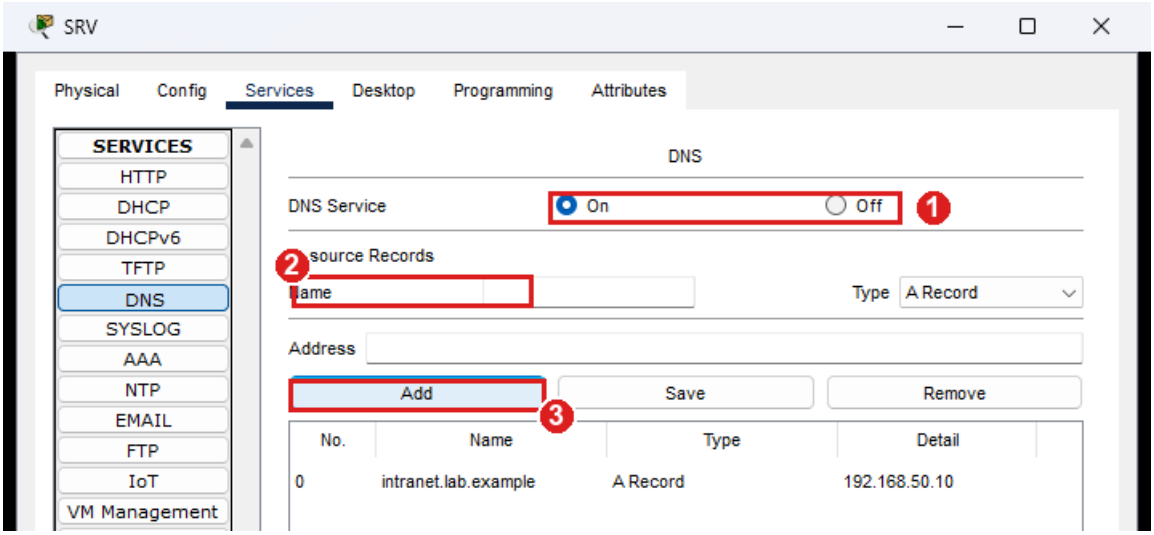
Настроить имя сервера, опубликовать учебную страницу и разделить диагностику DNS, TCP и HTTP.

Стенд и адреса

Используйте 20. SRV 192.168.50.10 предоставляет DNS и HTTP. У клиента DNS должен быть 192.168.50.10. Домен курса – `lab.test`, имя страницы – `intranet.lab.test`; они используются внутри учебного DNS.

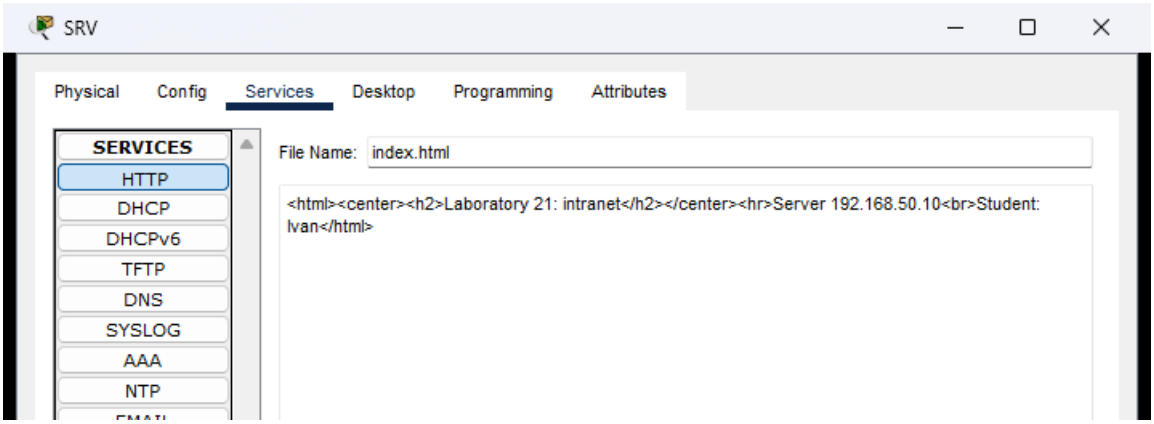
Порядок работы

1. На SRV откройте *Services / DNS*. Переключатель *DNS Service* стоит в положении *Off*: включите *On*. В блоке *Resource Records* введите *Name* – `intranet.lab.test`, *Type* оставьте *A Record*, *Address* – `192.168.50.10` и нажмите **Add**. Запись появится в таблице ниже; кнопка *Save* нужна только для правки существующей записи.



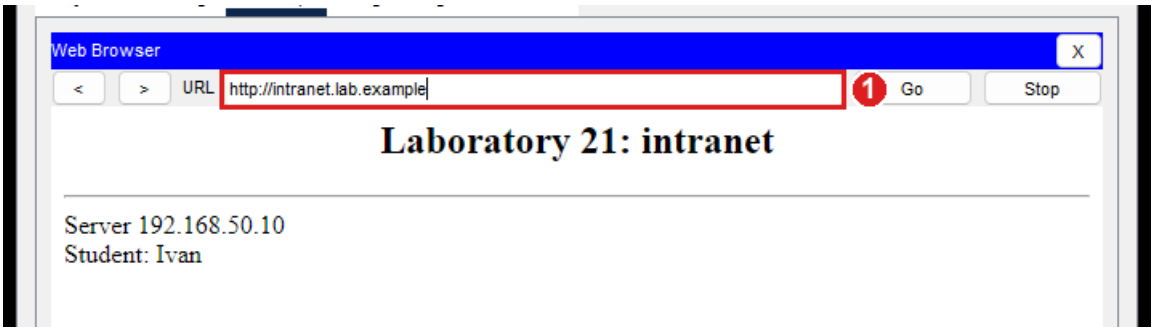
Служба DNS: 1 – включена, 2 – ввод имени и адреса, 3 – кнопка Add.

2. Откройте *Services / HTTP*. Службы HTTP и HTTPS по умолчанию уже включены (*On*). В таблице *File Manager* найдите `index.html` и нажмите *(edit)*. Замените текст страницы: добавьте название лабораторной, адрес сервера и своё имя или псевдоним, нажмите *Save*. Программа спросит: «There is already a file with the same name, overwrite?» – ответьте *Yes*.



Редактор страницы index.html: 1 – кнопка Save.

3. На PC-A откройте *Desktop / Web Browser*, введите в строку *URL* адрес `http://192.168.50.10` и нажмите *Go*, затем `http://intranet.lab.test`. Перед опытом по имени убедитесь, что DNS клиента правильный (`192.168.50.10`: он приходит от DHCP или вводится вручную в *IP Configuration*).



Web Browser на PC-A: страница открыта по имени.

4. В Simulation выберите DNS, TCP и HTTP. Повторите открытие по имени; если разрешение уже закэшировано, используйте новое имя с дополнительной A-записью или новый клиент.
5. Найдите DNS query/response, три шага TCP-handshake и HTTP-запрос. Запишите адреса и порты для каждого этапа.

Почему домен – lab.test, а не lab.test

В исходной редакции курса использовалось имя lab.test. DNS Packet Tracer такое имя разрешает без проблем, но почтовый клиент из работы 22 принимает адреса только с последней частью домена не длиннее четырёх букв: `alice@lab.test` отвергается сообщением «Invalid email address entered». Поэтому во всём курсе используется зарезервированный для тестов домен `lab.test`.

Проверка результата

Оба URL возвращают вашу страницу. DNS-ответ содержит ожидаемый IPv4. TCP соединяет клиента с серверным портом 80; динамический порт клиента в разных опытах может отличаться. Не требуйте фиксированного значения клиентского порта в отчёте.

Управляемая неисправность

Измените только A-запись на свободный адрес 192.168.50.99. При свежем запросе имя ведёт к неверному узлу; открытие по правильному IP продолжает работать. Исправьте запись и исключите влияние кэша повторным опытом.

Что сдать и объяснить

Сдайте страницу, запись DNS и временную шкалу событий. Почему успешный ping сервера не гарантирует, что HTTP включён? Почему правильный DNS не гарантирует TCP-связность?

Самостоятельное усложнение. На втором Server-PT разместите другую страницу по адресу 192.168.50.20. Создайте `help.lab.test` и проверьте, что имена ведут к разным серверам.

Лабораторная 22. FTP и учебная электронная почта

Время: 105 минут. **Подготовка:** 21.

Какую проблему решает работа

В компании нужно передавать файлы и обмениваться почтой между сотрудниками, причём так, чтобы администратор понимал, какие службы и порты за этим стоят. Работа настраивает FTP и учебную электронную почту и показывает, как диагностировать каждую службу и чем эти протоколы небезопасны.

Что вы научитесь делать

Различать служебное соединение, передачу данных и прикладную аутентификацию.

Стенд и адреса

Стенд 21, два ПК-клиента. SRV 192.168.50.10. Для наблюдения используются упрощённые FTP и EMAIL службы Packet Tracer; это демонстрация протоколов, а не рекомендация передавать реальные пароли без шифрования.

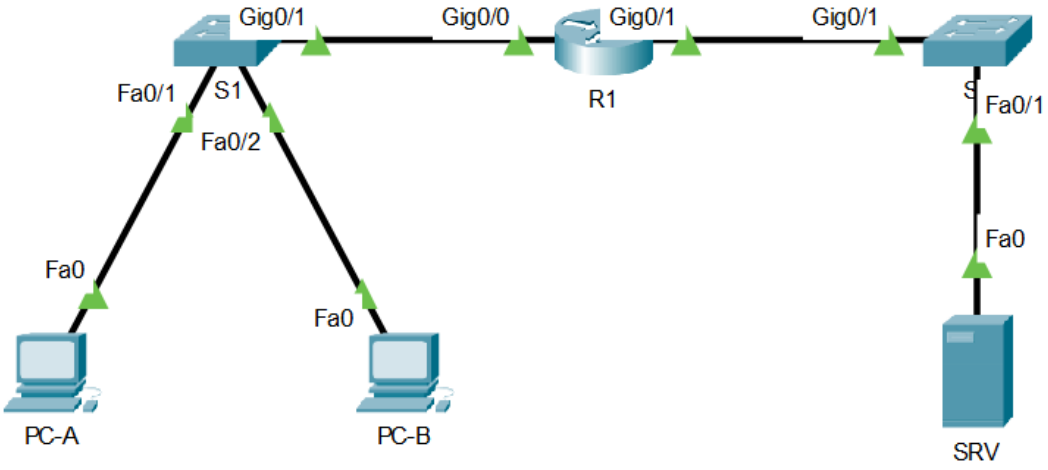


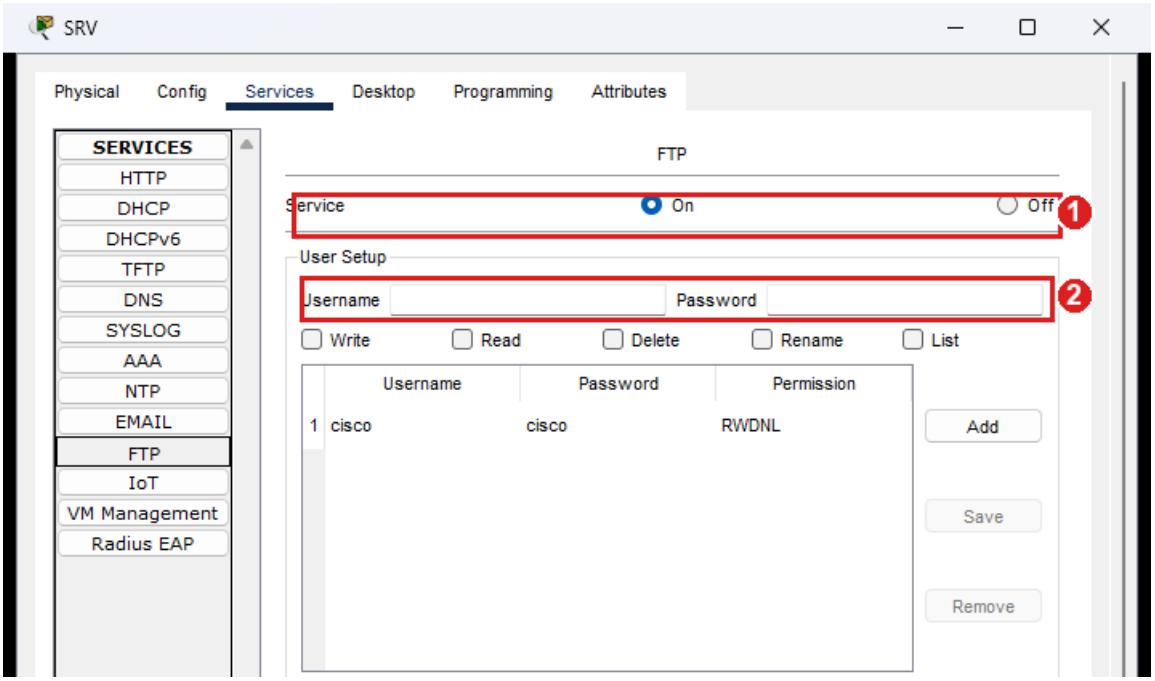
Схема работы 22: два клиента и сервер SRV за маршрутизатором R1.

Таблица соединений (проверена построением схемы в Packet Tracer 8.2.1). Порты записаны полными именами, как в списке портов программы.

Устройство	Порт	Устройство	Порт	Кабель
PC-A	FastEthernet0	S1	FastEthernet0/1	прямой
PC-B	FastEthernet0	S1	FastEthernet0/2	прямой
S1	GigabitEthernet0/1	R1	GigabitEthernet0/0	прямой
R1	GigabitEthernet0/1	S2	GigabitEthernet0/1	прямой
S2	FastEthernet0/1	SRV	FastEthernet0	прямой

Порядок работы

1. На SRV откройте *Services / FTP*. Служба по умолчанию включена. В списке пользователей уже есть учётная запись `cisco` с паролем `cisco` и всеми правами (*RWDNL*); в учебной сети оставьте её только для первого эксперимента, в отчёте её не показывайте. Создайте пользователя: *Username* – `student`, *Password* – `LabFtp42`, поставьте галочки *Read* и *List* (для передачи файла на сервер добавьте *Write* осознанно). Нажмите **Add**.

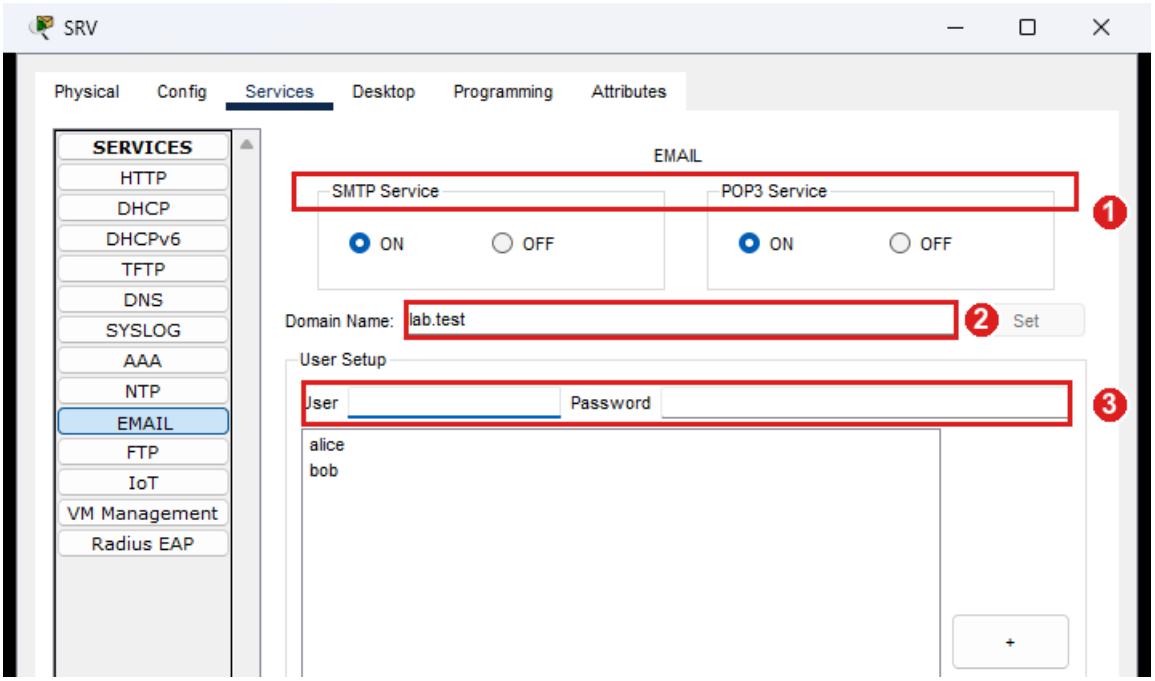


Служба FTP: 1 – включена, 2 – ввод пользователя и прав, нижняя таблица – файлы сервера.

2. На ПК откройте *Desktop / Command Prompt* и введите `ftp 192.168.50.10`. После строки `Username:` введите имя, после `Password:` – пароль (пароль на экране не показывается). Успешный вход выглядит так:

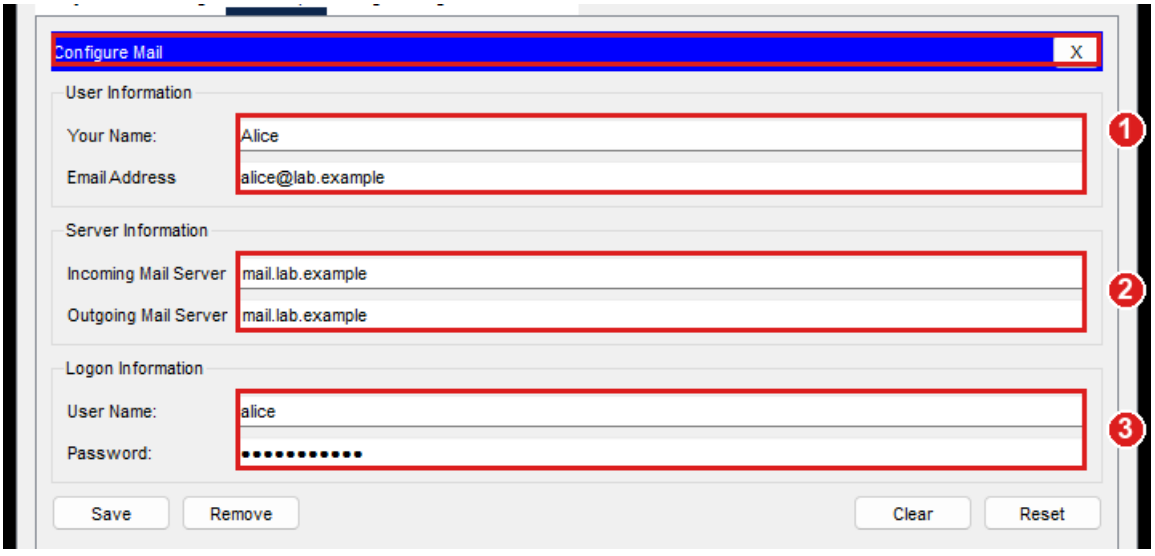
```
C:\>ftp 192.168.50.10
Trying to connect...192.168.50.10
Connected to 192.168.50.10
220- Welcome to PT Ftp server
Username:student
331- Username ok, need password
Password:
230- Logged in
(passive mode On)
ftp>
```

3. В приглашении `ftp>` введите `help` или `?`, затем `dir`: сервер покажет нумерованный список файлов (прошивки Cisco). Выберите существующий файл и выполните `get` с его фактическим именем, например `get asa842-k8.bin`. Выход – `quit`.
4. На SRV откройте *Services / EMAIL*. Службы SMTP и POP3 по умолчанию включены (*ON*). В поле *Domain Name* введите `lab.test` и нажмите *Set*. В блоке *User Setup* введите *User* – `alice`, *Password* – `AlicePass42` и нажмите кнопку **+**; затем повторите для `bob` с паролем `BobPass42`. Имена появятся в списке.



Служба EMAIL: 1 – SMTP и POP3, 2 – домен и кнопка Set, 3 – пользователи.

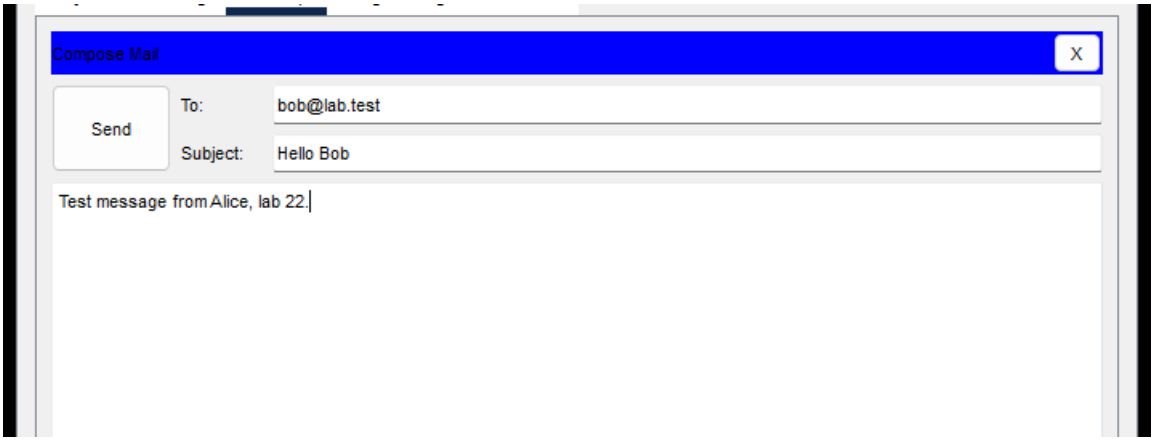
5. В *Services / DNS* (служба включена, как в работе 21) добавьте A-запись `mail.lab.test` с адресом `192.168.50.10` (кнопка Add). У клиентов в поле DNS должен быть `192.168.50.10`.
6. На PC-A откройте *Desktop / Email*. При первом запуске откроется форма *Configure Mail*: *Your Name* – `Alice`; *Email Address* – `alice@lab.test`; *Incoming Mail Server* и *Outgoing Mail Server* – `mail.lab.test`; *User Name* – `alice`; *Password*. Нажмите *Save*. Если появится окно «Invalid email address entered», проверьте домен: почтовый клиент не принимает адреса вида alice@lab.test (последняя часть длиннее четырёх букв).



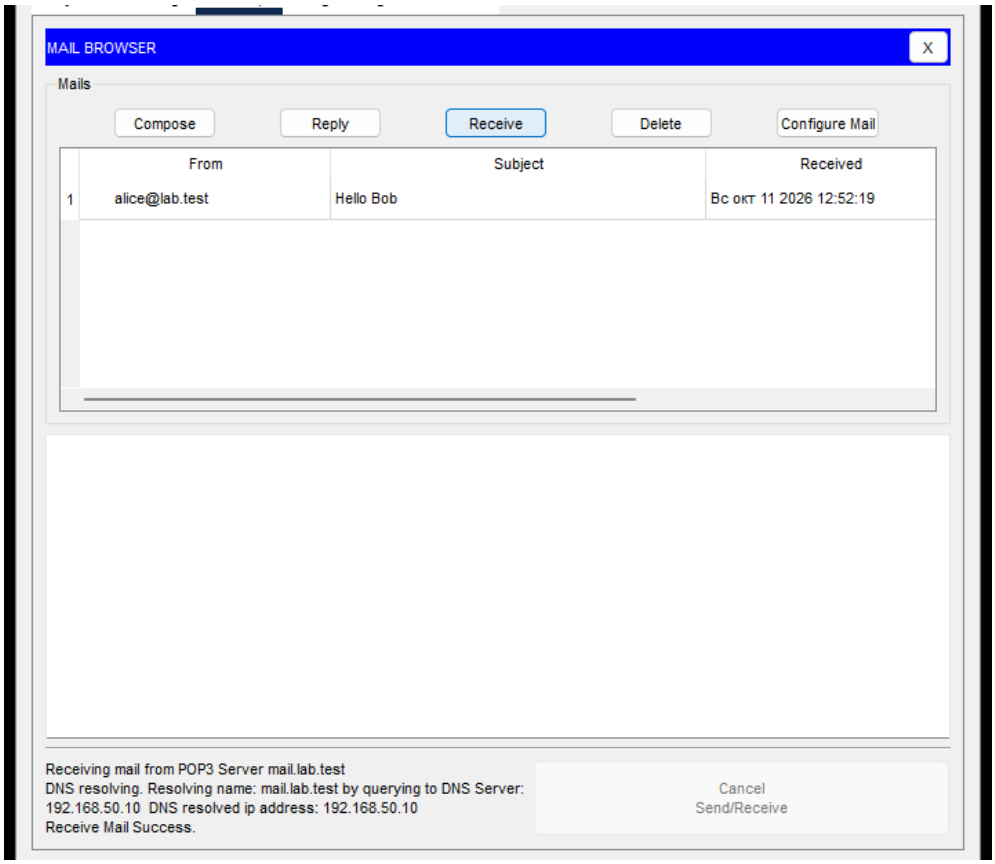
Форма Configure Mail: 1 – заголовок, 2 – серверы, 3 – учётные данные.

7. Закройте форму кнопкой X и откройте *Email* ещё раз: теперь виден *MAIL BROWSER* с кнопками *Compose, Reply, Receive, Delete, Configure Mail*. На PC-B настройте почту для `bob` (`bob@lab.test`) так же.

8. На PC-A нажмите *Compose*, заполните *To* (`bob@lab.test`), *Subject* и текст, нажмите *Send*. В нижней строке окна появятся сообщения о разрешении имени `mail.lab.test` по DNS и об отправке. На PC-B нажмите *Receive*: письмо появится в списке, а внизу будет надпись «Receive Mail Success.». Ответьте Alice кнопкой *Reply*. В Simulation проследите SMTP и POP3 по отдельности.



Окно Compose Mail на PC-A.



PC-B получил письмо: в списке сообщение от alice@lab.test.

Проверка результата

FTP позволяет просмотреть список и получить файл; при правах только чтения не должен разрешать неоговорённую запись. Письмо проходит в обе стороны при корректных настройках серверов и пользователей. SMTP передаёт письмо серверу, POP3 получает его клиентом в этой учебной модели. DNS A-записи

достаточно для клиентов с вручную указанным сервером; опыт не моделирует полноценно всю систему междоменной доставки.

Управляемая неисправность

На Bob укажите неправильный пароль. Подтвердите, что сервер пингуется и имя разрешается, но получение письма отвергается. Исправьте только пароль. Прикладной отказ аутентификации не следует исправлять заменой маршрутов.

Что сдать и объяснить

Сдайте список серверных пользователей без публикации личных паролей, пример письма и журнал FTP. Назовите привычные порты FTP control 21, SMTP 25 и POP3 110; объясните, почему FTP может использовать отдельное соединение для данных.

Самостоятельное усложнение. Ограничьте права второго FTP-пользователя и сравните доступные операции. Если нужен файл для put, используйте существующий локальный файл, показываемый клиентом, либо создайте его через доступный Desktop Text Editor; ориентируйтесь на help текущей модели.

Глава 30

ACL: формулируем и проверяем политику

30.1 Правило первого совпадения

ACL просматривается сверху вниз; действует первое подходящее правило. В конце обычно есть неявный запрет. Standard IPv4 ACL проверяет прежде всего адрес источника; extended ACL может учитывать протокол, источник, назначение и транспортные порты. ACL не работает на интерфейсе только потому, что была создана: её нужно применить с направлением `in` или `out` [6].

Wildcard маска задаёт биты, которые не нужно сравнивать: 0 означает сравнивать, 1 – игнорировать. Для непрерывной подсети /24 wildcard равна 0.0.0.255. Для /27 – 0.0.0.31. В других задачах wildcard может быть непрерывной не всегда, поэтому не считайте её просто альтернативным написанием сетевой маски. `host` означает один адрес, `any` – любой.

Направление выбирается относительно интерфейса маршрутизатора. `in` – пакет вошёл через этот интерфейс; `out` – выходит. Сначала нарисуйте стрелку запроса и ответа, затем выбирайте место фильтра. Проверяйте разрешённые и запрещённые потоки: полный отказ сети не является успешной политикой безопасности.

Лабораторная 23. Standard ACL и место применения

Время: 90 минут. **Подготовка:** 09.

Какую проблему решает работа

Нужно запретить одному отделу доступ к серверу, не отрезав остальных. Правило фильтрации, поставленное не там, где нужно, либо блокирует лишнее, либо ничего не блокирует. Работа решает задачу «где и как поставить standard ACL» и показывает, почему место применения правила так же важно, как его текст.

Что вы научитесь делать

Разрешить доступ в целевую LAN только одной исходной сети, сохранив остальную связность.

Стенд и адреса

Вернитесь к исправной цепочке 09 без резервных линий. Политика: в LAN 30 разрешён трафик из LAN 10; трафик из LAN 20 запрещён. Standard ACL не различает назначения, поэтому применяем её близко к целевой LAN: R3 G0/0, направление out.

Порядок работы

```
enable
configure terminal
ip access-list standard TO-LAN30
 permit 192.168.10.0 0.0.0.255
 deny any
```

```
exit
interface gigabitEthernet 0/0
 ip access-group TO-LAN30 out
exit
end
show access-lists
show ip interface gigabitEthernet 0/0
```

1. Проверьте PC-A (LAN 10) к PC-C (LAN 30): разрешено. PC-B к PC-C: запрещено. PC-A к PC-B: разрешено.
2. После каждого опыта сравните счётчики ACL, если модель их отображает. Тесты генерируют дополнительный ICMP-трафик, поэтому объясняйте изменение, а не требуйте заранее фиксированного числа.
3. Для проверки маршрутов на время снимите только применение ACL: `no ip access-group TO-LAN30 out`. Убедитесь, что исходная связность возвращается, затем верните применение.

Проверка результата

Поведение совпадает с тремя строками матрицы. ACL на выходе в LAN 30 видит источники запросов LAN 10/20. Это правило не является полноценным firewall с состоянием: ответ на инициативный запрос из LAN 30 также рассматривается по его источнику. **Эталонный вывод** после проверок A–C (разрешено) и B–C (запрещено):

```
R3#show access-lists
Standard IP access list TO-LAN30
    10 permit 192.168.10.0 0.0.0.255 (6 match(es))
    20 deny any (8 match(es))
```

При запрещённом обмене PC-B получает от маршрутизатора `Reply from 10.0.23.2: Destination host unreachable` (запрет сообщается явно), а не просто тайм-аут. Числа совпадений у вас будут другими.

Управляемая неисправность

В отдельной копии перенесите эту standard ACL на R1 G0/0 in. Объясните, почему правило с разрешением только LAN 10 там не реализует запрет LAN 20 к LAN 30, и почему выбор места определяет результат.

Что сдать и объяснить

Сдайте матрицу «источник / назначение / прогноз / факт», место ACL и счётчики. Объясните первый матч и неявный запрет.

Самостоятельное усложнение. Добавьте в разрешение один конкретный ПК LAN 20 через `permit host 192.168.20.10` перед deny. Проверьте второй ПК той же LAN и различие результатов.

Лабораторная 24. Extended ACL для гостевого доступа

Время: 120 минут. **Подготовка:** 13, 21, 23.

Какую проблему решает работа

Гостям нужен только веб, а в локальную сеть и на внутренние серверы им нельзя. Standard ACL фильтрует только по адресу источника, а здесь нужна точность по назначению и порту. Работа решает проблему extended ACL: разрешить нужное, запретить остальное и проверить обе стороны политики.

Что вы научитесь делать

Разрешить гостям DNS и веб, запретить доступ к другой внутренней сети и сохранить оговорённые исключения.

Стенд и адреса

Новый стенд: R1 G0/0–S1–GUEST-PC, сеть 192.168.30.0/24, R1 .1; R1 G0/1–S2–SRV, сеть 192.168.50.0/24, R1 .1, SRV .10; R1 G0/2–S3–STAFF-PC, сеть 192.168.10.0/24, R1 .1. ПК .10, шлюзы .1. На SRV включите DNS и HTTP как в 21; DNS гостя – .50.10.

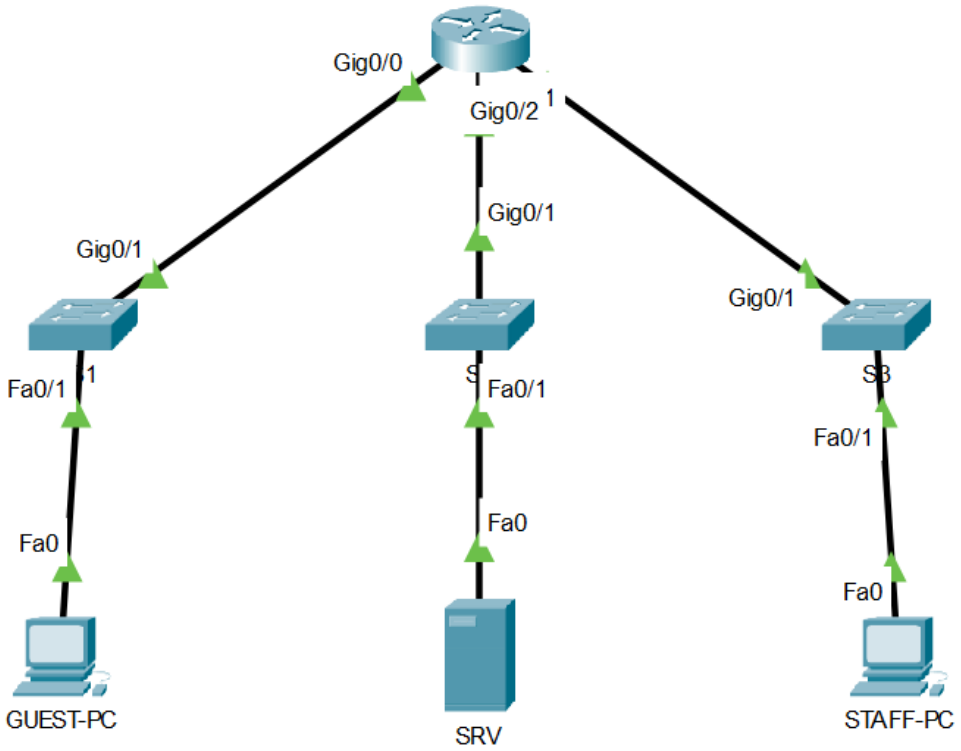


Схема работы 24: три LAN на трёх интерфейсах R1.

Таблица соединений (проверена построением схемы в Packet Tracer 8.2.1). Порты записаны полными именами, как в списке портов программы.

Устройство	Порт	Устройство	Порт	Кабель
GUEST-PC	FastEthernet0	S1	FastEthernet0/1	прямой
S1	GigabitEthernet0/1	R1	GigabitEthernet0/0	прямой
SRV	FastEthernet0	S2	FastEthernet0/1	прямой
S2	GigabitEthernet0/1	R1	GigabitEthernet0/1	прямой
STAFF-PC	FastEthernet0	S3	FastEthernet0/1	прямой
S3	GigabitEthernet0/1	R1	GigabitEthernet0/2	прямой

Порядок работы

Сначала докажите связность без ACL. Затем на R1:

```
enable
configure terminal
ip access-list extended GUEST-IN
 permit udp 192.168.30.0 0.0.0.255 host 192.168.50.10 eq 53
 permit tcp 192.168.30.0 0.0.0.255 host 192.168.50.10 eq 53
 permit tcp 192.168.30.0 0.0.0.255 host 192.168.50.10 eq 80
 permit icmp 192.168.30.0 0.0.0.255 host 192.168.30.1 echo
 deny ip 192.168.30.0 0.0.0.255 192.168.10.0 0.0.0.255
 deny ip any any
exit
interface gigabitEthernet 0/0
 ip access-group GUEST-IN in
exit
end
show access-lists
```

Проверка результата

Гость открывает веб по IP и имени, пингует свой шлюз, но не пингует STAFF-PC и SRV. Запрет ICMP к SRV является ожидаемым: разрешён HTTP, а не все протоколы к серверу. STAFF-PC продолжает работать с сервером, потому что ACL находится на гостевом входе. Ответы сервера выходят к гостю через G0/0 и не проходят эту ACL in в обратном направлении. **Эталонный вывод.** Packet Tracer показывает порты по именам служб (`domain`, `www`):

```
R1#show access-lists
Extended IP access list GUEST-IN
    10 permit udp 192.168.30.0 0.0.0.255 host 192.168.50.10 eq domain
    20 permit tcp 192.168.30.0 0.0.0.255 host 192.168.50.10 eq domain
    30 permit tcp 192.168.30.0 0.0.0.255 host 192.168.50.10 eq www
    40 permit icmp 192.168.30.0 0.0.0.255 host 192.168.30.1 echo
    50 deny ip 192.168.30.0 0.0.0.255 192.168.10.0 0.0.0.255
    60 deny ip any any
```

Ping гостя к собственному шлюзу проходит; ping к STAFF-PC и SRV получает `Destination host unreachable` от 192.168.30.1.

Управляемая неисправность

В отдельной копии поставьте `deny ip any any` перед permit-правилами. Проверьте полную блокировку и счётчик первого правила. Восстановите правильный порядок, лучше удалив и заново создав ACL после временного снятия её применения. Если модель поддерживает sequence numbers, покажите безопасное точечное редактирование.

Что сдать и объяснить

Сдайте минимум шесть тестов: DNS, HTTP по имени, HTTP по IP, ping шлюза, ping STAFF, ping SRV. Для каждого укажите, какое правило должно сработать. Не используйте единственный ping в качестве проверки HTTP-политики.

Самостоятельное усложнение. Разрешите FTP к отдельному серверу в LAN 50, предварительно изучив необходимость управляющего и канала данных. Сформулируйте политику до команд; простого permit TCP/21 может быть недостаточно для полного обмена.

Глава 31

NAT и учебный провайдер

31.1 Адреса до и после трансляции

Static NAT постоянно сопоставляет один внутренний адрес одному внешнему. PAT, или NAT overload, позволяет нескольким внутренним соединениям использовать один внешний адрес, различая их транспортные порты или идентификаторы. Inside local – исходный внутренний адрес; inside global – представление этого адреса снаружи. Таблица трансляций помогает увидеть сопоставление, но сама трансляция не создаёт маршруты.

Пометки `ip nat inside` и `ip nat outside` определяют стороны преобразования. ACL, используемая в `ip nat inside source list`, выбирает адреса для трансляции; пока её не применили через `ip access-group`, это не интерфейсный фильтр. NAT не следует считать полноценной политикой безопасности [7].

Лабораторная 25. Static NAT и публикация сервера

Время: 120 минут. **Подготовка:** 09, 21.

Какую проблему решает работа

Сервер во внутренней сети имеет частный адрес, который в Интернете не маршрутизируется, но клиентам снаружи нужно открывать его сайт. Работа решает задачу публикации: static NAT связывает внутренний адрес с внешним, и вы проверяете таблицу трансляций и доступность сервера.

Что вы научитесь делать

Опубликовать внутренний веб-сервер через отдельный внешний адрес и проверить преобразование.

Стенд и адреса

R-EDGE G0/0–S1–SRV: 192.168.10.0/24, EDGE .1, SRV .100, шлюз .1. R-EDGE G0/1 203.0.113.2/24–ISP G0/0 203.0.113.1/24. ISP G0/1 198.51.100.1/24–S2–OUT-PC 198.51.100.10/24, шлюз .1. SRV предоставляет HTTP. Публикуемый адрес – 203.0.113.100.



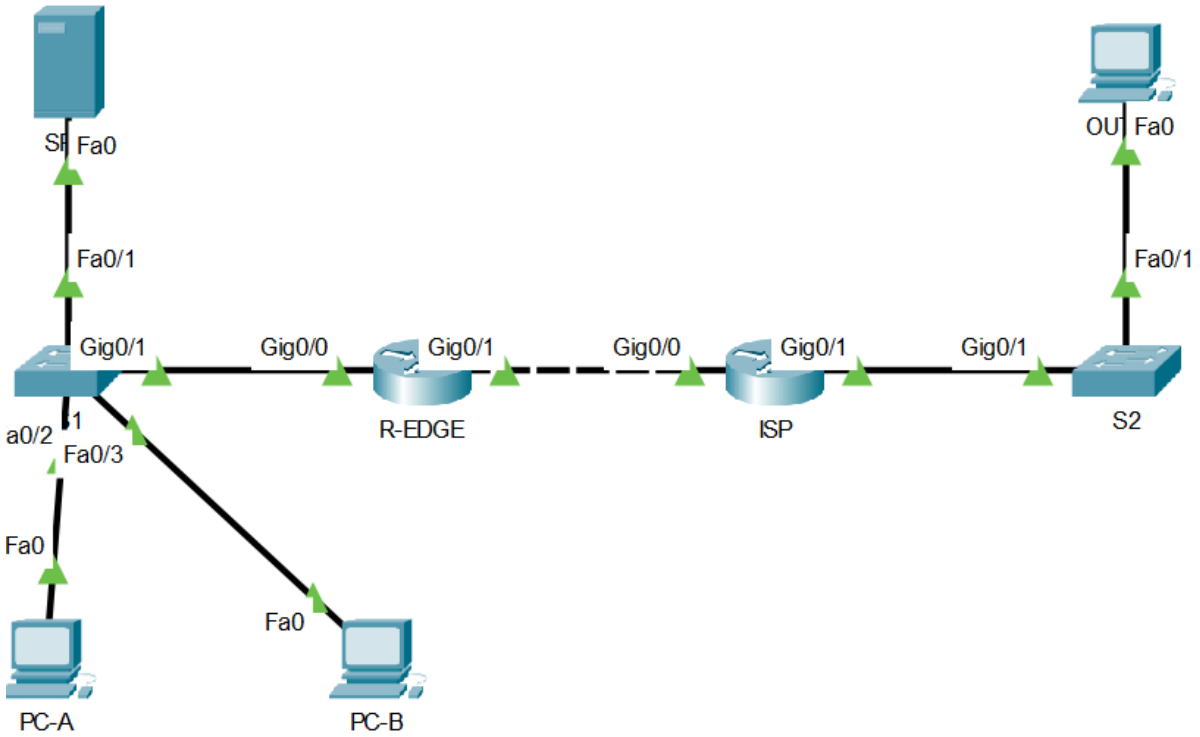


Схема работы 25 (и 26): внутренняя LAN, R-EDGE, ISP и внешний компьютер OUT-PC.

Таблица соединений (проверена построением схемы в Packet Tracer 8.2.1). Порты записаны полными именами, как в списке портов программы.

Устройство	Порт	Устройство	Порт	Кабель
SRV	FastEthernet0	S1	FastEthernet0/1	прямой
PC-A	FastEthernet0	S1	FastEthernet0/2	прямой
PC-B	FastEthernet0	S1	FastEthernet0/3	прямой
S1	GigabitEthernet0/1	R-EDGE	GigabitEthernet0/0	прямой
R-EDGE	GigabitEthernet0/1	ISP	GigabitEthernet0/0	перекрёстный (crossover)
ISP	GigabitEthernet0/1	S2	GigabitEthernet0/1	прямой
OUT-PC	FastEthernet0	S2	FastEthernet0/1	прямой

Порядок работы

Настройте адреса и включите все интерфейсы. На EDGE в глобальном режиме:

```
interface gigabitEthernet 0/0
 ip nat inside
exit
interface gigabitEthernet 0/1
 ip nat outside
exit
ip nat inside source static 192.168.10.100 203.0.113.100
ip route 0.0.0.0 0.0.0.0 203.0.113.1
end
show ip nat translations
```

На ISP добавьте маршрут именно к опубликованному адресу:

```
ip route 203.0.113.100 255.255.255.255 203.0.113.2
```

Этот /32 задаёт явный следующий узел и не требует полагаться на proxy ARP для адреса .100. На OUT-PC откройте `http://203.0.113.100`; на SRV или внутреннем ПК сначала убедитесь, что HTTP работает локально.

Проверка результата

Внешний клиент получает страницу внутреннего SRV. В NAT-таблице видно постоянное сопоставление 192.168.10.100–203.0.113.100. В Simulation найдите изменение destination IP входящего запроса и source IP ответа. ISP не нужен маршрут к 192.168.10.0/24 для этого опубликованного обмена. **Эталонный вывод** после ping внутреннего SRV из внешнего OUT-PC на 203.0.113.100:

```
R-EDGE#show ip nat translations
Pro  Inside global     Inside local       Outside local      Outside global
icmp 203.0.113.100:2   192.168.10.100:2   198.51.100.10:2    198.51.100.10:2
---  203.0.113.100     192.168.10.100     ---                ---
```

Нижняя строка со знаками `---` – постоянное статическое соответствие. Ответы SRV идут с TTL=126: между OUT-PC и SRV два маршрутизатора.

Управляемая неисправность

Удалите default route на EDGE. Приём запроса к серверу может происходить, но ответ клиенту 198.51.100.10 теряет путь. Верните маршрут и подтвердите роль обратного направления.

Что сдать и объяснить

Сдайте схему, NAT-таблицу, маршруты EDGE/ISP и заголовки до/после трансляции. Назовите inside local и inside global в вашем стенде.

Самостоятельное усложнение. Добавьте второй внутренний сервер .101 и публикацию 203.0.113.101. На ISP добавьте отдельный /32. Проверьте два разных сайта.

Лабораторная 26. PAT: несколько клиентов за одним адресом

Время: 105 минут. **Подготовка:** 25.

Какую проблему решает работа

Публичных адресов у организации мало, а компьютеров много, и все они выходят в Интернет через один адрес. Работа решает проблему экономии адресов: PAT различает клиентов по номерам портов, а вы учитесь читать таблицу трансляций и объяснять, откуда берутся ответы.

Что вы научитесь делать

Разрешить двум внутренним клиентам пользоваться внешней службой через адрес outside-интерфейса.

Стенд и адреса

Копия 25. Внутри добавьте PC-A 192.168.10.10 и PC-B .20, шлюзы .1. Снаружи добавьте OUT-SRV 198.51.100.100/24, шлюз 198.51.100.1, HTTP On. Статическую публикацию SRV можно оставить: она не совпадает с outside-адресом EDGE .2.

Порядок работы

На EDGE:

```
enable
configure terminal
access-list 1 permit 192.168.10.0 0.0.0.255
ip nat inside source list 1 interface gigabitEthernet 0/1 overload
end
show ip nat statistics
```

Проверьте стороны inside/outside и default route, унаследованные из 25. На обоих ПК откройте `http://198.51.100.100`; затем сразу выведите `show ip nat translations`. Для сохранения записей повторите обращения, если динамические записи уже истекли.

Проверка результата

Оба клиента получают страницу. Динамические записи используют inside global 203.0.113.2 с различающимися параметрами сессий. ISP не имеет обратного маршрута к частным адресам клиентов, но возвращает ответ к внешнему адресу EDGE. Внутри исходные адреса ПК не меняются. **Эталонный вывод** после ping из PC-A и PC-B к 198.51.100.10:

```
R-EDGE#show ip nat translations
Pro  Inside global     Inside local       Outside local      Outside global
icmp 203.0.113.2:1024  192.168.10.20:1    198.51.100.10:1    198.51.100.10:1024
icmp 203.0.113.2:1     192.168.10.10:1    198.51.100.10:1    198.51.100.10:1
---  203.0.113.100     192.168.10.100     ---                ---
R-EDGE#show ip nat statistics
Total translations: 17 (1 static, 16 dynamic, 16 extended)
Outside Interfaces: GigabitEthernet0/1
Inside Interfaces: GigabitEthernet0/0
```

У всех сеансов один внешний адрес `203.0.113.2`, а различаются номера в поле порта. Листинг этой работы уже начинается с `enable` и `configure terminal`: если вы вводите его в режиме глобальной конфигурации, эти две строки дадут сообщение об ошибке, на остальные команды оно не влияет.

Управляемая неисправность

Измените ACL 1 так, чтобы она выбирала только host 192.168.10.10. Создайте новые соединения после удаления или истечения старых динамических трансляций; если `clear ip nat translation *` поддерживается, применяйте её только в этом учебном стенде. PC-A должен работать, PC-B – нет. Верните исходный выбор подсети.

Что сдать и объяснить

Сдайте две записи одновременных сеансов и объясните, как различаются ответы двум клиентам с одним внешним адресом. Почему добавление overload не исправляет отсутствующий маршрут по умолчанию?

Самостоятельное усложнение. Добавьте DNS на OUT-SRV, настройте имя и параметры клиентов. Проверьте отдельно трансляцию DNS и TCP/HTTP; при ACL-фильтрации разрешите нужные протоколы явно.

Часть IV

Динамические маршруты, IPv6 и беспроводные сети

Глава 32

OSPF и альтернативные протоколы

32.1 Соседство, база топологии и маршрут

OSPF – протокол состояния каналов. Маршрутизаторы находят соседей, обмениваются сведениями о топологии и вычисляют пути по стоимости. Для нашего single-area стенда все каналы принадлежат area 0. Router ID должен быть уникален; это идентификатор процесса, который не обязан совпадать с адресом физического порта.

Команда `network` в OSPF выбирает локальные интерфейсы по адресу и wildcard и назначает им область. Она не означает «создать произвольную сеть в таблице». Passive-interface не формирует соседство на пользовательском сегменте, но не запрещает рекламу соответствующего префикса. Номер процесса OSPF локален: на соседях он может различаться; номер области на общем канале должен согласовываться.

В Ethernet-сегменте могут выбираться DR и BDR. Состояние FULL ожидается на наших двухузловых маршрутизаторных каналах после сходимости. В большой общей Ethernet-сети не всякая пара DROTHER обязана иметь FULL между собой: состояние 2-WAY может быть штатным. Не обобщайте критерий маленькой лабораторной на все OSPF-топологии.

32.2 Почему маршрутизатор не стал соседом

Начните с up/up и адресов одной подсети. Затем проверьте area, маски, таймеры и passive-interface. Убедитесь, что router ID уникальны. Если соседство есть, но LAN не рекламируется, проверьте выбор её интерфейса командой network. Если маршрут появился, но ping не работает, проверьте шлюзы ПК, ACL и обратную передачу.

Лабораторная 27. Single-area OSPF

Время: 120 минут. **Подготовка:** 09.

Какую проблему решает работа

В сети из десяти маршрутизаторов прописывать маршруты вручную невозможно, а при каждом изменении схемы — тем более. Работа решает проблему автоматизации: протокол OSPF сам находит соседей, обменивается сведениями о сетях и строит таблицы маршрутов.

Что вы научитесь делать

Заменить статические пути динамическими и подтвердить соседства и рекламу LAN.

Стенд и адреса

Исправная цепочка 09: те же адреса и порты. Сначала сохраните отдельную копию. Удалите шесть статических маршрутов к удалённым LAN командами `no ip route`, повторяя точные строки 09 с префиксом no. Уберите и default route,

если выполняли усложнение 09.

Порядок работы

На R1:

```
router ospf 1
 router-id 1.1.1.1
 network 192.168.10.0 0.0.0.255 area 0
 network 10.0.12.0 0.0.0.3 area 0
 passive-interface gigabitEthernet 0/0
exit
```

На R2:

```
router ospf 1
 router-id 2.2.2.2
 network 192.168.20.0 0.0.0.255 area 0
 network 10.0.12.0 0.0.0.3 area 0
 network 10.0.23.0 0.0.0.3 area 0
 passive-interface gigabitEthernet 0/0
exit
```

На R3:

```
router ospf 1
 router-id 3.3.3.3
 network 192.168.30.0 0.0.0.255 area 0
 network 10.0.23.0 0.0.0.3 area 0
 passive-interface gigabitEthernet 0/0
exit
```

Все команды групп вводятся из глобальной конфигурации. После end:

```
show ip ospf neighbor
show ip ospf interface
show ip protocols
show ip route
```

Проверка результата

R1 и R3 имеют по одному соседу, R2 – двух. Удалённые LAN имеют код O; для основной OSPF-информации дистанция обычно 110. Все ПК связаны. Учитывайте, что непосредственно подключённая LAN остаётся C на своём маршрутизаторе, хотя рекламируется остальным. **Эталонный вывод** (через 1–2 минуты модельного времени):

```
R2#show ip ospf neighbor
Neighbor ID     Pri   State           Dead Time   Address         Interface
3.3.3.3           1   FULL/DR         00:00:30    10.0.23.2
GigabitEthernet0/2
1.1.1.1           1   FULL/BDR        00:00:30    10.0.12.1
GigabitEthernet0/1

R1#show ip route ospf
O       10.0.23.0 [110/2] via 10.0.12.2, 00:02:05, GigabitEthernet0/1
O    192.168.20.0 [110/2] via 10.0.12.2, 00:02:15, GigabitEthernet0/1
O    192.168.30.0 [110/3] via 10.0.12.2, 00:02:05, GigabitEthernet0/1
```

Какой из двух соседей станет DR на канале, зависит от Router ID и порядка включения; важно состояние FULL.

Управляемая неисправность

На R2 сделайте G0/2 passive. Соседство с R3 исчезнет после обработки изменения; маршрут к LAN 30 у R1 исчезнет, если другого пути нет. Выполните `no passive-interface gigabitEthernet 0/2`, дождитесь восстановления.

Что сдать и объяснить

Сдайте соседства, протоколы и удалённые O-маршруты. Почему одинаковый номер процесса на всех трёх маршрутизаторах удобен, но не обязателен?

Самостоятельное усложнение. На одном краю используйте OSPF process 7 вместо 1, предварительно удалив старый процесс. Сохраните тот же router ID, области и сети; подтвердите восстановление соседства.

Лабораторная 28. Стоимость OSPF и отказ удалённой линии

Время: 105 минут. **Подготовка:** 27, 10.

Какую проблему решает работа

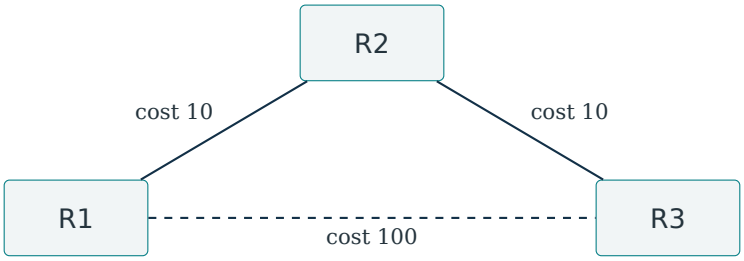
OSPF выбирает путь по стоимости, и при отказе линии трафик должен уйти на запасной путь, но не обязательно туда, куда ожидает администратор. Работа показывает, как стоимость влияет на выбор маршрута и как сеть перестраивается после отказа удалённой линии.

Что вы научитесь делать

Управлять выбором пути и наблюдать распространение сведений об отказе.

Стенд и адреса

Копия 27. Добавьте R1 G0/2–R3 G0/2, адреса 10.0.13.1 и .2/30. На R1 и R3 включите эту сеть в OSPF area 0. Статических маршрутов к LAN быть не должно.



Порядок работы

1. На обоих концах прямой линии R1–R3 задайте `ip ospf cost 100`. На четырёх интерфейсах каналов R1–R2 и R2–R3 задайте `ip ospf cost 10`. Стоимость задаётся в интерфейсной конфигурации.
2. Посмотрите `show ip ospf interface` и маршрут к LAN 30 на R1. При одинаковой стоимости последнего LAN-перехода путь через R2 дешевле прямого.
3. Проверьте tracert PC-A–PC-C. Выключите только R2 G0/2: для R1 отказ находится дальше непосредственного соседа R2.
4. Наблюдайте изменение таблицы R1 и переключение на канал к R3. Запишите потери во время перехода и состояние после сходимости.
5. Включите G0/2 на R2 и проверьте возврат более дешёвого пути.

Проверка результата

В отличие от простого статического next hop в работе 10, OSPF распространяет изменение топологии и R1 может переключиться, даже сохраняя физическую связь с R2. На R1 после сходимости next hop к LAN 30 – 10.0.13.2. Ответный путь проверяйте отдельно на R3. **Эталонный вывод.** Стоимости 10 на каналах через R2 и 100 на прямом канале; до отказа и после выключения R2 G0/2 (на R1):

```
R1#show ip route ospf
O       10.0.23.0 [110/20] via 10.0.12.2, 00:02:50, GigabitEthernet0/1
O    192.168.20.0 [110/11] via 10.0.12.2, 00:02:50, GigabitEthernet0/1
O    192.168.30.0 [110/21] via 10.0.12.2, 00:02:50, GigabitEthernet0/1
   ... shutdown на R2 G0/2, ожидание сходимости ...
R1#show ip route ospf
O    192.168.20.0 [110/11] via 10.0.12.2, 00:05:50, GigabitEthernet0/1
O    192.168.30.0 [110/101] via 10.0.13.2, 00:02:55, GigabitEthernet0/2
```

Метрика 21 = 10 + 10 + 1 (стоимость LAN-интерфейса R3 Gig0/0); метрика 101 = 100 + 1. После отказа tracert идёт через 10.0.13.2 (TTL=126). Первый ping серии после переключения теряет один запрос.

Управляемая неисправность

Уберите OSPF network 10.0.13.0/30 на одном конце. Физическая линия остаётся, но резервного OSPF-соседства нет. Повторите удалённый отказ и объясните потерю связности; восстановите включение сети.

Что сдать и объяснить

Сдайте три маршрута: до отказа, во время переключения и после восстановления. Укажите стоимость выбранного пути из фактического вывода, а не из предполагаемой скорости кабеля.

Самостоятельное усложнение. Сделайте прямой путь дешевле: стоимость 5 на обоих концах. Подтвердите изменение next hop без отказа линий. Сравнивайте суммарные стоимости, а не только одну величину.

Лабораторная 29. Две области OSPF

Время: 105 минут. **Подготовка:** 27.

Какую проблему решает работа

В большой сети все маршрутизаторы в одной области получают слишком много сведений и тратят ресурсы на пересчёт при любом изменении. Работа решает проблему масштабирования: деление сети на области OSPF и граничный маршрутизатор между ними.

Что вы научитесь делать

Выделить backbone и вторую область и найти межобластные маршруты.

Стенд и адреса

Возьмите цепочку 27 без прямой линии R1–R3. R1 LAN 10 и канал R1–R2 относятся к area 0; R2 LAN 20 – area 0. Канал R2–R3 и R3 LAN 30 – area 10. R2 становится ABR.

Порядок работы

На R2 в OSPF-процессе измените назначение сети 10.0.23.0/30:

```
no network 10.0.23.0 0.0.0.3 area 0
network 10.0.23.0 0.0.0.3 area 10
```

На R3 измените обе строки:

```
no network 10.0.23.0 0.0.0.3 area 0
no network 192.168.30.0 0.0.0.255 area 0
network 10.0.23.0 0.0.0.3 area 10
network 192.168.30.0 0.0.0.255 area 10
```

На R1 ничего менять не требуется. Проверьте `show ip ospf`, `show ip ospf neighbor` и `show ip route`; при поддержке модели изучите `show ip ospf database`.

Проверка результата

Соседства восстановлены: общая область на каждой паре согласована. На R1 LAN 30 приходит как O IA; на R3 LAN 10 и LAN 20 – как O IA. R2 связан и с backbone, и с area 10. В этой простой схеме не нужны virtual link, redistribution и суммаризация. **Эталонный вывод** (на R1 после смены областей и сходимости):

```
R1#show ip route ospf
O IA    10.0.23.0 [110/20] via 10.0.12.2, 00:04:04, GigabitEthernet0/1
O    192.168.20.0 [110/11] via 10.0.12.2, 00:25:00, GigabitEthernet0/1
O IA 192.168.30.0 [110/21] via 10.0.12.2, 00:04:04, GigabitEthernet0/1
```

Если после ввода команд на R2 или R3 соседство не восстановилось, проверьте, что команды действительно выполнены в режиме `router ospf 1` (приглашение `(config-router)##`) и что интерфейс показывает новую область: `show ip ospf interface gigabitEthernet 0/2` должна содержать `Area 10`. Пока на одном конце остаётся прежняя область, соседство не образуется.

Управляемая неисправность

Измените область только на R3 для канала 10.0.23.0/30 обратно на 0. На R2 оставьте 10. Соседство на общем канале не формируется. Исправьте согласование областей.

Что сдать и объяснить

Сдайте рисунок областей, указание ABR и примеры O/O IA. Почему область выбирается для интерфейса, а не для произвольного имени устройства?

Самостоятельное усложнение. Добавьте ещё одну LAN к R3 в area 10. Проверьте её появление как межобластного префикса на R1. Суммаризацию исследуйте только после сохранения обычных маршрутов и проверки точного диапазона.

Лабораторная 30. Сравнение EIGRP и RIPv2

Время: 120–150 минут. **Подготовка:** 27.

Какую проблему решает работа

Выбор протокола маршрутизации — инженерное решение: у EIGRP и RIPv2 разные метрики, скорость реакции на отказ и ограничения. Работа сравнивает их на одной схеме, чтобы вы могли обосновать выбор по наблюдениям, а не по привычке.

Что вы научитесь делать

Увидеть разные источники маршрутов на одной топологии и отличать их метрики.

Стенд и адреса

Две отдельные копии цепочки 09 с настроенными интерфейсами и без статических маршрутов. В этих копиях удалите прежний OSPF через `no router ospf 1`, если он присутствует. Не смешивайте протоколы и не делайте redistribution в этом опыте.

Порядок работы

В копии EIGRP на R1:

```
router eigrp 100
 no auto-summary
 network 192.168.10.0 0.0.0.255
 network 10.0.12.0 0.0.0.3
 passive-interface gigabitEthernet 0/0
exit
```

На R2 используйте тот же AS 100, добавив 192.168.20.0/24, 10.0.12.0/30 и 10.0.23.0/30 с соответствующими wildcard. На R3 – 192.168.30.0/24 и 10.0.23.0/30. На всех оставьте LAN G0/0 passive. Проверьте `show ip eigrp neighbors`, `show ip protocols` и D-маршруты.

В копии RIP на R1:

```
router rip
 version 2
 no auto-summary
 network 10.0.0.0
 network 192.168.10.0
 passive-interface gigabitEthernet 0/0
exit
```

На R2 вместо 192.168.10.0 укажите 192.168.20.0, на R3 – 192.168.30.0; строка network 10.0.0.0 остаётся у всех. В классическом RIP network задаётся по основным сетям, без wildcard как у OSPF. Проверьте `show ip protocols` и R-маршруты после ожидания обновлений.

Проверка результата

В обеих копиях LAN связаны. EIGRP использует собственную составную метрику, RIP – число переходов; числа метрик разных протоколов напрямую не сравнивают. Обычно внутренний EIGRP имеет дистанцию 90, RIP – 120. В этих опытах нет борьбы источников: каждый запускается в отдельной копии. **Эталонный вывод.** Таблицы на R1 для EIGRP и RIP (дистанции 90 и 120):

```
R1#show ip eigrp neighbors
H   Address         Interface      Hold Uptime    SRTT   RTO   Q   Seq
0   10.0.12.2       Gig0/1         10   00:04:29  40     1000  0   5
R1#show ip route eigrp
D       10.0.23.0/30 [90/3072] via 10.0.12.2, 00:04:30, GigabitEthernet0/1
D    192.168.20.0/24 [90/3072] via 10.0.12.2, 00:04:30, GigabitEthernet0/1
D    192.168.30.0/24 [90/3328] via 10.0.12.2, 00:04:30, GigabitEthernet0/1

R1#show ip route rip
R       10.0.23.0/30 [120/1] via 10.0.12.2, 00:00:24, GigabitEthernet0/1
R    192.168.20.0/24 [120/1] via 10.0.12.2, 00:00:24, GigabitEthernet0/1
R    192.168.30.0/24 [120/2] via 10.0.12.2, 00:00:24, GigabitEthernet0/1
```

Метрика RIP – число переходов (1 и 2), EIGRP – составное число (3072, 3328).

Управляемая неисправность

В EIGRP измените AS только на R3 на 200. Соседство с R2 не возникает; верните 100. В RIP-копии удалите network для LAN 30, сохранив транзитную: R3 участвует в обмене, но LAN 30 не объявляется.

Что сдать и объяснить

Составьте таблицу OSPF/EIGRP/RIP: код маршрута, механизм обмена, смысл метрики, команда проверки. Объясните, почему RIP с ограничением числа переходов подходит не для любой большой сети.

Самостоятельное усложнение. Сравните наблюдаемое восстановление после выключения линии. Не делайте универсальный рейтинг протоколов по одному запуску симулятора: важны топология, таймеры и реализованная модель.

Глава 33

IPv6 и параллельная работа стеков

33.1 Адреса и Neighbor Discovery

IPv6 использует 128 бит, записанных шестнадцатеричными группами. Начальные нули группы можно опускать; последовательность нулевых групп сокращается :: не более одного раза. Например, 2001:db8:10:0:0:0:0:1 превращается в 2001:db8:10::1. Диапазон 2001:db8::/32 зарезервирован для документации; используем его в стендах.

В типичной пользовательской LAN IPv6 применяется /64. У IPv6 нет широковещательного broadcast как у IPv4. Neighbor Discovery использует ICMPv6 для поиска соседей и маршрутизаторов. Link-local адреса действуют на одном канале; одинаковый fe80::1 на разных независимых LAN допустим. При маршруте через link-local next hop необходимо учитывать выходной интерфейс.

SLAAC позволяет клиенту сформировать адрес на основе Router Advertisement. Статический IPv6 и SLAAC – разные способы настройки узла. Dual stack означает одновременную работу IPv4 и IPv6, а не автоматический перевод одного протокола в другой. ACL IPv4 не следует считать политикой для IPv6.

Лабораторная 31. IPv6: две LAN и SLAAC

Время: 105 минут. **Подготовка:** 08.

Какую проблему решает работа

Адресов IPv4 не хватает, сети постепенно переходят на IPv6, но он должен работать вместе со старым протоколом. Работа показывает, как компьютеры получают адрес автоматически (SLAAC) и как две локальные сети связываются по IPv6.

Что вы научитесь делать

Назначить IPv6-адреса, включить пересылку и увидеть Neighbor Discovery вместо ARP.

Стенд и адреса

R1 2911 между двумя 2960 как в 08. G0/0: 2001:db8:10::1/64; G0/1: 2001:db8:20::1/64. На каждом интерфейсе link-local fe80::1. PC-A: 2001:db8:10::10/64; PC-B: 2001:db8:20::10/64; IPv6-шлюз каждого – fe80::1 на его LAN.

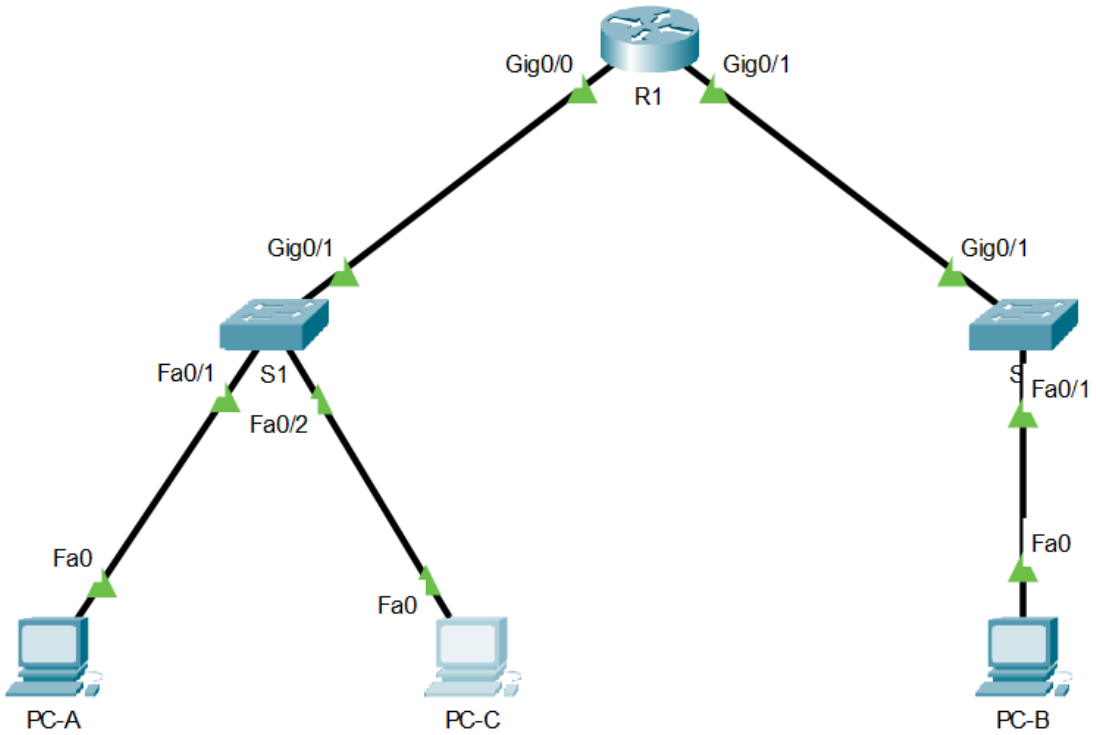


Схема работы 31: две LAN и компьютер PC-C для проверки SLAAC.

Таблица соединений (проверена построением схемы в Packet Tracer 8.2.1). Порты записаны полными именами, как в списке портов программы.

Устройство	Порт	Устройство	Порт	Кабель
PC-A	FastEthernet0	S1	FastEthernet0/1	прямой
PC-C	FastEthernet0	S1	FastEthernet0/2	прямой
S1	GigabitEthernet0/1	R1	GigabitEthernet0/0	прямой
R1	GigabitEthernet0/1	S2	GigabitEthernet0/1	прямой
S2	FastEthernet0/1	PC-B	FastEthernet0	прямой

Порядок работы

```
enable
configure terminal
ipv6 unicast-routing
interface gigabitEthernet 0/0
 ipv6 address 2001:db8:10::1/64
 ipv6 address fe80::1 link-local
 no shutdown
exit
interface gigabitEthernet 0/1
 ipv6 address 2001:db8:20::1/64
 ipv6 address fe80::1 link-local
 no shutdown
exit
end
show ipv6 interface brief
show ipv6 route
show ipv6 neighbors
```

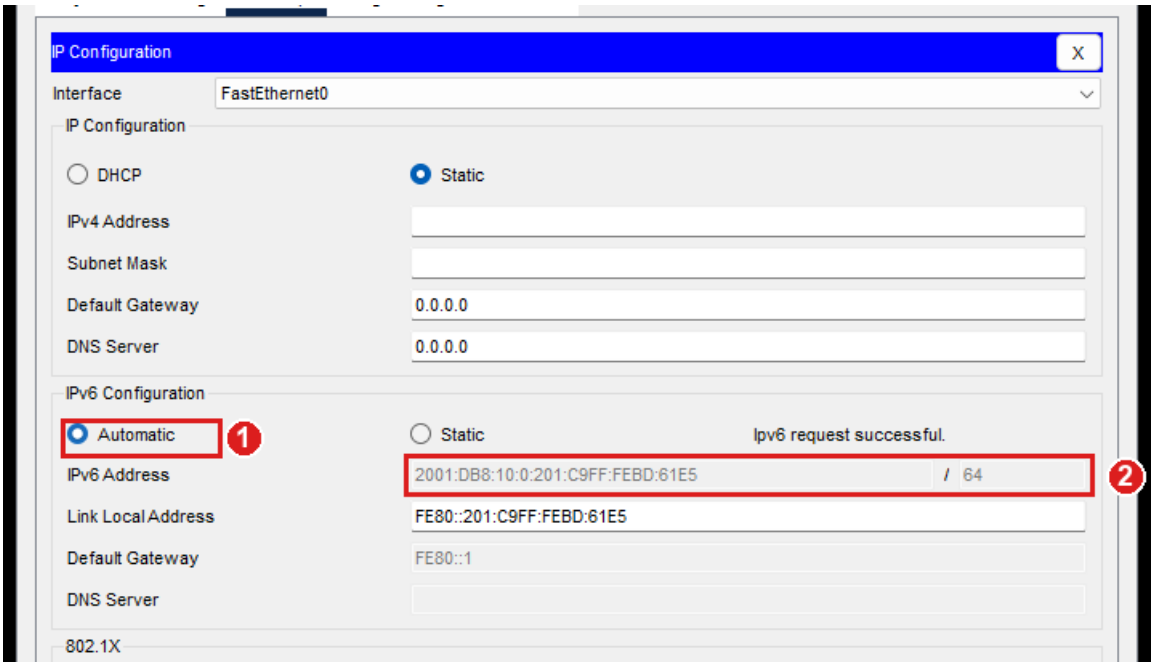
На ПК в IP Configuration заполните поля IPv6 Static. Проверьте ping глобального адреса собственного шлюза и удалённого ПК. Добавьте PC-C в LAN 10 и в блоке IPv6 выберите переключатель *Automatic*. Сравните его глобальный адрес, префикс и шлюз с вручную настроенным PC-A.

Проверка результата

IPv6 работает между LAN; таблица соседей IPv6 заполняется после трафика. PC-C получает адрес из префикса LAN 10 через SLAAC при наличии RA. В Simulation найдите ICMPv6 ND/RA события по фильтрам доступной версии; отсутствие ARP для IPv6 – штатное поведение. **Эталонный вывод.**

```
Router#show ipv6 route
C   2001:DB8:10::/64 [0/0]
     via GigabitEthernet0/0, directly connected
L   2001:DB8:10::1/128 [0/0]
     via GigabitEthernet0/0, receive
C   2001:DB8:20::/64 [0/0]
     via GigabitEthernet0/1, directly connected
L   2001:DB8:20::1/128 [0/0]
     via GigabitEthernet0/1, receive
L   FF00::/8 [0/0]
     via Null0, receive
```

Ответы PC-B на ping из PC-A приходят с TTL=127. В окне *IP Configuration* клиента с переключателем *Automatic* в блоке IPv6 появляются сообщение «ipv6 request successful.», адрес с префиксом сети LAN 10 и шлюз FE80::1.



PC-C после выбора Automatic: адрес из префикса 2001:DB8:10::/64, шлюз FE80::1.

Управляемая неисправность

Уберите `ipv6 unicast-routing`. Локальные тесты не обязаны исчезнуть, но межсетевой forwarding прекращается; RA также могут измениться. Восстановите команду и дождитесь автонастройки клиента.

Что сдать и объяснить

Сдайте адресную таблицу, маршруты и соседей. Почему одинаковый link-local шлюз в двух LAN не является конфликтом? Почему нельзя предсказать точный интерфейсный идентификатор SLAAC только по названию PC-C?

Самостоятельное усложнение. Добавьте IPv4 из 08 параллельно. Проверьте оба стека независимо и продемонстрируйте, что неверный IPv4-шлюз не обязан ломать исправный IPv6-путь.

Лабораторная 32. Статические маршруты IPv6

Время: 105 минут. **Подготовка:** 31, 09.

Какую проблему решает работа

Протокол IPv6 не использует таблицу маршрутов IPv4: для него нужны отдельные маршруты, и ошибки здесь выглядят иначе. Работа решает задачу связности между IPv6-сетями с помощью статических маршрутов и учит находить типичные ошибки.

Что вы научитесь делать

Организовать межмаршрутизаторный IPv6-путь и проверить обратную маршрутизацию.

Стенд и адреса

R1 G0/0 LAN-A 2001:db8:10::1/64; R2 G0/0 LAN-B 2001:db8:20::1/64. Между R1 G0/1 и R2 G0/1 сеть 2001:db8:12::/64: R1 ::1, R2 ::2. ПК ::10 своей LAN, шлюз глобальный ::1 своей LAN либо link-local маршрутизатора.



Порядок работы

На обоих маршрутизаторах включите `ipv6 unicast-routing`, задайте адреса интерфейсов и `no shutdown`. На R1:

```
ipv6 route 2001:db8:20::/64 2001:db8:12::2
```

На R2:

```
ipv6 route 2001:db8:10::/64 2001:db8:12::1
```

В EXEC проверьте `show ipv6 route`, соседние transit IP и ping между ПК. Для traceroute IPv6 используйте поддерживаемый синтаксис, который показывает `traceroute ?`; набор функций PC-PT и IOS отличается.

Проверка результата

Удалённые префиксы представлены S-маршрутами, локальные – подключёнными. ПК видят друг друга. Для варианта с link-local next hop маршрут включает интерфейс, например на R1 `ipv6 route 2001:db8:20::/64 gigabitEthernet 0/1 fe80::2`, если R2 G0/1 имеет именно этот link-local.

Управляемая неисправность

Удалите маршрут R2 к LAN-A. Проверьте ping между транзитными интерфейсами и отдельно между ПК. Первый работает, второй не обязан: ответ конечному узлу требует своего маршрута.

Что сдать и объяснить

Сдайте оба статических маршрута, таблицы и результат удаления обратного пути. Чем IPv6 route ::/0 соответствует IPv4 route 0.0.0.0/0?

Самостоятельное усложнение. Замените маршрут R1 default route `ipv6 route ::/0 2001:db8:12::2`. На R2 сохраните специфичный обратный маршрут. Не добавляйте взаимные defaults без понимания несуществующих назначений.

Глава 34

Wi-Fi и Интернет вещей

34.1 Ассоциация, адрес и приложение

Для беспроводного клиента есть несколько отдельных этапов: увидеть SSID, пройти согласованную защиту, получить IP-параметры и установить прикладной обмен. SSID является именем сети, а не паролем. При неправильной маске клиент может успешно подключиться к Wi-Fi и всё же не достичь удалённого сервера.

Домашний wireless router совмещает точку доступа, коммутатор, DHCP и нередко NAT. Простая access point мостит беспроводной и проводной сегменты. Lightweight AP управляется контроллером WLC. Эти роли не взаимозаменяемы. В Packet Tracer качество радиомодели и набор способов защиты ограничены; не используйте симулятор как точный инструмент обследования покрытия или подтверждения безопасности современной WLAN.

Лабораторная 33. Домашняя WLAN с WPA2 Personal

Время: 105 минут. **Подготовка:** 19, 21.

Какую проблему решает работа

В офисе нужна беспроводная сеть, но открытая сеть без пароля даёт доступ любому, а слабая защита взламывается. Работа настраивает домашнюю WLAN с шифрованием WPA2 Personal: имя сети, пароль, параметры, подключение клиента и проверку связи.

Что вы научитесь делать

Подключить ноутбук к защищённой WLAN и проверить отдельно Wi-Fi и IP-параметры.

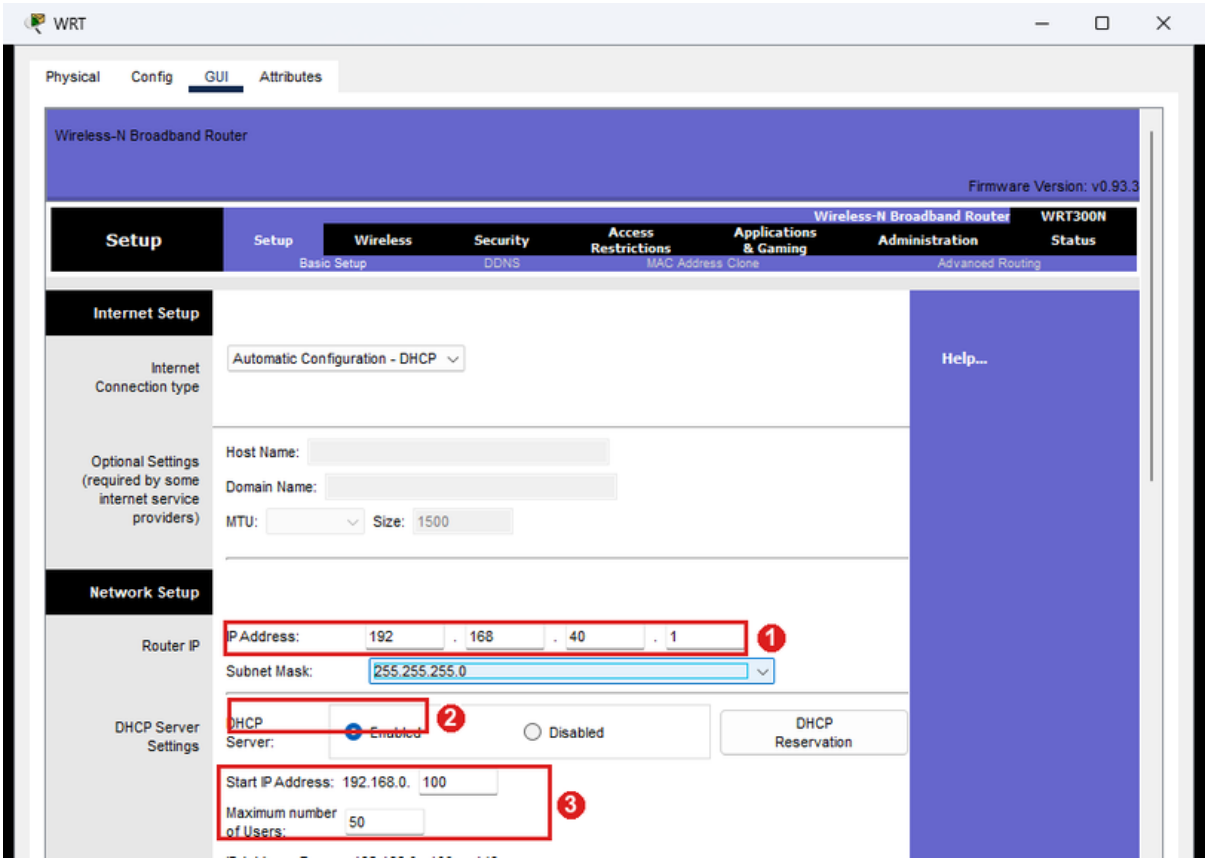
Стенд и адреса

WRT300N, Laptop-PT и PC-A по кабелю в один из LAN Ethernet-портов WRT. Интернет-порт пока свободен. LAN 192.168.40.0/24: WRT .1, DHCP .100–.149, PC-A .10/24. SSID LAB-WIFI, учебная passphrase LabWireless42.

Порядок работы

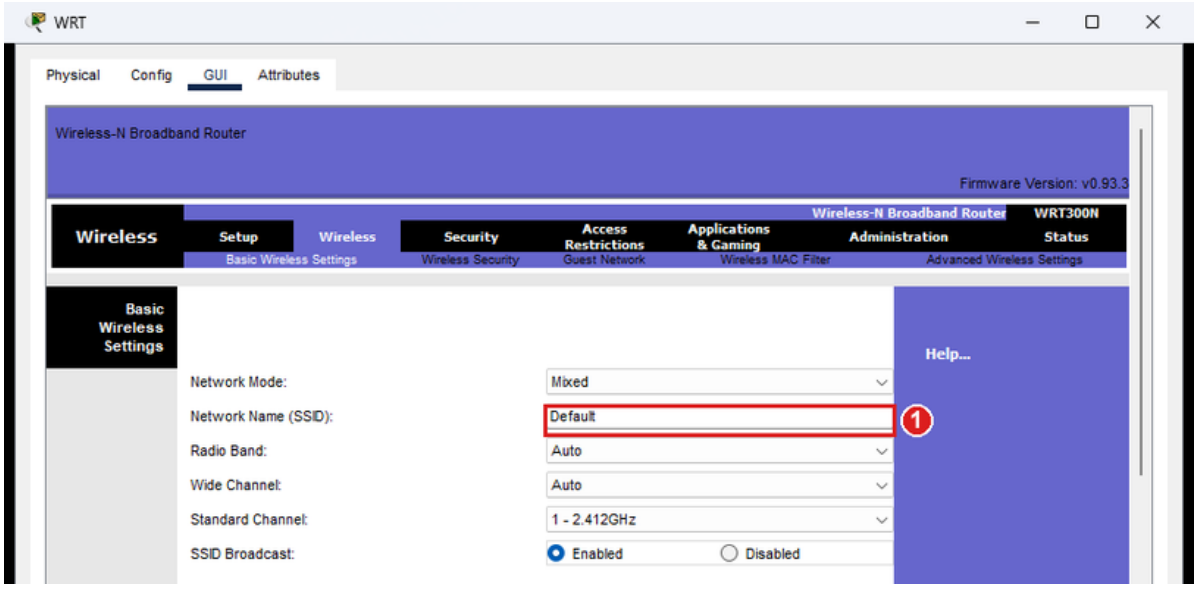
1. Соберите схему. Wireless Devices – четвёртый значок второго ряда палитры Network Devices; в списке моделей выберите беспроводной маршрутизатор *Linksys-WRT300N*. Ноутбук – *Laptop* в End Devices. Соедините PC-A прямым кабелем: `FastEthernet0` у PC-A и порт `Ethernet 1` у WRT (в списке портов WRT: `Internet`, `Ethernet 1`–`Ethernet 4`). Порт Internet не используем. Дайте PC-A адрес `192.168.40.10/24`, шлюз `192.168.40.1`.
2. Щёлкните по WRT и откройте вкладку *GUI* – это веб-интерфейс домашнего маршрутизатора. На странице *Setup / Basic Setup* в блоке *Network Setup* в поле *Router IP Address* (четыре окошка) измените третье число на `40`, чтобы получилось 192.168.40.1; маска `255.255.255.0`; *DHCP Server* – *Enabled*; *Start IP Address* – `192.168.40.100`, *Maximum number of Users* – `50`. Кнопка **Save Settings**

находится в самом низу страницы: прокрутите её полосой прокрутки у правого края (колесо мыши может прокручивать список маски, а не страницу). Без нажатия Save Settings изменения не применяются, и после каждой вкладки Save нужно нажимать заново.

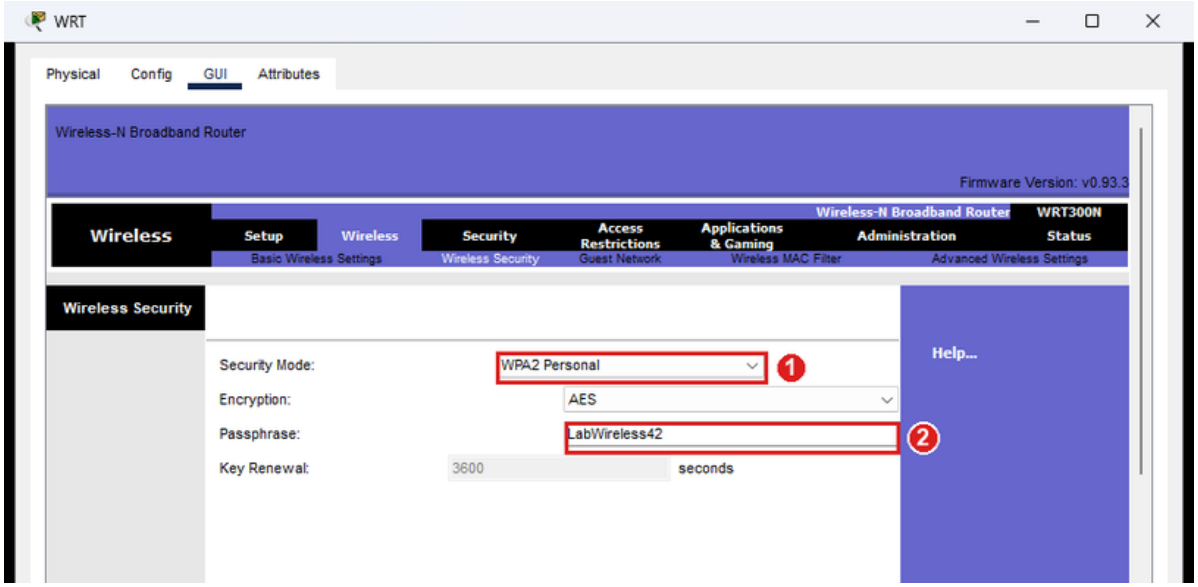


WRT, Setup / Basic Setup: 1 – адрес маршрутизатора, 2 – маска, 3 – DHCP.

3. Откройте *Wireless / Basic Wireless Settings* и в поле *Network Name (SSID)* введите `LAB-WIFI`; остальное оставьте. Прокрутите вниз и нажмите *Save Settings*. Затем откройте подвкладку *Wireless Security*: в списке *Security Mode* выберите `WPA2 Personal`, *Encryption* оставьте `AES`, в поле *Passphrase* введите `LabWireless42`, снова *Save Settings*.



Wireless / Basic Wireless Settings: 1 – SSID.

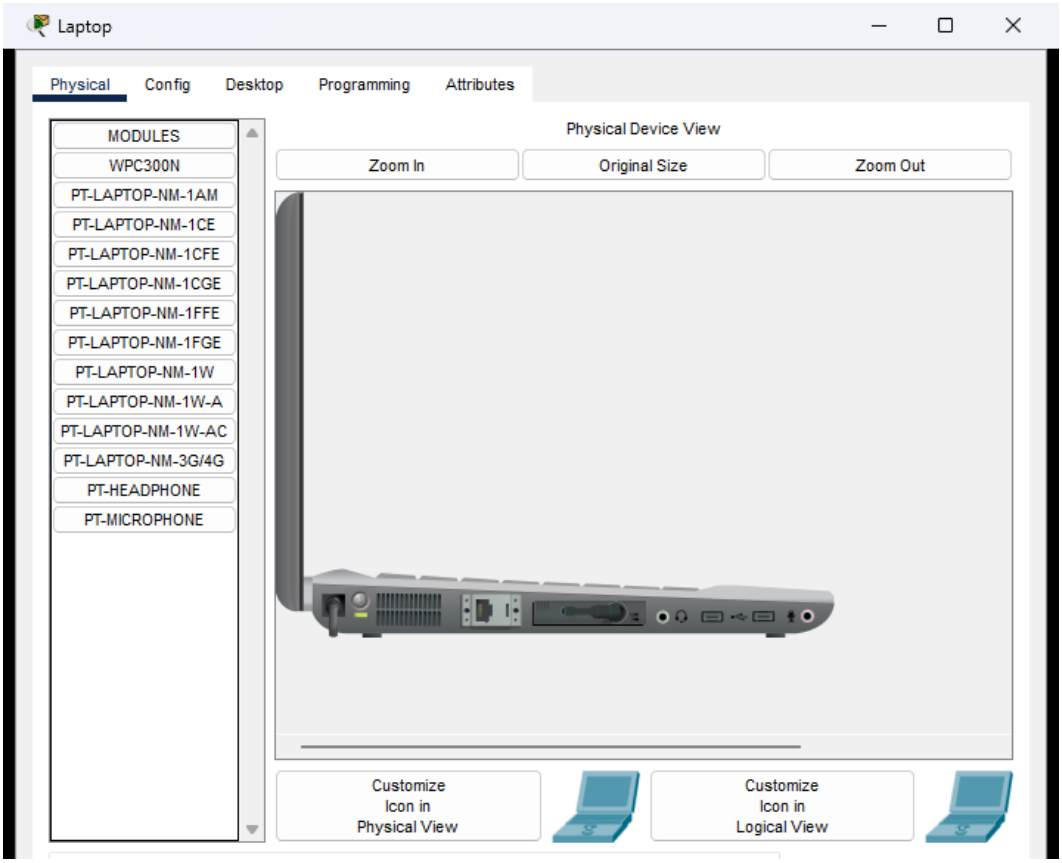


Wireless Security: 1 – Security Mode, 2 – Passphrase.

4. Ноутбуку нужен беспроводной модуль: по умолчанию у него только проводной порт `FastEthernet0`. Щёлкните по ноутбуку, откройте вкладку *Physical*. Выключите питание: круглая кнопка слева на изображении ноутбука (светодиод рядом погаснет). Перетащите установленный в чёрный слот (справа, с разъёмом RJ45) модуль в список модулей слева, затем перетащите из списка модуль *WPC300N* в освободившийся чёрный слот и снова включите питание. В списке портов (команда `ipconfig`) вместо FastEthernet0 появится `Wireless0`.



Корпус ноутбука (Physical): чёрный слот справа – место сменного модуля.

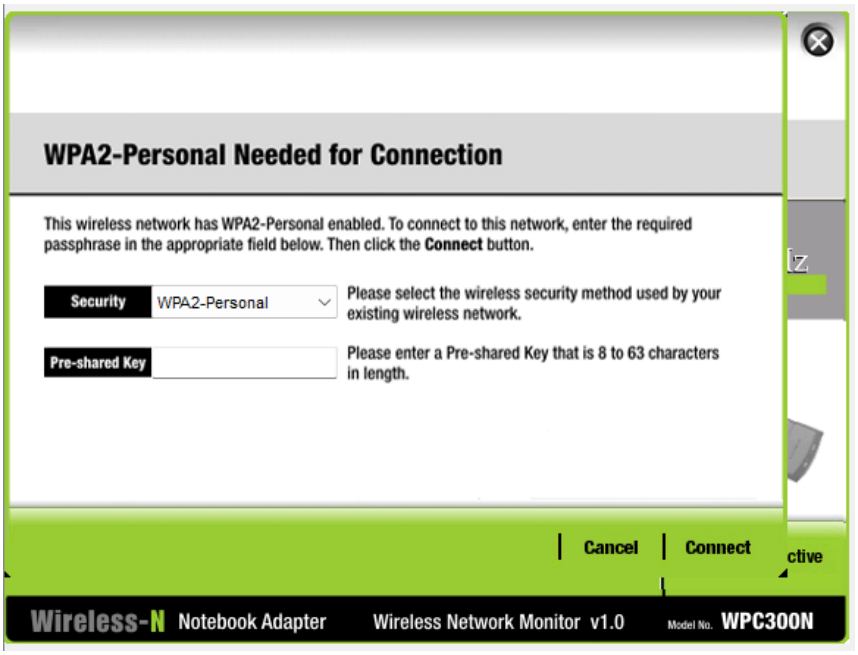


После установки WPC300N и включения питания: в слоте видна антенная карта.

5. На ноутбуке откройте *Desktop / PC Wireless*. Вкладка *Connect* покажет найденные сети: `LAB-WIFI`, канал 1, сигнал 100%, защита WPA2-PSK. Выделите строку и нажмите *Connect*; в окне *WPA2-Personal Needed for Connection* введите passphrase в поле *Pre-shared Key* и нажмите *Connect*. Справа внизу появится надпись «Adapter is Active».



PC Wireless, вкладка Connect: найденная сеть LAB-WIFI.



Запрос ключа WPA2-Personal.

6. Беспроводной интерфейс по умолчанию настроен на DHCP и получает адрес от WRT (в проверенном прогоне – 192.168.40.100). Проверьте адрес командой `ipconfig` в Command Prompt ноутбука (смотрите раздел *Wireless0 Connection*, а не Bluetooth), затем `ping 192.168.40.1` и `ping 192.168.40.10`. Сравните положение устройств в *Physical Workspace* при доступности соответствующей модели покрытия.

Проверка результата

Ноутбук ассоциирован с нужным SSID, имеет адрес из пула и пингует проводной ПК. Успех этого теста не требует внешнего Интернет-порта. Если виден SSID, но адрес отсутствует, отдельно проверьте защиту и DHCP.

Управляемая неисправность

Измените passphrase только на ноутбуке. Проверьте потерю ассоциации и исправьте. Затем в отдельном опыте задайте неправильный статический шлюз при правильном Wi-Fi: локальные пары могут оставаться достижимыми.

Что сдать и объяснить

Сдайте настройки SSID/защиты, состояние подключения и IP-параметры. В чём отличие проверки «видит SSID» от «открывает веб-сайт»?

Самостоятельное усложнение. Подключите Internet-порт WRT к отдельному учебному ISP и внешнему серверу, согласовав outside-подсеть и маршруты. Не подключайте его к собственному LAN-коммутатору, создавая неоднозначную схему.

Лабораторная 34. Централизованная WLAN через WLC

Время: 120–150 минут. **Подготовка:** 12, 19, 33.

Какую проблему решает работа

Десяток точек доступа настраивать по отдельности трудно: на каждой свои параметры, одну забыли — и в здании появляются «мёртвые зоны» или небезопасная точка. Работа решает проблему централизованным управлением точками через контроллер WLC.

Что вы научитесь делать

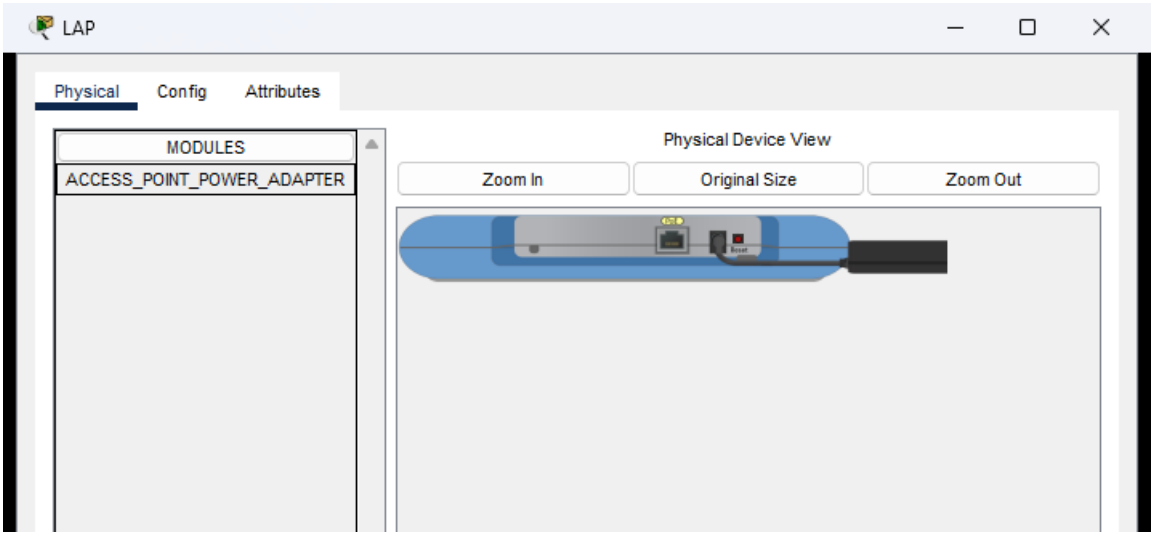
Разделить контроллер и точку доступа и связать WLAN с клиентским адресным планом.

Стенд и адреса

WLC-PT или WLC-2504, LAP-PT или 3702i, 2960, R1 и Laptop. Для первого опыта все подключения в одной VLAN 1 и сети 192.168.60.0/24: R1 .1, WLC management .2, проводной admin PC .10, LAP .3 статически либо DHCP, клиентские выдачи .100–.149. Такой старт исключает неопределённость VLAN-тегов. Подключите WLC порт 1 и LAP к разным портам S1; обеспечьте LAP питание, если модель требует адаптер.

Порядок работы

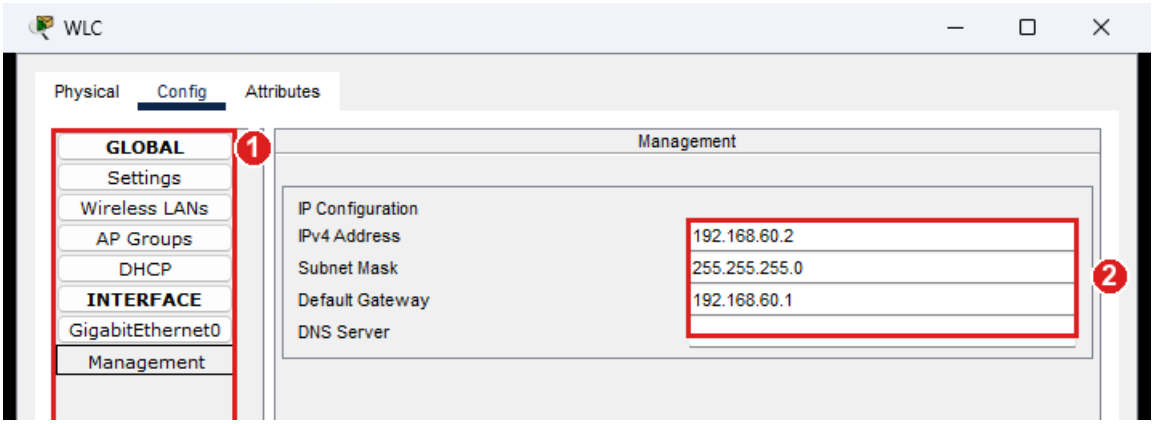
1. На R1 задайте G0/0 .1/24 и DHCP для этой сети с исключением .1–.99, как в работе 19 (пул WLAN: `network 192.168.60.0 255.255.255.0`, `default-router 192.168.60.1`). Включите интерфейс (`no shutdown`). На admin PC задайте .10/24, шлюз .1. Соберите схему: все устройства подключаются прямыми кабелями к S1 (контроллер – `GigabitEthernet0`, точка доступа – `GigabitEthernet0`; в списке портов WLC и LAP нет порта с именем «port 1»).
2. **Питание точки доступа.** LAP-PT при размещении выключена: попытка открыть её настройки даёт ошибку «Device must be powered on». Щёлкните по LAP, откройте вкладку *Physical*, перетащите модуль `ACCESS_POINT_POWER_ADAPTER` из списка слева на гнездо питания на изображении точки. Индикатор порта на схеме станет зелёным.



LAP-PT, вкладка Physical: слева единственный модуль – адаптер питания.

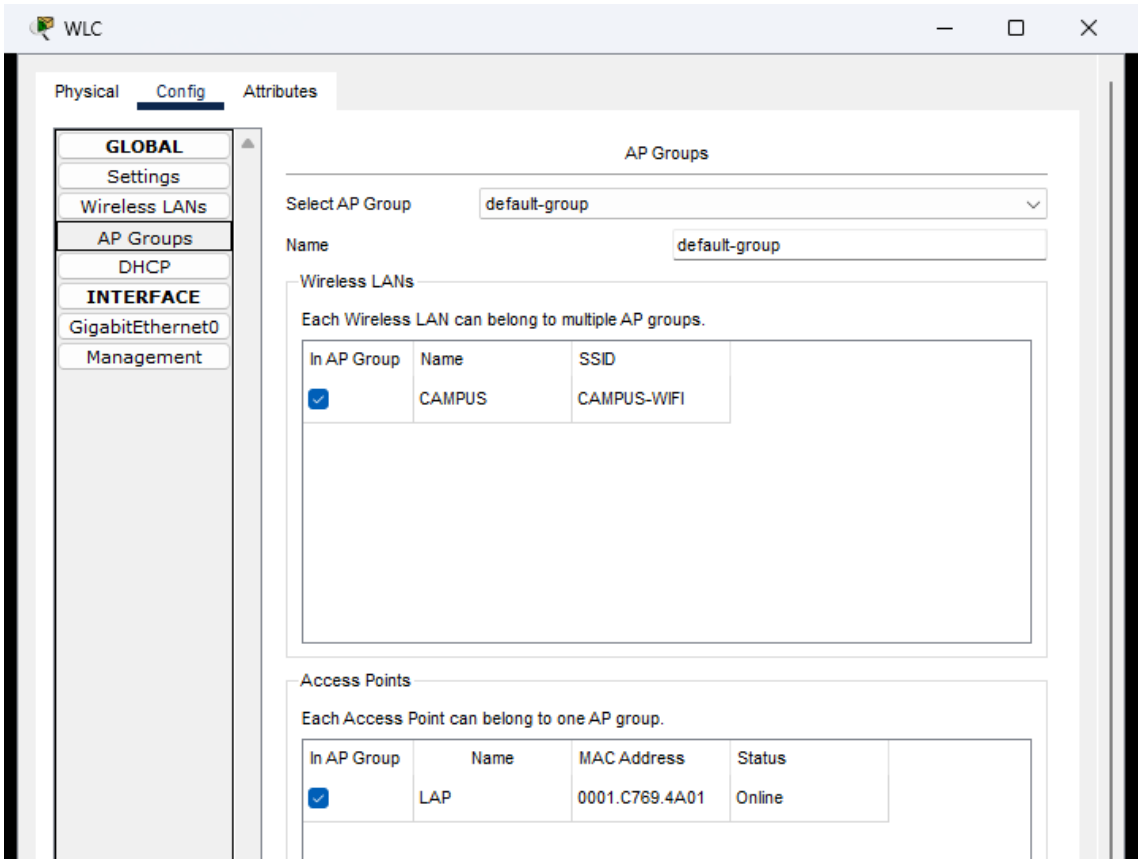
3. **Контроллер WLC-PT настраивается во вкладке Config, а не через веб-**

интерфейс. Щёлкните по WLC, откройте *Config*. Слева: *GLOBAL – Settings, Wireless LANs, AP Groups, DHCP; INTERFACE – GigabitEthernet0, Management*. Выберите *Management* и введите *IPv4 Address* `192.168.60.2`, *Subnet Mask* `255.255.255.0`, *Default Gateway* `192.168.60.1`. Отдельного мастера создания учётной записи у WLC-PT нет; веб-интерфейс с мастером имеет более полная модель WLC-2504 (в этом прогоне она не проверялась).



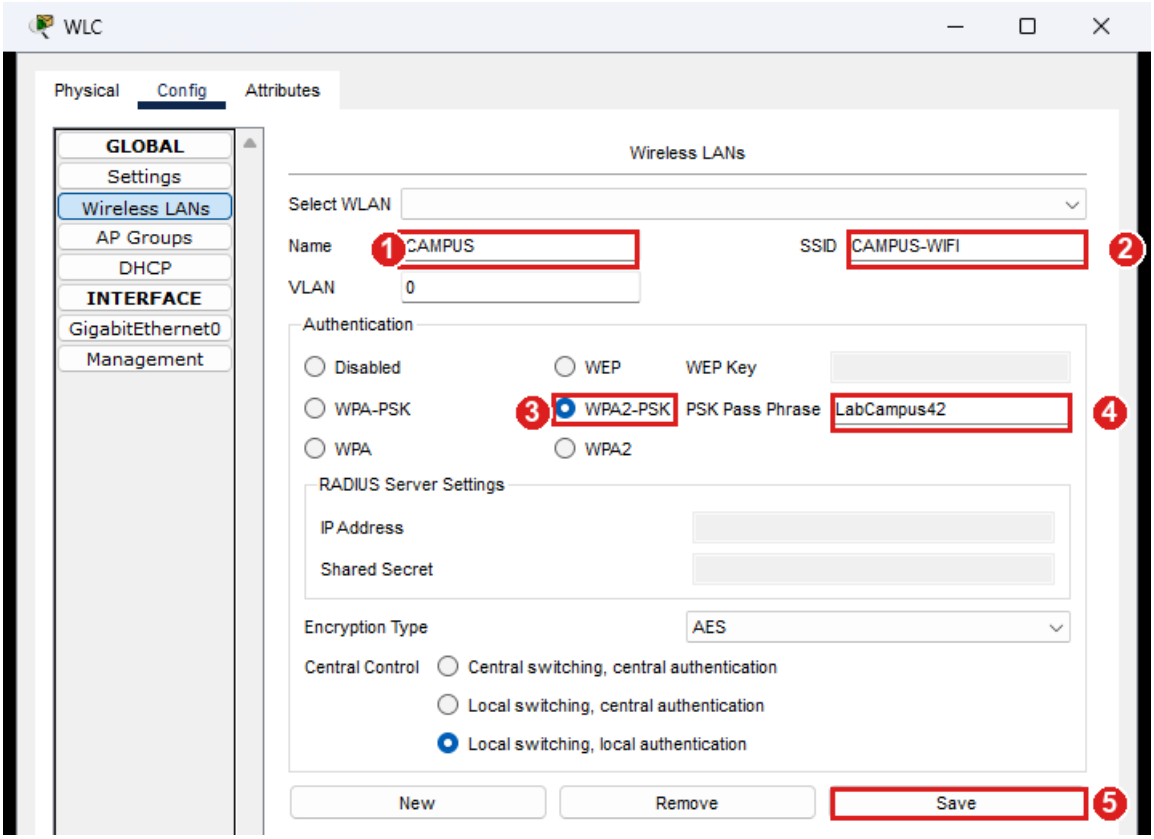
WLC-PT, Config / Management: 1 – раздел Management, 2 – IP-параметры.

4. На LAP откройте *Config / Settings* и выберите *DHCP*. Подождите или нажмите Fast Forward Time: точка получит адрес из пула (в проверке – 192.168.60.100). Поля «Primary Controller IP» у LAP-PT нет: точка находит контроллер сама. Убедитесь на WLC: *Config / AP Groups*, внизу блок *Access Points* – в нём строка LAP с MAC-адресом и состоянием *Online*.



AP Groups: внизу – точка LAP в состоянии Online, вверху – WLAN, входящие в группу default-group.

5. На WLC откройте *Config / Wireless LANs* и нажмите *New*. Заполните: *Name* – `CAMPUS`, *SSID* – `CAMPUS-WIFI`, *VLAN* – `0` (оставьте), *Authentication* – *WPA2-PSK*, *PSK Pass Phrase* – `LabCampus42`. *Encryption Type* автоматически станет *AES*. Режим *Central Control* оставьте по умолчанию (*Local switching, local authentication*). Нажмите **Save**. Новая WLAN сразу появится отмеченной в группе *default-group* на странице *AP Groups*; привязывать её к интерфейсу вручную не нужно.



Wireless LANs: 1 – имя, 2 – SSID, 3 – WPA2-PSK, 4 – пароль, 5 – Save.

6. Подключите ноутбук с модулем WPC300N так же, как в работе 33 (Desktop / PC Wireless / Connect, сеть `CAMPUS-WIFI`, пароль `LabCampus42`), получите адрес по DHCP и проверьте .1 и admin PC. Этот последний шаг в данной редакции не воспроизводился; остальные шаги работы проверены построением схемы.

Проверка результата

Контроллер видит LAP, ноутбук ассоциирован и получил корректный адрес. Если конкретный WLC не позволяет использовать management для клиентов, создайте dynamic interface отдельной клиентской VLAN по дополнительному заданию; укажите это отличие в отчёте. Диагностируйте этапы отдельно: IP LAP, регистрация AP, WLAN Enabled, security, interface mapping, DHCP.

Управляемая неисправность

Отключите только WLAN на контроллере. Сам WLC остаётся доступен проводному администратору, но клиентское подключение перестаёт работать. Включите WLAN и подтвердите восстановление.

Что сдать и объяснить

Сдайте данные management-интерфейса, список AP, профиль WLAN и IP клиента. Объясните, почему работа проводного интерфейса управления ещё не доказывает работу клиентского WLAN.

Самостоятельное усложнение. В сохранённой копии выделите management VLAN 60 (192.168.60.0/24) и client VLAN 70 (192.168.70.0/24). WLC-порт S1 сделайте trunk с VLAN 60,70; management WLC тегируйте 60, dynamic interface – 70 с адресом .2, шлюзом .1 и DHCP server 192.168.70.1. LAP-порт оставьте access 60. На R1 создайте подинтерфейсы, два DHCP pool и согласованный trunk. Привяжите WLAN к dynamic interface 70; проверяйте наличие этой возможности по модели.

Лабораторная 35. IoT: регистрация и простое правило

Время: 105–120 минут. **Подготовка:** 19, 21, 33.

Какую проблему решает работа

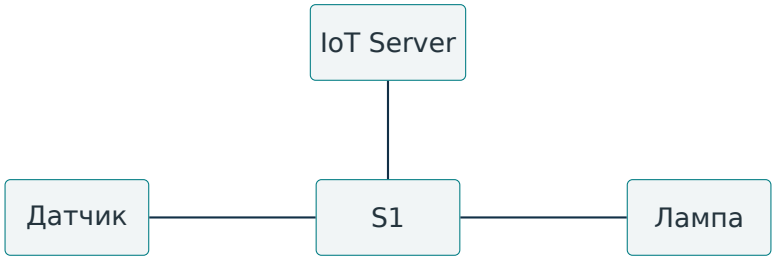
Датчики и умные устройства подключаются к сети, и ими нужно управлять централизованно, но они не должны получать лишних прав. Работа решает задачу регистрации устройств интернета вещей и простого правила, которое ими управляет, и показывает, как проверить результат.

Что вы научитесь делать

Подключить Smart Things к registration server и проверить правило «условие – действие».

Стенд и адреса

S1, R1 .1/24, Server-PT .10/24 и admin PC .20/24 в 192.168.80.0/24. Выберите сетевые Smart Things: Motion Detector и Light (названия вещей на схеме можно задать любые, ниже – Motion и Lamp), которые поддерживают регистрацию. Простые компоненты без сетевого интерфейса для этого стенда не подходят. Для них понадобился бы отдельный MCU/SBC и программирование.

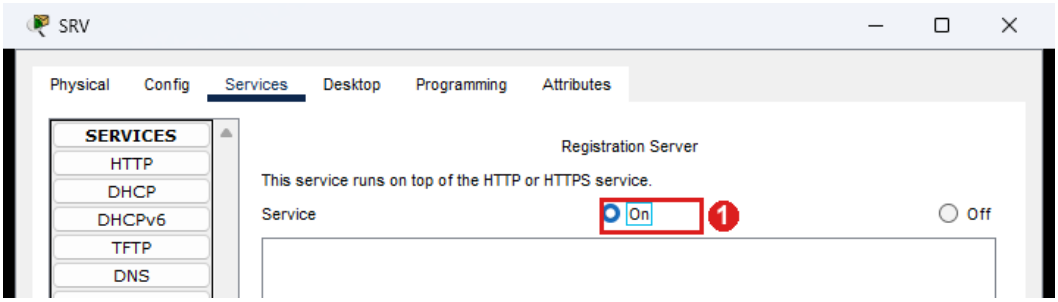


Порядок работы

1. **Что нужно знать о Smart Things.** Motion Detector и Light (в палитре и в окне сервера – «Light», слова «Lamp» в Packet Tracer 8.2.1 нет) имеют только беспроводной интерфейс `Wireless0`; у них нет Ethernet-порта. По умолчанию такая вещь ищет открытую сеть с SSID `HomeGateway`. Поэтому в схеме нужен беспроводной маршрутизатор или точка доступа. Проверенный вариант: *Linksys-WRT300N*; сеть 192.168.0.0/24 по умолчанию.
2. Соберите схему: WRT300N (Network Devices / Wireless Devices), Server-PT и ПК администратора. Кабель SRV `FastEthernet0` – WRT `Ethernet 1`, admin PC `FastEthernet0` – WRT `Ethernet 2`. SRV: `192.168.0.10/24`, шлюз `192.168.0.1`; admin

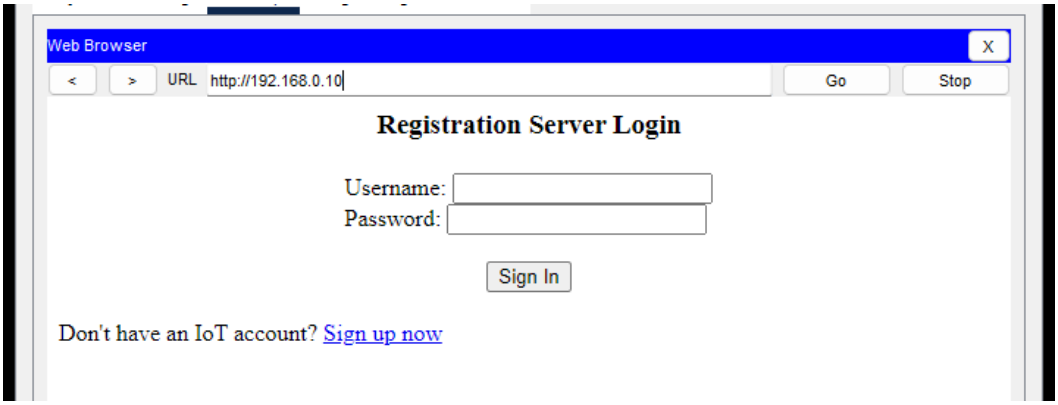
PC: `192.168.0.20/24`, шлюз `192.168.0.1`. Пул DHCP маршрутизатора начинается с .100, поэтому статические адреса не конфликтуют с ним.

3. Откройте WRT, вкладку *GUI*, *Wireless / Basic Wireless Settings*: в поле *Network Name (SSID)* введите `HomeGateway` (защиту оставьте *Disabled*), прокрутите страницу вниз и нажмите *Save Settings*.
4. На SRV откройте *Services / IoT*: страница называется *Registration Server* («работает поверх HTTP или HTTPS»). Включите *Service – On*.



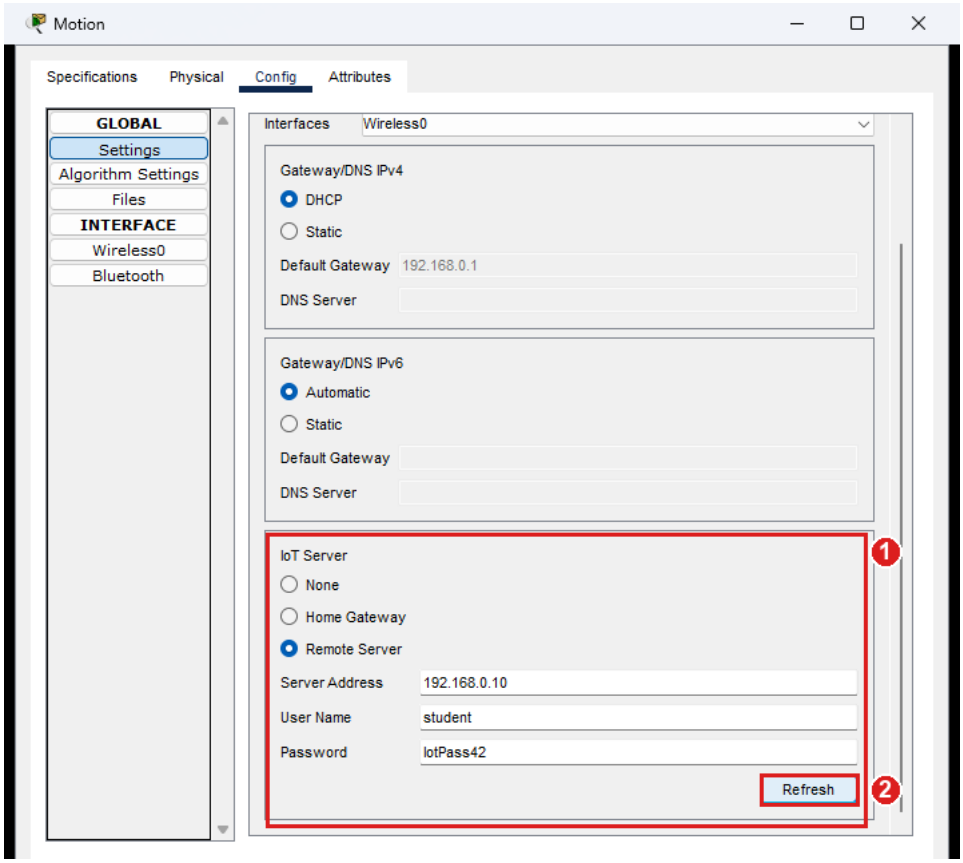
Служба IoT на сервере: переключатель Service по умолчанию Off.

5. На admin PC откройте *Desktop / Web Browser* и адрес `http://192.168.0.10`. Появится страница *Registration Server Login*. Нажмите ссылку *Sign up now*, введите имя и пароль (например `student` и `IotPass42`), нажмите *Create*, затем войдите теми же данными кнопкой *Sign In*. Если вход вернул «Wrong username or password», введите данные ещё раз. Откроется страница *IoT Server – Devices* с меню *Home | Conditions | Editor | Log Out*; список устройств пока пуст.



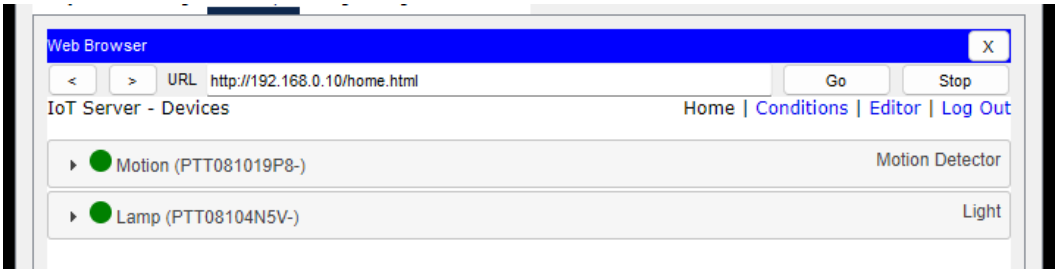
Страница входа Registration Server.

6. Добавьте Smart Things: подкласс *Home* класса End Devices (второй значок во втором ряду палитры), модели *Motion Detector* и *Light*. Щёлкните по вещи: окно имеет вкладки *Specifications, Physical, Config, Attributes*. На вкладке *Specifications* описано, как взаимодействовать с вещью (у Motion Detector – «Alt-mouse move»: перемещение мыши над значком при нажатой клавише Alt; через 5 секунд без движения датчик отключается). Откройте *Config / Settings*, прокрутите окно вниз до блока *IoT Server*: выберите *Remote Server*, *Server Address* – `192.168.0.10`, *User Name* и *Password* – данные вашей учётной записи, нажмите *Connect*. Кнопка сменит название на *Refresh*. IPv4 у вещи должен быть *DHCP*: адрес придёт от WRT (проверено: .100 и .101).



Config / Settings вещи: блок IoT Server с выбранным Remote Server и введёнными данными.

7. Обновите страницу *Home* на admin PC (ссылка Home): в списке появятся обе вещи с зелёными индикаторами и типами *Motion Detector* и *Light*.



Страница IoT Server – Devices: зарегистрированы Motion и Lamp.

8. Откройте *Conditions*: таблица условий (*Actions, Enabled, Name, Condition*) и кнопка *Add*. Создайте правило: при срабатывании Motion Detector включать Light; затем обратное правило для выключения, если модель предоставляет соответствующее состояние датчика. (Окно создания условия в проверочном прогоне не открывалось автоматически; названия его полей сверяйте с экраном.)
9. Измените состояние датчика доступным способом (Alt-mouse move над значком датчика на схеме) и наблюдайте лампу.

Проверка результата

Оба устройства зарегистрированы под нужной учётной записью; состояние лампы меняется согласно условию, а не только при ручном включении. Запишите точное имя свойства и значения из фактического UI: у разных Things они различаются.

Управляемая неисправность

Укажите неправильный пароль регистрации только на Lamp. Она может иметь корректный IP и отвечать на сетевые проверки, но не участвовать в серверном управлении. Исправьте аутентификацию и повторите правило.

Что сдать и объяснить

Сдайте список устройств, таблицу «состояние датчика / ожидаемая лампа / фактическая лампа» и правило. Чем сетевое Smart Thing отличается от датчика, подключённого к MCU через контакт?

Самостоятельное усложнение. Используйте датчик температуры и Fan: порог включения выше порога выключения, например 28 и 25 градусов. Это гистерезис, уменьшающий частое переключение. Выбирайте только свойства, реально реализованные выбранными устройствами.

Часть V

Обслуживание, диагностика и проекты

Глава 35

Наблюдаемость и восстановление

35.1 Что сеть должна сообщать администратору

Ping показывает достижимость в выбранных условиях, но не объясняет всю историю. Системный журнал содержит события интерфейсов и конфигурации; NTP помогает согласовать время; резервная копия облегчает восстановление. Точный порядок действий и сохранённые настройки важнее случайного набора скриншотов.

Syslog использует уровни от 0 до 7: Emergency, Alert, Critical, Error, Warning, Notice, Informational, Debug. Настройка порога определяет, какие сообщения отправляются. Не путайте severity сообщения с номером лабораторной и количеством потерянных пакетов. TFTP является простой передачей без полноценной защиты и аутентификации; здесь он используется в изолированном учебном сегменте.

Лабораторная 36. NTP, Syslog и резервная копия TFTP

Время: 120 минут. **Подготовка:** 06, 21.

Какую проблему решает работа

Когда случилась авария, нужно знать, что и когда произошло: часы устройств должны совпадать, журналы должны храниться не только на самих устройствах, а конфигурация — быть сохранена вне них. Работа решает проблему наблюдаемости и восстановления: NTP, Syslog и копия конфигурации на сервере TFTP.

Что вы научитесь делать

Согласовать время, собрать событие на сервере и сохранить конфигурацию вне устройства.

Стенд и адреса

R1 G0/0 192.168.90.1/24–S1–SRV .10/24; admin PC .20/24. Шлюз SRV и ПК .1. G0/1 R1 используется как свободный тестовый интерфейс; не выключайте путь к серверу для первого события.

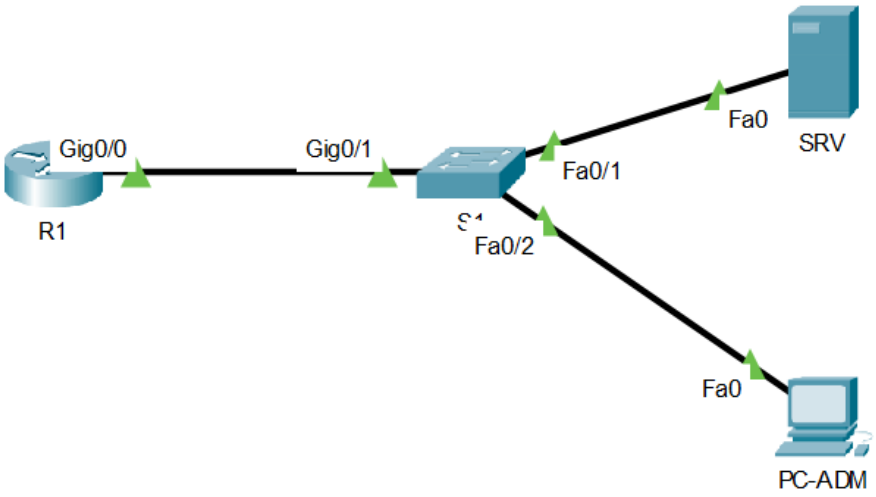


Схема работы 36: маршрутизатор R1, сервер SRV и компьютер администратора.

Таблица соединений (проверена построением схемы в Packet Tracer 8.2.1). Порты записаны полными именами, как в списке портов программы.

Устройство	Порт	Устройство	Порт	Кабель
R1	GigabitEthernet0/0	S1	GigabitEthernet0/1	прямой
SRV	FastEthernet0	S1	FastEthernet0/1	прямой
PC-ADM	FastEthernet0	S1	FastEthernet0/2	прямой

Порядок работы

На SRV включите Services / NTP, Syslog и TFTP. Установите понятное учебное время NTP, если форма позволяет. На R1:

```
enable
configure terminal
service timestamps log datetime msec
ntp server 192.168.90.10
logging 192.168.90.10
logging trap informational
interface gigabitEthernet 0/1
 description TEST-EVENT
 no shutdown
 shutdown
exit
end
show clock
show logging
show ntp associations
show ntp status
copy running-config startup-config
copy running-config tftp:
```

При копировании задайте Remote host 192.168.90.10 и Destination filename `R1-L36.cfg`. Промпты интерактивны: отвечайте на них после команды, не вставляйте ответы как IOS-команды. Просмотрите список файлов TFTP на сервере.

Проверка результата

В Syslog есть событие тестового интерфейса, временные метки интерпретируемы. NTP может требовать времени на синхронизацию; сначала подтвердите IP-связность, затем associations/status. Некоторые модели отображают лишь часть

диагностических команд: если одна не реализована, укажите это и используйте доступное состояние, не объявляя синхронизацию только по введённой строке ntp server.

Для проверки копии измените на R1 описание тестового интерфейса. Выполните `copy tftp: running-config`, указав host и filename. Проверьте возвращённое описание. Копирование в running-config обычно **сливает** настройки, а не гарантированно заменяет всё; лишние новые строки могут сохраниться. Полное восстановление через startup-config и перезапуск выполняйте только в отдельной учебной копии после сохранения проекта. **Эталонный вывод.** Пример диалога копирования (после первой команды `copy` нажмите Enter, и только потом вводите вторую):

```
R1#copy running-config startup-config
Destination filename [startup-config]?
Building configuration...
[OK]
R1#copy running-config tftp:
Address or name of remote host []? 192.168.90.10
Destination filename [R1-confg]? R1-L36.cfg
Writing running-config...
```

Сразу после настройки `show ntp status` может показывать `Clock is unsynchronized, stratum 16`: синхронизации нужно время. Если вставить блок команд из листинга целиком, вторая команда `copy` будет принята как имя файла для первой и закончится ошибкой `Invalid argument`.

Управляемая неисправность

Выключите TFTP-службу, не меняя IP. Ping сервера остаётся успешным, а копирование не проходит. Включите службу и повторите. Не заменяйте маршруты для исправления выключенного приложения.

Что сдать и объяснить

Сдайте сообщение Syslog, состояние NTP, имя резервного файла и проверку восстановленного описания. Почему восстановление конфигурации нужно проверять опытом, а не только наличием файла?

Самостоятельное усложнение. Подключите S1 к серверу с SVI управления и шлюзом. Добавьте Syslog и отдельную TFTP-копию его настроек; соблюдайте уникальные имена файлов и устройств.

Лабораторная 37. Практикум поиска неисправностей

Время: 150–180 минут. **Подготовка:** 09, 13, 20–28.

Какую проблему решает работа

В реальной жизни сеть приходит к администратору уже сломанной и без подсказки о причине. Работа решает проблему метода: по симптому определить уровень, проверить гипотезу, исправить минимальным изменением и убедиться, что исправление не сломало другое.

Что вы научитесь делать

Локализовать сбой по слоям, доказать причину и исправить её минимальным изменением.

Стенд и адреса

Используйте несколько сохранённых исправных стендов. Для каждого случая делайте отдельную копию. Если работаете самостоятельно, внесите указанную ошибку, закройте задание на несколько минут и затем расследуйте только по наблюдениям. Для работы с преподавателем он может выбрать ошибку без подсказки студенту.

№	Исходный стенд	Внести одну ошибку	Первое наблюдение
1	08	На R1 G0/1: shutdown	PC-A не достигает PC-B.
2	08	У PC-B удалить шлюз	Локальный шлюз доступен.
3	11	Fa0/2 перенести в VLAN 20	Связь одной пары пропала.
4	12	Убрать VLAN 20 из allowed S2	VLAN 10 работает через trunk.
5	13	dot1Q 200 вместо 20	Физический uplink активен.
6	09	Удалить обратный маршрут R3	Запрос и ответ ведут себя различно.
7	20	Удалить helper на G0/0	Статический адрес работает, DHCP нет.
8	21	Неверная A-запись DNS	По IP страница доступна.
9	24	deny any первым правилом	Приложения гостя заблокированы.
10	26	Outside-интерфейс отметить inside	Частная LAN исправна, NAT нет.
11	27	Transit-интерфейс сделать passive	Нет ожидаемого соседства.
12	31	Отключить ipv6 unicast-routing	Локальные IPv6-тесты проходят.

Порядок работы

Для каждого случая примените один и тот же журнал расследования:

1. Сформулируйте симптом: откуда, куда, каким протоколом, когда, с какой ожидаемой политикой. «Не работает сеть» слишком неопределённо.
2. Проверьте питание, кабель и интерфейсное состояние. Затем адрес/маску/шлюз, VLAN/access/trunk и маршруты туда и обратно.
3. Если сетевой обмен есть, проверьте ACL, NAT и конкретную службу. Для проблемы имени сравните имя и IP. Для DHCP сравните динамическую и временную статическую настройку.
4. Сформулируйте две гипотезы и выберите команду, различающую их. До изменения запишите результат. Для наблюдения одного направления используйте Simulation.
5. Внесите минимальную правку. Повторите исходный тест и соседний контрольный тест, чтобы исключить побочное нарушение.
6. Сохраните исправный проект и документируйте конкретную строку или поле, которое изменили.

Проверка результата

Выберите минимум восемь случаев из разных тем; полный маршрут – все двенадцать. Для каждого нужны доказательство до правки, подтверждённая причина и тест после. Простое повторение конфигурации всего устройства не считается локализацией.

Управляемая неисправность

Усложнённый раунд: одновременно ошибочная VLAN и шлюз ПК. После первой правки симптом может сохраниться. Остановитесь и заново оцените остаточный отказ; не отменяйте уже доказанное исправление только потому, что сеть ещё не полностью заработала.

Что сдать и объяснить

Сдайте таблицу «симптом / гипотезы / различающая проверка / причина / правка / результат». На защите преподаватель задаёт один новый симптом, а вы вслух выбираете следующий тест.

Самостоятельное усложнение. Создайте три собственных неисправности и отдельный ключ ответов. Каждая должна иметь воспроизводимый симптом и однозначную проверку; не используйте случайную перегрузку симулятора как учебную ошибку.

Глава 36

Комплексные проекты

Проекты отличаются от пошаговых работ: задан результат и ограничения, а детали конфигурации выбираете вы. Сначала нарисуйте схему и адресный план, согласуйте политику, затем строите минимальный работающий фрагмент. После каждого этапа сохраняйте версию: физическая сборка, VLAN, шлюзы, маршруты, службы, политики, резервирование. Так сложный отказ можно сравнить с последним исправным состоянием.

Лабораторная 38. Кампус: три отдела и серверная

Время: 6–8 часов. **Подготовка:** 11–24, 27, 36–37.

Какую проблему решает работа

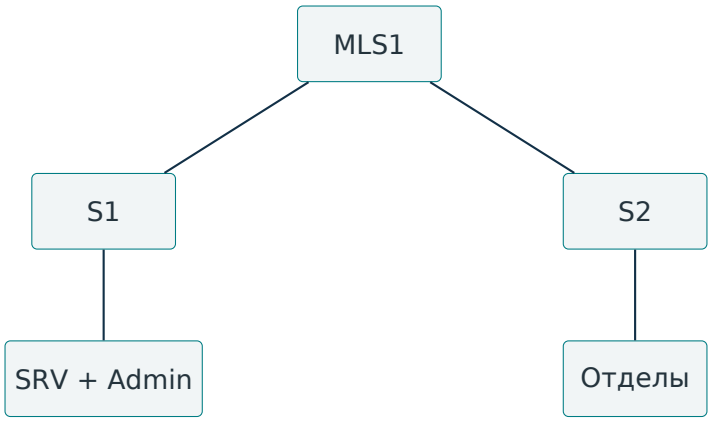
Отдельные приёмы ничего не стоят, пока они не собраны в сеть организации: три отдела, серверная, выход в Интернет, защита и наблюдение. Работа решает проблему целостного проектирования: единый адресный план, согласованные VLAN, маршрутизация и политики.

Что вы научитесь делать

Спроектировать небольшую сеть кампуса, связать VLAN, организовать службы и проверить политику гостей.

Стенд и адреса

MLS1 3560, S1 и S2 2960, три группы ПК и Server-PT. MLS1 Fa0/1–S1 G0/1; MLS1 Fa0/2–S2 G0/1. На каждом access-коммутаторе Fa0/1 – STAFF, Fa0/2 – STUDENTS, Fa0/3 – GUEST. SRV на S1 Fa0/10 VLAN 50. Admin PC на S1 Fa0/11 VLAN 99.



VLAN	Подсеть	Шлюз MLS1	Назначение
10	10.38.10.0/24	10.38.10.1	STAFF
20	10.38.20.0/24	10.38.20.1	STUDENTS
30	10.38.30.0/24	10.38.30.1	GUEST
50	10.38.50.0/24	10.38.50.1	SRV .10
99	10.38.99.0/24	10.38.99.1	S1 .2, S2 .3, Admin .10
999	Нет IP	Нет	Native без пользователей

Обязательный результат

Все пользовательские ПК получают адреса DHCP с отдельными пулами; SRV предоставляет DHCP, DNS и HTTP. STAFF и STUDENTS достигают сервера. Гости используют только DNS и HTTP сервера и могут пинговать свой шлюз, но не достигают STAFF, STUDENTS и сети управления. Admin управляет S1/S2 по SSH. На пользовательских access-портах настроен PortFast; защита портов не применяется к uplink. Сохранены конфигурации и адресный план.

Порядок работы

1. Создайте VLAN на всех нужных устройствах. На uplink согласуйте trunk, native 999 и allowed 10,20,30,50,99,999. Настройте access-порты по ролям.
2. На MLS1 создайте SVI 10,20,30,50,99, включите ip routing. На SRV и Admin задайте статику. На S1/S2 настройте SVI 99 и default-gateway 10.38.99.1.
3. На SRV создайте три DHCP pool с выдачей .100–.149 и DNS 10.38.50.10; на SVI 10,20,30 добавьте helper к серверу.
4. Создайте имя `portal.lab.test`, страницу и SSH как в предыдущих работах. Проверьте исходную связность до ACL.
5. На SVI 30 примените extended ACL in. Сначала разрешите DHCP клиента: UDP с source port 68 на destination port 67, включая первоначальный источник 0.0.0.0. Затем DNS TCP/UDP 53 и HTTP TCP/80 к SRV, ping собственного шлюза, после – запрет остального.
6. Включите журналирование на SRV и резервное копирование на TFTP по модели 36. Добавляйте функции по одной и повторяйте приёмочные тесты.

Проверка результата

Пример ключевого DHCP-разрешения гостевой ACL:

```
permit udp any eq 68 any eq 67
```

Это разрешение важно перед правилами с source 10.38.30.0/24: у первоначального Discover адрес источника ещё не выдан. Ответ relay к клиенту выходит через SVI и не проверяется этой ACL in. В приложении дан полный ориентир гостевой ACL; адаптируйте его к своим точным адресам.

Источник	Тест	Ожидается
Все 3 пользовательские VLAN	Получить DHCP	Правильные адрес/маска/шлюз/DNS
STAFF, STUDENTS	HTTP по имени	Страница открывается
GUEST	DNS + HTTP SRV	Разрешено
GUEST	Ping собственного шлюза	Разрешено
GUEST	Ping к SRV, STAFF, STUDENTS и Admin	Запрещено
Admin	SSH на S1 и S2	Успешный вход
Admin	Проверить Syslog/TFTP	Событие и две разные копии

Управляемая неисправность

Выберите две ошибки из 37: одну L2 и одну служб/политики. Исправьте с журналом расследования. Не снимайте всю гостевую ACL для финального теста DHCP.

Что сдать и объяснить

Сдайте .pkt, схему, адресный план, экспорт конфигураций, матрицу проверок и журнал двух ремонтов. Защита: объясните путь DNS-запроса и HTTP-ответа гостя, включая helper и направление ACL.

Самостоятельное усложнение. Добавьте LACP между MLS1 и S2 на двух совместимых портах. Не оставляйте старый отдельный uplink параллельно без про-

думанной STP-схемы. Проверьте отказ одного участника и сохранение политики.

Лабораторная 39. Головной офис, филиал и учебный ISP

Время: 6–8 часов. **Подготовка:** 09–10, 25–29, 38.

Какую проблему решает работа

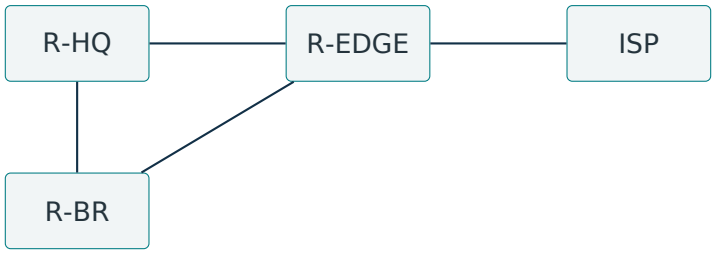
Организации растут, и у головного офиса появляется филиал: нужны маршруты между площадками, общий адресный план и выход в Интернет через провайдера. Работа решает проблему объединения площадок в одну управляемую сеть.

Что вы научитесь делать

Объединить две площадки, организовать динамические внутренние маршруты и PAT на выходе.

Стенд и адреса

R-HQ, R-BR, R-EDGE, ISP – 2911. В отличие от 38, здесь HQ может иметь одну LAN; сервером DNS/HTTP служит HQ-SRV. Используйте Ethernet и схему ниже.



Участок	Адреса	Порты
HQ LAN	10.39.10.0/24: HQ .1, SRV .10, PC .100	HQ G0/0
BR LAN	10.39.20.0/24: BR .1, PC .100	BR G0/0
HQ–EDGE	10.39.1.0/30: HQ .1, EDGE .2	HQ G0/1–EDGE G0/0
HQ–BR	10.39.1.4/30: HQ .5, BR .6	HQ G0/2–BR G0/1
BR–EDGE	10.39.1.8/30: BR .9, EDGE .10	BR G0/2–EDGE G0/1
EDGE–ISP	203.0.113.0/30: ISP .1, EDGE .2	EDGE G0/2–ISP G0/0
ISP server LAN	198.51.100.0/24: ISP .1, SRV .100	ISP G0/1

Порядок работы

1. Настройте интерфейсы и локальные шлюзы. В HQ и BR выдавайте DHCP на своих маршрутизаторах; DNS в обоих пулах – 10.39.10.10. У HQ-SRV статический адрес.
2. Включите OSPF area 0 на HQ, BR и EDGE для трёх внутренних /30 и пользовательских LAN. LAN-интерфейсы passive; outside EDGE G0/2 не включайте в OSPF.
3. На EDGE добавьте default route к 203.0.113.1. В OSPF-процессе EDGE добавьте `default-information originate`. В обычном варианте он объявляет default при наличии такого маршрута в таблице.
4. На EDGE G0/0 и G0/1 отметьте NAT inside, G0/2 – outside. ACL для PAT должна

выбирать обе пользовательские сети 10.39.10.0/24 и 10.39.20.0/24. Настройте overload на G0/2.
5. На внешнем Server-PT включите HTTP. ISP не включайте во внутренний OSPF и не добавляйте на нём маршруты к частным LAN: ответы PAT возвращаются к outside-адресу EDGE.
6. Для основного пути BR–HQ назначьте cost 10 на обоих концах; на BR–EDGE cost 100. На HQ–EDGE cost 10. Проверьте, что выбранный путь соответствует сумме стоимостей.

Проверка результата

HQ и BR открывают внутренний сайт по имени и внешний сайт по IP. На HQ/BR виден изученный default OSPF, на EDGE – NAT-записи обеих LAN. При отказе HQ–BR внутренняя связь после сходимости проходит через EDGE; внешний доступ филиала продолжает работать. Проверяйте это новыми сеансами и сохранёнными ACL.

Управляемая неисправность

Выключите HQ G0/2. Доказательно покажите новый next hop и путь. Затем снимите NAT inside только с EDGE G0/1: резервный маршрут есть, но внешние сеансы филиала могут перестать транслироваться. Исправьте стороны NAT, сохранив правильный OSPF.

Что сдать и объяснить

Сдайте адресный план, OSPF-соседства, default route, два NAT-сеанса и два маршрута до/после отказа. Почему наличие OSPF-маршрута через резервную линию не гарантирует, что на ней правильно настроен NAT?

Самостоятельное усложнение. Опубликуйте HQ-SRV через отдельный внешний адрес. Поскольку outside /30 содержит только .1 и .2, не назначайте дополнительный адрес внутри этого /30. Выделите отдельный документальный префикс, например 192.0.2.100/32, и статический маршрут ISP к EDGE .2. Настройте static NAT и отдельную политику публикации.

Лабораторная 40. Итоговый проект: сеть учебного центра

Время: 10–14 часов. **Подготовка:** полный курс.

Какую проблему решает работа

Итоговая работа проверяет, можете ли вы самостоятельно спроектировать, настроить, проверить и описать сеть учебного центра. Она решает проблему перехода от выполнения заданий по образцу к самостоятельной инженерной работе.

Что вы научитесь делать

Самостоятельно пройти цикл «требования – проект – настройка – проверка – отказ – восстановление».

Стенд и адреса

Учебный центр имеет головную площадку и филиал. На головной площадке 60 рабочих мест преподавателей, 100 студентов, 20 гостей, 10 серверов и 8 управляемых устройств; в филиале 25 рабочих мест и 4 устройства инфраструктуры. Для LAN выдан 10.40.0.0/22. Для внутренних транзитов выдан отдельный 10.40.4.0/27. В стенде достаточно 2–3 представителей каждой группы. IPv6-пул 2001:db8:40::/48; по /64 на пользовательский сегмент.

Требования к результату

VLAN преподавателей, студентов, гостей, серверов и управления разделены. Есть централизованный DHCP с relay, DNS и внутренний веб-портал. Между площадками OSPF с резервным путём. Во внешнюю учебную сеть используется PAT. Гости достигают только разрешённых DNS/HTTP и своего шлюза, сохраняя DHCP. Администратор управляет IOS-узлами по SSH. Есть Syslog, согласованное время и проверенная резервная копия. Преподавательская и серверная LAN поддерживают IPv6 с маршрутом между ними. WLAN для гостей входит в гостевую политику, если выбранная AP-модель поддерживает нужную привязку VLAN.

Порядок работы

1. Преобразуйте количество устройств в VLSM: добавьте шлюзы и разумный запас. Составьте таблицу блоков и проверьте отсутствие пересечений. Транзиты не включайте в LAN-пул.
2. Выберите архитектуру шлюзов: router-on-a-stick или L3-коммутатор. Обоснуйте её для учебной модели и количества интерфейсов. Укажите все порты и native/allowed VLAN.
3. Постройте минимальный рабочий путь одного клиента к серверу. Затем добавляйте остальные VLAN и филиал. На каждом шаге сохраняйте версию.
4. Реализуйте адресные службы и внутреннюю маршрутизацию. Включите резерв и проведите управляемый отказ до ACL/NAT, чтобы подтвердить отдельно маршруты.
5. Настройте гостевую политику, внешний PAT, SSH и обслуживание. Проверьте и разрешения, и запреты. Для WLAN допустима отдельная AP-модель в access VLAN гостей; не заменяйте её случайным домашним маршрутизатором с дополнительным NAT.
6. Добавьте IPv6 в две LAN. IPv4 ACL не защищает автоматически IPv6: либо отдельно реализуйте поддерживаемые IPv6-политики, либо явно ограничьте

IPv6 только двумя негостевыми LAN и зафиксируйте это в проекте.
7. Проведите итоговую приёмку и восстановление из резервной копии в отдельном клоне проекта. Сделайте журнал решения трёх неисправностей разных уровней.

Проверка результата

Минимальная приёмка включает 15 проверок: DHCP каждой пользовательской VLAN (минимум 4), DNS, HTTP по имени, разрешённый гостевой HTTP, запрещённый доступ гостей к управлению и преподавателям, SSH администратора, два внешних PAT-сеанса, отказ основного межплощадочного пути, Syslog, восстановление конфигурации и IPv6 между двумя LAN. Если несколько подпунктов объединены в одну строку отчёта, приложите отдельные наблюдения каждого.

Управляемая неисправность

Три обязательных сценария: неправильный allowed VLAN; отсутствующий обратный маршрут или OSPF-реклама; ошибка в DNS/DHCP/ACL. К каждому приложите рабочую, сломанную и исправленную версии либо понятный журнал изменений.

Что сдать и объяснить

Комплект сдачи: финальный .pkt, схема с портами, IPv4/IPv6-план, матрица политик, конфигурации, журнал приёмки, резервные копии и краткая записка с ограничениями модели. Защита проходит на собственном стенде: преподаватель выбирает два разрешённых и два запрещённых потока, а студент объясняет ожидаемый путь до запуска теста.

Самостоятельное усложнение. Добавьте второй сервер, раздельные OSPF-области или контроллер WLAN. Сначала полностью выполните обязательные требования. BGP, MPLS, производственная VPN и сложные firewall не являются обязательной частью этого курса: их модели требуют отдельного подбора среды и самостоятельной проверки.

Часть VI

Отказоустойчивость, защита и мониторинг

Глава 37

Расширенные работы 41–48

37.1 Как читать эти работы

Работы 41–48 расширяют основной курс темами, которые в настоящих сетях встречаются почти всегда: резервный шлюз, защита коммутатора от посторонних серверов, быстрая сходимость остовного дерева, мониторинг по SNMP, защищённый туннель, централизованная проверка прав, телефония с приоритетом голоса и обмен маршрутами между организациями. Работы построены по тому же плану, что и 01–40: сначала названа проблема, затем цель, схема, шаги, проверка, неисправность, отчёт и расширение.

Статус проверки

Все восемь схем собраны и прогнаны в Packet Tracer 8.2.1 (13 октября 2026 года): команды из листингов введены в консоли, результаты сверены командами `show`, ping и tracert, рисунки схем сняты с программы. Там, где учебная модель устройства не принимает команду настоящего IOS, в тексте указана замена и помечено, что именно в Packet Tracer недоступно. Подробности — в разделе «Что проверено в Packet Tracer» в конце книги.

Три правила для вставки листингов

1) Листинг, который вставляют в консоль целиком, начинайте строкой `enable`: после перезагрузки или долгого простоя консоль возвращается в режим `Router>`. 2) Добавьте в начало `no ip domain-lookup`: иначе опечатка или слово `end` в пользовательском режиме запускают DNS-поиск на полминуты. 3) Если в листинге есть `exit`, а предыдущая команда была отвергнута, `exit` выбросит вас на уровень выше, и все остальные строки уйдут не туда. После вставки всегда смотрите `show running-config`.

Лабораторная 41. Резервный шлюз: HSRP

Время: 90–120 минут. **Подготовка:** 25–26 (OSPF), 09.

Какую проблему решает работа

Все компьютеры сети указывают один шлюз по умолчанию. Если маршрутизатор-шлюз выходит из строя, работа всего отдела останавливается, хотя второй маршрутизатор исправен. Работа решает проблему *единой точки отказа шлюза*: два маршрутизатора делят один виртуальный адрес, и при отказе активного второй принимает его на себя без перенастройки компьютеров.

Что вы научитесь делать

Настроить HSRP на двух маршрутизаторах, увидеть переход роли `Active` при отказе и вернуть её обратно.

Стенд и адреса

Устройства: R1, R2, R3 (маршрутизаторы 2911), S1 (2960), PC1. Адреса: LAN 192.168.10.0/24, виртуальный шлюз 192.168.10.1, R1 .2, R2 .3, PC1 .50. Каналы R1–R3 10.0.13.0/30 (R1 .1, R3 .2) и R2–R3 10.0.23.0/30 (R2 .1, R3 .2). На R3 интерфейс Loopback0 198.51.100.1/32 изображает «внешний мир».

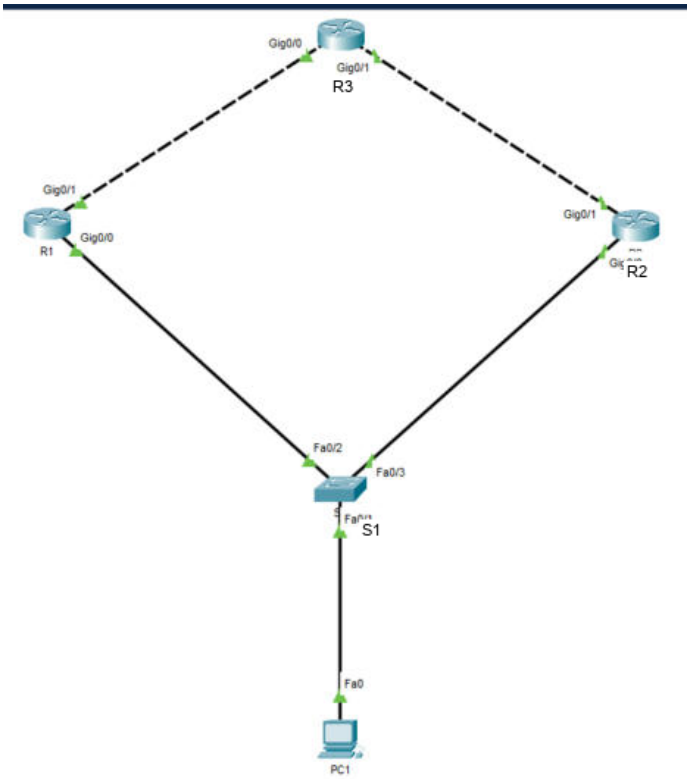


Схема работы 41: R3 вверху, R1 и R2 делят виртуальный шлюз, PC1 за S1.

Таблица соединений. Если соединять инструментом автоподбора кабеля в этом порядке, порты совпадут с таблицей.

Устройство	Порт	Устройство	Порт	Кабель
PC1	FastEthernet0	S1	FastEthernet0/1	прямой
R1	GigabitEthernet0/0	S1	FastEthernet0/2	прямой
R2	GigabitEthernet0/0	S1	FastEthernet0/3	прямой
R1	GigabitEthernet0/1	R3	GigabitEthernet0/0	перекрёстный
R2	GigabitEthernet0/1	R3	GigabitEthernet0/1	перекрёстный

Порядок работы

Настройте адреса интерфейсов, затем OSPF (одна область) на R1, R2, R3: R1 и R2 объявляют сеть 192.168.10.0/24 и свой канал к R3; интерфейсы G0/0 делают пассивными; R3 объявляет оба канала и Loopback0. На PC1 задайте шлюз 192.168.10.1. Затем HSRP:

```
! R1
interface gigabitEthernet 0/0
 ip address 192.168.10.2 255.255.255.0
 standby 1 ip 192.168.10.1
 standby 1 priority 110
 standby 1 preempt
 no shutdown
! R2
interface gigabitEthernet 0/0
 ip address 192.168.10.3 255.255.255.0
 standby 1 ip 192.168.10.1
 no shutdown
```

Проверка результата

`show standby brief` на R1 показывает `Active`, на R2 — `Standby`. Эталонный вывод на R1:

```
Interface   Grp  Pri P State   Active     Standby       Virtual IP
Gig0/0      1    110 P Active  local      192.168.10.3  192.168.10.1
```

С PC1 доступен адрес 198.51.100.1 (первый запрос может потеряться на ARP). Запустите на PC1 `ping -t 198.51.100.1`, выключите на R1 интерфейс G0/0 командой `shutdown`: в консоли R1 появится `\%HSRP-6-STATECHANGE: ... Grp 1 state Active -> Init`, а R2 через несколько секунд станет `Active`. При проверке потерялись два ответа подряд. Включите интерфейс обратно: R1 проходит состояния `Speak -> Standby -> Active` (около 20–25 секунд), и благодаря `preempt` снова становится активным; при возврате ещё несколько ответов теряются вперемежку с удачными. Итог проверки: 123 запроса, 7 потеряно.

Управляемая неисправность

На R2 уберите группу 1 и задайте группу 2 с тем же адресом: `no standby 1 ip 192.168.10.1`, затем `standby 2 ip 192.168.10.1`. Роли перестают согласовываться: на обоих маршрутизаторах `Active`, а в консоли R2 сыплются сообщения `\%IP-4-DUPADDR: Duplicate address 192.168.10.1 on GigabitEthernet0/0, sourced by 0000.0C07.AC01`. Последние две цифры виртуального MAC-адреса — номер группы: так и находят расхождение. Исправьте номер группы.

Что сдать и объяснить

Сдайте вывод `show standby brief` до отказа, во время отказа и после возвращения; число потерянных пакетов. Объясните, зачем нужен `preempt` и когда он вреден.

Самостоятельное усложнение. Добавьте отслеживание канала к R3. Учебная модель 2911 принимает команду только в такой форме — с пробелом между типом и номером интерфейса и без величины уменьшения (по умолчанию 10):

```
! R1
interface gigabitEthernet 0/0
 standby 1 track GigabitEthernet 0/1
! R2: чтобы перехватить роль, приоритет между 100 и 110 и preempt
interface gigabitEthernet 0/0
 standby 1 priority 105
 standby 1 preempt
```

Форма `standby 1 track GigabitEthernet0/1 20` из документации IOS в Packet Tracer отвергается. Выключите на R1 интерфейс `G0/1`: приоритет R1 падает до 100, R2 с приоритетом 105 становится `Active`; после `no shutdown` R1 возвращает роль.

Лабораторная 42. Защита от чужого DHCP-сервера

Время: 90 минут. **Подготовка:** 15–16, 20.

Какую проблему решает работа

Любой подключённый к порту коммутатора компьютер или забытый домашний маршрутизатор может начать раздавать адреса и шлюз вместо настоящего DHCP-сервера. Клиенты получают чужие настройки и теряют связь или отправляют трафик злоумышленнику. Работа решает проблему *недоверенных DHCP-ответов*: коммутатор принимает ответы сервера только с доверенного порта.

Что вы научитесь делать

Включить DHCP Snooping на коммутаторе, показать, что ложный сервер перестаёт влиять на клиентов, и ограничить число адресов на порту.

Стенд и адреса

Устройства: S1 (2960), SRV-DHCP (сервер, 192.168.50.10/24, пул с 192.168.50.100, шлюз 192.168.50.1), ROGUE (второй сервер, 192.168.50.66/24, «чужой»: раздаёт адреса с 192.168.50.200 и шлюз 192.168.50.66 — то есть себя), PC1, PC2. Все порты — в VLAN 50. SRV-DHCP подключён к `FastEthernet0/24`, ROGUE — к `FastEthernet0/2`, PC1 — к `FastEthernet0/1`, PC2 — к `FastEthernet0/3`. Порт `Fa0/24` выбирают вручную: кабель «Copper Straight-Through», щелчок по серверу, затем по коммутатору и пункт `FastEthernet0/24` в списке.

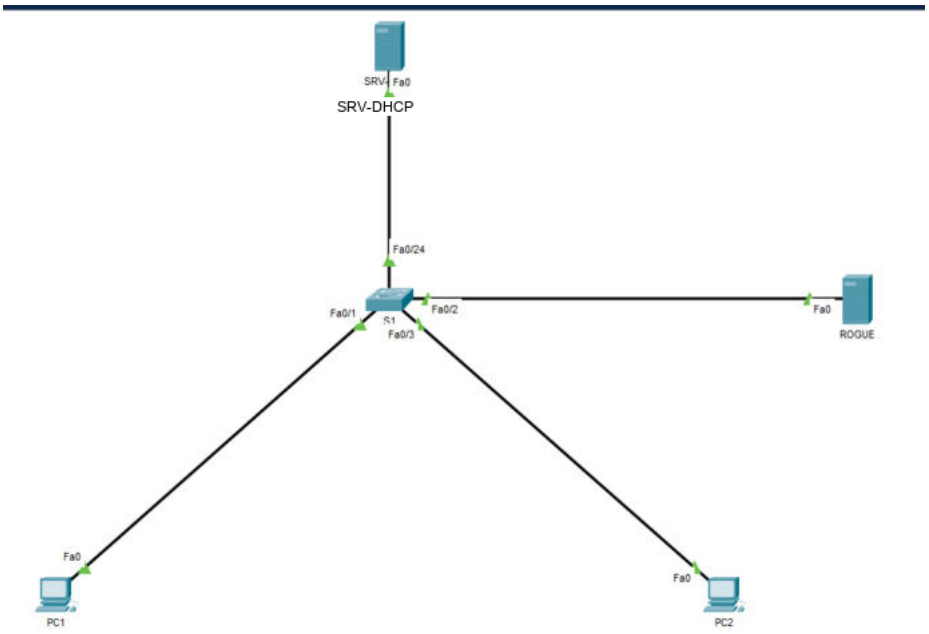


Схема работы 42: настоящий сервер на Fa0/24, чужой — на Fa0/2.

Почему чужой пул в той же подсети

Сервер Packet Tracer раздаёт только пул, совпадающий с подсетью его собственного интерфейса: сервер с адресом 192.168.50.66 и пулом 10.66.0.0/24 не отвечает вовсе. Поэтому чужой сервер здесь отличается не сетью, а *шлюзом*: клиент, получивший шлюз 192.168.50.66, отправит весь трафик злоумышленнику. Именно по шлюзу в выводе `ipconfig` и узнают, кто ответил.

Порядок работы

Настройте серверы: `Desktop / IP Configuration` — адреса; `Services / DHCP` — в пуле `serverPool` поля Default Gateway, Start IP Address (четыре окошка) и Maximum Number of Users, кнопка `Save`, затем переключатель `Service: On`. Ошибка в поле шлюза даёт окно «Invalid IP address entered for gateway».

Сначала *без защиты*. Адрес на ПК удобнее запрашивать командой `ipconfig /renew` в `Command Prompt`: она печатает полученные адрес и шлюз; дождитесь завершения (до 15 секунд). Когда включены оба сервера, в Packet Tracer стабильно отвечает настоящий. Чтобы увидеть эффект чужого, временно выключите службу DHCP на SRV-DHCP (`Services / DHCP / Off`) и повторите `ipconfig /renew`: придёт адрес 192.168.50.200 и шлюз 192.168.50.66. Включите службу обратно. Затем включите защиту:

```
enable
configure terminal
vlan 50
exit
interface range fastEthernet 0/1 - 3
 switchport mode access
 switchport access vlan 50
 ip dhcp snooping limit rate 10
exit
interface fastEthernet 0/24
 switchport mode access
 switchport access vlan 50
 ip dhcp snooping trust
exit
ip dhcp snooping
ip dhcp snooping vlan 50
no ip dhcp snooping information option
end
show ip dhcp snooping
show ip dhcp snooping binding
```

Снова выполните `ipconfig /renew` на PC1 и PC2.

Проверка результата

`show ip dhcp snooping` перечисляет порты: `FastEthernet0/24 yes unlimited`, остальные `no 10`. Клиенты получают адреса от настоящего сервера (шлюз 192.168.50.1). `show ip dhcp snooping binding` перечисляет выданные адреса:

```
MacAddress          IpAddress        Lease(sec)  Type           VLAN  Interface
00:01:64:C8:A0:C8   192.168.50.100   86400       dhcp-snooping  50
FastEthernet0/1
00:60:5C:C2:DD:A7   192.168.50.204   86400       dhcp-snooping  50
FastEthernet0/3
```

Решающая проверка: выключите службу DHCP на SRV-DHCP и выполните `ipconfig /renew` — теперь ответ `DHCP request failed`: чужой сервер на недоверенном порту отброшен коммутатором. Включите службу обратно. Строка `no ip dhcp snooping information option` нужна, потому что между клиентом и сервером нет ретранслятора, а сервер может отбрасывать запросы с добавленной коммутатором опцией 82.

Управляемая неисправность

Снимите `ip dhcp snooping trust` с порта сервера. Клиенты перестают получать адреса вообще (`DHCP request failed`): ответы настоящего сервера теперь то-

же считаются недоверенными. Верните доверие.

Что сдать и объяснить

Сдайте вывод `show ip dhcp snooping binding` и объяснение, какие порты доверенные и почему только они. Укажите, что произойдёт при подключении второго коммутатора к порту без доверия.

Самостоятельное усложнение. Добавьте на порты `port-security` (`switchport port-security, maximum 2, violation restrict`) и проверьте счётчики командой `show port-security interface fastEthernet 0/1`: `Port Status: Secure-up`, `Violation Mode: Restrict`, `Maximum MAC Addresses: 2`.

Лабораторная 43. Быстрое остовное дерево и корневой мост

Время: 90–120 минут. **Подготовка:** 17–19.

Какую проблему решает работа

Классический STP при обрыве канала восстанавливает связь за 30–50 секунд, а корневой мост выбирается по наименьшему MAC-адресу, то есть случайно: роль может достаться самому старому коммутатору в углу. Работа решает две проблемы: *медленную сходимость* и *непредсказуемый корень*.

Что вы научитесь делать

Включить Rapid PVST+, назначить корень и запасной корень, защитить порты доступа.

Стенд и адреса

Устройства: S1, S2, S3 (2960) в треугольнике, PC1 у S2 и PC2 у S3 в VLAN 10 (адреса 192.168.10.11 и .12). Каналы S1–S2 (`Fa0/1–Fa0/1`), S1–S3 (`Fa0/2–Fa0/1`), S2–S3 (`Fa0/2–Fa0/2`), транки с VLAN 10. PC1 на `Fa0/10` S2, PC2 на `Fa0/10` S3 (порт выбирают вручную в списке).

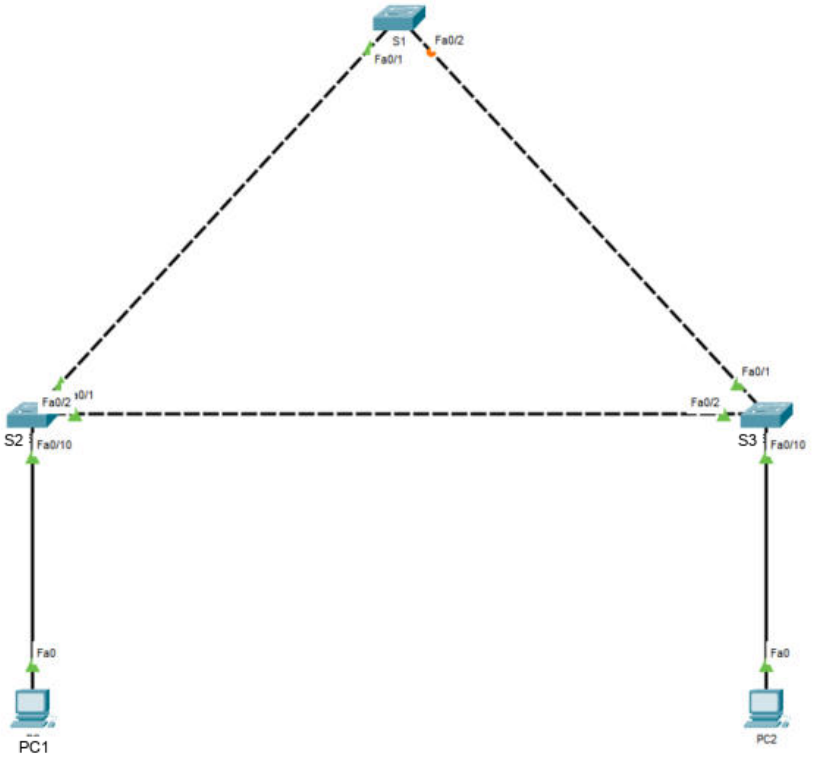


Схема работы 43: треугольник коммутаторов, компьютеры на Fa0/10.

Порядок работы

Настройте VLAN 10 и транки. Пока транк настроен только с одной стороны, коммутаторы пишут `\%SPANTREE-2-RECV\_PVID\_ERR` и `BLOCK\_PVID\_LOCAL` для VLAN 1 — это проходит само. Посмотрите `show spanning-tree vlan 10` на всех коммутаторах: кто корень и какой порт заблокирован (при проверке корнем оказался S2 с MAC 0001.C7D6.6948 — наименьшим из трёх; на S3 порт `Fa0/1` был `Altn BLK`; строка `Spanning tree enabled protocol ieee`). Затем:

```
! на всех трёх коммутаторах
spanning-tree mode rapid-pvst
! S1 - основной корень, S2 - запасной
spanning-tree vlan 10 root primary
spanning-tree vlan 10 root secondary
! на портах доступа S2 и S3
interface fastEthernet 0/10
 spanning-tree portfast
 spanning-tree bpduguard enable
```

Первую из команд `root` вводят на S1, вторую — на S2. Команда `portfast` печатает предупреждение — это нормально.

Проверка результата

`show spanning-tree vlan 10` на S1 содержит `Spanning tree enabled protocol rstp`, `Priority 24586` и `This bridge is the root`; у S2 приоритет 28682. Порты S1 после смены корня около 20 секунд находятся в состоянии `LRN`. Запустите `ping -t` с PC1 на PC2 и оборвите *рабочий* канал S1–S2 (на S2: `interface fa0/1`, `shutdown`). Обрыв канала S2–S3 ничего не покажет — он и так заблокирован. При проверке за время обрыва и восстановления не потерялось ни одного ответа (79 отправлено, 0 потеряно): трафик сразу пошёл через S2–S3.

Особенность Packet Tracer

После `no shutdown` канала коммутаторы могут «разойтись»: S1 и S2 одновременно пишут `This bridge is the root`, а S3 видит корнем S2. Связь при этом есть. Лечится переключением режима на корневом коммутаторе: `spanning-tree mode pvst`, затем снова `spanning-tree mode rapid-pvst`; после этого S2 показывает `Root ID Priority 24586` и `Fa0/1 Root FWD`.

Управляемая неисправность

Задайте на S3 `spanning-tree vlan 10 priority 0`. Через 20 секунд S3 сообщает `This bridge is the root` (приоритет 10), путь трафика меняется. Верните порядок командой `no spanning-tree vlan 10 priority` и включите `spanning-tree guard root` на портах S1, обращённых к S2 и S3 (`interface range fastEthernet 0/1 - 2`). Повторите неисправность: теперь S1 остаётся корнем, его порт `Fa0/2` переходит в `Desg BLK`, а в консоли — `\%SPANTREE-2-ROOTGUARDBLOCK: Port 0/2 tried to become non-designated in VLAN 10. Moved to root-inconsistent state.` Команда `show spanning-tree inconsistentports` в Packet Tracer показывает 0 — ориентируйтесь на сообщение и состояние `BLK`. Снова верните приоритет S3.

Что сдать и объяснить

Сдайте таблицу «коммутатор — роль — приоритет — состояние портов» до и после изменений и время простоя при обрыве. Объясните назначение `portfast` и `bpduguard`.

Самостоятельное усложнение. Порт `Fa0/10` занят компьютером, поэтому настройте так же порт `Fa0/11` на S2 (`switchport mode access`, `switchport access vlan 10`, `portfast`, `bpduguard enable`) и подключите к нему перекрёстным кабелем четвёртый коммутатор. В консоли S2: `\%SPANTREE-2-BLOCK\_BPDUGUARD: Received BPDU on port FastEthernet0/11 with BPDU Guard enabled. Disabling port.` и `\%PM-4-ERR\_DISABLE; show interfaces fa0/11 | include line` даёт `(err-disabled)`. Удалите чужой коммутатор, затем верните порт командами `shutdown` и `no shutdown`.

Лабораторная 44. Мониторинг по SNMP

Время: 60–90 минут. **Подготовка:** 36, модуль 17 практикума.

Какую проблему решает работа

Администратор узнаёт о проблемах от пользователей, потому что никто не измеряет состояние устройств. Чтобы мониторинг заметил перегрузку или отказ раньше людей, он должен уметь спрашивать устройства о состоянии стандартным способом. Работа решает проблему *отсутствия измерений* и знакомит с SNMP.

Что вы научитесь делать

Включить SNMP на маршрутизаторе, прочитать значения с компьютера администратора через MIB Browser и убедиться, что сообщество работает как пароль.

Стенд и адреса

Устройства: R1 (192.168.1.1/24), S1, PC-ADM (192.168.1.20), SRV (192.168.1.10). Все адреса одной сети; PC-ADM подключён к `Fa0/1` S1, SRV — к `Fa0/2`, R1 — к `Fa0/3`.

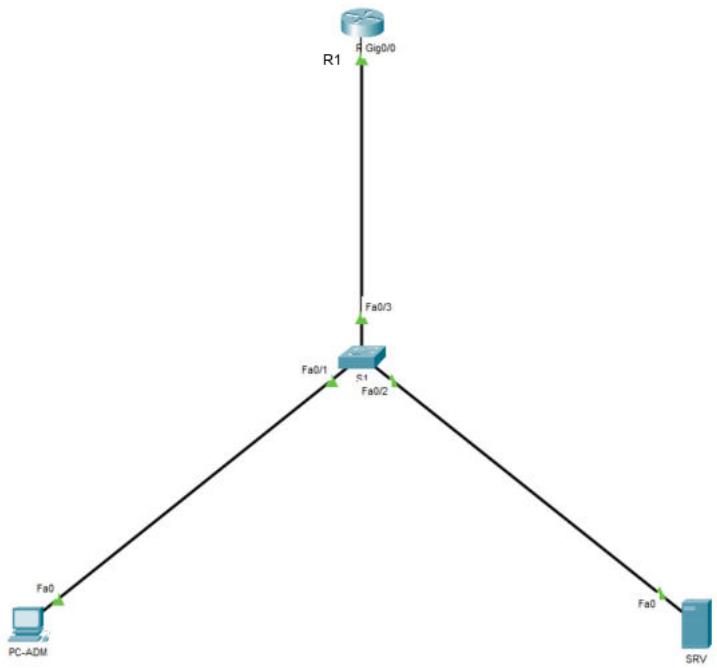


Схема работы 44.

Границы учебной модели

На маршрутизаторе 2911 в Packet Tracer команда `snmp-server` знает только `community <строка> ro|rw`. Ограничение по списку доступа (`ro 10`), `location`, `contact`, `enable traps` и `host` отвергаются как `Invalid input`. Операция `Set` в MIB Browser не работает (сразу «Value for set dialog cannot be empty»). Поэтому здесь проверяют чтение, а защиту по адресу и ловушки обсуждают как то, что есть на настоящем IOS.

Порядок работы

На R1:

```
snmp-server community SEVER-RO ro
snmp-server community SEVER-RW rw
```

На PC-ADM откройте `Desktop / MIB Browser`. Кнопка `Advanced...`: Address 192.168.1.1, Port 161, Read Community `SEVER-RO`, Write Community `SEVER-RW`, SNMP Version `v2`, `OK`. В поле OID введите `.1.3.6.1.2.1.1.1.0` (sysDescr), операция `Get`, кнопка `GO`; затем `.1.3.6.1.2.1.1.5.0` (sysName). Для таблицы интерфейсов введите `.1.3.6.1.2.1.2.2.1.2` (ifDescr) и операцию `Get Bulk`. Тот же OID можно выбрать в дереве слева: двойным щелчком раскрывают `router\_std MIBs / .iso / .org / .dod / .internet / .mgmt / .mib-2 / .system`.

Проверка результата

В таблице Result: sysDescr — `Cisco IOS Software, C2900 Software (C2900-UNIVERSALK9-M), Version 15.1(4)M4...`, sysName — `R1` (сравните с `show version`); Get Bulk по ifDescr даёт `Vlan1`, `GigabitEthernet0/0`, `0/1`, `0/2`. При версии v1 кнопка Get Bulk отвечает «Version 1 does not support getbulk». Повторите те же запросы с SRV: с верным сообществом он получает те же ответы — без списка доступа сообщество остаётся единственным «паролем».

Управляемая неисправность

Измените в `Advanced...` Read Community на `public`: таблица результата остаётся пустой — ответа нет, сообщений об ошибке тоже. Верните верное значение.

Что сдать и объяснить

Сдайте считанные значения, настройки R1 и ответ: почему сообщество SNMP версии 2c нельзя считать надёжной защитой, что используют вместо него (SNMPv3) и как на настоящем IOS ограничивают доступ по адресу (`snmp-server community ... ro 10` со списком доступа 10).

Самостоятельное усложнение. Свяжите работу с модулем 17 практикума: перечислите, какие величины из мониторинга сервера (нагрузка, память, диски) вы запрашивали бы у маршрутизатора для тех же целей, и найдите в дереве MIB соответствующие ветви (`.interfaces`, `.ip`).

Лабораторная 45. Туннель GRE и защита IPsec

Время: 150–180 минут. **Подготовка:** 25–29, 31.

Какую проблему решает работа

Два филиала связаны через интернет, который не принадлежит организации: частные адреса там не маршрутизируются, а чужие маршрутизаторы видят трафик. Работа решает проблему *связи частных сетей через общую сеть*: GRE создаёт виртуальный канал между офисами, IPsec шифрует его содержимое.

Что вы научитесь делать

Построить туннель GRE между двумя офисами, пустить по нему трафик частных сетей и затем защитить его IPsec.

Стенд и адреса

Устройства: R1 (офис А), R2 (интернет-провайдер), R3 (офис Б) — все 2911, PC-A, PC-B. Сети: офис А 192.168.1.0/24 (R1 `G0/0`), офис Б 192.168.3.0/24 (R3 `G0/1`), R1–R2 203.0.113.0/30 (R1 `G0/1` .1, R2 `G0/0` .2), R2–R3 198.51.100.0/30 (R2 `G0/1` .1, R3 `G0/0` .2). Туннель 10.255.0.0/30: R1 .1, R3 .2. На R2 маршрутов в частные сети нет. Обратите внимание: внешний интерфейс R1 — `G0/1`, а R3 — `G0/0`.



Схема работы 45: R2 изображает провайдера.

Порядок работы

Настройте адреса и статические маршруты по умолчанию на R1 и R3 в сторону R2. Проверьте, что `ping 203.0.113.1` с R3 проходит, а пинг из сети А в сеть Б нет. Туннель (источник задают *именем интерфейса*: форму `tunnel source 203.0.113.1` учебная модель отвергает):

```
! R1
interface tunnel 0
 ip address 10.255.0.1 255.255.255.252
 tunnel source gigabitEthernet 0/1
 tunnel destination 198.51.100.2
exit
ip route 192.168.3.0 255.255.255.0 10.255.0.2
! R3 (зеркально)
interface tunnel 0
```

```
 ip address 10.255.0.2 255.255.255.252
 tunnel source gigabitEthernet 0/0
 tunnel destination 203.0.113.1
exit
ip route 192.168.1.0 255.255.255.0 10.255.0.1
```

Защита. На 2911 для команд crypto нужен пакет securityk9. Введите license boot module c2900 technology-package securityk9, на вопрос ACCEPT? [yes/no] ответьте yes, сохраните конфигурацию (copy running-config startup-config) и перезагрузите маршрутизатор командой reload, подтвердив вопрос Proceed with reload? [confirm] клавишей Enter *вручную* — в составе вставленного листинга подтверждение теряется, и перезагрузка не происходит. Перезагрузка длится около минуты; после неё show version | include security должен показать securityk9 в столбце текущего пакета, а crypto ? в режиме настройки — не только key. Затем на R1 (на R3 — зеркально: адрес пира 203.0.113.1, список доступа с переставленными адресами, crypto map на G0/0):

```
crypto isakmp policy 10
 encryption aes
 hash sha
 authentication pre-share
 group 2
exit
crypto isakmp key SEVER-KEY address 198.51.100.2
crypto ipsec transform-set TS esp-aes esp-sha-hmac
access-list 110 permit gre host 203.0.113.1 host 198.51.100.2
crypto map CMAP 10 ipsec-isakmp
 set peer 198.51.100.2
 set transform-set TS
 match address 110
exit
interface gigabitEthernet 0/1
 crypto map CMAP
```

Команда crypto ipsec transform-set в Packet Tracer не открывает подрежим, поэтому строки mode transport здесь нет (режим остаётся туннельным); если её ввести, следующий exit выбросит вас из режима настройки. После crypto map CMAP на интерфейсе появляется \%CRYPTO-6-ISAKMP_ON_OFF: ISAKMP is ON.

Проверка результата

До IPsec: PC-A достигает PC-B, а tracert 192.168.3.10 показывает три шага — 192.168.1.1, 10.255.0.2 (туннель), 192.168.3.10; маршрутизатор провайдера не виден. На R3 show ip route содержит C 10.255.0.0/30 ... Tunnel0 и S 192.168.1.0/24 ... via 10.255.0.1. После IPsec первый ping 10.255.0.1 с R3 может пройти с потерями (идёт согласование; в show crypto isakmp sa состояние MM_NO_STATE), второй — успешно, и show crypto isakmp sa показывает QM_IDLE ... ACTIVE на обеих сторонах. show crypto ipsec sa (без фильтра) показывает local ident (198.51.100.2/255.255.255.255/47/0), PERMIT, flags=\{origin_is_acl,\} и растущие \#pkts encaps / \#pkts decaps. show crypto isakmp policy перечисляет AES, SHA, Pre-Shared Key, group 2.

Управляемая неисправность

Укажите на R3 другой ключ: no crypto isakmp key SEVER-KEY address 203.0.113.1, затем crypto isakmp key WRONG-KEY address 203.0.113.1. Уже установленная сессия продолжает работать (время жизни 86400 секунд), а команд clear crypto isakmp и clear crypto sa в Packet Tracer нет — сбросьте её,

сняв и вернув crypto map на внешнем интерфейсе (`no crypto map CMAP`, `crypto map CMAP`). Теперь `ping 10.255.0.1` даёт 0 из 5, сессия не поднимается. Верните ключ и снова снимите-верните crypto map.

Что сдать и объяснить

Сдайте схему, настройки обеих сторон, вывод проверок и объяснение: что видит провайдер до и после IPsec, чем отличаются транспортный и туннельный режимы.

Самостоятельное усложнение. Замените статические маршруты через туннель на OSPF и сравните таблицы маршрутизации.

Лабораторная 46. Централизованная проверка прав: AAA

Время: 90–120 минут. **Подготовка:** 23, 34.

Какую проблему решает работа

Если на каждом устройстве свои локальные пароли, при увольнении сотрудника их приходится менять везде, а все администраторы имеют одинаковые права. Работа решает проблему *учёта и разграничения доступа*: вход проверяет центральный сервер, а помощник администратора получает только часть команд.

Что вы научитесь делать

Настроить аутентификацию через RADIUS с запасным локальным входом и привилегии разного уровня.

Стенд и адреса

Устройства: R1 (192.168.1.1/24), S1, SRV (RADIUS-сервер 192.168.1.10), PC-ADM (192.168.1.20). Подключения к S1 по прямым кабелям: PC-ADM — `Fa0/1`, SRV — `Fa0/2`, R1 — `Fa0/3`.

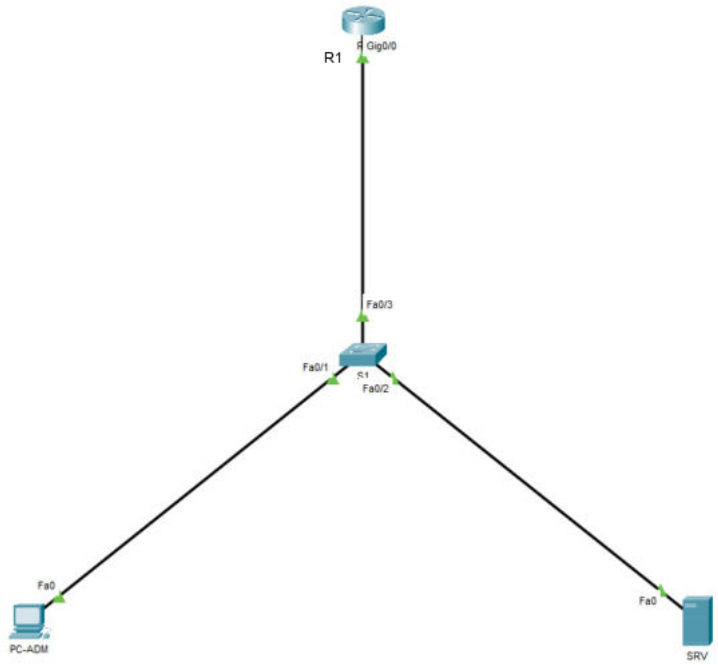


Схема работы 46.

Порядок работы

На SRV откройте `Services / AAA`: переключатель `Service: On`, порт RADIUS 1645 (по умолчанию). В блоке Network Configuration введите Client Name `R1`, Client IP 192.168.1.1 (щёлкните в середину поля), Secret `RADKEY`, тип `Radius` и нажмите `Add` справа от таблицы. В блоке User Setup добавьте пользователя `ivanov` с паролем `Ivan0v!`. На R1:

```
hostname R1
ip domain-name sever.local
username admin privilege 15 secret Adm1n!
enable secret Adm1n!
```

```
enable secret level 5 H3lp!
privilege exec level 5 ping
privilege exec level 5 show ip interface brief
crypto key generate rsa
1024
aaa new-model
radius-server host 192.168.1.10 key RADKEY
aaa authentication login VTY-AUTH group radius local
line vty 0 4
 login authentication VTY-AUTH
 transport input ssh
```

Строка `1024` — ответ на вопрос о длине ключа. В `show running-config` команда `radius-server host` превращается в `radius server 192.168.1.10`, а `privilege exec level 5 show ip interface brief` раскладывается на четыре строки — так и должно быть. Консоль не переводите на AAA, пока не проверили вход: ошибка на консоли оставит без доступа.

Почему уровни показывают через enable 5

Пользователь, вошедший через RADIUS, получает уровень 1: сервер Packet Tracer не передаёт уровень привилегий. А локальная запись `helpdesk privilege 5` по SSH не сработает: пока сервер RADIUS доступен и отвечает «нет такого пользователя», IOS к локальной базе не переходит (ответ `\% Login invalid`). Локальный вход включается только когда сервер *недоступен*. Поэтому уровень 5 выдают паролем `enable secret level 5`.

Проверка результата

С PC-ADM `ssh -l ivanov 192.168.1.1`, пароль `Ivan0v!`: приглашение `R1>`. Команда `enable 5` с паролем `H3lp!`: `show privilege` отвечает `Current privilege level is 5`; `show ip interface brief` и `ping 192.168.1.10` работают, `show running-config` отклоняется (`Invalid input`). Выйдите и выключите службу AAA на SRV: `ssh -l admin 192.168.1.1` с паролем `Adm1n!` после паузы в 10–15 секунд (тайм-аут RADIUS) входит по локальной записи, а `ssh -l ivanov` отвечает `\% Login invalid`. Включите службу обратно. Три неверных пароля подряд заканчиваются сообщением `\% Password: timeout expired!` и разрывом.

Управляемая неисправность

Укажите на R1 другой секрет RADIUS (`no radius-server host 192.168.1.10 key RADKEY`, затем `radius-server host 192.168.1.10 key WRONGKEY`): пользователь `ivanov` не входит (`\% Login invalid` после паузы), а `admin` входит. Найдите причину по расхождению секретов и исправьте.

Что сдать и объяснить

Сдайте протокол входов трёх проверок (ivanov через RADIUS, ivanov с уровнем 5, admin при выключенном сервере), перечень разрешённых на уровне 5 команд и объяснение, зачем нужен запасной локальный вход и чем он опасен.

Самостоятельное усложнение. Добавьте `aaa authorization exec default group radius local` и `aaa accounting exec default start-stop group radius` (обе команды принимаются) и опишите, какие дополнительные сведения собираются на настоящем сервере.

Лабораторная 47. IP-телефония: голосовая VLAN и приоритет

Время: 120–150 минут. **Подготовка:** 15–18, 20.

Какую проблему решает работа

Телефонный разговор чувствителен к задержкам: если голос идёт в одной очереди с загрузкой файлов, он рвётся. Кроме того, телефон и компьютер часто подключены к одному порту, а их нужно разделить. Работа решает две проблемы: *смешивание голоса и данных* и *отсутствие приоритета у голоса*.

Что вы научитесь делать

Настроить голосовую VLAN, получение настроек телефонами и приоритезацию голоса на выходе маршрутизатора.

Стенд и адреса

Устройства: R1 — маршрутизатор **2811** (у 2911 в Packet Tracer нет команды `telephony-service`; 2811 стоит в конце прокручиваемого ряда маршрутизаторов), S1 (2960), два IP-телефона 7960 (в палитре End Devices седьмой значок, не путать с соседним Network Controller), PC1 и PC2 за телефонами. VLAN 10 — данные, 192.168.10.0/24; VLAN 20 — голос, 192.168.20.0/24. Телефоны на `Fa0/1` и `Fa0/2` S1, R1 (`Fa0/0`) — на `Fa0/3` S1 (транк); компьютеры — в порт PC телефона.

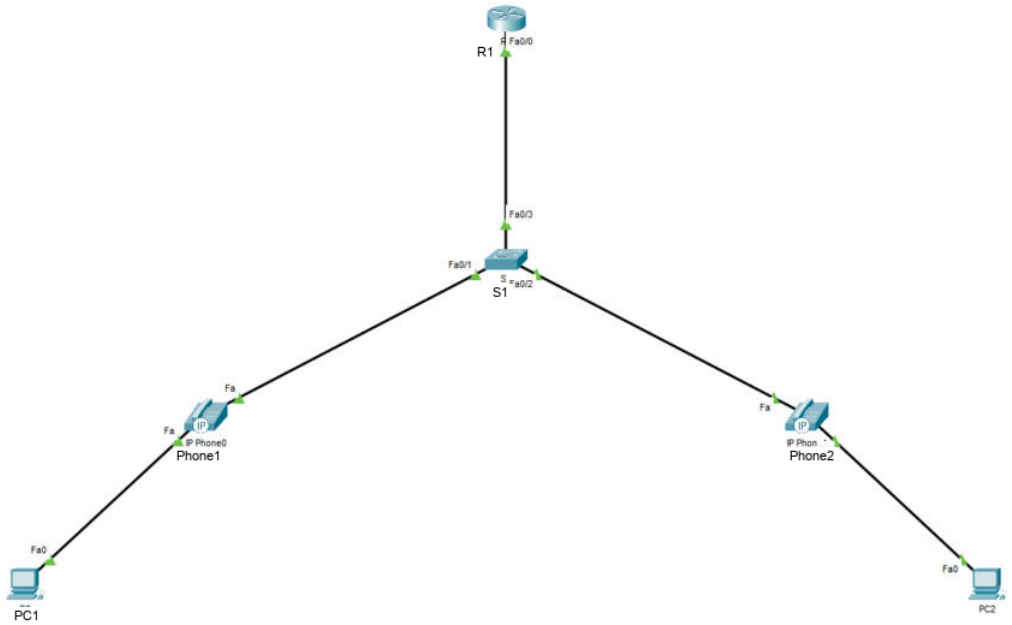


Схема работы 47: компьютеры подключены через телефоны.

Питание телефона

Пока телефон без питания, его канал к коммутатору красный. Откройте телефон, вкладка `Physical`: внизу справа картинка адаптера IP_PHONE_POWER_ADAPTER — перетащите её на круглый разъём питания на задней панели (слева внизу, левее трёх гнёзд RJ-45). Появится шнур, канал станет зелёным.

Порядок работы

Коммутатор и маршрутизатор:

```
! S1
vlan 10
vlan 20
interface range fastEthernet 0/1 - 2
 switchport mode access
 switchport access vlan 10
 switchport voice vlan 20
 spanning-tree portfast
interface fastEthernet 0/3
 switchport mode trunk
! R1 (2811)
interface fastEthernet 0/0
 no shutdown
interface fastEthernet 0/0.10
 encapsulation dot1Q 10
 ip address 192.168.10.1 255.255.255.0
interface fastEthernet 0/0.20
 encapsulation dot1Q 20
 ip address 192.168.20.1 255.255.255.0
ip dhcp pool VOICE
 network 192.168.20.0 255.255.255.0
 default-router 192.168.20.1
 option 150 ip 192.168.20.1
ip dhcp pool DATA
 network 192.168.10.0 255.255.255.0
 default-router 192.168.10.1
telephony-service
 max-ephones 5
 max-dn 5
 ip source-address 192.168.20.1 port 2000
 auto assign 1 to 5
ephone-dn 1
 number 1001
ephone-dn 2
 number 1002
```

Приоритет голоса на выходе маршрутизатора:

```
class-map match-all VOICE
 match ip dscp ef
policy-map WAN-OUT
 class VOICE
  priority 128
 class class-default
  fair-queue
interface fastEthernet 0/1
 service-policy output WAN-OUT
```

Проверка результата

После `ephone-dn` в консоли R1 появляется `\%LINK-3-UPDOWN: Interface ephone\_dsp DN 1.1, changed state to up`, а через полминуты — `\%IPPHONE-6-REGISTER: ephone-1 IP:192.168.20.2 ... has registered`. Команды `show ephone` в Packet Tracer нет; регистрацию смотрят в этих сообщениях и на вкладке GUI телефона: сначала «Configuring IP», затем номер в правом верхнем углу экрана (номера раздаются в порядке регистрации). Снимите трубку (щелчок по ней), наберите на клавиатуре номер второго телефона: на экране `To: 1001`, `Ring Out`; на втором телефоне `From: 1002`, после снятия

трубки — Connected. Компьютеры за телефонами получают адреса VLAN 10 (ipconfig /renew: 192.168.10.2, шлюз 192.168.10.1). show policy-map interface fastEthernet 0/1 показывает класс VOICE с Match: ip dscp ef (46), Queueing Strict Priority, Bandwidth 128 (kbps) и класс class-default с Flow Based Fair Queueing; счётчики нулевые, пока через Fa0/1 нет трафика.

Управляемая неисправность

Уберите switchport voice vlan 20 на Fa0/1: телефон оказывается в сети данных, не находит option 150, на экране снова «Configuring IP», регистрации нет. Верните настройку — через полминуты телефон регистрируется заново.

Что сдать и объяснить

Сдайте таблицу VLAN и адресов, настройку телефонии и вывод проверок. Объясните, что такое DSCP EF и почему приоритет применяют на медленном канале.

Самостоятельное усложнение. Добавьте телефон в третьей VLAN и звонок через шлюз с изменением номера.

Лабораторная 48. Обмен маршрутами между организациями: основы BGP

Время: 120 минут. **Подготовка:** 25–27.

Какую проблему решает работа

Внутри организации достаточно OSPF, но между организациями и провайдерами протоколы внутренней маршрутизации не применяют: у каждой стороны свои правила и она не доверяет соседу. Работа решает проблему *обмена маршрутами между независимыми сетями* с помощью протокола BGP.

Что вы научитесь делать

Установить сессии eBGP между тремя автономными системами, объявить свои сети и проследить путь.

Стенд и адреса

Устройства: R1 (AS 65001, сеть 192.168.1.0/24 на `G0/0`), R2 (AS 65002, транзитная; свою сеть 192.168.2.0/24 изобразите интерфейсом Loopback0 192.168.2.1/24), R3 (AS 65003, сеть 192.168.3.0/24 на `G0/1`), PC1 (192.168.1.10) за R1 и PC3 (192.168.3.10) за R3. Каналы R1–R2 10.0.12.0/30 (R1 `G0/1` .1, R2 `G0/0` .2), R2–R3 10.0.23.0/30 (R2 `G0/1` .1, R3 `G0/0` .2). Адреса и интерфейсы настройте самостоятельно по схеме.



Схема работы 48: три автономные системы.

Порядок работы

```
! R1
router bgp 65001
 neighbor 10.0.12.2 remote-as 65002
 network 192.168.1.0 mask 255.255.255.0
! R2
router bgp 65002
 neighbor 10.0.12.1 remote-as 65001
 neighbor 10.0.23.2 remote-as 65003
 network 192.168.2.0 mask 255.255.255.0
! R3
router bgp 65003
 neighbor 10.0.23.1 remote-as 65002
```

```
network 192.168.3.0 mask 255.255.255.0
```

Проверка результата

Примерно через минуту после настройки обеих сторон в консоли появляется `\%BGP-5-ADJCHANGE: neighbor 10.0.12.2 Up`. `show ip bgp` на R1 содержит сеть 192.168.3.0/24 с путём `65002 65003`:

```
   Network          Next Hop   Metric LocPrf Weight Path
*> 192.168.1.0/24   0.0.0.0         0         32768 i
*> 192.168.2.0/24   10.0.12.2       0             0 65002 i
*> 192.168.3.0/24   10.0.12.2       0             0 65002 65003 i
```

В `show ip bgp summary` учебная модель вместо числа префиксов или состояния `Idle/Active` печатает в столбце `State/PfxRcd` число 4 — судите о сессии по счётчику `MsgRcvd` и по сообщениям в консоли. `show ip route bgp` показывает `B 192.168.3.0/24 [20/0] via 10.0.12.2`. PC1 достигает PC3; `tracert` проходит 192.168.1.1, 10.0.12.2, 10.0.23.2.

Управляемая неисправность

Укажите на R3 неверный номер автономной системы соседа (`no neighbor 10.0.23.1 remote-as 65002`, затем `neighbor 10.0.23.1 remote-as 65009`). В консоли R3: `\%BGP-5-ADJCHANGE: neighbor 10.0.23.1 Down Neighbor deleted`, затем повторяющиеся `\%BGP-3-NOTIFICATION: sent to neighbor 10.0.23.1 2/2 (peer in wrong AS)`; у R1 сеть 192.168.3.0/24 пропадает. Найдите причину по этим сообщениям и исправьте.

Что сдать и объяснить

Сдайте таблицы BGP на всех маршрутизаторах и объяснение: чем путь AS отличается от метрики OSPF, и почему BGP нельзя «просто включить» на границе с провайдером без фильтров.

Самостоятельное усложнение. В Packet Tracer у команды `neighbor` есть только `remote-as` и `next-hop-self`; списков префиксов (`ip prefix-list`, `neighbor ... prefix-list in`) нет. Покажите, зачем нужны фильтры, без них: объявите на R2 служебный канал (`network 10.0.12.0 mask 255.255.255.252`) и убедитесь, что этот частный префикс появился в таблицах R1 и R3; затем уберите объявление. Опишите, как на настоящем IOS такие префиксы отфильтровали бы списком `prefix-list` на входе.

Часть VII

Системы в сети: проекты на стыке двух книг

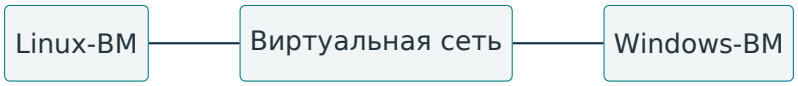
Глава 38

От модели сети к настоящей службе

38.1 Два стенда и одна проверяемая задача

Packet Tracer объясняет движение сообщений через модели устройств. Виртуальные машины дают настоящие процессы, файлы, журналы и права. Одинаковая подсеть на двух рисунках не соединяет эти среды физически. Мы используем два самостоятельных стенда: сетевую модель и виртуальную сеть гипервизора, а затем сопоставляем их поведение.

Для общей практики у каждой ВМ остаётся первый адаптер NAT для установки программ. Второй адаптер подключается к одной **внутренней сети** гипервизора без выхода наружу. Если гипервизор предлагает Host-only, сеть также включает хост; не путайте это с полной изоляцией. В любом варианте обе ВМ должны быть подключены к одной сети одного типа. Подключение к рабочей сети через bridge для этих работ не требуется.



Участник	Учебный адрес	Назначение
Linux-сервер	192.168.56.10/24	SSH, файлы и веб-служба.
Windows-клиент	192.168.56.20/24	Проверка служб и прав пользователей.
Сеть	192.168.56.0/24	Только второй адаптер, без шлюза по умолчанию.

Если этот диапазон уже используется на хосте или в другой виртуальной сети, выберите свободный частный /24 и последовательно замените адреса во всех таблицах и правилах I01–I05. Два адаптера одной ВМ не должны оказаться в одной IP-подсети этого опыта. На втором интерфейсе не нужен отдельный внешний DNS: для числового обращения он не используется.

Лабораторная I01. Две ОС в одной учебной сети

Время: 105–120 минут. **Подготовка:** модули 11–12 практикума; сетевые 01–04.

Какую проблему решает работа

Модель в Packet Tracer — не настоящая сеть: реальные системы устроены сложнее, и ошибки в них выглядят иначе. Работа решает проблему перехода от модели к практике: две настоящие виртуальные машины связываются по сети, и вы сопоставляете реальное TCP-соединение с тем, что видели в симуляции.

Что вы научитесь делать

Получить реальное TCP-соединение между ВМ и сопоставить его с моделью.

Стенд и адреса

Выключите ВМ, добавьте вторые адаптеры в одну изолированную сеть через настройки гипервизора и запустите. Сверьте виртуальные MAC с выводом ip link и Get-NetAdapter, чтобы отличить учебный адаптер от NAT.

Порядок работы

В Ubuntu Server используем Netplan: YAML-файл описывает постоянные настройки, отступы задаются пробелами. Скопируйте текущие файлы /etc/netplan в отдельное место. После проверки реального имени второго интерфейса создайте /etc/netplan/70-course-lab.yaml; enp0s8 ниже – пример, который нужно заменить при несовпадении:

```
network:
  version: 2
  ethernets:
    enp0s8:
      dhcp4: false
      addresses:
        - 192.168.56.10/24
```

Установите этому файлу права 600, проверьте `sudo netplan generate`, затем из консоли ВМ выполните `sudo netplan try`. Проверьте адрес и подтверждайте конфигурацию только после успешного теста; отсутствие подтверждения позволяет вернуть прежнее состояние. Другие файлы Netplan не должны задавать конфликтующие параметры того же интерфейса [47]. Если используется Ubuntu Desktop с NetworkManager, настройте тот же адрес в свойствах *второго* адаптера и запишите способ.

В административной PowerShell Windows сначала прочитайте Get-NetAdapter, затем настройте точно выбранный второй адаптер:

```
$labAdapter = 'Ethernet 2'
New-NetIPAddress -InterfaceAlias $labAdapter -IPAddress 192.168.56.20
-PrefixLength 24
Get-NetIPConfiguration -InterfaceAlias $labAdapter
Test-NetConnection 192.168.56.10 -Port 22
```

Имя Ethernet 2 – пример, не автоматическое обнаружение. Если адрес уже существует, не создавайте дубликат. Gateway второго адаптера пустой; маршрут по умолчанию остаётся у NAT. Linux должен слушать SSH и разрешать клиент по правилу из модуля 12 практикума. Если Windows не имеет ssh.exe, установите штатный OpenSSH Client через дополнительные компоненты ОС, затем проверьте Get-Command. Подключитесь `ssh labanna@192.168.56.10`, сверьте ключ сервера и выполните whoami/hostname на Linux.

Проверка результата

TcpTestSucceeded – True; внутри удалённого сеанса видны Linux и labanna. Ping может не получить ответа из-за правил ICMP, но это не отменяет отдельно проверенный TCP.

Управляемая неисправность

Подключите второй адаптер одной ВМ к другой *виртуальной* сети того же типа. Сравните отсутствие соседнего обмена с неизменным числовым планом, затем восстановите имя сети.

Что сдать и объяснить

Адресный план, виртуальные MAC, реальные имена интерфейсов, маршрут и протокол SSH. В Packet Tracer создайте отдельный аналог «два ПК–S1» и опишите, что в нём моделируется, а что проверено в ВМ.

Самостоятельное усложнение. В обеих средах измените адрес одной конечной машины на другую подсеть и сравните решения о локальном получателе. Не переносите настройку bridge на рабочую сеть.

38.2 Samba: сетевой ресурс и права на данные

SMB – протокол доступа к общим файлам. **Samba** позволяет Linux предоставлять такой ресурс Windows-клиенту. Сетевой путь `\\192.168.56.10\CourseTeam` состоит из сервера и имени ресурса; это не физический Linux-путь. Сервер сопоставляет ресурс с каталогом, проверяет учётную запись и политику публикации, затем действуют права файловой системы [40].

В автономной Samba база паролей SMB отличается от обычного пароля Linux. Для этого курса соответствующая Unix-запись уже существует, а smbpasswd добавляет учётные данные сетевой службы. Локальная группа Windows LabEditors не выдаёт автоматически права в Linux labteam: это разные базы ролей. Анонимный доступ не нужен, проверяем именные записи.

Лабораторная I02. Linux-файловый сервер и Windows-клиент

Время: 150–180 минут. **Подготовка:** I01, модули 2–4 практикума.

Какую проблему решает работа

Сотрудникам нужна общая папка на сервере, но не все должны делать с ней одно и то же: одним нужно читать, другим менять, третьим — ничего. Работа решает задачу настройки файлового сервера на Linux с правами доступа, которые проверяются с Windows-клиента.

Что вы научитесь делать

Проверить разрешённые и запрещённые операции через настоящую сетевую службу.

Стенд и адреса

Ubuntu, Samba из штатного пакета samba, labteam и три Linux-пользователя. Каталог /srv/course-share новый и отдельный от документов прежних работ.

Порядок работы

Подготовьте папку: владелец root, группа labteam, режим 2770. По образцу модуля 3 практикума задайте default ACL для группового rwx и отсутствия others. Дайте labguest проход и чтение, в том числе наследуемое чтение новых объектов:

```
sudo mkdir -p /srv/course-share
sudo chown root:labteam /srv/course-share
sudo chmod 2770 /srv/course-share
sudo setfacl -m d:u::rwx,d:g::rwx,d:o::--- /srv/course-share
sudo setfacl -m u:labguest:rx,d:u:labguest:r-x /srv/course-share
sudo smbpasswd -a labanna
```

```
sudo smbpasswd -a labboris
sudo smbpasswd -a labguest
```

Сохраните копию /etc/samba/smb.conf. Добавьте отдельный раздел, не меняя чужие ресурсы:

```
[CourseTeam]
    path = /srv/course-share
    browseable = yes
    guest ok = no
    valid users = @labteam labguest
    read only = yes
    write list = @labteam
    create mask = 0660
    directory mask = 2770
```

Проверьте `testparm -s`, затем `sudo systemctl restart smbd`. Убедитесь в прослушивании TCP 445 и разрешите `sudo ufw allow from 192.168.56.20 to any port 445 proto tcp`, если UFW активен. Из отдельного Windows-сеанса подключите ресурс; пример ниже – **CMD**, звёздочка запрашивает пароль интерактивно:

```
net use Z: \\192.168.56.10\CourseTeam /user:labboris *
dir Z:\
echo created by editor > Z:\report.txt
type Z:\report.txt
net use Z: /delete
```

В другом обычном сеансе Windows подключитесь как labguest: чтение report.txt разрешено, изменение и удаление запрещены. Windows может сохранять SMB-сеансы и не допускать разные учётные данные к одному серверу в одном сеансе; используйте отдельные входы и завершайте только учебное подключение, а не все сетевые диски компьютера.

Проверка результата

Редактор создаёт и изменяет, читатель читает без изменения, неверные учётные данные дают отказ. На Linux проверены владельцы, группа и эффективные ACL созданного файла.

Управляемая неисправность

Временно уберите групповую запись в ACL учебного каталога и повторите создание файла редактором. Разрешение Samba не отменяет запрет файловой системы. Восстановите ACL и проверьте.

Что сдать и объяснить

Политика ресурса, ACL, результаты четырёх операций для каждой роли, порт и журнал smbd. Учётные пароли не включаются.

Самостоятельное усложнение. Добавьте второй ресурс только для чтения и сравните его сетевую политику с теми же правами диска.

38.3 Веб-служба: процесс, порт, конфигурация и файл

HTTP-клиент обращается к адресу и порту, сервер выбирает ресурс и читает файл от имени служебной записи. Поэтому успешное TCP-соединение может завершиться HTTP 403 при отказе доступа или 404 при отсутствии ресурса. Код

200 означает успешный HTTP-ответ, но ожидаемое содержимое страницы проверяют отдельно. Учебный HTTP без TLS используется только внутри изолированной сети.

Лабораторная I03. Linux: сайт, доступ и диагностика отказа

Время: 150 минут. **Подготовка:** I01, модули 7 и 11–12 практикума; сетевые 21 и 24.

Какую проблему решает работа

Сайт «не открывается», а виновных много: служба не запущена, неверный путь, нет прав, закрыт порт на брандмауэре. Работа решает проблему диагностики отказа: один запрос проверяется по цепочке возможных причин, и вы учитесь исключать их по очереди.

Что вы научитесь делать

Показать, как состояние службы, путь, права и брандмауэр влияют на один запрос.

Стенд и адреса

Linux, пакет nginx, новый /var/www/course-site. Через sudo mkdir создайте каталог, права 755; в index.html поместите текст Course site ready, владелец root, права 644. Никаких личных файлов здесь не публикуем.

Порядок работы

В новом /etc/nginx/sites-available/course-site создайте:

```
server {
    listen 192.168.56.10:8080;
    server_name _;
    root /var/www/course-site;
    index index.html;
    location / {
        try_files $uri $uri/ =404;
    }
}
```

Создайте ссылку на этот файл в sites-enabled, проверьте `sudo nginx -t`, только после успеха выполните `sudo systemctl reload nginx`. Проверьте listening через ss. Штатный сайт пакета на 80 может оставаться включённым: учтите его в паспорте, не считайте конфигурацию 8080 описанием всех слушающих портов. Разрешите UFW TCP 8080 только от Windows-клиента, как в модуле 12 практикума. В Windows проверьте Test-NetConnection с Port 8080, затем откройте `http://192.168.56.10:8080/` в браузере и прочитайте ожидаемый текст. В Linux сравните status службы, access.log и error.log Nginx [41].

Для проверки правила Windows из модуля 12 практикума создайте там временную службу на 8080: после установки Python 3 в учебной Windows-ВМ запустите:

```
python -m http.server 8080 --bind 192.168.56.20 --directory
"$env:USERPROFILE\admin-course\site"
```

Это отдельный опыт только с учебными документами и без аутентификации; проверяйте с Linux, создавая и удаляя ровно правило Course-Lab-HTTP. Прервите сервер Ctrl+C после сравнения. Без установленного Python эту дополнительную часть пропустите и отметьте границу проверки.

Проверка результата

Подтверждены реальное соединение, HTTP-ответ и нужное содержимое. Наличие записи access.log сопоставлено со временем запроса и адресом клиента.

Управляемая неисправность

Измените права только учебного index.html с 644 на 600 при владельце root. Процесс nginx не может прочитать файл; проверьте HTTP-ошибку и error.log, затем восстановите 644. Не «исправляйте» отказ запуском всего веб-сервера от root.

Что сдать и объяснить

Цепочка «клиент–интерфейс–правило–сокет–процесс–файл», конфигурация, код ответа и записи журнала до/после.

Самостоятельное усложнение. Уберите одну точку с запятой в копии настройки и получите отказ nginx -t. Объясните, почему после неуспешной проверки нельзя перезагружать рабочую конфигурацию.

Глава 39

Восстановление и передача инфраструктуры

39.1 Сохранить данные вместе с политикой доступа

В проекте копируются не только байты документа. Нужны роли, идентификаторы, права, ACL, конфигурации публикации, состояние ключей и инструкция запуска. Восстановление на другую систему требует согласования UID/GID и служебных записей: одинаковые имена не гарантируют одинаковые числовые владельцы. Секреты копируют в защищённое место, а не в общий отчёт.

Лабораторная I04. Восстановление общего ресурса и прав

Время: 150–180 минут. **Подготовка:** I02, модули 9–11 практикума.

Какую проблему решает работа

Аварии случаются: пользователь удалил папку, диск отказал. Мало вернуть файлы — нужно вернуть и права доступа, иначе данные окажутся недоступны или, хуже, доступны всем. Работа решает проблему восстановления вместе с матрицей доступа и проверки результата с клиента.

Что вы научитесь делать

Вернуть данные и матрицу доступа, затем проверить их с Windows.

Стенд и адреса

Отдельный новый /var/backups/course, доступ только root, и отдельный /srv/course-restore. На время снимка данных участники прекращают изменения учебного ресурса; в реальном приложении способ согласования выбирают по его требованиям.

Порядок работы

Создайте копию с ACL и расширенными атрибутами:

```
sudo mkdir -p /var/backups/course
sudo chmod 700 /var/backups/course
sudo tar --acls --xattrs -czf /var/backups/course/course-data.tar.gz -C /srv
course-share
sudo chmod 600 /var/backups/course/course-data.tar.gz
sudo tar -tzf /var/backups/course/course-data.tar.gz
sudo mkdir -p /srv/course-restore
sudo tar --acls --xattrs -xzpf /var/backups/course/course-data.tar.gz -C
/srv/course-restore
```

Сравните хеши конкретных файлов и ACL через getfacl. От имени labboris и labguest выполните соответствующие разрешённые и запрещённые операции в восстановленном дереве. Затем добавьте отдельный Samba-ресурс CourseRestore по образцу I02 с новым path, той же политикой и проверкой

testparm. Через Windows повторите матрицу доступа. Первоначальный ресурс не переключается автоматически: сначала доказывается пригодность восстановленного.

Проверка результата

Данные совпадают, роли сохранились, редактор изменяет восстановленный документ, читатель не может изменить. Проверены путь, владельцы, ACL и доступ через сеть.

Управляемая неисправность

Повредите только исходный report.txt после создания копии. Покажите возможность получить прежнюю версию и измерьте длительность. Затем в отдельном опыте уберите один ACL у восстановленного файла и обнаружьте потерю политики при сохранённых байтах.

Что сдать и объяснить

План возврата, версия копии, хеши, права, фактические RPO/RTO и вывод о проверенных границах. Архив на том же диске ВМ не защищает от потери всей ВМ: укажите внешний носитель или отдельное хранилище для независимой копии.

Самостоятельное усложнение. Составьте процедуру восстановления на новую ВМ: какие роли и идентификаторы нужно подготовить до извлечения данных, какие конфигурации добавить и чем доказать результат.

Лабораторная I05. Итог: инфраструктура учебного отдела

Время: 12–16 часов. **Подготовка:** модуль 14 практикума, сетевые 38–40, I01–I04.

Какую проблему решает работа

Администратор уходит в отпуск или на другую работу, и инфраструктуру принимает другой человек. Если нет описания, проверок и копий, ему придётся всё выяснять заново. Работа решает проблему передачи: понятный, проверенный и восстанавливаемый комплект, по которому можно работать, не зная автора.

Что вы научитесь делать

Передать следующему администратору понятный, проверенный и восстанавливаемый комплект.

Стенд и адреса

Организация имеет сотрудников, читателя, администратора и посетителя. Реальный стенд – Linux-сервер и Windows-клиент в изолированной сети. Отдельный Packet Tracer-проект изображает две пользовательские VLAN, серверную зону и внешний учебный сегмент. Маршрутизируемая часть проверяется в симуляторе; службы и роли – в ВМ.

Порядок работы

Сначала сформулируйте требования: кто читает и изменяет файлы, откуда доступен SSH, кому доступен сайт, сколько данных допустимо потерять и за какое время восстановить. Составьте адресный план и матрицу разрешённых потоков.

В Packet Tracer настройте VLAN, маршруты, адресацию и правила согласно уже изученным лабораторным. В ВМ реализуйте файловый ресурс, сайт, SSH-ключ, узкие правила доступа, версии копий и журнал запуска. Не предполагайте, что ACL Packet Tracer уже фильтрует трафик реальных ВМ.

Проверьте каждого пользователя и поток; для запретов также сохраните результат. Восстановите ресурс в отдельное место, затем предложите процедуру переключения. Партнёр вносит две *раздельные* неисправности: одну в модель сети, другую в ОС. Для каждой запишите симптом, проверку, подтверждённую причину, минимальное исправление и повторный тест. Передайте комплект другому человеку для повторения основных проверок без устных подсказок.

Проверка результата

Работают требуемые службы, запрещены описанные операции, копия восстановлена, оба стенда имеют паспорта. Ни один результат симуляции не выдаётся за проверку настоящей ОС. Полномочия сотрудников ограничены их ролью.

Что сдать и объяснить

Схемы, адреса и роли; конфигурации; сценарии; матрицы доступа и сетевых проверок; журналы изменений; протокол восстановления; инструкция запуска и отключения; перечень известных ограничений.

Самостоятельное усложнение. Предложите дальнейшее развитие: централизованные записи через AD/LDAP, управление конфигурацией Ansible, мониторинг и контейнеры. Для каждого пункта объясните решаемую проблему, новую точку отказа и проверку, а не только название технологии.

ПРИЛОЖЕНИЯ

Книга I. Системное администрирование

Методические указания и эталонные решения для преподавателя, модель учебных команд, отчёт о проверке полигонов, справочник команд.

Приложение А

Книга преподавателя, модель и проверка полигонов

A.1 Книга преподавателя: практикум «Базовые команды Linux»

1. Состав комплекта

Файл	Адресат	Содержание
`setup.sh`	обучающийся	скрипт создания учебного полигона `~/linux-lab` и программы проверки `proverka`
`chast-1-navigaciya.md`	обучающийся	часть 1: разделы 1–5 (`pwd`, `cd`, `ls`, `mkdir`, `touch`)
`chast-2-rabota-s-failami.md`	обучающийся	часть 2: разделы 6–10 (`cp`, `mv`, `rm`, `cat`, `less`)
`chast-3-poisk-i-prava.md`	обучающийся	часть 3: разделы 11–18 (`head`, `tail`, `echo`, `grep`, `find`, `chmod`, `man`, итоговое задание)
`uchitel.md`	преподаватель	эталонные решения, проверяемые результаты, методические замечания

Каждый раздел содержит теоретические сведения и блок из десяти заданий. Всего 170 заданий и итоговое задание из 7 шагов.

2. Принцип построения заданий

Каждое задание завершается **наблюдаемым результатом** в файловой системе:

- создан, скопирован, перемещён или удалён файл или каталог;
- изменены права доступа или дата изменения;
- вывод команды сохранён в файл ответа `otvety/N` с помощью `>`;
- значение, найденное при интерактивной работе (`less`, `man`), записано в файл ответа командой `echo`.

Результат проверяется программой `~/linux-lab/proverka`, которую устанавливает скрипт `setup.sh`. Обучающийся сразу видит, выполнено ли задание, и может исправить ошибку без участия преподавателя. Задания вида «ответьте на вопрос» или «определите без выполнения» в практикуме отсутствуют.

2.1. Программа проверки

Команда	Действие
`~/linux-lab/proverka`	сводка: количество выполненных заданий по каждому разделу

Команда	Действие
`~/linux-lab/proverka 4`	подробная проверка раздела 4 (допустимо `04`, `04_mkdir`, `mkdir`)
`~/linux-lab/proverka 4 5`	проверка нескольких разделов

Пример вывода:

```
Раздел 4 (04_mkdir)
  [выполнено]    4.1    каталог sandbox
  [не выполнено] 4.2    каталоги red, green, blue
                        результат не соответствует условию задания
  ...
  Итог: 1 из 10

Раздел 1 (01_pwd)
  [не выполнено] 1.2    otvety/2: путь каталога zoo/birds
                        нет файла с ответом: otvety/2
```

Программа не выводит правильных решений: при ошибке сообщается только причина (отсутствует файл ответа либо результат не соответствует условию). Проверка выполняется по фактическому состоянию каталога, поэтому её можно запускать сколько угодно раз.

Сводка `~/linux-lab/proverka` удобна для контроля: обучающийся показывает её в конце занятия.

Ограничение. Программа `proverka` — обычный сценарий bash, и эталонные значения в нём открыты. Заинтересованный обучающийся может прочитать его командой `cat`. Это допустимо, поскольку чтение сценария само по себе является упражнением; для контрольных работ рекомендуется устный опрос по сохранённым ответам.

2.2. Особенности отдельных заданий

- **15.10.** Файл `treasure.txt` при каждом создании полигона помещается в случайный каталог и содержит случайное слово, поэтому ответ у каждого обучающегося свой.
- **15.1.** Список `.txt` включает `treasure.txt`, поэтому в нём 16 файлов, а не 15.
- **9.8.** Файл `sandbox/story_copy.txt` изначально содержит неполную копию (3 строки). Задание засчитывается после восстановления полного текста.
- **7.4.** Содержимое файлов `left.txt` и `right.txt` перепутано намеренно.

3. Организация занятий

3.1. Рекомендуемое распределение

Занятие	Материал	Ориентировочная продолжительность
1	часть 1	2 академических часа
2	часть 2	2 академических часа
3	часть 3	2–3 академических часа

3.2. Учебный полигон

Скрипт `setup.sh` запускается один раз перед первым занятием. Он создаёт каталог `~/linux-lab`, в нём 18 рабочих каталогов (`01_pwd` … `18_final`) и программу

proverka. Каждый рабочий каталог содержит одинаковый набор учебных файлов и пустой каталог otvety. Для разделов 5–9, 13 создаётся каталог sandbox, в разделах 7–9 и 16 — дополнительные файлы для заданий.

Разделы независимы: действия в одном рабочем каталоге не влияют на другие.

Команда	Действие
`bash setup.sh`	создать полигон; если он уже существует, скрипт запрашивает подтверждение полного пересоздания
`bash setup.sh 7`	пересоздать только рабочий каталог раздела 7 (результаты этого раздела удаляются)
`bash setup.sh 7 12`	пересоздать несколько разделов
`bash setup.sh --force`	пересоздать весь полигон без запроса
`bash setup.sh --help`	вывести справку

Восстановление раздела требуется, если обучающийся необратимо испортил данные (например, удалил лишние файлы в разделе 8). Скрипт изменяет только содержимое ~/linux-lab и перед удалением возвращает владельцу права, поэтому восстановление возможно и после раздела 16. Каталог ~/linux-lab/06_cp_backup (задание 6.10) удаляется только при полном пересоздании.

3.3. Требования к рабочей среде

1. **Linux с bash и GNU coreutils** (Ubuntu, Debian, Fedora, Linux Mint, WSL). В macOS ряд параметров отличается (`head -n -N`, `touch -d`, `stat`), поэтому практикум и программа проверки рассчитаны на Linux.
2. **Учётная запись обычного пользователя.** На суперпользователя ограничения прав не распространяются. При запуске от имени root скрипт выводит предупреждение.
3. **Кодировка UTF-8.** Без неё `grep -i` не сопоставляет строчные и прописные буквы кириллицы, а `head -c` и `rev` обрабатывают русский текст некорректно. Скрипт проверяет кодировку.
4. **Справочные страницы.** В минимизированных системах (контейнеры Docker, некоторые облачные образы) справочные страницы удалены. Для раздела 17 необходимо выполнить `sudo unminimize` или установить пакеты `man-db` и `manpages`, затем `sudo mandb` для работы `man -k`.

3.4. Проверка материалов

Эталонные решения из раздела 4 выполнены автоматическим тестом на Ubuntu (bash 5.2, GNU coreutils 9.4, локаль ru_RU.UTF-8) от имени обычного пользователя на полигоне, созданном одним запуском скрипта. Результаты теста:

- на чистом полигоне программа proverka засчитывает 0 из 177 проверок, то есть без выполнения заданий ничего не засчитывается;
- после выполнения эталонных решений засчитываются 177 из 177;
- типичные ошибки не засчитываются: неверный параметр grep, двойные кавычки вместо одинарных, > вместо >>, удаление лишних файлов, избыточные права;
- восстановление отдельного раздела и полное пересоздание полигона работают корректно.

Тест повторялся трижды, чтобы проверить случайное размещение файла

`treasure.txt`. Интерактивные действия (клавиши `less`, второй терминал) заменены в тесте записью эталонных значений; сами эталонные значения сверены с данными полигона.

3.5. Методические замечания

1. **Язык сообщений.** В системах с русской локализацией сообщения об ошибках выводятся на русском языке, например «Нет такого файла или каталога» вместо `No such file or directory`.
2. **Путь к файлу ответа.** Типичная ошибка разделов 1 и 2 — команда `pwd > otvety/2` после перехода в подкаталог. Оболочка сообщает `otvety/2: No such file or directory`. Это хороший повод повторить относительные и абсолютные пути.
3. **Восстановление каталога.** Если раздел восстанавливается, когда обучающийся находится внутри его рабочего каталога, оболочка выводит `getcwd: cannot access parent directories`. После восстановления необходимо повторно выполнить `cd`.
4. **Формат `ls -l`** зависит от локали: `5,0M` или `5.0M`, `сен 23` или `Sep 23`. Программа проверки это учитывает.

4. Эталонные решения

В таблицах приведены эталонные команды и результат, который проверяет программа `proverka`. Допускаются и другие решения, дающие тот же результат. `D` — рабочий каталог раздела.

Раздел 1. `pwd` (D = `~/linux-lab/01_pwd`)

№	Эталонное решение	Проверяемый результат
1	`cd ~/linux-lab/01_pwd`, `pwd > otvety/1`	`otvety/1` = путь D
2	`cd zoo/birds`, `pwd > ~/linux-lab/01_pwd/otvety/2`	`otvety/2` = `D/zoo/birds`
3	`cd /`, `pwd > ~/linux-lab/01_pwd/otvety/3`	`otvety/3` = `/`
4	`cd`, `pwd > ~/linux-lab/01_pwd/otvety/4`	`otvety/4` = домашний каталог
5	`cd /usr/share`, `cd ../lib`, `pwd > ~/linux-lab/01_pwd/otvety/5`	`otvety/5` = `/usr/lib`
6	`cd ~/linux-lab/01_pwd/space/planets`, `echo "Я нахожусь в $(pwd)" > ~/linux-lab/01_pwd/otvety/6`	`otvety/6` = `Я нахожусь в D/space/planets`
7	`cd ~/linux-lab/01_pwd`, `ln -s space sky`	`sky` — символическая ссылка на `space`
8	`cd sky`, `pwd > ../otvety/8a`, `pwd -P > ../otvety/8b`	`8a` = `D/sky`, `8b` = `D/space`
9	`cd ~/linux-lab/01_pwd/kitchen`, `echo $PWD > ../otvety/9`	`otvety/9` = `D/kitchen`
10	`cd ~/linux-lab/01_pwd/logs`, `realpath . > ../otvety/10`	`otvety/10` = `D/logs`

Раздел 2. `cd` (D = `~/linux-lab/02_cd`)

№	Эталонное решение	Проверяемый результат
1	`cd zoo/mammals`, `cd ..`, `cd ..`, `pwd > otvety/1`	D
2	`cd zoo/birds`, `cd ../../space/planets`, `cd ../../kitchen/recipes`, `pwd > ../../otvety/2`	D/kitchen/recipes
3	`cd projects/site/css`, `cd ../../..`, `pwd > otvety/3`	D
4	`cd D/zoo`, `cd D/space`, 3 × `cd -`, `pwd > D/otvety/4`	D/zoo
5	`cd projects/site`, `pwd > ../../otvety/5`	D/projects/site
6	`cd "my photos"`, `pwd > ../otvety/6`	D/my photos
7	`cd projects/site/img` (с Tab), `pwd > ../../../otvety/7`	D/projects/site/img
8	`cd .secret`, `pwd > ../otvety/8`	D/.secret
9	из `projects/site/img`: `cd ../../../zoo/reptiles`, `pwd > ../../otvety/9`	D/zoo/reptiles
10	последовательность из задания, `pwd > D/otvety/10`	D/kitchen

Раздел 3. `ls` (D = `~/linux-lab/03_ls`)

№	Эталонное решение	Проверяемый результат
1	`ls -a > otvety/1`	содержит `.hidden_note` и `.secret`
2	`ls -a .secret > otvety/2`	содержит `hint.txt`
3	`ls -lh space > otvety/3`	строка `telescope.raw` с размером в M
4	`ls -S space > otvety/4`	первая строка `telescope.raw`
5	`ls -R zoo > otvety/5`	содержит `zoo/birds:`
6	`ls ?????.txt > otvety/6`	`story.txt`
7	`ls -d */ > otvety/7`	только каталоги
8	`ls archive/*/*.bak > otvety/8`	4 строки
9	`ls -l hello.sh > otvety/9`	строка начинается с `-rw` (права на выполнение нет)
10	`ls -1 archive/2024 | wc -l > otvety/10`	4

Раздел 4. `mkdir` (задания 2–10 в `D/sandbox`)

№	Эталонное решение	Проверяемый результат
1	`mkdir sandbox`, `cd sandbox`	каталог `sandbox`
2	`mkdir red green blue`	три каталога
3	`mkdir -p one/two/three`	цепочка каталогов
4	`mkdir "my folder"`	каталог `my folder`
5	`mkdir month{01..12}`	12 каталогов
6	`mkdir -p school/{class1,class2}/{math,art}`	4 конечных каталога
7	`mkdir .invisible`, `ls -a`	каталог `.invisible`

№	Эталонное решение	Проверяемый результат
8	`mkdir -pv band/{albums/{2024,2025},concerts,photos}`	4 конечных каталога
9	`mkdir -p 1/2/3/4/5/6/7/8/9/10`	цепочка каталогов
10	`mkdir -m 700 private`	права `700`

Раздел 5. `touch` (в `D/sandbox`)

№	Эталонное решение	Проверяемый результат
1	`touch note.txt`	пустой файл
2	`touch day{1..7}.txt`	7 файлов
3	`touch -d "2000-01-01 12:00" old.txt`	год изменения 2000
4	`touch -d "2030-01-01" future.txt`	год изменения 2030
5	`mkdir newdir`, `touch newdir/file.txt`	файл существует
6	`touch "my notes.txt"`	файл существует
7	`mkdir ../kitchen/fridge_items`, `touch ../kitchen/fridge_items/{молоко,сыр,яйца,масло,йогурт}`	не менее 5 файлов
8	`touch .todo`	файл существует
9	`touch file{001..100}.txt`	100 файлов
10	`touch -r old.txt same_time.txt`	время совпадает с `old.txt`

Раздел 6. `cp` (D = `~/linux-lab/06_cp`)

№	Эталонное решение	Проверяемый результат
1	`cp story.txt story_backup.txt`	файлы идентичны
2	`cp kitchen/recipes/*.txt sandbox/`	три рецепта в `sandbox`
3	`cp -r zoo zoo_copy`	копия каталога
4	`cp cities.txt space`	`space/cities.txt`
5	`cp -r projects/site projects/site_v2`, `echo "<p>v2</p>" >> projects/site_v2/index.html`	оригинал — 1 строка, копия — 2
6	`cp .secret/hint.txt clue.txt`	файлы идентичны
7	`mkdir sandbox/pictures`, `cp projects/site/img/*.[jJ][pP][gG] sandbox/pictures/`	`cat.jpg` и `Cat2.JPG`
8	`cp -v cities.txt story.txt config.conf sandbox/ > otvety/8`	3 строки со знаком `->`
9	`cp -p logs/old.log sandbox/`	время изменения совпадает
10	`cp -r ~/linux-lab/06_cp ~/linux-lab/06_cp_backup`	скрытые файлы в копии

Раздел 7. `mv` (в `D/sandbox`)

№	Эталонное решение	Проверяемый результат
1	`mv note.txt notes.txt`	`notes.txt` есть, `note.txt` нет
2	`mkdir week`, `mv day*.txt week/`	7 файлов в `week`, в `sandbox` нет
3	`mv ../zoo/reptiles ../zoo/reptilia`	каталог переименован
4	`mv left.txt tmp.txt`, `mv right.txt left.txt`, `mv tmp.txt right.txt`	`left.txt` = «лево», `right.txt` = «право»
5	`mv ../archive/2025/*.txt ../archive/2024/`	`2025` пуст, в `2024` 4 файла `.txt`
6	`mv ../archive/2024/report.bak ../archive/2024/report_2024.bak` и т. д.	4 файла `*_2024.bak`
7	`cd mess`, `mkdir images music docs video`, `mv *.jpg images/`, `mv *.mp3 music/`, `mv *.txt docs/`, `mv *.mp4 video/`	2, 2, 2 и 1 файл
8	`mkdir letters`, `mv a.txt b.txt letters/`	файлы в `letters`
9	`mv final.txt final_old.txt`, `mv plan.txt final.txt`	`final.txt` = «план», `final_old.txt` = «итог»
10	`mv ../projects/site/img ../space/images`	`space/images/cat.jpg`

Раздел 8. `rm` (в `D/sandbox`)

№	Эталонное решение	Проверяемый результат
1	`rm story_backup.txt`	файл удалён
2	`rm -i week/*` (n для `day7.txt`)	в `week` только `day7.txt`
3	`rmdir red`	каталог удалён
4	`rm -r box`	каталог удалён
5	`ls file0*.txt`, `rm file0*.txt`	из `file*.txt` остался `file100.txt`
6	`ls ../archive/2024/*.bak`, `rm ../archive/2024/*.bak`	`.bak` нет, `archive/2025` не тронут
7	`rm "my notes.txt"`	файл удалён
8	`rm -- -oops` или `rm ./-oops`	файл удалён
9	`rm -r 1`	цепочка удалена
10	`rm day[1-6].txt`	из `day*.txt` остался `day7.txt`

Если обучающийся удалил лишнее (например, `rm file*.txt` в задании 5), задание невозможно выполнить без восстановления: `bash setup.sh 8`.

Раздел 9. `cat` (D = `~/linux-lab/09_cat`)

№	Эталонное решение	Проверяемый результат
1	`cat kitchen/recipes/*.txt > sandbox/cookbook.txt`	22 строки
2	`cat -n cities.txt > sandbox/cities_numbered.txt`	строка `3<Tab>Сочи`

№	Эталонное решение	Проверяемый результат
3	`cat .secret/hint.txt > sandbox/secret_copy.txt`	копия идентична
4	`cat .hidden_note story.txt > sandbox/two.txt`	9 строк, первая из `.hidden_note`
5	`cat > sandbox/diary.txt`, ввод, Ctrl+D	не менее 3 непустых строк
6	`cat logs/old.log > sandbox/all_logs.txt`, `cat logs/app.log >> sandbox/all_logs.txt`	41 строка, первая — `old day`
7	`cat -E config.conf > sandbox/config_marked.txt`	содержит `port=8080$`
8	`cat sandbox/story_copy.txt > sandbox/story_copy.txt`, `cat story.txt > sandbox/story_copy.txt`	совпадает со `story.txt`
9	`tac cities.txt > sandbox/cities_reverse.txt`	12 строк, первая — Омск
10	`rev story.txt > sandbox/story_reversed.txt`	8 строк, первая оканчивается на «…йевтаМ»

Раздел 10. `less` (D = `~/linux-lab/10_less`)

№	Действия	Проверяемый результат
1	`less logs/app.log`, `/ERROR`, `n`; `echo 5 > otvety/1`	5
2	`G`; `echo 16:40 > otvety/2`	16:40
3	`less -N story.txt`, `/ракету`; `echo 5 > otvety/3`	5
4	`/Азия`, `n`; `echo -e "панда\nкобра\nтигр" > otvety/4`	три названия, по одному в строке
5	`:n`, `/Марс`; `echo 4 > otvety/5`	4
6	`grep -i error logs/app.log | less`; `echo 09:35 > otvety/6`	09:35
7	`h`; `echo "=" > otvety/7`	=
8	`&INFO`, `G`; `echo 15:33 > otvety/8`	15:33
9	`G`, `?WARNING`; `echo 15:26 > otvety/9`	15:26
10	`less +F …` и `echo "2026-09-20 17:00 ERROR test" >> …`	в журнале есть строка `ERROR test`

Задание 10 выполняется последним, так как добавленная запись изменяет ответ задания 2.

Раздел 11. `head` (D = `~/linux-lab/11_head`)

№	Эталонное решение	Проверяемый результат
1	`head cities.txt > otvety/1`	10 строк, первая — Москва
2	`head -n 3 space/planets.txt > otvety/2`	3 строки, первая — Меркурий
3	`head -c 20 story.txt > otvety/3`	`Жил-был кот` (кириллица — 2 байта на символ)
4	`head -n -5 cities.txt > otvety/4`	7 строк
5	`head -q -n 1 kitchen/recipes/*.txt > otvety/5`	3 строки без `==>`
6	`head -n 5 cities.txt | tail -n 1 > otvety/6`	Екатеринбург
7	`head -n 45 numbers.txt | tail -n 5 > otvety/7`	41–45
8	`head -n 95 numbers.txt | tail -n 6 > otvety/8`	90–95
9	`touch config.conf, ls -t | head -n 3 > otvety/9`	3 строки, среди них `config.conf`
10	`head -n 1 zoo/animals.txt > otvety/10`	`лев;млекопитающее;Африка`

В задании 9 первым может оказаться каталог `otvety`: оболочка создаёт файл ответа до выполнения `ls`, и время изменения каталога обновляется. Это наглядно показывает порядок обработки перенаправления.

Раздел 12. `tail` (D = `~/linux-lab/12_tail`)

№	Эталонное решение	Проверяемый результат
1	`tail cities.txt > otvety/1`	10 строк, первая — Сочи
2	`tail -n 1 space/planets.txt > otvety/2`	Нептун
3	`tail -n +10 cities.txt > otvety/3`	Томск, Самара, Омск
4	`tail -f …` и 3 × `echo "TEST" >> …`	не менее 3 строк `TEST` в журнале
5	`tail -n 3 numbers.txt | head -n 1 > otvety/5`	`98`
6	`grep -i error logs/app.log | tail -n 1 > otvety/6`	запись `16:40 error …`
7	`tail -n 2 logs/app.log logs/old.log > otvety/7`	заголовки `==> … <==`
8	`tail -n 1 story.txt > otvety/8`	«Кот вернулся на подоконник и уснул. Конец.»
9	`grep INFO logs/app.log | tail -n 3 > otvety/9`	3 строки, последняя — 15:33
10	`grep WARNING logs/app.log | head -n 1 > otvety/10`	запись `08:21 WARNING …`

Раздел 13. echo (D = ~/linux-lab/13_echo)

№	Эталонное решение	Проверяемый результат
1	`echo "Привет, Linux!" > otvety/1`	строка совпадает
2	`echo "$USER $HOME" > otvety/2`	имя и домашний каталог
3	`echo … > sandbox/diary.txt`, 2 × `echo … >> sandbox/diary.txt`	3 строки
4	`echo B > sandbox/list.txt`, `echo C >> sandbox/list.txt`	ровно B и C
5	`echo -e "раз\nдва\tтри" > otvety/5`	2 строки, во второй табуляция
6	`echo 'Мой дом: $HOME' > otvety/6`	текст `$HOME` без подстановки
7	`echo * > otvety/7`	список имён файлов каталога
8	`echo "молоко" >> kitchen/fridge.txt`	слово в файле
9	`echo "Список покупок" > kitchen/shopping.txt`, 5 × `echo "N. …" >> kitchen/shopping.txt`	заголовок и 5 пунктов
10	`echo {a..z} > otvety/10`	26 букв через пробел

Раздел 14. grep (D = ~/linux-lab/14_grep)

№	Эталонное решение	Проверяемый результат
1	`grep Африка zoo/animals.txt > otvety/1`	4 строки
2	`grep -c птица zoo/animals.txt > otvety/2`	5
3	`grep -c кот story.txt > otvety/3a`; `grep -ic … > otvety/3b`; `grep -wic … > otvety/3c`	3, 5, 3
4	`grep -c ERROR logs/app.log > otvety/4a`; `grep -ic error logs/app.log > otvety/4b`	5, 11
5	`grep -v "^#" config.conf | grep -v "^$" > otvety/5`	5 строк без #
6	`grep -rl картофель kitchen > otvety/6`	borsch.txt, olivie.txt
7	`grep -n Марс space/planets.txt > otvety/7`	4:Марс
8	`grep "disk usage" logs/app.log > otvety/8`	6 строк
9	`grep "^К" cities.txt > otvety/9`	Казань, Калининград
10	`grep млекопитающее zoo/animals.txt | grep -c Африка > otvety/10`	2

Пояснение к заданию 3: без `-i` найдены строки 1, 3, 4; с `-i` добавлены строки 2 («КОТ») и 8 («Кот»); с `-w` исключены строки, где «кот» входит в другое слово («котом»). Параметр `-c` считает строки, а не вхождения.

Раздел 15. find (D = ~/linux-lab/15_find)

№	Эталонное решение	Проверяемый результат
1	`find . -name "*.txt" > otvety/1`	16 строк, включая `treasure.txt`
2	`find zoo -type d > otvety/2`	4 строки
3	`find . -iname "cat*" > otvety/3`	`cat.jpg`, `Cat2.JPG`
4	`find . -size +1M > otvety/4`	`./space/telescope.raw`
5	`find archive projects kitchen -empty -type f > otvety/5`	12 строк
6	`find . -name "*.bak" | wc -l > otvety/6`	4
7	`touch cities.txt`, `find . -mmin -1 -name "*.txt" > otvety/7`	`./cities.txt`
8	`find . -name "*.log" > otvety/8`	2 строки
9	`find . -name "*.bak"`, затем `find . -name "*.bak" -delete`	файлов `.bak` нет
10	`find . -name treasure.txt`, `cat <путь> > otvety/10`	совпадает с содержимым `treasure.txt`

Задание 5 ограничено каталогами `archive`, `projects`, `kitchen`: при поиске по `.` в результат попадал бы сам пустой файл ответа, который оболочка создаёт до запуска `find`. Этот эффект полезно продемонстрировать обучающимся.

Раздел 16. `chmod` (D = `~/linux-lab/16_chmod`)

№	Эталонное решение	Проверяемый результат
1	`chmod +x hello.sh`	файл исполняемый
2	`./hello.sh > otvety/2`	«Привет из скрипта! …»
3	`chmod 750 p1`, `chmod 644 p2`, `chmod 664 p3`, `chmod 711 p4`	права `750`, `644`, `664`, `711`
4	`chmod u=rw,g=r,o= secret.txt`	`640`
5	`chmod a-w readonly.txt`	нет права `w` ни у одной категории
6	`chmod a-x closed`	нет права `x` ни у одной категории
7	`chmod go= diary.txt`	`600`
8	`chmod -R go-w projects/site`	нет объектов с `w` у группы и остальных
9	`mkdir safe`, `chmod 700 safe`	`700`
10	`echo "золото" > safe/gold.txt`, `chmod 400 safe/gold.txt`	`400`, файл не пуст

В задании 10 следует обратить внимание обучающихся, что владелец может вернуть себе право записи командой `chmod`. Следовательно, такая настройка защищает от случайного изменения, но не от намеренного.

Раздел 17. `man` (D = `~/linux-lab/17_man`)

№	Эталонное решение	Проверяемый результат
1	`echo "-S" > otvety/1`	`-S`
2	`echo "-r" > otvety/2`	`-r` или `-R`
3	`echo "-q" > otvety/3`	`-q`
4	`echo "-p" > otvety/4`	`-p` или `-F`
5	`whatis ls > otvety/5`	описание `ls`
6	`type cd > otvety/6`	`cd is a shell builtin` (или русский вариант)
7	`man -k copy > otvety/7`	содержит строку `cp (1)` …
8	`sort cities.txt > otvety/8`	12 строк, первая — Владивосток
9	`wc -w < story.txt > otvety/9` (или `wc -w story.txt`)	49
10	`echo "-h" > otvety/10`	`-h`

Задание 7 требует индекса справочной системы. Если `man -k` выводит `nothing appropriate`, выполните `sudo mandb`.

Раздел 18. Итоговое задание (D = ~/linux-lab/18_final)

```
mkdir report
grep -i error logs/app.log > report/errors.txt
echo "Всего ошибок: $(grep -ic error logs/app.log)" >> report/errors.txt
find . -type f -size +100k -not -path "./report/*" -exec cp {} report/ \;
chmod 400 report/errors.txt
tail -n 3 report/errors.txt > otvety/6
ls report
rm report/telescope.raw
```

Шаг	Проверяемый результат
1	каталог `report`
2	в `report/errors.txt` не менее 11 строк с `error`
3	последняя строка — `Всего ошибок: 11`
4	`report/photo_moon.jpg` идентичен исходному
5	права `errors.txt` — `400`
6	`otvety/6` — 3 строки, последняя `Всего ошибок: 11`
7	`telescope.raw` удалён, `photo_moon.jpg` на месте

Замечания к итоговому заданию:

- На шаге 4 допускается двухэтапное решение: сначала `find . -size +100k`, затем `cp` найденных файлов.
- Условие `-not -path "./report/*"` исключает повторную обработку уже скопированных файлов. Без него возможна ошибка `are the same file`.
- Файл `telescope.raw` удаляется без запроса. Если же попытаться удалить `report/errors.txt`, команда `rm` выведет запрос `remove write-protected regular file?`, так как у файла права `400`.

A.2 Книга преподавателя: эталонные решения полигонов

Решения даны для проверки работы ученика и для разбора затруднений. Каждое решение выполняется автоматической самопроверкой полигона (`selftest`): на чистом полигоне выполнено 0 заданий, после применения эталонных решений — все. Решение — один из корректных вариантов, ученик может справиться иначе; проверяется результат, а не способ. Для заданий, где ученик записывает найденное значение в `otvety`, значение зависит от `.seed` ученика, поэтому в этом приложении приводится способ его получить.

Модуль 1. Сервер и его паспорт

Linux.

1.1. Версия операционной системы. *Подсказка ученику: grep найдёт строку, `cut -d= -f2` отделит значение после знака `=`, `tr -d` удалит кавычки.*

```
grep '^PRETTY_NAME=' srv/etc/os-release | cut -d= -f2 | tr -d '"' > otvety/1
```

1.2. Имя сервера. *Подсказка ученику: Файл состоит из одной строки; достаточно перенаправить вывод cat.*

```
cat srv/etc/hostname > otvety/2
```

1.3. Версия ядра. *Подсказка ученику: Слова разделены пробелами: `cut -d' ' -f3`.*

```
cut -d' ' -f3 srv/proc/version > otvety/3
```

1.4. Число процессорных ядер. *Подсказка ученику: `grep -c` считает строки, а не вхождения: каждая строка `processor` — одно ядро.*

```
grep -c '^processor' srv/proc/cpuinfo > otvety/4
```

1.5. Оперативная память. *Подсказка ученику: awk умеет считать: `awk '/^MemTotal/{print int($2/1024)}'`.*

```
awk '/^MemTotal/{print int($2/1024)}' srv/proc/meminfo > otvety/5
```

1.6. Сколько суток работает сервер. *Подсказка ученику: В сутках 86400 секунд; дробную часть отбросьте функцией int.*

```
awk '{print int($1/86400)}' srv/proc/uptime > otvety/6
```

1.7. Загрузка сервера. *Подсказка ученику: Сравните третье число loadavg с результатом задания 4; awk сравнивает числа через `>`.*

```
cut -d' ' -f3 srv/proc/loadavg > otvety/7a
awk -v c="$(grep -c '^processor' srv/proc/cpuinfo)" '{print ($3 > c) ? "да" :
"нет"}' srv/proc/loadavg > otvety/7b
```

1.8. Файловые системы. *Подсказка ученику: Отберите строки, где первый столбец начинается с /dev/, выведите второй столбец и добавьте sort.*

```
awk '$1 ~ "^/dev/" {print $2}' srv/proc/mounts | sort > otvety/8
```

1.9. Где что лежит. *Подсказка ученику: `find srv -name имя` выводит полный путь файла; нужная часть — путь без имени файла и без первого `srv/`.*

```
find srv -name sshd_config -printf '%h\n' | sed 's|^srv/||' > otvety/9a
find srv -name meminfo -printf '%h\n' | sed 's|^srv/||' > otvety/9b
find srv -name start.sh -printf '%h\n' | sed 's|^srv/||' > otvety/9c
```

1.10. Паспорт сервера. *Подсказка ученику: Соберите файл командой echo с перенаправлением > для первой строки и >> для остальных, подставляя содержимое файлов ответов через $(cat ...).*

```
{ echo "host: $(cat otvety/2)"; echo "os: $(cat otvety/1)"; echo "kernel: $(cat
otvety/3)"
  echo "cpu: $(cat otvety/4)"; echo "ram_mb: $(cat otvety/5)"; echo "uptime_days:
  $(cat otvety/6)"; } > pasport.txt
```

Windows.

1.1. Версия системы. *Подсказка ученику: `Get-ComputerInfo -Property OsName` выводит только нужное свойство. Значение можно взять через `(Get-ComputerInfo).OsName`.*

```
(Get-ComputerInfo).OsName | Set-Content -Encoding UTF8 otvety\1
```

1.2. Сколько процессоров. *Подсказка ученику: Поле `CsNumberOfLogicalProcessors` уже содержит число; ничего считать не нужно.*

```
(Get-ComputerInfo).CsNumberOfLogicalProcessors | Set-Content -Encoding UTF8
otvety\2
```

1.3. Объём памяти. *Подсказка ученику: Разделите байты на `1GB` и округлите: `[math]::Round(…)`.*

```
[math]::Round((Get-ComputerInfo).CsTotalPhysicalMemory / 1GB) | Set-Content
-Encoding UTF8 otvety\3
```

1.4. Сколько работает без перезагрузки. *Подсказка ученику: Разность двух дат — объект `TimeSpan`, у него есть свойство `TotalDays`. Приведите его к `[int]` (отбросив дробную часть).*

```
[int]((Get-Date) - (Get-ComputerInfo).OsLastBootUpTime).TotalDays | Set-Content
-Encoding UTF8 otvety\4
```

1.5. Свободное место на C:. *Подсказка ученику: У `Get-Volume` есть параметр `-DriveLetter`. Размер в байтах делите на `1GB`.*

```
[math]::Round((Get-Volume -DriveLetter C).SizeRemaining / 1GB) | Set-Content
-Encoding UTF8 otvety\5
```

1.6. Паспорт сервера. *Подсказка ученику: Соберите строки в массив и запишите `Set-Content`. Значения подставляйте из объекта `Get-ComputerInfo`.*

```
$i = Get-ComputerInfo
@("HOST=$($i.CsName)", "CORES=$($i.CsNumberOfLogicalProcessors)",
"RAMGB=$([math]::Round($i.CsTotalPhysicalMemory / 1GB))", "OS=$($i.OsName)") |
Set-Content -Encoding UTF8 pasport.txt
```

Модуль 2. Учётные записи и группы

Linux.

2.1. Кто у нас обычный пользователь. *Подсказка ученику: Третье поле passwd — UID. `awk -F: '$3 >= 1000 && $3 < 65534 {print $1}'`.*

```
awk -F: '$3 >= 1000 && $3 < 65534 {print $1}' srv/etc/passwd | sort > otvety/1
```

2.2. Второй root. *Подсказка ученику: UID 0 — это полномочия root независимо от имени. Отберите строки passwd, где третье поле равно 0.*

```
awk -F: '$3 == 0 && $1 != "root" {print $1}' srv/etc/passwd > otvety/2
```

2.3. Обезвредить второй root. *Подсказка ученику: Два действия: `usermod -L имя` блокирует пароль, `usermod -s /usr/sbin/nologin имя` запрещает оболочку.*

```
n=$(cat otvety/2); usermod -L "$n"; usermod -s /usr/sbin/nologin "$n"
```

2.4. Учётные записи без пароля. *Подсказка ученику: Второе поле shadow — пароль. Пустое поле: `$2 == ""`.*

```
awk -F: '$2 == "" {print $1}' srv/etc/shadow | sort > otvety/4
```

2.5. Закрыть дыру. *Подсказка ученику: Команда одна и та же для каждого имени: `passwd -l имя`. Цикл `for n in $(cat otvety/4)` избавит от повторов.*

```
for n in $(cat otvety/4); do passwd -l "$n"; done
```

2.6. Новый сотрудник. *Подсказка ученику: `useradd -m -s … -c "…" -G dev nikita`: -m создаёт домашний каталог, -s задаёт оболочку, -c комментарий, -G добавляет в группу.*

```
useradd -m -s /bin/bash -c "Никита Орлов" -G dev nikita
```

2.7. Политика паролей. *Подсказка ученику: `chage -M` — максимум, `-m` — минимум, `-W` — предупреждение, `-d 0` — «пароль устарел» (сменить при входе). Проверка: `chage -l nikita`.*

```
chage -M 90 -m 1 -W 14 nikita; chage -d 0 nikita
```

2.8. Борис меняет отдел. *Подсказка ученику: Ключ `-G` без `-a` заменяет весь список дополнительных групп; перечислите нужные. Проверка: `id boris`.*

```
usermod -G design,sudo boris
```

2.9. Увольнение. *Подсказка ученику: `usermod -L -e ГГГГ-ММ-ДД имя`; дату в прошлом выдаст `date -d '2 days ago' +%F`.*

```
usermod -L -e "$(date -d '2 days ago' +%F)" gleb
```

2.10. Убрать временную запись. *Подсказка ученику: `userdel -r имя` удаляет запись, личную группу и домашний каталог. Затем проверьте: `getent passwd temp`.*

```
userdel -r temp
```

Windows.

2.1. Кто у нас обычный пользователь. *Подсказка ученику: `Get-LocalUser` выдаёт объекты со свойствами `Name` и `Enabled`. Отберите их через `Where-Object`, имена выведите по одному.*

```
Get-LocalUser | Where-Object { $_.Enabled -and $_.Name -notin
'Administrator','Guest','DefaultAccount','WDAGUtilityAccount' } | Sort-Object Name
| ForEach-Object { $_.Name } | Set-Content -Encoding UTF8 otvety\1
```

2.2. Лишний администратор. *Подсказка ученику: `Get-LocalGroupMember -Group Administrators` возвращает имена в виде `SRV01\имя`. Часть после обратной косой — имя пользователя.*

```
(Get-LocalGroupMember -Group Administrators).Name | ForEach-Object { ($_ -split
'\\')[-1] } | Where-Object { $_ -notin 'Administrator','dmitri' } | Set-Content
-Encoding UTF8 otvety\2
```

2.3. Обезвредить. *Подсказка ученику: Два действия: отключить и убрать из группы. Имя возьмите из своего ответа к заданию 2.*

```
$n = Get-Content otvety\2
Disable-LocalUser -Name $n
Remove-LocalGroupMember -Group Administrators -Member $n
```

2.4. Учётные записи без пароля. *Подсказка ученику: Условие в `Where-Object: $_.Enabled -and -not $_.PasswordRequired`. Встроенные отключённые записи сюда не попадают.*

```
Get-LocalUser | Where-Object { $_.Enabled -and -not $_.PasswordRequired } |
Sort-Object Name | ForEach-Object { $_.Name } | Set-Content -Encoding UTF8
otvety\4
```

2.5. Закрыть дыру. *Подсказка ученику: Цикл по именам из файла ответа: `Get-Content otvety\4 | ForEach-Object { … }`. Допустимо, если вы удалили запись, как в задании 8.*

```
Get-Content otvety\4 | ForEach-Object { Disable-LocalUser -Name $_ }
```

2.6. Новый сотрудник. *Подсказка ученику: Пароль нужно сначала превратить в безопасную строку: `ConvertTo-SecureString 'текст' -AsPlainText -Force`.*

```
$pw = ConvertTo-SecureString 'Start-Passw0rd!' -AsPlainText -Force
New-LocalUser -Name nikita -Password $pw -FullName 'Никита Орлов' -Description
'разработчик' | Out-Null
Add-LocalGroupMember -Group dev -Member nikita
```

2.7. Увольнение. *Подсказка ученику: `Set-LocalUser -Name имя -AccountExpires (Get-Date).AddDays(-1)` — дата «вчера». Проверка: `Get-LocalUser gleb | Format-List *`.*

```
Disable-LocalUser -Name gleb
Set-LocalUser -Name gleb -AccountExpires (Get-Date).AddDays(-1)
```

2.8. Убрать временную запись. *Подсказка ученику: Профиль — обычный каталог: удаляйте его командой `Remove-Item -Recurse` отдельно от учётной записи.*

```
Remove-LocalUser -Name temp
Remove-Item -LiteralPath 'srv\Users\temp' -Recurse -Force
```

Модуль 3. Права доступа: chmod, umask, setgid, ACL

Linux.

3.1. Ключи SSH. *Подсказка ученику: Числовой вид: 7 = rwx, 6 = rw-, 4 = r–. Три цифры — владелец, группа, остальные.*

```
chmod 700 srv/home/anna/.ssh; chmod 600 srv/home/anna/.ssh/id_ed25519
srv/home/anna/.ssh/authorized_keys; chmod 644 srv/home/anna/.ssh/id_ed25519.pub
```

3.2. Отчёт бухгалтерии. *Подсказка ученику: `chmod u=rw,g=r,o= файл`: знак = задаёт права точно, пустое значение для `o` снимает все права.*

```
chmod u=rw,g=r,o= srv/home/olga/otchety/year.pdf
```

3.3. Сценарий резервного копирования. *Подсказка ученику: Право x даёт запуск как программы. 7 = rwx для владельца, 5 = r-x для группы, 0 — остальным.*

```
chmod 750 srv/opt/app/backup.sh
```

3.4. Читать права. *Подсказка ученику: Формат `stat` задаётся ключом `-c`: `%a` — число, `%A` — строка прав. Или посмотрите `ls -l`.*

```
stat -c %a srv/etc/app/secret.key > otvety/4a; stat -c %A srv/etc/app/secret.key > otvety/4b
```

3.5. Каталог загрузки сайта. *Подсказка ученику: Для каталога `r` — список имён, `w` — создание и удаление, `x` — вход. Уберите `w` у группы и остальных.*

```
chmod 755 srv/srv/www/uploads
```

3.6. Общая папка команды. *Подсказка ученику: Первая цифра 2 — бит setgid (`chmod g+s`). Остальные три — как обычно. Проверьте: `ls -ld` покажет `s` в правах группы.*

```
chmod 2770 srv/srv/projects/team
```

3.7. Обменный каталог. *Подсказка ученику: Sticky-бит (`chmod +t`) показывается буквой `t` в конце строки прав. Так устроен `/tmp`.*

```
chmod 1777 srv/tmp/exchange
```

3.8. umask. *Подсказка ученику: Файлы создаются с `666 & ~umask`, каталоги — с `777 & ~umask`. Подоболочка `( … )` не меняет umask вашего терминала.*

```
( cd "$(mktemp -d)"; umask 027; touch f; mkdir d; stat -c %a f d ) > otvety/8
```

3.9. ACL для аудитора. *Подсказка ученику: Если `setfacl` не найден, установите пакет `acl` (на Ubuntu: `sudo apt install acl`). Записи ACL — `u:имя:права`.*

```
setfacl -m u:nobody:r-- srv/srv/projects/report.txt
```

3.10. Файлы, открытые всем на запись. *Подсказка ученику: `-perm -002` означает «в правах есть как минимум o+w». Затем `find … -exec chmod o-w {} +`.*

```
find srv -type f -perm -002 | sort > otvety/10; find srv -type f -perm -002 -exec chmod o-w {} +
```

Windows.

3.1. Файлы, открытые всем. *Подсказка ученику: `(Get-Acl файл).Access` — список записей ACL. Или посмотрите `icacls файл`: ищите запись «Все:(M)» (Modify, изменение).*

```
Get-ChildItem srv\Finance, srv\Public -File | Where-Object {
    (Get-Acl $_.FullName).Access | Where-Object { (ConvertTo-Sid
    $_.IdentityReference) -eq 'S-1-1-0' -and $_.AccessControlType -eq 'Allow' }
} | Sort-Object Name | ForEach-Object { $_.Name } | Set-Content -Encoding UTF8
otvety\1
```

3.2. Закрыть лишнее. *Подсказка ученику: Обойдите файлы циклом `ForEach-Object`. SID в `icacls` пишется со звёздочкой; в PowerShell заключите аргумент в апострофы: `'*S-1-1-0'`.*

```
Get-ChildItem srv\Finance, srv\Public -File | ForEach-Object { icacls $_.FullName
/remove '*S-1-1-0' | Out-Null }
```

3.3. Папка бухгалтерии: отключить наследование. *Подсказка ученику: Порядок важен: сначала `/inheritance:d` (копирует унаследованное в явные записи), затем `/remove`. Проверьте результат: `icacls srv\Finance`.*

```
icacls srv\Finance /inheritance:d | Out-Null
icacls srv\Finance /remove '*S-1-5-32-545' | Out-Null
```

3.4. Общая папка для чтения. *Подсказка ученику: Флаги `(OI)` и `(CI)` означают наследование объектами (файлами) и контейнерами (папками). `RX` — чтение и выполнение.*

```
icacls srv\Public /grant '*S-1-5-32-545:(OI)(CI)RX' | Out-Null
```

3.5. Запрет записи гостям. *Подсказка ученику: Запрещающие записи (`Deny`) проверяются раньше разрешающих и сильнее их. Применяйте запрет осмысленно: он может заблокировать и нужных людей.*

```
icacls 'srv\Public\readonly' /deny '*S-1-5-32-546:(OI)(CI)W' | Out-Null
```

3.6. Где отключено наследование. *Подсказка ученику: `Get-ChildItem -Recurse -File` перечислит файлы, `Get-Acl` вернёт ACL, а свойство `AreAccessRulesProtected` показывает, что наследование отключено.*

```
Get-ChildItem srv\Shared, srv\Public -Recurse -File | Where-Object { (Get-Acl
$_.FullName).AreAccessRulesProtected } | Sort-Object Name | ForEach-Object {
$_.Name } | Set-Content -Encoding UTF8 otvety\6
```

Модуль 4. sudo и минимальные полномочия

Linux.

4.1. Кто в группе sudo. *Подсказка ученику: Четвёртое поле строки группы — список членов через запятую; `tr ',' '\n'` превратит его в столбец.*

```
getent group sudo | cut -d: -f4 | tr ',' '\n' | sort > otvety/1
```

4.2. Что разрешено Борису. *Подсказка ученику: Команда `sudo -l -U пользователь` показывает правила для указанного пользователя.*

```
sudo -l -U boris > otvety/2
```

4.3. Опасное правило. *Подсказка ученику: `grep -l шаблон файлы` печатает только имена подходящих файлов; `basename` отрезает путь.*

```
grep -l 'NOPASSWD: *ALL' srv/etc/sudoers.d/* | xargs -n1 basename > otvety/3
```

4.4. Только перезапуск nginx. *Подсказка ученику: Правило: `пользователь хост=(от_кого) NOPASSWD: полная команда с аргументами.` Конкретная команда вместо `ALL`.*

```
printf '%s\n' '# Правила разработчиков' 'anna ALL=(root) NOPASSWD:
/usr/bin/systemctl restart nginx' > srv/etc/sudoers.d/90-dev
```

4.5. Ошибка в правиле. *Подсказка ученику: После метки `NOPASSWD` обязателен символ `:`. Проверка: `visudo -c -f файл` до и после правки.*

```
sed -i 's|NOPASSWD /usr/bin/apt|NOPASSWD: /usr/bin/apt|' srv/etc/sudoers.d/99-oops
```

4.6. Группа веб-администраторов. *Подсказка ученику: Группу в sudoers записывают со знаком процента: `%webadmins`. Права файлов в sudoers.d — 0440 (`chmod 440`).*

```
groupadd webadmins; usermod -aG webadmins vera
echo '%webadmins ALL=(root) NOPASSWD: /usr/bin/systemctl reload nginx' >
srv/etc/sudoers.d/60-webadmins
chmod 440 srv/etc/sudoers.d/60-webadmins
```

4.7. Редакторы с побегом в оболочку. *Подсказка ученику: Нужны файлы, где в списке разрешённых команд встречается один из перечисленных путей: `grep -lE`.*

```
grep -lE '/(vim|less|bash|sh|find|awk)([ ,]|$)' srv/etc/sudoers.d/* | xargs -n1
basename | sort > otvety/7
```

4.8. sudoedit вместо vim. *Подсказка ученику: `sudoedit` запускается так же, как обычная команда в правиле: `vera ALL=(root) sudoedit /путь/к/файлу`.*

```
printf '%s\n' '# Дизайнеры правят конфигурации' 'vera ALL=(root) sudoedit
/etc/nginx/nginx.conf' > srv/etc/sudoers.d/92-editors
```

4.9. Журнал и защита терминала. *Подсказка ученику: Каждая настройка — отдельная строка, начинающаяся со слова `Defaults`; значения с косой чертой берут в кавычки.*

```
printf '%s\n' 'Defaults use_pty' 'Defaults logfile="/var/log/sudo.log"' >
srv/etc/sudoers.d/00-defaults
chmod 440 srv/etc/sudoers.d/00-defaults
```

4.10. Контрольная: кто остался с полным root. *Подсказка ученику: В цикле перебирайте имена из passwd и спрашивайте `sudo -l -U имя /bin/bash`; код выхода 0 означает «разрешено».*

```
for u in $(cut -d: -f1 srv/etc/passwd); do sudo -l -U "$u" /bin/bash >/dev/null
2>&1 && echo "$u"; done | sort > sudo-audit.txt
```

Windows.

4.1. Кто администратор. *Подсказка ученику: Имена членов группы имеют вид `SRV01\имя`. Уберите префикс через `-split '\\'` и исключите двух законных администраторов.*

```
(Get-LocalGroupMember -Group Administrators).Name | ForEach-Object { ($_ -split
'\\')[-1] } | Where-Object { $_ -notin 'Administrator','dmitri' } | Sort-Object |
Set-Content -Encoding UTF8 otvety\1
```

4.2. Лишить полномочий. *Подсказка ученику: `Remove-LocalGroupMember -Group Administrators -Member имя` убирает члена. Имена возьмите из своего ответа к заданию 1.*

```
Get-Content otvety\1 | ForEach-Object { Remove-LocalGroupMember -Group
Administrators -Member $_ }
```

4.3. Узкие группы вместо администратора. *Подсказка ученику: Принцип минимальных полномочий: чем уже группа, тем безопаснее. `New-LocalGroup`, затем `Add-LocalGroupMember`.*

```
New-LocalGroup -Name webadmins -Description 'Администраторы веб-сервера' |
Out-Null
Add-LocalGroupMember -Group webadmins -Member vera
Add-LocalGroupMember -Group 'Event Log Readers' -Member anna
```

4.4. Опасные правила делегирования. *Подсказка ученику: `Import-Csv` превращает строки таблицы в объекты со свойствами `Principal`, `Command`, `Password`. Отберите нужные через `Where-Object`.*

```
Import-Csv srv\etc\delegation.csv | Where-Object { $_.Command -eq '*' -and
$_.Password -eq 'NoPassword' } | ForEach-Object { $_.Principal } | Set-Content
-Encoding UTF8 otvety\4
```

4.5. Сузить правило. *Подсказка ученику: Измените свойство объекта в цикле и сохраните весь массив обратно командой `Export-Csv`.*

```
$r = Import-Csv srv\etc\delegation.csv
foreach ($x in $r) { if ($x.Principal -eq 'boris') { $x.Command = 'Restart-Service
nginx' } }
$r | Export-Csv srv\etc\delegation.csv -NoTypeInformation -Encoding UTF8
```

4.6. Контрольная: акт по полномочиям. *Подсказка ученику: Список имён склейте оператором `-join ','`. Правила с оболочкой отберите по регулярному выражению: `-match 'powershell\.exe|cmd\.exe'`.*

```
$a = (Get-LocalGroupMember -Group Administrators).Name | ForEach-Object { ($_
-split '\\')[-1] } | Sort-Object
$n = @(Import-Csv srv\etc\delegation.csv | Where-Object { $_.Command -match
'powershell\.exe|cmd\.exe' }).Count
@("ADMINS=$($a -join ',')", "SHELL_RULES=$n") | Set-Content -Encoding UTF8
audit-admin.txt
```

Модуль 5. Процессы и ресурсы

Linux.

5.1. Найти процесс. *Подсказка ученику: `ps -ef` показывает все процессы; отфильтруйте нужный через `grep` (приём `grep '[r]eport-gen'` исключает сам `grep`). PID — второй столбец.*

```
ps -ef | grep '[r]eport-gen' | awk '{print $2}' | head -n 1 > otvety/1
```

5.2. Процесс из временного каталога. *Подсказка ученику: В выводе `ps -ef` смотрите на столбец с командой: нужен путь, содержащий `srv/tmp/.cache`. Имя файла — последняя часть пути (`basename`).*

```
ps -ef | grep -o '[^ ]*srv/tmp/[^ ]*' | head -n 1 | xargs basename > otvety/2
```

5.3. Вежливое завершение. *Подсказка ученику: `kill PID` по умолчанию посылает сигнал 15 (SIGTERM). Проверьте результат: `ps -ef | grep report-gen`.*

```
kill "$(cat .snap/pid_report)"; sleep 1
```

5.4. Упрямый процесс. *Подсказка ученику: SIGKILL (9) нельзя перехватить или проигнорировать, но программа не успеет сохранить данные — поэтому его применяют последним.*

```
p=$(cat .snap/pid_stuck); kill "$p"; sleep 1; kill -9 "$p"; sleep 1
```

5.5. Устранить вредителя. *Подсказка ученику: Порядок: сначала завершить процесс (`kill -9`), затем удалить файл. Проверьте, что в `srv/tmp/.cache` ничего не осталось.*

```
kill -9 "$(cat .snap/pid_rogue)"; sleep 1; rm -f "srv/tmp/.cache/$(cat
.snap/rogue)"
```

5.6. Понизить приоритет. *Подсказка ученику: Чем больше значение nice, тем ниже приоритет. Обычный пользователь может только повышать nice своих процессов. Проверка: `ps -o pid,ni,cmd -p PID`.*

```
renice -n 10 -p "$(cat .snap/pid_backup)"
```

5.7. Задача в фоне. *Подсказка ученику: В bash PID последнего фонового процесса хранится в специальной переменной `$!`.*

```
sleep 300 > /dev/null 2>&1 &
echo $! > otvety/7
```

5.8. Пережить выход из системы. *Подсказка ученику: `nohup команда > файл 2>&1 &`: nohup защищает от сигнала SIGHUP при выходе, & уводит в фон, перенаправление — в файл.*

```
nohup srv/opt/job/long-job.sh > srv/var/log/long-job.out 2>&1 &
sleep 3
```

5.9. Завершить группу процессов. *Подсказка ученику: `pkill имя` отправляет SIGTERM всем процессам с таким именем. Сначала проверьте, кого заденет: `pgrep -l worker-pool`.*

```
kill $(cat .snap/pid_pool); sleep 1
```

5.10. Контрольная: что осталось. *Подсказка ученику: Из вывода `ps -ef` выберите пути с `srv/opt/app`, оставьте имя файла (`basename`) и уберите повторы (`sort -u`).*

```
ps -ef | grep -o '[^ ]*srv/opt/app/[^ ]*' | xargs -n1 basename | sort -u >
processes.txt
```

Windows.

5.1. Найти процесс. *Подсказка ученику: `Get-Process -Name имя` возвращает объект процесса; у него есть свойства `Id`, `Path`, `CPU`, `PriorityClass`.*

```
(Get-Process -Name report-gen).Id | Set-Content -Encoding UTF8 otvety\1
```

5.2. Процесс из временного каталога. *Подсказка ученику: Для некоторых системных процессов `Path` пуст или недоступен; используйте `Where-Object { $_.Path -like '*шаблон*' }`.*

```
Get-Process | Where-Object { $_.Path -like '*\srv\Temp\*' } | ForEach-Object {
$_.Name } | Select-Object -First 1 | Set-Content -Encoding UTF8 otvety\2
```

5.3. Завершить процесс. *Подсказка ученику: Можно по PID (`-Id`) или по имени (`-Name`). Проверьте результат: `Get-Process -Name report-gen`. Ключ `-Force` нужен, если процесс не отвечает.*

```
Stop-Process -Id ([int](Get-Content otvety\1)) -Force
```

5.4. Устранить вредителя. *Подсказка ученику: Сначала `Stop-Process`, затем `Remove-Item`. Если файл ещё занят, подождите секунду.*

```
$r = Get-Content otvety\2
Stop-Process -Name $r -Force
Start-Sleep -Seconds 1
Remove-Item "srv\Temp\.cache\$r.exe" -Force
```

5.5. Понизить приоритет. *Подсказка ученику: Приоритет — свойство объекта процесса, его можно присвоить: `$p.PriorityClass = 'BelowNormal'`. Допустимые значения: Idle, BelowNormal, Normal, AboveNormal, High.*

```
(Get-Process -Name nightly-backup).PriorityClass = 'BelowNormal'
```

5.6. Завершить группу процессов. *Подсказка ученику: `Stop-Process -Name` имя завершает все процессы с таким именем. Сначала посмотрите, кого заденет: `Get-Process -Name worker-pool`.*

```
Stop-Process -Name worker-pool -Force
```

5.7. Фоновая программа с выводом в файл. *Подсказка ученику: `Start-Process -FilePath ping.exe -ArgumentList '-n','300','127.0.0.1' -WindowStyle Hidden -RedirectStandardOutput файл`. Путь файла лучше указать полным.*

```
$o = (Resolve-Path 'srv\var\log').Path + '\ping.out'
Start-Process -FilePath ping.exe -ArgumentList '-n', '300', '127.0.0.1'
-WindowStyle Hidden -RedirectStandardOutput $o
Start-Sleep -Seconds 3
```

5.8. Контрольная: что осталось. *Подсказка ученику: Отберите процессы по `Path`, возьмите `Name`, уберите повторы `Sort-Object -Unique`.*

```
Get-Process | Where-Object { $_.Path -like '*\srv\app\*' } | ForEach-Object {
$_.Name } | Sort-Object -Unique | Set-Content -Encoding UTF8 processes.txt
```

Модуль 6. Пакеты и репозитории

Linux.

6.1. Сколько пакетов установлено. *Подсказка ученику: Строки пакетов в `dpkg -l` начинаются с `ii`; посчитайте их через `grep -c`.*

```
dpkg -l | grep -c '^ii' > otvety/1
```

6.2. Чей это файл. *Подсказка ученику: `dpkg -S путь` печатает «пакет: путь». Для неустановленных пакетов будет сообщение об ошибке.*

```
dpkg -S /usr/bin/htop /usr/bin/tree /usr/bin/ncdu 2>/dev/null | cut -d: -f1 >
otvety/2
```

6.3. Починить источник. *Подсказка ученику: Запустите `apt update` и прочитайте сообщение: проблема в адресе источника. Правьте `srv/etc/apt/sources.list`.*

```
sed -i 's|old-releases.example|archive.example|' srv/etc/apt/sources.list
```

6.4. Установить nginx и tmux. *Подсказка ученику: `apt install имя1 имя2`. Ключ `-y` отвечает «да» на вопросы.*

```
apt install -y nginx tmux
```

6.5. Что подтянулось само. *Подсказка ученику: Пакеты, которые вы не называли, а они пришли как зависимости, помечены «auto». Покажет `apt-mark showauto`.*

```
apt-mark showauto | sort > otvety/5
```

6.6. Убрать telnet. *Подсказка ученику: `apt purge` удаляет пакет вместе с настройками, `apt remove` оставляет настройки.*

```
apt purge -y telnet
```

6.7. Что нужно обновить. *Подсказка ученику: Вывод `apt list --upgradable` содержит строки вида `имя/stable версия … [upgradable from …]`; имя — до символа `/`.*

```
apt list --upgradable 2>/dev/null | grep '/stable' | cut -d/ -f1 | sort > otvety/7
```

6.8. Обновить всё, кроме openssl. *Подсказка ученику: Сначала `apt-mark hold имя`, потом `apt upgrade`. Проверьте: `apt-mark showhold`.*

```
apt-mark hold openssl; apt upgrade -y
```

6.9. Файлы конфигурации пакета. *Подсказка ученику: `dpkg -L пакет` перечисляет файлы пакета; отфильтруйте их по началу строки: `grep '^/etc'`.*

```
dpkg -L nginx-common | grep '^/etc' | sort > otvety/9
```

6.10. Контрольная: список установленного вручную. *Подсказка ученику: Пометку «manual» хранит `apt-mark`: `apt-mark showmanual`. Не забудьте отсортировать вывод.*

```
apt-mark showmanual | sort > packages.txt
```

Windows.

6.1. Сколько программ установлено. *Подсказка ученику: Вывод `winget list` — текст: первая строка заголовок. Пропустите её (`Select-Object -Skip 1`) и посчитайте остальные (`.Count`).*

```
(winget list | Select-Object -Skip 1 | Where-Object { $_.Trim() }).Count |
Set-Content -Encoding UTF8 otvety\1
```

6.2. Найти нужный пакет. *Подсказка ученику: Идентификатор вида `Издатель.Программа` — самый надёжный способ указать пакет: `winget install --id …`.*

```
'Notepad++.Notepad++' | Set-Content -Encoding UTF8 otvety\2
```

6.3. Установить программы. *Подсказка ученику: Одна команда на один пакет: `winget install --id Идентификатор`.*

```
winget install --id Notepad++.Notepad++
winget install --id 7zip.7zip
```

6.4. Убрать старую Java. *Подсказка ученику: Перед удалением убедитесь, что Java не нужна другим программам; в настоящем сервере это отдельный вопрос.*

```
winget uninstall --id Oracle.JavaRuntimeEnvironment
```

6.5. Что нужно обновить. *Подсказка ученику: Идентификатор — первое слово строки таблицы. Строки с обновлениями содержат точку в первом слове.*

```
winget upgrade | Select-String -Pattern '^\S+\.\S+\s' | ForEach-Object { ($_.Line
-split '\s+')[0] } | Sort-Object | Set-Content -Encoding UTF8 otvety\5
```

6.6. Обновить всё, кроме PowerShell. *Подсказка ученику: Сначала `winget pin add --id …`, потом `winget upgrade --all`. Проверка: `winget pin list` и `winget upgrade`.*

```
winget pin add --id Microsoft.PowerShell
winget upgrade --all
```

6.7. Контрольная: список для аудита. *Подсказка ученику: Идентификатор — первое слово каждой строки `winget list`; отсортируйте и запишите в файл.*

```
winget list | Select-Object -Skip 1 | Where-Object { $_.Trim() } | ForEach-Object
{ ($_ -split '\s+')[0] } | Sort-Object | Set-Content -Encoding UTF8 packages.txt
```

Модуль 7. Службы и автозапуск (systemd)

Linux.

7.1. Какие службы упали. *Подсказка ученику: `systemctl list-units --failed` печатает только упавшие юниты. Имена вида `имя.service` вытащите через `grep -o`.*

```
systemctl list-units --failed | grep -o '[a-z-]*\.service' | sed 's/\.service//' |
sort -u > otvety/1
```

7.2. Код ошибки магазина. *Подсказка ученику: Код `203/EXEC` означает, что systemd не смог выполнить программу, указанную в `ExecStart`. Посмотрите также `journalctl -u shop-api`.*

```
systemctl status shop-api | grep -o 'status=[0-9A-Z/]*' | cut -d= -f2 > otvety/2
```

7.3. Исправить магазин. *Подсказка ученику: Сравните путь в `ExecStart=` с содержимым `srv/opt/app`. После правки unit-файла всегда `systemctl daemon-reload`.*

```
sed -i 's|/opt/app/shop-api-v2|/opt/app/shop-api|'
srv/etc/systemd/system/shop-api.service
systemctl daemon-reload; systemctl start shop-api
```

7.4. Служба без пользователя. *Подсказка ученику: Служебным пользователям задают оболочку `/usr/sbin/nologin`, ключ `-r` создаёт системную учётную запись.*

```
useradd -r -s /usr/sbin/nologin svc-report; systemctl start report-worker
```

7.5. Нет файла настроек. *Подсказка ученику: В unit-файле найдите строку `EnvironmentFile=` — это путь, который нужно создать (внутри `srv`).*

```
echo 'SYNC_HOST=backup.local' > srv/etc/legacy-sync.env; systemctl start
legacy-sync
```

7.6. Неудачные входы по SSH. *Подсказка ученику: `journalctl -u имя` выводит журнал службы; посчитайте нужные строки через `grep -c`.*

```
journalctl -u ssh --no-pager | grep -c 'Failed password' > otvety/6
```

7.7. Новая служба hello. *Подсказка ученику: Три раздела: `[Unit]`, `[Service]`, `[Install]`. `systemctl enable --now` включает автозапуск и сразу запускает.*

```
printf '%s\n' '[Unit]' 'Description=Hello' '' '[Service]'
'ExecStart=/usr/local/bin/hello.sh' 'Restart=on-failure' '' '[Install]'
'WantedBy=multi-user.target' > srv/etc/systemd/system/hello.service
systemctl daemon-reload; systemctl enable --now hello
```

7.8. Перезапуск при сбое. *Подсказка ученику: Две строки в разделе `[Service]`. Запомните порядок: изменили файл → `daemon-reload` → (при необходимости) `restart`.*

```
sed -i 's/^Restart=no$/Restart=on-failure/' srv/etc/systemd/system/flaky.service
sed -i '/^Restart=on-failure$/a RestartSec=5' srv/etc/systemd/system/flaky.service
systemctl daemon-reload
```

7.9. Отключить лишнее. *Подсказка ученику: `disable` убирает автозапуск при загрузке, `stop` останавливает сейчас. Ключ `--now` делает оба действия.*

```
systemctl disable --now bluetooth
```

7.10. Контрольная: что стартует при загрузке. *Подсказка ученику: Сначала `systemctl list-unit-files --state=enabled`, затем отберите столбец имён и уберите `.service`.*

```
systemctl list-unit-files --state=enabled | grep -o '[a-z-]*\.service' | sed
's/\.service//' | sort > services.txt
```

Windows.

7.1. Какие службы не работают. *Подсказка ученику: `Get-Service` выдаёт объекты со свойствами `Name`, `Status`, `StartType`. Отберите нужные через `Where-Object`.*

```
Get-Service | Where-Object { $_.StartType -eq 'Automatic' -and $_.Status -eq
'Stopped' } | Sort-Object Name | ForEach-Object { $_.Name } | Set-Content
-Encoding UTF8 otvety\1
```

7.2. Код ошибки магазина. *Подсказка ученику: В `FilterHashtable` можно указать `LogName`, `Id`, `Level` (2 — ошибка), `ProviderName`. Текст причины находится в свойстве `Message`.*

```
(Get-WinEvent -FilterHashtable @{ LogName = 'System'; Level = 2 } | Where-Object {
$_.Message -like '*shop-api*' } | Select-Object -First 1).Id | Set-Content
-Encoding UTF8 otvety\2
```

7.3. Исправить магазин. *Подсказка ученику: Синтаксис `sc.exe` необычен: `параметр=` пишется слитно, а значение — после пробела. Проверка: `sc.exe qc shop-api` и `Get-Service shop-api`.*

```
sc.exe config shop-api binPath= "C:\Program Files\Shop\shop-api.exe"
Start-Service shop-api
```

7.4. Служба без пользователя. *Подсказка ученику: Пароль нужно сначала превратить в безопасную строку: `ConvertTo-SecureString 'текст' -AsPlainText -Force`.*

```
New-LocalUser -Name svc-report -Password (ConvertTo-SecureString 'Svc-Passw0rd!'
-AsPlainText -Force) -Description 'Служебная запись' | Out-Null
Start-Service report-worker
```

7.5. Зависимая служба. *Подсказка ученику: Зависимости перечислены в `sc.exe qc legacy-sync` (поле DEPENDENCIES). Сначала запускают то, от чего зависят.*

```
Start-Service queue-svc
Start-Service legacy-sync
```

7.6. Неудачные входы. *Подсказка ученику: Заключите результат в `@( … )`, чтобы `.Count` работал и для единственной записи.*

```
@(Get-WinEvent -FilterHashtable @{ LogName = 'Security'; Id = 4625 }).Count |
Set-Content -Encoding UTF8 otvety\6
```

7.7. Перезапуск при сбое. *Подсказка ученику: Время в `actions=` задаётся в миллисекундах (5000 = 5 секунд), а в `reset=` — в секундах (86400 = сутки).*

```
sc.exe failure flaky reset= 86400 actions= restart/5000
```

7.8. Отключить лишнее. *Подсказка ученику: Остановка действует «сейчас», тип запуска — при следующей загрузке. Нужны оба действия.*

```
Stop-Service bluetooth
Set-Service bluetooth -StartupType Disabled
```

7.9. Контрольная: что стартует при загрузке. *Подсказка ученику: Отберите службы с `StartType -eq 'Automatic'`, возьмите имена, отсортируйте.*

```
Get-Service | Where-Object { $_.StartType -eq 'Automatic' } | Sort-Object Name |
ForEach-Object { $_.Name } | Set-Content -Encoding UTF8 services-auto.txt
```

Модуль 8. Журналы и расследование

Linux.

8.1. Сколько было запросов. *Подсказка ученику: Строк в файле столько же, сколько запросов: `wc -l`.*

```
wc -l < srv/var/log/nginx/access.log > otvety/1
```

8.2. Самый активный адрес. *Подсказка ученику: Классическая цепочка: взять первое поле, `sort`, `uniq -c`, `sort -rn`, `head -n 1`.*

```
awk '{print $1}' srv/var/log/nginx/access.log | sort | uniq -c | sort -rn | head
-n 1 | awk '{print $2}' > otvety/2
```

8.3. Сколько страниц не найдено. *Подсказка ученику: Код ответа — девятое поле. `awk '$9 == 404'` отберёт строки, `wc -l` их посчитает.*

```
awk '$9 == 404' srv/var/log/nginx/access.log | wc -l > otvety/3
```

8.4. Самый популярный путь. *Подсказка ученику: Та же цепочка, что и в задании 2, но по седьмому полю.*

```
awk '{print $7}' srv/var/log/nginx/access.log | sort | uniq -c | sort -rn | head
-n 1 | awk '{print $2}' > otvety/4
```

8.5. Подбор паролей на сайте. *Подсказка ученику: Сначала `grep 'POST /login'`, затем отбор по коду 401, подсчёт по адресам и порог `$1 > 20`. Адрес с единичными ошибками — просто забывчивый пользователь.*

```
grep 'POST /login' srv/var/log/nginx/access.log | awk '$9 == 401 {print $1}' |
sort | uniq -c | awk '$1 > 20 {print $2}' | sort > otvety/5
```

8.6. Сканер уязвимостей. *Подсказка ученику: Почти то же, что в задании 5, но фильтр — код 404 и порог 50; результат сохраните в файл.*

```
awk '$9 == 404 {print $1}' srv/var/log/nginx/access.log | sort | uniq -c | awk '$1
> 50 {print $2}' | sort > blocklist.txt
```

8.7. Самый медленный запрос. *Подсказка ученику: `sort -k11 -n -r` упорядочит по одиннадцатому полю как по числу, по убыванию. Первая строка — самая медленная.*

```
sort -k11 -n -r srv/var/log/nginx/access.log | head -n 1 | awk '{print $7}' >
otvety/7
```

8.8. Объём трафика. *Подсказка ученику: В `awk` накапливайте сумму в переменной и печатайте её в блоке `END`.*

```
awk '{s += $10} END {print s}' srv/var/log/nginx/access.log > otvety/8
```

8.9. Кого подбирали по SSH. *Подсказка ученику: Имя пользователя — девятое поле строки с `Failed password for`. Дальше знакомая цепочка `sort | uniq -c | sort -rn`.*

```
grep 'Failed password' srv/var/log/auth.log | awk '{print $9}' | sort | uniq -c |
sort -rn | head -n 1 | awk '{print $2}' > otvety/9
```

8.10. Ротация журналов. *Подсказка ученику: Блок в фигурных скобках: путь, `{`, директивы по одной в строке, `}`. Старую директиву `daily` нужно заменить, а не дополнить.*

```
printf '%s\n' '/var/log/shop/*.log {' '    weekly' '    rotate 8' '    compress' '
missingok' '    notifempty' '}' > srv/etc/logrotate.d/shop
```

Windows.

8.1. Сколько было запросов. *Подсказка ученику: `Get-Content файл | Measure-Object -Line` считает строки; можно и `(Get-Content файл).Count`.*

```
(Get-Content srv\var\log\iis\access.log | Measure-Object -Line).Lines |
Set-Content -Encoding UTF8 otvety\1
```

8.2. Самый активный адрес. *Подсказка ученику: `Group-Object Ip` считает записи по каждому адресу (свойство `Count`), `Sort-Object Count -Descending` ставит самый частый первым.*

```
$l = Get-Content srv\var\log\iis\access.log | ForEach-Object { $f = $_ -split ' ';
[pscustomobject]@{ Ip = $f[2]; Method = $f[3]; Path = $f[4]; Status = [int]$f[5];
Bytes = [int64]$f[6]; Time = [double]$f[7] } }
    ($l | Group-Object Ip | Sort-Object Count -Descending | Select-Object -First
    1).Name | Set-Content -Encoding UTF8 otvety\2
```

8.3. Сколько страниц не найдено. *Подсказка ученику: Код ответа — свойство `Status` (индекс 5 после `-split ' '`). `Where-Object Status -eq 404`, затем `.Count`.*

```
$l = Get-Content srv\var\log\iis\access.log | ForEach-Object { $f = $_ -split ' ';
[pscustomobject]@{ Ip = $f[2]; Method = $f[3]; Path = $f[4]; Status = [int]$f[5];
Bytes = [int64]$f[6]; Time = [double]$f[7] } }
    @($l | Where-Object Status -eq 404).Count | Set-Content -Encoding UTF8
    otvety\3
```

8.4. Самый популярный путь. *Подсказка ученику: То же, что в задании 2, но группировать по `Path`.*

```
$l = Get-Content srv\var\log\iis\access.log | ForEach-Object { $f = $_ -split ' ';
[pscustomobject]@{ Ip = $f[2]; Method = $f[3]; Path = $f[4]; Status = [int]$f[5];
Bytes = [int64]$f[6]; Time = [double]$f[7] } }
    ($l | Group-Object Path | Sort-Object Count -Descending | Select-Object -First
    1).Name | Set-Content -Encoding UTF8 otvety\4
```

8.5. Подбор паролей на сайте. *Подсказка ученику: Фильтр по методу, пути и коду, группировка по адресу и порог `Where-Object Count -gt 20`.*

```
$l = Get-Content srv\var\log\iis\access.log | ForEach-Object { $f = $_ -split ' ';
[pscustomobject]@{ Ip = $f[2]; Method = $f[3]; Path = $f[4]; Status = [int]$f[5];
Bytes = [int64]$f[6]; Time = [double]$f[7] } }
    $l | Where-Object { $_.Method -eq 'POST' -and $_.Path -eq '/login' -and
    $_.Status -eq 401 } | Group-Object Ip | Where-Object Count -gt 20 |
    ForEach-Object { $_.Name } | Sort-Object | Set-Content -Encoding UTF8 otvety\5
```

8.6. Сканер уязвимостей. *Подсказка ученику: Почти то же, что в задании 5, но фильтр — код 404 и порог 50; результат сохраните в файл.*

```
$l = Get-Content srv\var\log\iis\access.log | ForEach-Object { $f = $_ -split ' ';
[pscustomobject]@{ Ip = $f[2]; Method = $f[3]; Path = $f[4]; Status = [int]$f[5];
Bytes = [int64]$f[6]; Time = [double]$f[7] } }
    $l | Where-Object Status -eq 404 | Group-Object Ip | Where-Object Count -gt 50
    | ForEach-Object { $_.Name } | Sort-Object | Set-Content -Encoding UTF8
    blocklist.txt
```

8.7. Самый медленный запрос. *Подсказка ученику: Время — число; свойство `Time` сортируется как число (`[double]`), если преобразовать его при разборе строки.*

```
$l = Get-Content srv\var\log\iis\access.log | ForEach-Object { $f = $_ -split ' ';
[pscustomobject]@{ Ip = $f[2]; Method = $f[3]; Path = $f[4]; Status = [int]$f[5];
Bytes = [int64]$f[6]; Time = [double]$f[7] } }
    ($l | Sort-Object Time -Descending | Select-Object -First 1).Path |
    Set-Content -Encoding UTF8 otvety\7
```

8.8. Объём трафика. *Подсказка ученику: `Measure-Object Bytes -Sum` возвращает объект со свойством `Sum`.*

```
$l = Get-Content srv\var\log\iis\access.log | ForEach-Object { $f = $_ -split ' ';
[pscustomobject]@{ Ip = $f[2]; Method = $f[3]; Path = $f[4]; Status = [int]$f[5];
Bytes = [int64]$f[6]; Time = [double]$f[7] } }
    ($l | Measure-Object Bytes -Sum).Sum | Set-Content -Encoding UTF8 otvety\8
```

8.9. Кого подбирали по паролю. *Подсказка ученику: Имя из сообщения достаньте регулярным выражением: `if ($_.Message -match 'Имя: ([^;]+);') { $Matches[1] }`.*

```
Get-WinEvent -FilterHashtable @{ LogName = 'Security'; Id = 4625 } |
ForEach-Object { if ($_.Message -match 'Имя: ([^;]+);') { $Matches[1] } } |
Group-Object | Sort-Object Count -Descending | Select-Object -First 1 |
ForEach-Object { $_.Name } | Set-Content -Encoding UTF8 otvety\9
```

8.10. Архив старых журналов. *Подсказка ученику: Отберите файлы по `LastWriteTime`, передайте их пути в `Compress-Archive -Path … -DestinationPath …`, затем удалите исходные.*

```
$old = Get-ChildItem srv\var\log\old -Filter *.log | Where-Object {
$_.LastWriteTime -lt (Get-Date).AddDays(-30) }
Compress-Archive -Path $old.FullName -DestinationPath srv\var\log\old\archive.zip
$old | Remove-Item
```

Модуль 9. Диски, место и файловые системы

Linux.

9.1. Что занимает место. *Подсказка ученику: `du -sk путь/*` печатает размер каждого элемента в килобайтах; результат упорядочьте `sort -rn` и возьмите первую строку.*

```
du -sk srv/var/* | sort -rn | head -n 1 | awk '{print $2}' | xargs basename >
otvety/1
```

9.2. Три самых больших файла. *Подсказка ученику: `stat -c '%s %n'` даёт размер и имя; упорядочьте по размеру в обратном порядке и возьмите первые три.*

```
find srv/var -type f -exec stat -c '%s %n' {} + | sort -rn | head -n 3 | awk
'{print $2}' | xargs -n1 basename | sort > otvety/2
```

9.3. Где кончается место. *Подсказка ученику: Столбец `Use%` — пятый, точка монтирования — шестой. Нужна строка, где процент не меньше 90.*

```
df -h | awk 'NR > 1 {gsub("%","",$5); if ($5 >= 90) print $6}' | head -n 1 >
otvety/3
```

9.4. Свободное место есть, а писать нельзя. *Подсказка ученику: Ключ `-i` показывает не байты, а индексные дескрипторы (inode): один маленький файл расходует один inode.*

```
df -i | awk '$5 == "100%" {print $6}' > otvety/4
```

9.5. Сколько файлов в очереди. *Подсказка ученику: `ls каталог | wc -l` считает имена файлов. Каждая строка вывода `ls` — один файл.*

```
ls srv/var/spool/mailq | wc -l > otvety/5
```

9.6. Очистить очередь. *Подсказка ученику: `find каталог -type f -delete` удаляет только файлы. Когда файлов тысячи, `rm *` может отказаться: «слишком длинный список аргументов».*

```
find srv/var/spool/mailq -type f -delete
```

9.7. Старые архивы. *Подсказка ученику: Сначала выведите кандидатов без `-delete` и убедитесь, что их ровно шесть. `-mtime +30` — старше 30 суток.*

```
find srv/var/log/app -name '*.gz' -mtime +30 -delete
```

9.8. UUID нового диска. *Подсказка ученику: `blkid` печатает пары `ключ="значение"`. Вытащите `UUID="…"`, затем уберите кавычки (`cut -d'"' -f2`).*

```
blkid | grep '^/dev/sdb1' | grep -o 'UUID="[^"]*"' | cut -d'"' -f2 > otvety/8
```

9.9. Подключение по UUID. *Подсказка ученику: Шесть полей через пробел: что, куда, тип, параметры, dump (0), порядок проверки (2). `nofail` не даёт серверу «зависнуть» при загрузке, если диска нет.*

```
u=$(cat otvety/8); mkdir -p srv/data; echo "UUID=$u /data ext4 defaults,nofail 0
2" >> srv/etc/fstab
```

9.10. Контрольная: отчёт для руководителя. *Подсказка ученику: Ответы заданий 1, 3 и 4 лежат в `otvety/`. Соберите файл командой `printf` или тремя `echo`.*

```
printf 'BIGGEST=%s\nFULL=%s\nINODES=%s\n' "$(cat otvety/1)" "$(cat otvety/3)"
"$(cat otvety/4)" > disk-report.txt
```

Windows.

9.1. Что занимает место. *Подсказка ученику: Для каждого подкаталога посчитайте сумму `Length` его файлов и отсортируйте по убыванию.*

```
Get-ChildItem srv\var -Directory | ForEach-Object { [pscustomobject]@{ Name =
$_.Name; Size = (Get-ChildItem $_.FullName -Recurse -File | Measure-Object Length
-Sum).Sum } } | Sort-Object Size -Descending | Select-Object -First 1 |
ForEach-Object { $_.Name } | Set-Content -Encoding UTF8 otvety\1
```

9.2. Три самых больших файла. *Подсказка ученику: `Select-Object -First 3` после сортировки по убыванию даёт три самых больших; затем имена в алфавитном порядке.*

```
Get-ChildItem srv\var -Recurse -File | Sort-Object Length -Descending |
Select-Object -First 3 | ForEach-Object { $_.Name } | Sort-Object | Set-Content
-Encoding UTF8 otvety\2
```

9.3. Где кончается место. *Подсказка ученику: Доля свободного места — `SizeRemaining / Size`. Нужен том, где она меньше 0,1.*

```
Get-Volume | Where-Object { $_.DriveLetter -ne 'C' -and $_.SizeRemaining / $_.Size
-lt 0.1 } | ForEach-Object { $_.DriveLetter } | Set-Content -Encoding UTF8
otvety\3
```

9.4. Сколько файлов в очереди. *Подсказка ученику: `@(Get-ChildItem каталог -File).Count` — скобки `@()` гарантируют массив даже при одном файле.*

```
@(Get-ChildItem srv\var\spool\mailq -File).Count | Set-Content -Encoding UTF8
otvety\4
```

9.5. Очистить очередь. *Подсказка ученику: `Get-ChildItem каталог -File | Remove-Item`: удаляются файлы, а не сам каталог. Сначала проверьте список без `Remove-Item`.*

```
Get-ChildItem srv\var\spool\mailq -File | Remove-Item -Force
```

9.6. Старые архивы. *Подсказка ученику: Сначала выведите кандидатов без `Remove-Item` и убедитесь, что их ровно шесть. Граница — `(Get-Date).AddDays(-30)`.*

```
Get-ChildItem srv\var\log\app -Filter *.zip | Where-Object { $_.LastWriteTime -lt
(Get-Date).AddDays(-30) } | Remove-Item
```

9.7. Контрольная: отчёт для руководителя. *Подсказка ученику: Ответы заданий 1 и 3 лежат в `otvety\`. Соберите две строки в массив и запишите `Set-Content`.*

```
@("BIGGEST=$(Get-Content otvety\1)", "FULL=$(Get-Content otvety\3)") | Set-Content
-Encoding UTF8 disk-report.txt
```

Модуль 10. Резервное копирование и восстановление

Linux.

10.1. Полная копия. *Подсказка ученику: `tar czf архив -C каталог_откуда имя`: ключ `-C` переходит в каталог до добавления файлов.*

```
tar czf backup/full.tar.gz -C srv/srv shop
```

10.2. Копия без мусора. *Подсказка ученику: `--exclude='шаблон'` ставится до перечня того, что архивируется. Проверьте результат: `tar tzf backup/clean.tar.gz`.*

```
tar czf backup/clean.tar.gz --exclude='*.tmp' --exclude='cache' -C srv/srv shop
```

10.3. Что в ночном архиве. *Подсказка ученику: `tar tzf` печатает список содержимого. Имена каталогов оканчиваются на `/`, их можно исключить через `grep -v`.*

```
tar tzf backup/nightly.tar.gz | grep -vc '/$' > otvety/3
```

10.4. Контрольная сумма. *Подсказка ученику: Выполняйте `sha256sum` из каталога `backup` и указывайте имя без пути: иначе в файле сумм будет путь, которого при проверке не найдётся.*

```
( cd backup && sha256sum clean.tar.gz > clean.sha256 )
```

10.5. Какой архив испорчен. *Подсказка ученику: Проверку надо запускать в каталоге с архивами. Строка с проблемой заканчивается словом `FAILED`.*

```
( cd backup/set && sha256sum -c SHA256SUMS 2>&1 | grep FAILED | cut -d: -f1 ) >
otvety/5
```

10.6. Восстановление в другое место. *Подсказка ученику: `tar xzf архив -C куда`. Если каталога «куда» нет, `tar` откажется: создайте его заранее (`mkdir -p`).*

```
mkdir -p srv/restore; tar xzf backup/clean.tar.gz -C srv/restore
```

10.7. Восстановить один файл. *Подсказка ученику: После архива в `tar x` можно перечислить нужные пути (посмотрите их в `tar tzf`). Пути в архиве начинаются с `shop/`, значит, распаковывать надо в `srv/srv`.*

```
tar xzf backup/nightly.tar.gz -C srv/srv shop/config/settings.ini
```

10.8. Что изменилось после отметки. *Подсказка ученику: `find каталог -type f -newer файл-отметка` выводит файлы, изменённые позже отметки. Затем оставьте имена без путей.*

```
find srv/srv/shop -type f -newer backup/marker | xargs -n1 basename | sort >
otvety/8
```

10.9. Ротация копий. *Подсказка ученику: Даты в именах записаны в виде ГГГГ-ММ-ДД, поэтому алфавитный порядок совпадает с хронологическим. Исключите три последних имени и удалите остальные.*

```
ls backup/daily-*.tar.gz | sort | head -n -3 | xargs rm
```

10.10. Контрольная: отчёт о копии. *Подсказка ученику: Подстановка `$(…)` внутри `printf` подставит значения. Хеш — первое поле вывода `sha256sum`.*

```
printf 'FILES=%s\nSHA=%s\n' "$(tar tzf backup/clean.tar.gz | grep -vc '/$')"
"$(sha256sum backup/clean.tar.gz | cut -d' ' -f1)" > backup-report.txt
```

Windows.

10.1. Полная копия. *Подсказка ученику: `Compress-Archive -Path что -DestinationPath куда`. Если файл архива уже есть, добавьте `-Force`.*

```
Compress-Archive -Path srv\Shop -DestinationPath backup\full.zip -Force
```

10.2. Копия без мусора. *Подсказка ученику: Ключ `/E` копирует подкаталоги, `/XF` исключает файлы по маске, `/XD` — каталоги. Проверьте результат `Get-ChildItem backup\clean -Recurse`.*

```
robocopy srv\Shop backup\clean /E /XF *.tmp /XD cache | Out-Null
```

10.3. Что в ночном архиве. *Подсказка ученику: Записи архива — `$a.Entries`; у каталогов `FullName` оканчивается на `/` или `\`.*

```
Add-Type -AssemblyName System.IO.Compression.FileSystem
$a = [IO.Compression.ZipFile]::OpenRead((Resolve-Path backup\nightly.zip).Path)
@($a.Entries | Where-Object { -not ($_.FullName.EndsWith('/') -or
$_.FullName.EndsWith('\')) }).Count | Set-Content -Encoding UTF8 otvety\3
$a.Dispose()
```

10.4. Контрольная сумма. *Подсказка ученику: Хеш — свойство `Hash` результата `Get-FileHash`. Строку соберите подстановкой `"$h  full.zip"`.*

```
$h = (Get-FileHash backup\full.zip -Algorithm SHA256).Hash
"$h  full.zip" | Set-Content -Encoding ASCII backup\full.sha256
```

10.5. Какой архив испорчен. *Подсказка ученику: Разберите каждую строку файла сумм (хеш и имя), вычислите хеш файла и сравните без учёта регистра (`-ine`).*

```
Get-Content backup\set\SHA256SUMS.txt | ForEach-Object {
    $p = $_ -split '\s+'
    if ((Get-FileHash (Join-Path backup\set $p[1]) -Algorithm SHA256).Hash -ine
    $p[0]) { $p[1] }
} | Set-Content -Encoding UTF8 otvety\5
```

10.6. Восстановление в другое место. *Подсказка ученику: `Expand-Archive -Path архив -DestinationPath куда`. Каталог назначения будет создан сам.*

```
Expand-Archive -Path backup\full.zip -DestinationPath srv\restore -Force
```

10.7. Восстановить один файл. *Подсказка ученику: Распаковка во временный каталог (`$env:TEMP`) не затрагивает рабочие данные. Затем `Copy-Item` одного файла с `-Force`.*

```
$t = Join-Path $env:TEMP ('nightly-' + [guid]::NewGuid())
Expand-Archive -Path backup\nightly.zip -DestinationPath $t
Copy-Item (Join-Path $t 'Shop\config\settings.ini') srv\Shop\config\settings.ini
-Force
Remove-Item $t -Recurse -Force
```

10.8. Что изменилось после отметки. *Подсказка ученику: Сравнивайте `LastWriteTime` каждого файла с `LastWriteTime` отметки: `Where-Object { $_.LastWriteTime -gt $m }`.*

```
$m = (Get-Item backup\marker).LastWriteTime
Get-ChildItem srv\Shop -Recurse -File | Where-Object { $_.LastWriteTime -gt $m } |
Sort-Object Name | ForEach-Object { $_.Name } | Set-Content -Encoding UTF8
otvety\8
```

10.9. Ротация копий. *Подсказка ученику: Даты в именах записаны ГГГГ-ММ-ДД, поэтому алфавитный порядок совпадает с хронологическим. `Select-Object -SkipLast 3` пропускает три последних.*

```
Get-ChildItem backup -Filter 'daily-*.txt' | Sort-Object Name | Select-Object
-SkipLast 3 | Remove-Item
```

10.10. Контрольная: отчёт о копии. *Подсказка ученику: Две строки в массиве, значения подставьте командами из заданий 2 и 4.*

```
$n = @(Get-ChildItem backup\clean -Recurse -File).Count
@("FILES=$n", "SHA=$((Get-FileHash backup\full.zip -Algorithm SHA256).Hash)") |
Set-Content -Encoding UTF8 backup-report.txt
```

Модуль 11. Автоматизация: сценарии и cron

Linux.

11.1. Сценарий-приветствие. *Подсказка ученику: `$1` — первый аргумент, `$#` — число аргументов, `$0` — имя сценария. Сообщение об ошибке направьте в поток ошибок: `>&2`.*

```
printf '%s\n' '#!/bin/bash' 'if [ $# -lt 1 ]; then echo "Использование: $0 ИМЯ"
>&2; exit 1; fi' 'echo "Привет, $1!"' > srv/opt/scripts/greet.sh
```

11.2. Коды выхода. *Подсказка ученику: Проверки `-e` (существует) и `-s` (не пуст). Код последней команды доступен в `$?`, а `exit N` завершает сценарий с кодом N.*

```
printf '%s\n' '#!/bin/bash' '[ -e "$1" ] || exit 1' '[ -s "$1" ] || exit 2' 'exit
0' > srv/opt/scripts/check-file.sh
```

11.3. Починить count-lines.sh. *Подсказка ученику: Две ошибки: переменную нужно заключить в кавычки (`"$1"`), а имя файла из вывода `wc` убирают перенаправлением ввода `<`.*

```
printf '%s\n' '#!/bin/bash' 'wc -l < "$1"' > srv/opt/scripts/count-lines.sh
```

11.4. Замок и trap. *Подсказка ученику: `trap 'rm -f "$LOCKFILE"' EXIT` выполняется при выходе из сценария по любой причине: нормально, по `exit` или по ошибке.*

```
printf '%s\n' '#!/bin/bash' 'trap '"'"'rm -f "$LOCKFILE"'"'"' EXIT' 'touch
"$LOCKFILE"' '[ "$FAIL" == "1" ] && exit 3' 'sleep 1' 'exit 0' >
srv/opt/scripts/with-lock.sh
```

11.5. Сумма продаж. *Подсказка ученику: `awk -F, '{s += $2} END {print s}'` файл. Внутри сценария не забудьте кавычки вокруг `$1`.*

```
printf '%s\n' '#!/bin/bash' 'awk -F, '"'"'{s += $2} END {print s}'"'"' "$1"' >
srv/opt/scripts/sumcsv.sh
bash srv/opt/scripts/sumcsv.sh srv/var/lib/lab/sales.csv > otvety/5
```

11.6. Когда сработает. *Подсказка ученику: Выражение в кавычках передайте первым аргументом: `cron-next '0 3 * * 0'`. Не забывайте, что «сейчас» — воскресенье.*

```
{ cron-next '*/15 9-17 * * 1-5'; cron-next '0 3 * * 0'; cron-next '30 9 1 * *'; }
> otvety/6
```

11.7. Ночная копия. *Подсказка ученику: Формат строки: минута час день_месяца месяц день_недели команда. «Каждый день» — звёздочки в последних трёх полях.*

```
echo '30 2 * * * /opt/scripts/backup-shop.sh' > mycron.txt; crontab mycron.txt
```

11.8. Ошибка в расписании. *Подсказка ученику: Сообщение укажет номер строки. Часы в cron — от 0 до 23, так что значение 25 недопустимо. Задание «в 3:00» — это минута 0 и час 3.*

```
sed -i 's|^0 25 |0 3 |' broken.cron; crontab broken.cron
```

11.9. Свое выражение. *Подсказка ученику: Шаг `*/10` в поле минут, диапазон `8-19` в поле часов (последний запуск в 19:50), дни недели `1-6` (воскресенье — 0 или 7).*

```
echo '*/10 8-19 * * 1-6' > otvety/9
```

11.10. Контрольная: частые слова. *Подсказка ученику: Шаблон `getopts "n:" o` в цикле `while`, значение ключа — в `$OPTARG`, после цикла `shift $((OPTIND-1))` оставит только позиционные аргументы.*

```
  cat > srv/opt/scripts/top.sh <<'EOT'
#!/bin/bash
n=3
while getopts "n:" o; do
  case $o in n) n=$OPTARG ;; *) exit 2 ;; esac
done
shift $((OPTIND - 1))
[ -n "$1" ] || { echo "Использование: $0 [-n N] ФАЙЛ" >&2; exit 2; }
sort "$1" | uniq -c | sort -rn | head -n "$n" | awk '{print $1, $2}'
EOT
```

Windows.

11.1. Сценарий-приветствие. *Подсказка ученику: Параметр сценария объявляют строкой `param([string]$Name)` в начале. Код выхода задаёт `exit N`; после запуска он доступен в `$LASTEXITCODE`.*

```
Set-Content -Encoding UTF8 srv\scripts\greet.ps1 -Value @(
    'param([string]$Name)',
    'if (-not $Name) { Write-Error "Использование: greet.ps1 ИМЯ"; exit 1 }',
    'Write-Output "Привет, $Name!"')
```

11.2. Коды выхода. *Подсказка ученику: Проверки `Test-Path` (существует) и свойство файла `Length` (размер в байтах). Выход — `exit N`.*

```
Set-Content -Encoding UTF8 srv\scripts\check-file.ps1 -Value @(
    'param([string]$Path)',
    'if (-not (Test-Path -LiteralPath $Path)) { exit 1 }',
    'if ((Get-Item -LiteralPath $Path).Length -eq 0) { exit 2 }',
    'exit 0')
```

11.3. Починить count-lines.ps1. *Подсказка ученику: Нужное число — свойство `Lines` результата `Measure-Object`. `-LiteralPath` делает путь буквальным (без масок).*

```
Set-Content -Encoding UTF8 srv\scripts\count-lines.ps1 -Value
@('param([string]$Path)', '(Get-Content -LiteralPath $Path | Measure-Object
-Line).Lines')
```

11.4. Замок и finally. *Подсказка ученику: Блок `finally` выполняется при любом выходе из `try`, в том числе после `exit` или ошибки.*

```
Set-Content -Encoding UTF8 srv\scripts\with-lock.ps1 -Value @(
    'New-Item -ItemType File -Path $env:LOCKFILE -Force | Out-Null',
    'try {',
    '    if ($env:FAIL -eq "1") { exit 3 }',
    '    Start-Sleep -Seconds 1',
    '} finally {',
    '    Remove-Item $env:LOCKFILE -Force -ErrorAction SilentlyContinue',
    '}',
    'exit 0')
```

11.5. Сумма продаж. *Подсказка ученику: `Import-Csv -Header Name,Qty` задаёт имена столбцов, если заголовка в файле нет. Сумма — `Measure-Object Qty -Sum`.*

```
Set-Content -Encoding UTF8 srv\scripts\sumcsv.ps1 -Value @('param([string]$Path)',
'(Import-Csv -LiteralPath $Path -Header Name,Qty | Measure-Object Qty -Sum).Sum')
& srv\scripts\sumcsv.ps1 srv\var\lib\lab\sales.csv | Set-Content -Encoding UTF8
otvety\5
```

11.6. Когда сработает. *Подсказка ученику: Не забывайте, что «сейчас» — воскресенье 09:30: сегодняшнее воскресное время 03:00 уже прошло.*

```
@(
    Get-LabNextRun -Trigger (New-ScheduledTaskTrigger -Daily -At 02:30)
    Get-LabNextRun -Trigger (New-ScheduledTaskTrigger -Weekly -DaysOfWeek
    Monday,Wednesday,Friday -At 09:00)
    Get-LabNextRun -Trigger (New-ScheduledTaskTrigger -Weekly -DaysOfWeek Sunday
    -At 03:00)
) | Set-Content -Encoding UTF8 otvety\6
```

11.7. Ночная копия. *Подсказка ученику: Задание = действие (что запускать) + триггер (когда). Их создают отдельными командами и передают в `Register-ScheduledTask`.*

```
$a = New-ScheduledTaskAction -Execute 'powershell.exe' -Argument '-File
C:\scripts\backup-shop.ps1'
Register-ScheduledTask -TaskName BackupShop -Action $a -Trigger
(New-ScheduledTaskTrigger -Daily -At 02:30) | Out-Null
```

11.8. Ошибка в расписании. *Подсказка ученику: Часы — от 0 до 23, значение 25 недопустимо. «В 3 часа ночи» — `03:00`. После правки сценарий нужно запустить заново.*

```
(Get-Content register-all.ps1) -replace "25:00", "03:00" | Set-Content -Encoding
UTF8 register-all.ps1
& .\register-all.ps1
```

11.9. Рабочая смена. *Подсказка ученику: Дни недели перечисляются по-английски через запятую: `Monday,Tuesday,…`. Проверьте `Get-LabNextRun`.*

```
$a = New-ScheduledTaskAction -Execute 'powershell.exe' -Argument '-File
C:\scripts\check.ps1'
$t = New-ScheduledTaskTrigger -Weekly -DaysOfWeek
Monday,Tuesday,Wednesday,Thursday,Friday,Saturday -At 08:00
Register-ScheduledTask -TaskName Shift -Action $a -Trigger $t | Out-Null
```

11.10. Контрольная: частые слова. *Подсказка ученику: `param([int]$N = 3, [string]$Path)` задаёт значение по умолчанию. Частоты считает `Group-Object`.*

```
Set-Content -Encoding UTF8 srv\scripts\top.ps1 -Value @(
    'param([int]$N = 3, [string]$Path)',
    'if (-not $Path) { Write-Error "Использование: top.ps1 [-N число] -Path ФАЙЛ";
    exit 2 }',
    'Get-Content -LiteralPath $Path | Group-Object | Sort-Object Count -Descending
    | Select-Object -First $N | ForEach-Object { "$($_.Count) $($_.Name)" }')
```

Модуль 12. Сеть, SSH и межсетевой экран

Linux.

12.1. Кто слушает порт. *Подсказка ученику: Нужна строка с портом 8000–8999. Имя процесса указано в столбце Process: `users:(("имя",pid=…))`.*

```
ss -tlnp | grep -E ':8[0-9]{3} ' | grep -o '(("[^"]*"' | tr -d '("' > otvety/1
```

12.2. Что видно из сети. *Подсказка ученику: Адрес и порт стоят в четвёртом столбце как `адрес:порт`. Отберите строки с `0.0.0.0:`, возьмите порт после двоеточия.*

```
ss -tlnp | awk 'NR > 1 && $4 ~ /^0\.0\.0\.0:/ {split($4, a, ":"); print a[2]}' |
sort -n | paste -sd, - > otvety/2
```

12.3. Ключ для автоматизации. *Подсказка ученику: `ssh-keygen -t ed25519 -N '' -C комментарий -f путь`. Открытый ключ (`.pub`) дописывают в `authorized_keys` оператором `>>`, а не `>`.*

```
ssh-keygen -q -t ed25519 -N '' -C deploy@srv01 -f
srv/home/anna/.ssh/id_ed25519_deploy
cat srv/home/anna/.ssh/id_ed25519_deploy.pub >> srv/home/anna/.ssh/authorized_keys
```

12.4. Короткое имя для сервера. *Подсказка ученику: Ключ `-G` у `ssh` печатает итоговые параметры соединения, не подключаясь. Отступы в записи `Host` — для читаемости, они необязательны.*

```
printf '%s\n' 'Host web' '    HostName 10.0.0.5' '    User deploy' '    Port 2222'
'    IdentityFile ~/.ssh/id_ed25519_deploy' > srv/home/anna/.ssh/config
```

12.5. Закрутить гайки в sshd. *Подсказка ученику: Меняйте значения в неприкомментированных строках (`sed -i 's/…/…/'`). Прежде чем отключать пароли на настоящем сервере, убедитесь, что вход по ключу работает!*

```
sed -i 's/^PermitRootLogin yes/PermitRootLogin no/; s/^PasswordAuthentication
yes/PasswordAuthentication no/; s/^X11Forwarding yes/X11Forwarding no/'
srv/etc/ssh/sshd_config
```

12.6. Убрать лишние правила. *Подсказка ученику: Правило удаляют так же, как создавали, добавив `delete`: `ufw delete allow 23/tcp`.*

```
ufw delete allow 23/tcp; ufw delete allow 3306/tcp
```

12.7. Запрет по умолчанию. *Подсказка ученику: Сначала запрещаем всё, затем разрешаем нужное — принцип «минимальных открытых портов».*

```
ufw default deny incoming
```

12.8. Разрешить нужное и включить. *Подсказка ученику: Правило с источником: `ufw allow from ПОДСЕТЬ to any port 22 proto tcp`. Включайте экран последним, чтобы случайно не отрезать себя от сервера.*

```
ufw allow from 192.168.1.0/24 to any port 22 proto tcp; ufw allow 80/tcp; ufw
allow 443/tcp; ufw enable
```

12.9. Проверка правил. *Подсказка ученику: Вывод `fwtest` начинается со слова ALLOW или DENY. Для остальных случаев адрес источника по умолчанию — внешний.*

```
{ fwtest tcp 22 from 192.168.1.50; fwtest tcp 22 from 8.8.8.8; fwtest tcp 80;
fwtest tcp 3306; fwtest tcp 23; } | awk '{print $1}' > otvety/9
```

12.10. Контрольная: акт аудита. *Подсказка ученику: Все три значения можно взять командами из заданий 2, 5 и 8. Статус — второе слово первой строки `ufw status`.*

```
printf 'OPEN=%s\nFW=%s\nROOTLOGIN=%s\n' "$(cat otvety/2)" "$(ufw status | head -n
1 | awk '{print $2}')" "$(grep -v '^#' srv/etc/ssh/sshd_config | awk '$1 ==
"PermitRootLogin" {print $2; exit}')" > netaudit.txt
```

Windows.

12.1. Кто слушает порт. *Подсказка ученику: Отберите соединения по `LocalPort` в диапазоне и возьмите свойство `ProcessName`.*

```
(Get-NetTCPConnection -State Listen | Where-Object { $_.LocalPort -ge 8000 -and
$_.LocalPort -le 8999 }).ProcessName | Set-Content -Encoding UTF8 otvety\1
```

12.2. Что видно из сети. *Подсказка ученику: Отберите `LocalAddress -eq '0.0.0.0'`, отсортируйте по `LocalPort` и склейте оператором `-join ','`.*

```
(Get-NetTCPConnection -State Listen | Where-Object { $_.LocalAddress -eq '0.0.0.0'
} | Sort-Object LocalPort | ForEach-Object { $_.LocalPort }) -join ',' |
Set-Content -Encoding UTF8 otvety\2
```

12.3. Ключ для автоматизации. *Подсказка ученику: `ssh-keygen -t ed25519 -N '""' -C комментарий -f путь`. Открытый ключ (`.pub`) дописывают в `authorized_keys` командой `Add-Content`, а не `Set-Content`.*

```
$k = (Resolve-Path 'srv\Users\anna\.ssh').Path + '\id_ed25519_deploy'
ssh-keygen -q -t ed25519 -N '""' -C deploy@srv01 -f $k
Add-Content -Encoding ASCII srv\Users\anna\.ssh\authorized_keys -Value
(Get-Content "$k.pub" -TotalCount 1)
```

12.4. Короткое имя для сервера. *Подсказка ученику: Ключ `-G` у `ssh` печатает итоговые параметры соединения, не подключаясь. Файл лучше сохранить в кодировке ASCII/UTF-8 без BOM (`-Encoding ASCII`).*

```
Set-Content -Encoding ASCII srv\Users\anna\.ssh\config -Value @('Host web', '
HostName 10.0.0.5', '    User deploy', '    Port 2222', '    IdentityFile
~/.ssh/id_ed25519_deploy')
```

12.5. Закрутить гайки в sshd. *Подсказка ученику: Оператор `-replace` с регулярным выражением `^…` меняет только строки, начинающиеся с директивы. Прежде чем отключать пароли на настоящем сервере, убедитесь, что вход по ключу работает!*

```
(Get-Content srv\ProgramData\ssh\sshd_config) -replace '^PermitRootLogin yes',
'PermitRootLogin no' -replace '^PasswordAuthentication yes',
'PasswordAuthentication no' -replace '^X11Forwarding yes', 'X11Forwarding no' |
Set-Content -Encoding ASCII srv\ProgramData\ssh\sshd_config
```

12.6. Убрать лишние правила. *Подсказка ученику: `Remove-NetFirewallRule -DisplayName 'имя'`. Точные имена покажет `Get-NetFirewallRule | Format-Table DisplayName, LocalPort`.*

```
Remove-NetFirewallRule -DisplayName 'Telnet (23)'
Remove-NetFirewallRule -DisplayName 'MySQL (3306)'
```

12.7. Запрет по умолчанию. *Подсказка ученику: Сначала запрещаем всё, затем разрешаем нужное — принцип «минимальных открытых портов».*

```
Set-NetFirewallProfile -Profile Domain,Private,Public -Enabled True
-DefaultInboundAction Block
```

12.8. Разрешить нужное. *Подсказка ученику: Источник ограничивает параметр `-RemoteAddress` (адрес или подсеть в виде `192.168.1.0/24`). Проверьте `Test-LabFirewall`.*

```
New-NetFirewallRule -DisplayName 'SSH из локальной сети' -Direction Inbound
-Action Allow -Protocol TCP -LocalPort 22 -RemoteAddress 192.168.1.0/24 | Out-Null
New-NetFirewallRule -DisplayName 'HTTP' -Direction Inbound -Action Allow -Protocol
TCP -LocalPort 80 | Out-Null
New-NetFirewallRule -DisplayName 'HTTPS' -Direction Inbound -Action Allow
-Protocol TCP -LocalPort 443 | Out-Null
```

12.9. Проверка правил. *Подсказка ученику: Результат — объект со свойством `Verdict`. Для остальных случаев адрес источника по умолчанию — внешний.*

```
@(
    (Test-LabFirewall -LocalPort 22 -RemoteAddress 192.168.1.50).Verdict
    (Test-LabFirewall -LocalPort 22 -RemoteAddress 8.8.8.8).Verdict
    (Test-LabFirewall -LocalPort 80).Verdict
    (Test-LabFirewall -LocalPort 3306).Verdict
    (Test-LabFirewall -LocalPort 23).Verdict
) | Set-Content -Encoding UTF8 otvety\9
```

12.10. Контрольная: акт аудита. *Подсказка ученику: Все три значения можно взять командами из заданий 2, 5 и 7.*

```
$fw = (Get-NetFirewallProfile | Select-Object -First 1).Enabled
$rl = (Get-Content srv\ProgramData\ssh\sshd_config | Where-Object { $_ -notmatch
'^\s*#' -and $_ -match '^PermitRootLogin' } | Select-Object -First 1) -replace
'^\S+\s+', ''
@("OPEN=$(Get-Content otvety\2)", "FW=$fw", "ROOTLOGIN=$rl") | Set-Content
-Encoding UTF8 netaudit.txt
```

Модуль 13. Аудит безопасности

Linux.

13.1. Пароли в файлах. *Подсказка ученику: `grep -r` ищет рекурсивно, `-l` печатает только имена файлов, `-i` игнорирует регистр, `-E` включает расширенные выражения.*

```
grep -rliE '(password|passwd|secret|token)[[:space:]]*=[[:space:]]*[^[:space:]]+'
srv/home srv/opt | sort > otvety/1
```

13.2. Пароль в истории команд. *Подсказка ученику: Ищите `-p`, за которым сразу идут символы пароля. В `grep` такой шаблон начинается с дефиса: поставьте перед ним `--`.*

```
grep -l -- ' -p[A-Za-z]' srv/home/*/.bash_history | cut -d/ -f3 > otvety/2
```

13.3. Убрать секреты. *Подсказка ученику: `sed -i -E 's/(ключ = )значение/\\1CHANGEME/I' файл` — группа в скобках сохраняется, остаток заменяется. Сначала потренируйтесь без `-i`.*

```
for f in $(cat .snap/secrets); do sed -i -E
's/((password|passwd|secret|token)[[:space:]]*=[[:space:]]*)[^[:space:]]+/\1CHANGEME/I'
"$f"; done
```

13.4. Политика паролей. *Подсказка ученику: Меняйте только числа в строках с этими тремя именами. Остальные параметры (UMASK и т. д.) не трогайте.*

```
sed -i 's/^PASS_MAX_DAYS .*/PASS_MAX_DAYS   90/; s/^PASS_MIN_DAYS .*/PASS_MIN_DAYS
1/; s/^PASS_WARN_AGE .*/PASS_WARN_AGE   14/' srv/etc/login.defs
```

13.5. Чужое расписание. *Подсказка ученику: Программы из `/tmp` и скрытых каталогов (`/tmp/.x`) в расписании — повод для тревоги. `grep -l '/tmp/' файлы`.*

```
grep -l '/tmp/' srv/var/spool/cron/crontabs/* | xargs -n1 basename > otvety/5
```

13.6. Убрать задание. *Подсказка ученику: `sed -i '\|шаблон|d' файл` удаляет строки с шаблоном; в качестве разделителя здесь удобна не косая черта, а вертикальная.*

```
sed -i '\|/tmp/|d' "srv/var/spool/cron/crontabs/$(cat .snap/cronuser)"
```

13.7. Ключи уволенного. *Подсказка ученику: Комментарий в конце строки ключа — подпись владельца. `sed -i '/шаблон/d' файлы` удаляет строки с шаблоном сразу в нескольких файлах.*

```
sed -i '/oleg@/d' srv/home/*/.ssh/authorized_keys
```

13.8. Лишние службы. *Подсказка ученику: `systemctl disable --now имя1 имя2 …` умеет работать с несколькими службами сразу.*

```
systemctl disable --now telnet-server rsh-server cups
```

13.9. Служебные учётные записи с оболочкой. *Подсказка ученику: В `passwd` UID — третье поле, оболочка — седьмое. Условие в `awk`: `$3 > 0 && $3 < 1000 && $7 ~ /sh$/`.*

```
awk -F: '$3 > 0 && $3 < 1000 && $7 ~ /sh$/ {print $1}' srv/etc/passwd | sort >
otvety/9
```

13.10. Контрольная: исправить и составить акт. *Подсказка ученику: Цикл по именам из `otvety/9`. Числа в акте посчитайте по строкам файлов ответов (`grep -c ''`).*

```
for u in $(cat otvety/9); do usermod -s /usr/sbin/nologin "$u"; done
printf 'SECRETS=%s\nCRON=%s\nSHELLS=%s\n' "$(grep -c '' otvety/1)" "$(cat
otvety/5)" "$(grep -c '' otvety/9)" > audit.txt
```

Windows.

13.1. Пароли в файлах. *Подсказка ученику: `Get-ChildItem -Recurse -File | Select-String -Pattern 'регулярное_выражение'` ищет по содержимому (регистр по умолчанию не важен). У результата есть свойство `Path`.*

```
Get-ChildItem 'srv\Users', 'srv\Program Files' -Recurse -File | Select-String
-Pattern '(password|passwd|secret|token)\s*=\s*\S+' | ForEach-Object { Split-Path
$_.Path -Leaf } | Sort-Object -Unique | Set-Content -Encoding UTF8 otvety\1
```

13.2. Пароль в истории команд. *Подсказка ученику: Ищите `-p`, за которым сразу идут символы пароля. Имя пользователя — компонент пути сразу после `Users`.*

```
Get-ChildItem srv\Users -Recurse -Filter ConsoleHost_history.txt | Select-String
-Pattern ' -p[A-Za-z]' | ForEach-Object { $_.Path -replace
'^.*\\Users\\([^\\]+)\\.*$', '$1' } | Select-Object -First 1 | Set-Content
-Encoding UTF8 otvety\2
```

13.3. Убрать секреты. *Подсказка ученику: Оператор `-replace` с группой в скобках: `'(ключ\s*=\s*)\S+'` → `'$1CHANGEME'` сохраняет левую часть и заменяет значение.*

```
Get-ChildItem 'srv\Users', 'srv\Program Files' -Recurse -File | Select-String
-Pattern '(password|passwd|secret|token)\s*=\s*\S+' | ForEach-Object { $_.Path } |
Sort-Object -Unique | ForEach-Object {
    (Get-Content $_) -replace '((password|passwd|secret|token)\s*=\s*)\S+',
    '$1CHANGEME' | Set-Content -Encoding UTF8 $_
}
```

13.4. Политика паролей. *Подсказка ученику: Меняйте значения только в строках с этими четырьмя именами; цепочка операторов `-replace` справится одной командой.*

```
(Get-Content srv\etc\secpol.ini) -replace '^MaximumPasswordAge=.*',
'MaximumPasswordAge=90' -replace '^MinimumPasswordAge=.*', 'MinimumPasswordAge=1'
-replace '^MinimumPasswordLength=.*', 'MinimumPasswordLength=12' -replace
'^PasswordComplexity=.*', 'PasswordComplexity=1' | Set-Content -Encoding UTF8
srv\etc\secpol.ini
```

13.5. Чужое задание планировщика. *Подсказка ученику: Действие задания — `$_.Actions[0].Execute`. Подозрительны пути во временных и скрытых каталогах.*

```
Get-ScheduledTask | Where-Object { $_.Actions[0].Execute -like '*\Temp\*' } |
ForEach-Object { $_.TaskName } | Set-Content -Encoding UTF8 otvety\5
```

13.6. Убрать задание. *Подсказка ученику: Имя берите из своего ответа к заданию 5. `-Confirm:$false` отключает вопрос о подтверждении.*

```
Unregister-ScheduledTask -TaskName (Get-Content otvety\5) -Confirm:$false
```

13.7. Ключи уволенного. *Подсказка ученику: Комментарий в конце строки ключа — подпись владельца. Перезапишите каждый файл строками без `oleg@`.*

```
Get-ChildItem srv\Users -Recurse -Filter authorized_keys | ForEach-Object {
(Get-Content $_.FullName | Where-Object { $_ -notmatch 'oleg@' }) | Set-Content
-Encoding ASCII $_.FullName }
```

13.8. Лишние службы. *Подсказка ученику: Цикл по трём именам, в нём два действия: остановка и запрет запуска.*

```
foreach ($n in 'telnet-server', 'rsh-server', 'cups') { Stop-Service $n;
Set-Service $n -StartupType Disabled }
```

13.9. Служебные записи с удалённым входом. *Подсказка ученику: Перечислите членов группы, для каждого проверьте `(Get-LocalUser -Name имя).PasswordNeverExpires`.*

```
Get-LocalGroupMember -Group 'Remote Desktop Users' | ForEach-Object { ($_.Name
-split '\\')[-1] } | Where-Object { (Get-LocalUser -Name $_).PasswordNeverExpires }
| Sort-Object | Set-Content -Encoding UTF8 otvety\9
```

13.10. Контрольная: исправить и составить акт. *Подсказка ученику: Цикл по именам из `otvety\9`. Числа в акте посчитайте по строкам файлов ответов.*

```
foreach ($u in (Get-Content otvety\9)) { Remove-LocalGroupMember -Group 'Remote
Desktop Users' -Member $u }
@("SECRETS=$(@(Get-Content otvety\1).Count)", "TASK=$(Get-Content otvety\5)",
"RDP=$(@(Get-Content otvety\9).Count)") | Set-Content -Encoding UTF8 audit.txt
```

Модуль 14. Итоговый инцидент: «сервер тормозит»

Linux.

14.1. Подозрительный процесс. *Подсказка ученику: `ps -ef`, затем отберите строки с путём, содержащим `srv/tmp`. Имя — последняя часть пути.*

```
ps -ef | grep -o '[^ ]*srv/tmp/[^ ]*' | head -n 1 | xargs basename > otvety/1
```

14.2. Остановить и удалить. *Подсказка ученику: `kill PID`, если процесс не исчез — `kill -9 PID`. Только потом `rm` файла.*

```
p=$(ps -ef | grep '[s]rv/tmp/' | awk '{print $2}' | head -n 1); kill -9
"${p:-$(cat .snap/pids)}"; sleep 1; rm -f "srv/tmp/.cache/$(cat otvety/1)"
```

14.3. Куда делось место. *Подсказка ученику: `du -sk srv/var/* | sort -rn | head -n 1`; имя — последняя часть пути.*

```
du -sk srv/var/* | sort -rn | head -n 1 | awk '{print $2}' | xargs basename >
otvety/3
```

14.4. Освободить место. *Подсказка ученику: Сначала выведите кандидатов на удаление командой `find` без `-delete`, затем повторите с `-delete`.*

```
rm -f srv/var/log/app/debug.log; find srv/var/log/app -name '*.gz' -mtime +30
-delete
```

14.5. Почему упал магазин. *Подсказка ученику: В сообщении `Failed to load environment files` указан путь. В журнале есть ещё и ошибка про место на диске — она уже решена в задании 4.*

```
journalctl -u shop-api --no-pager | grep -o '/etc/[a-z.]*env' | head -n 1 >
otvety/5
```

14.6. Запустить магазин. *Подсказка ученику: Файл создаётся внутри `srv` (путь `/etc/shop.env` в unit-файле соответствует `srv/etc/shop.env`).*

```
echo 'SHOP_ENV=prod' > srv/etc/shop.env; systemctl start shop-api
```

14.7. Кто ломится. *Подсказка ученику: Адрес — одиннадцатое поле строки с `Failed password`. Знакомая цепочка `sort | uniq -c | sort -rn`.*

```
grep 'Failed password' srv/var/log/auth.log | awk '{print $11}' | sort | uniq -c |
sort -rn | head -n 1 | awk '{print $2}' > otvety/7
```

14.8. Заблокировать атакующего. *Подсказка ученику: Правила ufw проверяются сверху вниз, срабатывает первое подходящее. Простой `ufw deny from …` добавил бы правило в конец, после `allow 80`.*

```
ufw insert 1 deny from "$(cat otvety/7)"
```

14.9. Вернуть потерянный файл. *Подсказка ученику: Посмотрите состав архива `tar tzf …`, распакуйте в `srv` только один путь: `tar xzf архив -C куда путь`.*

```
tar xzf backup/nightly.tar.gz -C srv etc/shop.conf
```

14.10. Контрольная: отчёт директору. *Подсказка ученику: Значения лежат в ваших файлах ответов, а состояние службы выдаёт `systemctl is-active shop-api`.*

```
printf 'HOG=%s\nDISK=%s\nATTACKER=%s\nSERVICE=%s\n' "$(cat otvety/1)" "$(cat
otvety/3)" "$(cat otvety/7)" "$(systemctl is-active shop-api)" > incident.txt
```

Windows.

14.1. Подозрительный процесс. *Подсказка ученику: `Get-Process | Where-Object { $_.Path -like '*шаблон*' }`. У некоторых системных процессов `Path` недоступен — это нормально.*

```
(Get-Process | Where-Object { $_.Path -like '*\srv\Temp\.cache\*' } |
Select-Object -First 1).Name | Set-Content -Encoding UTF8 otvety\1
```

14.2. Остановить и удалить. *Подсказка ученику: Сначала `Stop-Process`, подождите секунду, затем `Remove-Item`.*

```
$r = Get-Content otvety\1
Stop-Process -Name $r -Force
Start-Sleep -Seconds 1
Remove-Item "srv\Temp\.cache\$r.exe" -Force
```

14.3. Куда делось место. *Подсказка ученику: Для каждого подкаталога посчитайте сумму `Length` его файлов и отсортируйте по убыванию.*

```
Get-ChildItem srv\var -Directory | ForEach-Object { [pscustomobject]@{ Name =
$_.Name; Size = (Get-ChildItem $_.FullName -Recurse -File | Measure-Object Length
-Sum).Sum } } | Sort-Object Size -Descending | Select-Object -First 1 |
ForEach-Object { $_.Name } | Set-Content -Encoding UTF8 otvety\3
```

14.4. Освободить место. *Подсказка ученику: Сначала выведите кандидатов на удаление без `Remove-Item`, затем повторите с ним.*

```
Remove-Item srv\var\log\app\debug.log
Get-ChildItem srv\var\log\app -Filter *.zip | Where-Object { $_.LastWriteTime -lt
(Get-Date).AddDays(-30) } | Remove-Item
```

14.5. Почему не запускается магазин. *Подсказка ученику: Имя учётной записи указано в тексте сообщения (`Message`) после `SRV01\`. В журнале Application есть ещё ошибка про место на диске — она решена заданием 4.*

```
Get-WinEvent -FilterHashtable @{ LogName = 'System'; Level = 2 } | Where-Object {
$_.Message -like '*shop-api*' } | ForEach-Object { if ($_.Message -match
'SRV01\\([\w-]+)') { $Matches[1] } } | Select-Object -First 1 | Set-Content
-Encoding UTF8 otvety\5
```

14.6. Запустить магазин. *Подсказка ученику: Имя учётной записи возьмите из своего ответа к заданию 5. Пароль — `ConvertTo-SecureString … -AsPlainText -Force`.*

```
New-LocalUser -Name (Get-Content otvety\5) -Password (ConvertTo-SecureString
'Svc-Passw0rd!' -AsPlainText -Force) | Out-Null
Start-Service shop-api
```

14.7. Кто ломится. *Подсказка ученику: Адрес достаньте регулярным выражением, затем `Group-Object | Sort-Object Count -Descending`.*

```
Get-WinEvent -FilterHashtable @{ LogName = 'Security'; Id = 4625 } |
ForEach-Object { if ($_.Message -match 'адрес источника: (\S+)') { $Matches[1] } }
| Group-Object | Sort-Object Count -Descending | Select-Object -First 1 |
ForEach-Object { $_.Name } | Set-Content -Encoding UTF8 otvety\7
```

14.8. Заблокировать атакующего. *Подсказка ученику: Если `-LocalPort` не указан, правило действует на все порты. Проверьте `Test-LabFirewall` для 80-го и 22-го портов.*

```
New-NetFirewallRule -DisplayName 'Блок атакующего' -Direction Inbound -Action
Block -Protocol TCP -RemoteAddress (Get-Content otvety\7) | Out-Null
```

14.9. Вернуть потерянный файл. *Подсказка ученику: Распаковка во временный каталог (`$env:TEMP`) не затрагивает рабочие данные; затем `Copy-Item` одного файла.*

```
$t = Join-Path $env:TEMP ('nightly-' + [guid]::NewGuid())
Expand-Archive -Path backup\nightly.zip -DestinationPath $t
Copy-Item (Join-Path $t 'etc\shop.conf') srv\etc\shop.conf -Force
Remove-Item $t -Recurse -Force
```

14.10. Контрольная: отчёт директору. *Подсказка ученику: Значения лежат в ваших файлах ответов, а состояние службы выдаёт `(Get-Service shop-api).Status`.*

```
@("HOG=$(Get-Content otvety\1)", "DISK=$(Get-Content otvety\3)",
"ATTACKER=$(Get-Content otvety\7)", "SERVICE=$((Get-Service shop-api).Status)") |
Set-Content -Encoding UTF8 incident.txt
```

Модуль 15. Сетевые настройки хоста

Linux.

15.1. Адрес сервера. *Подсказка ученику: В краткой форме `ip -br addr` третий столбец содержит адрес с длиной маски. `/24` означает маску 255.255.255.0.*

```
ip -br addr | awk '$1 == "eth0" {print $3}' > otvety/1
```

15.2. Шлюз по умолчанию. *Подсказка ученику: Маршрут по умолчанию применяется ко всем назначениям, для которых нет более точного маршрута. Адрес шлюза стоит после слова `via`.*

```
ip route | awk '$1 == "default" {print $3}' > otvety/2
```

15.3. Серверы DNS. *Подсказка ученику: `awk '$1 == "nameserver" {print $2}' файл` печатает второй столбец нужных строк. Сервера опрашиваются в порядке перечисления.*

```
awk '$1 == "nameserver" {print $2}' srv/etc/resolv.conf | sort > otvety/3
```

15.4. Сколько маршрутов. *Подсказка ученику: Каждый маршрут — отдельная строка вывода; число строк даёт `wc -l`.*

```
ip route | wc -l > otvety/4
```

15.5. Непосредственно подключённая сеть. *Подсказка ученику: Маршрут `proto kernel` создаётся ядром автоматически при назначении адреса интерфейсу; его первое поле — адрес сети.*

```
ip route | awk '/dev eth0 proto kernel/ {print $1}' > otvety/5
```

15.6. Имя для файлового сервера. *Подсказка ученику: Формат строки `hosts`: адрес, затем имена через пробел. Добавляйте строку оператором `>>`, чтобы не затереть остальные.*

```
echo "10.20.30.50 files.sever.local files" >> srv/etc/hosts
```

15.7. Статический адрес в netplan. *Подсказка ученику: В YAML отступы значимы. Адрес `addresses: [адрес/маска]`, маршрут `routes:` со списком из `to: default` и `via: шлюз`, DNS в `nameservers: addresses: [...]`. После правки на настоящем сервере выполняют `netplan try`.*

```
printf '%s\n' 'network:' '  version: 2' '  ethernets:' '    eth0:' '      dhcp4:
false' "      addresses: [$(cat .snap/ip)]" '      routes:' '        - to:
default' "          via: $(cat .snap/gw)" '      nameservers:' "        addresses:
[$(head -n 1 .snap/dns), $(tail -n 1 .snap/dns)]" > srv/etc/netplan/01-net.yaml
```

15.8. Имя сервера. *Подсказка ученику: Имя хранится в `hostname`, а локальное разрешение имени — в `hosts`; если они расходятся, служба при запуске может «зависнуть» на поиске собственного имени.*

```
echo web01 > srv/etc/hostname; sed -i 's/srv01/web01/' srv/etc/hosts
```

15.9. Интерфейс с потерями. *Подсказка ученику: Столбцы после имени интерфейса: bytes, packets, errs, drop. В `awk` с разделителем по умолчанию имя — `$1`, а `drop` приёма — `$5` (из-за двоеточия имени нужно `gsub`).*

```
awk 'NR > 2 {gsub(":", "", $1); print $5, $1}' srv/proc/net/dev | sort -rn | head
-n 1 | awk '{print $2}' > otvety/9
```

15.10. Контрольная: сетевой паспорт хоста. *Подсказка ученику: Значения лежат в ваших файлах ответов 1–3 и в `hostname`; склейте строки в нужном виде командой `printf`.*

```
printf 'IP=%s\nGW=%s\nDNS=%s\nHOST=%s\n' "$(cat otvety/1)" "$(cat otvety/2)"
"$(paste -sd, otvety/3)" "$(cat srv/etc/hostname)" > netreport.txt
```

Windows.

15.1. Адрес сервера. *Подсказка ученику: У объекта адреса есть свойства `IPAddress` и `PrefixLength`; склейте их через `/`. `/24` соответствует маске `255.255.255.0`.*

```
$a = Get-NetIPAddress -InterfaceAlias Ethernet
"$($a.IPAddress)/$($a.PrefixLength)" | Set-Content -Encoding UTF8 otvety\1
```

15.2. Шлюз по умолчанию. *Подсказка ученику: Маршрут `0.0.0.0/0` соответствует любому назначению, для которого нет более точного маршрута.*

```
(Get-NetRoute -DestinationPrefix 0.0.0.0/0).NextHop | Set-Content -Encoding UTF8
otvety\2
```

15.3. Серверы DNS. *Подсказка ученику: Свойство `ServerAddresses` — массив строк; `Sort-Object` упорядочит его.*

```
(Get-DnsClientServerAddress -InterfaceAlias Ethernet).ServerAddresses |
Sort-Object | Set-Content -Encoding UTF8 otvety\3
```

15.4. Сколько маршрутов. *Подсказка ученику: Заключите вывод в `@( … )`, чтобы `.Count` работал при любом числе строк.*

```
@(Get-NetRoute).Count | Set-Content -Encoding UTF8 otvety\4
```

15.5. Имя для файлового сервера. *Подсказка ученику: Файл `hosts` — текст: адрес и имена через пробел. Дописывайте строку командой `Add-Content`, не перезаписывайте файл.*

```
Add-Content -Encoding ASCII 'srv\Windows\System32\drivers\etc\hosts' -Value
'10.20.30.50 files.sever.local files'
```

15.6. Другие DNS-серверы. *Подсказка ученику: Адреса перечисляются через запятую; порядок задаёт порядок опроса серверов.*

```
Set-DnsClientServerAddress -InterfaceAlias Ethernet -ServerAddresses 10.20.30.2,
10.20.30.3
```

15.7. Маршрут в удалённую сеть. *Подсказка ученику: Адрес шлюза возьмите из своего ответа к заданию 2. Проверка: `Get-NetRoute`.*

```
New-NetRoute -DestinationPrefix 172.16.0.0/16 -NextHop (Get-Content otvety\2)
-InterfaceAlias Ethernet
```

15.8. Контрольная: сетевой паспорт хоста. *Подсказка ученику: Значения лежат в ваших файлах ответов 1–2, а число маршрутов и DNS-серверы нужно получить командами заново: после заданий 6–7 они изменились.*

```
$dns = (Get-DnsClientServerAddress -InterfaceAlias Ethernet).ServerAddresses |
Sort-Object
@("IP=$(Get-Content otvety\1)", "GW=$(Get-Content otvety\2)", "DNS=$($dns -join
',')", "ROUTES=$(@(Get-NetRoute).Count)") | Set-Content -Encoding UTF8
netreport.txt
```

Модуль 16. Хранилище: разделы, LVM, монтирование, swap

Linux.

16.1. Размер диска sdb. *Подсказка ученику: Размер — четвёртый столбец строки с именем диска. Строки разделов и томов LVM начинаются с рисунков дерева, имя диска — без них.*

```
lsblk | awk '$1 == "sdb" {print $4}' > otvety/1
```

16.2. Свободный диск. *Подсказка ученику: Разделы показаны под диском с отступом (├─, └─). Диск без вложенных строк свободен.*

```
lsblk | awk 'function f() { if (d != "" && p == 0) print d } /^sd/ { f(); d = $1;
p = 0; next } /^[^ ]*[├└]/ { p = 1 } END { f() }' > otvety/2
```

16.3. Свободное место в группе томов. *Подсказка ученику: В выводе `vgs` у группы томов столбцы: имя, число физических томов, число логических, ..., VSize (всего), VFree (свободно).*

```
vgs | awk '$1 == "vg0" {print $7}' > otvety/3
```

16.4. Крупнейший логический том. *Подсказка ученику: Размер — столбец `LSize`; отсортируйте строки по нему как по числу (`sort -rn`) и возьмите первую.*

```
lvs | awk 'NR > 1 {gsub("g", "", $4); print $4, $1}' | sort -rn | head -n 1 | awk
'{print $2}' > otvety/4
```

16.5. Тип файловой системы /var. *Подсказка ученику: В `mounts` поля: устройство, точка монтирования, тип, параметры, два нуля. Нужна строка, где второе поле равно `/var`.*

```
awk '$2 == "/var" {print $3}' srv/proc/mounts > otvety/5
```

16.6. Надёжное имя диска в fstab. *Подсказка ученику: `blkid` печатает UUID разделов. UUID не зависит от порядка подключения дисков, поэтому в `fstab` предпочтительнее имени `/dev/sdX`.*

```
u=$(blkid | grep '^/dev/sdc1' | awk -F'"' '{print $2}'); sed -i "s|^/dev/sdc1
|UUID=$u |" srv/etc/fstab
```

16.7. Опечатка в параметрах. *Подсказка ученику: Правильное значение — `defaults`. Ищите слово между пробелами, чтобы не задеть другие строки.*

```
sed -i 's/ default / defaults /' srv/etc/fstab
```

16.8. Файл подкачки. *Подсказка ученику: В `swapon --show` размер — третий столбец. Строку в `fstab` добавляйте оператором `>>`.*

```
swapon --show | awk 'NR > 1 {print $3}' > otvety/8
echo '/swapfile none swap sw 0 0' >> srv/etc/fstab
```

16.9. Сколько останется. *Подсказка ученику: Возьмите целую часть свободного места из задания 3 и вычтите 4: `$(( … ))` считает в оболочке.*

```
echo $(( $(cut -d. -f1 otvety/3) - 4 )) > otvety/9
```

16.10. Контрольная: паспорт хранилища. *Подсказка ученику: Все значения есть в ваших файлах ответов 2, 3 и 5.*

```
printf 'FREEDISK=%s\nVG=vg0\nVFREE=%s\nFSVAR=%s\n' "$(cat otvety/2)" "$(cat
otvety/3)" "$(cat otvety/5)" > storage.txt
```

Windows.

16.1. Сколько дисков. *Подсказка ученику: `@(Get-Disk).Count`*

```
@(Get-Disk).Count | Set-Content -Encoding UTF8 otvety\1
```

16.2. Новый диск. *Подсказка ученику: Новый диск без разметки имеет `PartitionStyle` равный `RAW`.*

```
(Get-Disk | Where-Object { $_.PartitionStyle -eq 'RAW' }).Number | Set-Content
-Encoding UTF8 otvety\2
```

16.3. Инициализировать. *Подсказка ученику: Номер диска возьмите из своего ответа к заданию 2. GPT — современный стиль таблицы разделов (диски больше 2 ТБ, надёжнее MBR).*

```
Initialize-Disk -Number (Get-Content otvety\2) -PartitionStyle GPT
```

16.4. Создать раздел. *Подсказка ученику: Если команда сообщает, что диск не инициализирован, вернитесь к заданию 3.*

```
New-Partition -DiskNumber (Get-Content otvety\2) -UseMaximumSize -DriveLetter E
```

16.5. Форматировать том. *Подсказка ученику: Форматирование уничтожает данные раздела: на настоящем сервере проверяйте букву диска дважды.*

```
Format-Volume -DriveLetter E -FileSystem NTFS -NewFileSystemLabel Data2
```

16.6. Файловая система тома D. *Подсказка ученику: `(Get-Volume -DriveLetter D).FileSystem`. ReFS — файловая система Microsoft для серверов с проверкой целостности данных.*

```
(Get-Volume -DriveLetter D).FileSystem | Set-Content -Encoding UTF8 otvety\6
```

16.7. Сколько свободно на D. *Подсказка ученику: Разделите свободное место на размер, умножьте на 100 и округлите `[math]::Round(…)`.*

```
$v = Get-Volume -DriveLetter D
[math]::Round($v.SizeRemaining / $v.Size * 100) | Set-Content -Encoding UTF8
otvety\7
```

16.8. Контрольная: паспорт хранилища. *Подсказка ученику: Количества получите командами `Get-Disk` и `Get-Volume`, остальное — из ваших ответов.*

```
@("DISKS=$(@(Get-Disk).Count)", "NEWDISK=$(Get-Content otvety\2)",
"VOLUMES=$(@(Get-Volume).Count)", "FREE_D=$(Get-Content otvety\7)") | Set-Content
-Encoding UTF8 storage.txt
```

Модуль 17. Мониторинг и производительность

Linux.

17.1. Кто занимает процессор. *Подсказка ученику: Строки процессов идут после пустой строки; `%CPU` — девятый столбец, команда — двенадцатый.*

```
top -b -n 1 | awk 'NR > 6 {print $9, $12}' | sort -rn | head -n 1 | awk '{print
$2}' > otvety/1
```

17.2. Кто занимает память. *Подсказка ученику: Тот же приём, что в задании 1, но по другому столбцу.*

```
top -b -n 1 | awk 'NR > 6 {print $10, $12}' | sort -rn | head -n 1 | awk '{print
$2}' > otvety/2
```

17.3. Доля занятой памяти. *Подсказка ученику: В строке `Mem:` второе поле — всего, третье — занято. `awk` умеет считать: `$3 * 100 / $2`; прибавьте 0,5 для округления.*

```
free -m | awk '/^Mem:/ {printf "%d\n", $3 * 100 / $2 + 0.5}' > otvety/3
```

17.4. Нагрузка за 15 минут. *Подсказка ученику: Три первых числа файла — нагрузка за 1, 5 и 15 минут.*

```
awk '{print $3}' srv/proc/loadavg > otvety/4
```

17.5. Ожидание диска. *Подсказка ученику: Шестнадцатый столбец — `wa`. Большие значения означают, что процессор простаивает в ожидании диска.*

```
vmstat | awk 'NR > 2 {print $16}' | sort -rn | head -n 1 > otvety/5
```

17.6. Загруженный диск. *Подсказка ученику: Значение около 100 % означает, что диск упирается в предел своей производительности.*

```
iostat | awk 'NR > 1 {print $7, $1}' | sort -rn | head -n 1 | awk '{print $2}' >
otvety/6
```

17.7. Проверка нагрузки. *Подсказка ученику: Сравнение дробного числа с целым в bash делают через awk: `awk -v l="$load" -v c="$cores" 'BEGIN { exit !(l < c) }'`. Число ядер — строки `processor` в `cpuinfo`.*

```
  cat > check-load.sh <<'EOT'
#!/bin/bash
d="$1"
cores=$(grep -c '^processor' "$d/proc/cpuinfo")
load=$(awk '{print $1}' "$d/proc/loadavg")
if awk -v l="$load" -v c="$cores" 'BEGIN { exit !(l < c) }'; then echo OK; exit 0;
else echo ALERT; exit 1; fi
EOT
```

17.8. Процессы-нарушители. *Подсказка ученику: Отберите строки процессов условием `$9 > 50`, выведите имя и значение, отсортируйте по второму столбцу как по числу.*

```
top -b -n 1 | awk 'NR > 6 && $9 > 50 {print $12, $9}' | sort -k2 -rn > alert.txt
```

17.9. Заполненные файловые системы. *Подсказка ученику: Процент занятости — пятый столбец; знак `%` убирают `gsub`, чтобы сравнивать как число.*

```
df -h | awk 'NR > 1 {gsub("%", "", $5); if ($5 >= 85) print $6}' | sort > otvety/9
```

17.10. Контрольная: сводка для дежурного. *Подсказка ученику: Все значения лежат в ваших файлах ответов 1–4.*

```
printf 'TOPCPU=%s\nTOPMEM=%s\nMEMPCT=%s\nLOAD15=%s\n' "$(cat otvety/1)" "$(cat
otvety/2)" "$(cat otvety/3)" "$(cat otvety/4)" > monitor.txt
```

Windows.

17.1. Кто занимает процессор. *Подсказка ученику: `Import-Csv` читает таблицу; сортируйте по числу, а не по тексту: `Sort-Object { [double]$_.CPU } -Descending`.*

```
(Import-Csv 'srv\var\lib\lab\procs.csv' | Sort-Object { [double]$_.CPU }
-Descending | Select-Object -First 1).Name | Set-Content -Encoding UTF8 otvety\1
```

17.2. Кто занимает память. *Подсказка ученику: Тот же приём, другой столбец.*

```
(Import-Csv 'srv\var\lib\lab\procs.csv' | Sort-Object { [double]$_.WorkingSetMB }
-Descending | Select-Object -First 1).Name | Set-Content -Encoding UTF8 otvety\2
```

17.3. Суммарная память. *Подсказка ученику: `Measure-Object -Sum` суммирует числа; значения из CSV — строки, преобразуйте в `[double]`.*

```
(Import-Csv 'srv\var\lib\lab\procs.csv' | ForEach-Object { [double]$_.WorkingSetMB
} | Measure-Object -Sum).Sum | Set-Content -Encoding UTF8 otvety\3
```

17.4. Сколько экземпляров svchost. *Подсказка ученику: Отфильтруйте `Where-Object` и посчитайте `@( … ).Count`.*

```
@(Import-Csv 'srv\var\lib\lab\procs.csv' | Where-Object { $_.Name -eq 'svchost'
}).Count | Set-Content -Encoding UTF8 otvety\4
```

17.5. Сколько «горячих» процессов. *Подсказка ученику: Условие `[double]$_.CPU -gt 50`; без преобразования сравнение шло бы как со строкой.*

```
@(Import-Csv 'srv\var\lib\lab\procs.csv' | Where-Object { [double]$_.CPU -gt 50
}).Count | Set-Content -Encoding UTF8 otvety\5
```

17.6. Процессы-нарушители. *Подсказка ученику: Фильтр, сортировка по убыванию, затем форматирование строки `"$($_.Name) $($_.CPU)"`.*

```
Import-Csv 'srv\var\lib\lab\procs.csv' | Where-Object { [double]$_.CPU -gt 50 } |
Sort-Object { [double]$_.CPU } -Descending | ForEach-Object { "$($_.Name)
$($_.CPU)" } | Set-Content -Encoding UTF8 alert.txt
```

17.7. Тома с малым запасом. *Подсказка ученику: Доля свободного места — `SizeRemaining / Size`; сравнивайте с `0.15`.*

```
Get-Volume | Where-Object { $_.SizeRemaining / $_.Size -lt 0.15 } | Sort-Object
DriveLetter | ForEach-Object { $_.DriveLetter } | Set-Content -Encoding UTF8
otvety\7
```

17.8. Контрольная: сводка для дежурного. *Подсказка ученику: Все значения лежат в ваших файлах ответов 1, 2, 3 и 5.*

```
@("TOPCPU=$(Get-Content otvety\1)", "TOPMEM=$(Get-Content otvety\2)",
"MEMTOTAL=$(Get-Content otvety\3)", "HOT=$(Get-Content otvety\5)") | Set-Content
-Encoding UTF8 monitor.txt
```

Модуль 18. Хеши, шифрование и сертификаты

Linux.

18.1. Хеш файла. *Подсказка ученику: `sha256sum` печатает хеш и имя файла; нужно только первое поле (`awk '{print $1}'`).*

```
sha256sum srv/var/www/files/f1.dat | awk '{print $1}' > otvety/1
```

18.2. Какой файл повреждён. *Подсказка ученику: Список хранит относительные имена, поэтому `sha256sum -c` нужно запускать внутри каталога. Нужные строки заканчиваются на `FAILED`.*

```
(cd srv/var/www/files && sha256sum -c SHA256SUMS 2>/dev/null | grep FAILED | cut
-d: -f1) > otvety/2
```

18.3. Раскодировать секрет. *Подсказка ученику: `base64` — это кодирование, а не шифрование: любой может раскодировать. Хранить так пароли нельзя.*

```
base64 -d srv/var/lib/lab/secret.b64 > otvety/3
```

18.4. Имя сертификата. *Подсказка ученику: Разделите строку по `CN=`, затем отрежьте всё после запятой.*

```
awk -F'CN=' '/Subject:/ {split($2, a, ","); print a[1]}' srv/etc/ssl/cert.txt >
otvety/4
```

18.5. Срок действия. *Подсказка ученику: Дата — последнее слово строки (`$NF` в `awk`).*

```
awk '/Not After/ {print $NF}' srv/etc/ssl/cert.txt > otvety/5
```

18.6. Сколько имён. *Подсказка ученику: `grep -o` печатает каждое совпадение отдельной строкой, их остаётся посчитать (`wc -l`).*

```
grep -o 'DNS:' srv/etc/ssl/cert.txt | wc -l | tr -d ' ' > otvety/6
```

18.7. Сценарий контрольных сумм. *Подсказка ученику: Старый `SHA256SUMS` удалите до вычисления, иначе он попадёт в список. Звёздочка раскрывается до создания выходного файла.*

```
  cat > mkhash.sh <<'EOT'
#!/bin/bash
cd "$1" || exit 1
rm -f SHA256SUMS
sha256sum -- * > SHA256SUMS
EOT
```

18.8. Контрольная: акт проверки. *Подсказка ученику: Все значения лежат в ваших ответах 2–5.*

```
printf 'BAD=%s\nSECRET=%s\nCN=%s\nEXPIRES=%s\n' "$(cat otvety/2)" "$(cat
otvety/3)" "$(cat otvety/4)" "$(cat otvety/5)" > crypto.txt
```

Windows.

18.1. Хеш файла. *Подсказка ученику: У результата есть свойство `Hash`; запишите именно его. Регистр букв оставьте как выдал `Get-FileHash`.*

```
(Get-FileHash -Algorithm SHA256 'srv\var\www\files\f1.dat').Hash | Set-Content
-Encoding UTF8 otvety\1
```

18.2. Какой файл повреждён. *Подсказка ученику: Разделите строку списка на хеш и имя (`-split`); `-ne` в PowerShell не различает регистр, так что сравнение безопасно.*

```
Get-Content 'srv\var\www\files\SHA256SUMS' | ForEach-Object {
    $h, $n = $_ -split '\s+', 2
    if ((Get-FileHash -Algorithm SHA256 "srv\var\www\files\$n").Hash -ne $h) { $n
    }
} | Set-Content -Encoding UTF8 otvety\2
```

18.3. Раскодировать секрет. *Подсказка ученику: `base64` — кодирование, а не шифрование: раскодировать может любой.*

```
[Text.Encoding]::UTF8.GetString([Convert]::FromBase64String((Get-Content
'srv\var\lib\lab\secret.b64').Trim())) | Set-Content -Encoding UTF8 otvety\3
```

18.4. Имя сертификата. *Подсказка ученику: Регулярное выражение `CN=([^,]+)` захватывает всё до запятой.*

```
$l = Get-Content 'srv\etc\ssl\cert.txt' | Where-Object { $_ -match 'Subject:' }
([regex]::Match($l, 'CN=([^,]+)')).Groups[1].Value | Set-Content -Encoding UTF8
otvety\4
```

18.5. Срок действия. *Подсказка ученику: Дата — последнее слово строки; индекс `[-1]` берёт последний элемент массива.*

```
$l = Get-Content 'srv\etc\ssl\cert.txt' | Where-Object { $_ -match 'Not After' }
($l -split '\s+')[-1] | Set-Content -Encoding UTF8 otvety\5
```

18.6. Сколько имён. *Подсказка ученику: `[regex]::Matches` возвращает все совпадения, их число — свойство `Count`.*

```
([regex]::Matches((Get-Content 'srv\etc\ssl\cert.txt' -Raw), 'DNS:')).Count |
Set-Content -Encoding UTF8 otvety\6
```

18.7. Список контрольных сумм. *Подсказка ученику: Конвейер: `Get-ChildItem`, `Sort-Object Name`, `ForEach-Object` с форматированием строки.*

```
Get-ChildItem 'srv\var\www\files\f*.dat' | Sort-Object Name | ForEach-Object {
"$($_.Name) $((Get-FileHash -Algorithm SHA256 $_.FullName).Hash)" } | Set-Content
-Encoding UTF8 hashes.txt
```

18.8. Контрольная: акт проверки. *Подсказка ученику: Все значения лежат в ваших ответах 2–5.*

```
@("BAD=$(Get-Content otvety\2)", "SECRET=$(Get-Content otvety\3)",
"CN=$(Get-Content otvety\4)", "EXPIRES=$(Get-Content otvety\5)") | Set-Content
-Encoding UTF8 crypto.txt
```

Модуль 19. Веб-сервер: конфигурация и журнал доступа

Linux.

19.1. Порт сайта. *Подсказка ученику: Директива заканчивается точкой с запятой; уберите её командой `gsub` или `tr -d ';'`.*

```
awk '$1 == "listen" {gsub(";", "", $2); print $2}' srv/etc/nginx/sites/portal.conf
> otvety/1
```

19.2. Корневой каталог. *Подсказка ученику: Тот же приём, что в задании 1.*

```
awk '$1 == "root" {gsub(";", "", $2); print $2}' srv/etc/nginx/sites/portal.conf >
otvety/2
```

19.3. Скрыть версию сервера. *Подсказка ученику: Директиву можно вставить сразу после `server_name`: `sed -i '/server_name/a …'`. Следите, чтобы скобки остались парными.*

```
sed -i '/server_name/a server_tokens off;' srv/etc/nginx/sites/portal.conf
```

19.4. Страницы не найдены. *Подсказка ученику: Фильтр `awk '$9 == 404'` оставляет нужные строки, `wc -l` их считает. Просто `grep 404` ошибётся: число 404 может встретиться и в других полях.*

```
awk '$9 == 404' srv/var/log/nginx/access.log | wc -l | tr -d ' ' > otvety/4
```

19.5. Самый активный клиент. *Подсказка ученику: Классическая связка: `sort | uniq -c | sort -rn | head -n 1`.*

```
awk '{print $1}' srv/var/log/nginx/access.log | sort | uniq -c | sort -rn | head
-n 1 | awk '{print $2}' > otvety/5
```

19.6. Ошибки сервера. *Подсказка ученику: Код начинается с цифры 5: условие `$9 ~ /^5/`.*

```
awk '$9 ~ /^5/' srv/var/log/nginx/access.log | wc -l | tr -d ' ' > otvety/6
```

19.7. Пропущенные точки с запятой. *Подсказка ученику: `grep -n` печатает номера строк, `-v` инвертирует условие: ищем строки, которые НЕ заканчиваются на `;`, `{` или `}`.*

```
grep -nvE '[;{}][[:space:]]*$' srv/etc/nginx/old.conf | cut -d: -f1 > otvety/7
```

19.8. Проверка конфигурации. *Подсказка ученику: Две проверки: парность скобок и окончание строк. Строки-исключения отфильтруйте `grep -v`, а `grep -q .` покажет, остались ли нарушители.*

```
  cat > check-conf.sh <<'EOT'
#!/bin/bash
f="$1"
o=$(grep -o '{' "$f" | wc -l)
c=$(grep -o '}' "$f" | wc -l)
if [ "$o" -ne "$c" ]; then echo BAD; exit 1; fi
if grep -vE '^[[:space:]]*(#|$)' "$f" | grep -vE '[;{}][[:space:]]*$' | grep -q .;
then echo BAD; exit 1; fi
echo GOOD
EOT
```

19.9. Контрольная: справка по порталу. *Подсказка ученику: Значения лежат в ваших ответах 1, 2, 4 и 5.*

```
printf 'PORT=%s\nROOT=%s\nNOTFOUND=%s\nTOPCLIENT=%s\n' "$(cat otvety/1)" "$(cat
otvety/2)" "$(cat otvety/4)" "$(cat otvety/5)" > web.txt
```

Windows.

19.1. Порт сайта. *Подсказка ученику: Из найденной строки оставьте только цифры: `-replace '[^0-9]', ''`.*

```
$l = Get-Content 'srv\etc\nginx\sites\portal.conf' | Where-Object { $_ -match
'^\s*listen\s' }
($l -replace '[^0-9]', '') | Set-Content -Encoding UTF8 otvety\1
```

19.2. Корневой каталог. *Подсказка ученику: Разбейте строку по пробелам (`-split`), возьмите второй элемент и отрежьте `;` (`TrimEnd`).*

```
$l = Get-Content 'srv\etc\nginx\sites\portal.conf' | Where-Object { $_ -match
'^\s*root\s' }
(($l.Trim() -split '\s+')[1]).TrimEnd(';') | Set-Content -Encoding UTF8 otvety\2
```

19.3. Скрыть версию сервера. *Подсказка ученику: Прочитайте файл, при проходе по строкам после `server_name` вставьте новую, запишите обратно.*

```
$c = Get-Content 'srv\etc\nginx\sites\portal.conf'
$o = foreach ($l in $c) { $l; if ($l -match 'server_name') { '    server_tokens
off;' } }
Set-Content -Encoding ASCII 'srv\etc\nginx\sites\portal.conf' $o
```

19.4. Страницы не найдены. *Подсказка ученику: `($_ -split ' ')[8]` — девятый элемент (нумерация с нуля). Поиск по тексту `404` может ошибиться из-за других полей.*

```
@(Get-Content 'srv\var\log\nginx\access.log' | Where-Object { ($_ -split ' ')[8]
-eq '404' }).Count | Set-Content -Encoding UTF8 otvety\4
```

19.5. Самый активный клиент. *Подсказка ученику: `Group-Object` группирует одинаковые значения, `Count` — размер группы.*

```
(Get-Content 'srv\var\log\nginx\access.log' | ForEach-Object { ($_ -split ' ')[0]
} | Group-Object | Sort-Object Count -Descending | Select-Object -First 1).Name |
Set-Content -Encoding UTF8 otvety\5
```

19.6. Ошибки сервера. *Подсказка ученику: Код начинается с 5: оператор `-like '5*'`.*

```
@(Get-Content 'srv\var\log\nginx\access.log' | Where-Object { ($_ -split ' ')[8]
-like '5*' }).Count | Set-Content -Encoding UTF8 otvety\6
```

19.7. Пропущенные точки с запятой. *Подсказка ученику: Заведите счётчик строк и выводите его, когда строка не оканчивается на `;`, `{` или `}`.*

```
$i = 0
Get-Content 'srv\etc\nginx\old.conf' | ForEach-Object { $i++; if ($_ -notmatch
'[;{}]\s*$') { $i } } | Set-Content -Encoding UTF8 otvety\7
```

19.8. Контрольная: справка по порталу. *Подсказка ученику: Значения лежат в ваших ответах 1, 2, 4 и 5.*

```
@("PORT=$(Get-Content otvety\1)", "ROOT=$(Get-Content otvety\2)",
"NOTFOUND=$(Get-Content otvety\4)", "TOPCLIENT=$(Get-Content otvety\5)") |
Set-Content -Encoding UTF8 web.txt
```

Модуль 20. Конфигурации под контролем версий

Linux.

20.1. Сколько коммитов. *Подсказка ученику: Ключ `-C КАТАЛОГ` выполняет команду `git` так, будто вы находитесь в этом каталоге.*

```
git -C srv/opt/infra rev-list --count HEAD > otvety/1
```

20.2. Первое сообщение. *Подсказка ученику: `git log --format=%s` печатает только темы коммитов, `--reverse` меняет порядок на хронологический.*

```
git -C srv/opt/infra log --reverse --format=%s | head -n 1 > otvety/2
```

20.3. Кто включил отладку. *Подсказка ученику: Ключ `-S строка` показывает коммиты, в которых число вхождений строки изменилось, то есть где она появилась или исчезла.*

```
git -C srv/opt/infra log -S debug=true --format=%s > otvety/3
```

20.4. Что не закоммичено. *Подсказка ученику: Вывод `--porcelain` предназначен для сценариев: код состояния и имя файла.*

```
git -C srv/opt/infra status --porcelain | awk '{print $2}' > otvety/4
```

20.5. Зафиксировать изменение. *Подсказка ученику: Ключ `-a` добавляет в коммит изменения отслеживаемых файлов, `-m` задаёт сообщение.*

```
git -C srv/opt/infra commit -q -am "Update $(cat otvety/4)"
```

20.6. Ветка для срочной правки. *Подсказка ученику: Ветка — лёгкий указатель на коммит; срочные правки делают в отдельной ветке, чтобы не мешать основной.*

```
git -C srv/opt/infra checkout -q -b hotfix
```

20.7. Идемпотентная настройка. *Подсказка ученику: `grep -qxF` ищет точное совпадение всей строки и не печатает ничего; код выхода сообщает, найдена ли строка.*

```
  cat > ensure-line.sh <<'EOT'
#!/bin/bash
f="$1"; line="$2"
grep -qxF -- "$line" "$f" || echo "$line" >> "$f"
EOT
```

20.8. Контрольная: паспорт репозитория. *Подсказка ученику: Первые три значения лежат в ваших ответах, ветку покажет `git rev-parse --abbrev-ref HEAD`.*

```
printf 'FIRST=%s\nDEBUG=%s\nCHANGED=%s\nBRANCH=%s\n' "$(cat otvety/2)" "$(cat
otvety/3)" "$(cat otvety/4)" "$(git -C srv/opt/infra rev-parse --abbrev-ref HEAD)"
> infra.txt
```

Windows.

20.1. Сколько коммитов. *Подсказка ученику: Ключ `-C КАТАЛОГ` выполняет команду `git` так, будто вы находитесь в этом каталоге.*

```
git -C srv\opt\infra rev-list --count HEAD | Set-Content -Encoding UTF8 otvety\1
```

20.2. Первое сообщение. *Подсказка ученику: `--format=%s` печатает только темы коммитов, `--reverse` меняет порядок на хронологический.*

```
git -C srv\opt\infra log --reverse --format=%s | Select-Object -First 1 |
Set-Content -Encoding UTF8 otvety\2
```

20.3. Кто включил отладку. *Подсказка ученику: Ключ `-S строка` показывает коммиты, в которых число вхождений строки изменилось.*

```
git -C srv\opt\infra log -S debug=true --format=%s | Set-Content -Encoding UTF8
otvety\3
```

20.4. Что не закоммичено. *Подсказка ученику: Строка `--porcelain` состоит из кода состояния и имени файла; код нужно отбросить.*

```
(git -C srv\opt\infra status --porcelain) -replace '^\s*\S+\s+', '' | Set-Content
-Encoding UTF8 otvety\4
```

20.5. Зафиксировать изменение. *Подсказка ученику: Ключ `-a` добавляет изменения отслеживаемых файлов, `-m` задаёт сообщение.*

```
git -C srv\opt\infra commit -q -am "Update $(Get-Content otvety\4)"
```

20.6. Ветка для срочной правки. *Подсказка ученику: Ветка — лёгкий указатель на коммит; срочные правки делают в отдельной ветке.*

```
git -C srv\opt\infra checkout -q -b hotfix
```

20.7. Идемпотентная настройка. *Подсказка ученику: `Select-String -Quiet` возвращает `$true` или `$false`; добавляйте строку только при `$false`.*

```
if (-not (Select-String -Path 'srv\opt\settings.conf' -Pattern '^retries=3$'
-Quiet)) { Add-Content -Encoding ASCII 'srv\opt\settings.conf' 'retries=3' }
```

20.8. Контрольная: паспорт репозитория. *Подсказка ученику: Первые три значения лежат в ваших ответах, ветку покажет `git rev-parse --abbrev-ref HEAD`.*

```
@("FIRST=$(Get-Content otvety\2)", "DEBUG=$(Get-Content otvety\3)",
"CHANGED=$(Get-Content otvety\4)", "BRANCH=$(git -C srv\opt\infra rev-parse
--abbrev-ref HEAD)") | Set-Content -Encoding UTF8 infra.txt
```

Модуль 21. Копии: ротация, инкременты и восстановление

Linux.

21.1. Сколько копий. *Подсказка ученику: `ls шаблон | wc -l`. Метка `.last` скрытая и под шаблон не подходит.*

```
ls srv/backup/data-*.tar.gz | wc -l | tr -d ' ' > otvety/1
```

21.2. Самая свежая копия. *Подсказка ученику: Дата в формате ГГГГ-ММ-ДД сортируется по алфавиту так же, как по времени: `sort | tail -n 1`.*

```
ls srv/backup | grep '^data-' | sort | tail -n 1 > otvety/2
```

21.3. Битый архив. *Подсказка ученику: `gzip -t файл` молчит, если архив исправен, и возвращает ненулевой код, если нет.*

```
for f in srv/backup/data-*.tar.gz; do gzip -t "$f" 2>/dev/null || basename "$f";
done > otvety/3
```

21.4. Восстановить один файл. *Подсказка ученику: Посмотреть содержимое можно командой `tar -tzf архив`. Если копия не читается, берите предыдущую. Имя исправной копии определите из заданий 2–3.*

```
tar -xzf srv/backup/$(cat .snap/goodname) -C restore ./report.txt
```

21.5. Оставить пять копий. *Подсказка ученику: `head -n -5` печатает все строки, кроме последних пяти; их остаётся передать `xargs rm`.*

```
ls srv/backup/data-*.tar.gz | sort | head -n -5 | xargs rm -f
```

21.6. Что изменилось с прошлой копии. *Подсказка ученику: `find КАТАЛОГ -type f -newer ЭТАЛОН` отбирает файлы новее эталонного; `-printf '%f\n'` печатает только имя.*

```
find srv/data -type f -newer srv/backup/.last -printf '%f\n' | sort > otvety/6
```

21.7. Сценарий ротации. *Подсказка ученику: Тот же приём, что в задании 5, но каталог и число берутся из параметров $1 и $2. Проверьте случай, когда файлов меньше N.*

```
  cat > rotate.sh <<'EOT'
#!/bin/bash
ls -1 "$1"/data-*.tar.gz 2>/dev/null | sort | head -n -"$2" | xargs -r rm -f
EOT
```

21.8. Контрольная: отчёт о копиях. *Подсказка ученику: Значения лежат в ваших ответах 1, 3, 6 и в восстановленном файле.*

```
printf 'COUNT=%s\nBAD=%s\nRESTORED=%s\nCHANGED=%s\n' "$(cat otvety/1)" "$(cat
otvety/3)" "$(head -n 1 restore/report.txt)" "$(wc -l < otvety/6 | tr -d ' ')" >
backup.txt
```

Windows.

21.1. Сколько копий. *Подсказка ученику: `Get-ChildItem шаблон`, затем `.Count` у массива.*

```
@(Get-ChildItem 'srv\backup\data-*.zip').Count | Set-Content -Encoding UTF8
otvety\1
```

21.2. Самая свежая копия. *Подсказка ученику: Имя вида ГГГГ-ММ-ДД сортируется по алфавиту так же, как по времени.*

```
(Get-ChildItem 'srv\backup\data-*.zip' | Sort-Object Name | Select-Object -Last
1).Name | Set-Content -Encoding UTF8 otvety\2
```

21.3. Битый архив. *Подсказка ученику: Исправный архив открывается без ошибки, повреждённый вызывает исключение: оберните `OpenRead` в `try … catch`.*

```
Add-Type -AssemblyName System.IO.Compression.FileSystem
Get-ChildItem 'srv\backup\data-*.zip' | ForEach-Object {
    $f = $_
    try { $z = [IO.Compression.ZipFile]::OpenRead($f.FullName); $z.Dispose() }
    catch { $f.Name }
} | Where-Object { $_ } | Set-Content -Encoding UTF8 otvety\3
```

21.4. Восстановить один файл. *Подсказка ученику: `Expand-Archive -Path архив -DestinationPath каталог -Force`. Имя исправной копии определите по заданиям 2–3 (для решения достаточно знать, что последняя копия может быть битой).*

```
Expand-Archive -Path ('srv\backup\' + (Get-Content '.snap\goodname'))
-DestinationPath restore -Force
```

21.5. Оставить пять копий. *Подсказка ученику: `Select-Object -SkipLast 5` пропускает последние пять элементов; всё остальное передайте в `Remove-Item`.*

```
Get-ChildItem 'srv\backup\data-*.zip' | Sort-Object Name | Select-Object -SkipLast
5 | Remove-Item
```

21.6. Что изменилось с прошлой копии. *Подсказка ученику: Сравните `LastWriteTime` каждого файла со временем метки.*

```
$m = (Get-Item 'srv\backup\last.txt').LastWriteTime
Get-ChildItem 'srv\data' -File | Where-Object { $_.LastWriteTime -gt $m } |
Sort-Object Name | ForEach-Object { $_.Name } | Set-Content -Encoding UTF8
otvety\6
```

21.7. Контрольная: отчёт о копиях. *Подсказка ученику: Значения лежат в ваших ответах 1, 3, 6 и в восстановленном файле.*

```
@("COUNT=$(Get-Content otvety\1)", "BAD=$(Get-Content otvety\3)",
"RESTORED=$(@(Get-Content restore\report.txt)[0])", "CHANGED=$(@(Get-Content
otvety\6).Count)") | Set-Content -Encoding UTF8 backup.txt
```

A.3 Учебные команды полигона и расхождения с настоящими

Учебные команды работают только с файлами каталога `srv` и ничего не меняют в системе. Вне полигона (там, где нет каталога `srv`) `df`, `lsblk`, `blkid` и `ss` вызывают настоящие программы. Ниже перечислено, что модель *умеет* и чего *не* умеет.

Команда	Что поддерживается	Чем отличается от настоящей
Linux		

useradd, usermod, userdel, groupadd, groupmod, groupdel, gpasswd, passwd, chpasswd, chage, id, getent	Ключи, которые встречаются в заданиях; проверка имён, групп, дат; сообщения об ошибках	Нет PAM, квот и шаблона на `/etc/skel` сверх базового набора; хеш пароля имитируется
sudo, visudo	`sudo -l [-U пользователь] [команда]`; `visudo -c [-f файл]`; псевдонимы, `Defaults`, `@includedir`, теги `NOPASSWD/PASSWD`, `sudoedit`	Команды не выполняются; нет проверки `requiretty`, `secure_path` и др.
apt, apt-get, apt-cache, apt-mark, dpkg	`update`, `install`, `remove`, `purge`, `autoremove`, `upgrade`, `list`, `search`, `show`; `dpkg -l/-L/-S/-s`; метки auto/manual, hold	Версии сравниваются на неравенство; нет подписей, Recommends и параллельных репозиториев
systemctl, journalctl, service	`status`, `start`, `stop`, `restart`, `reload`, `enable`, `disable`, `is-active`, `is-enabled`, `list-units`, `daemon-reload`; проверка `ExecStart`, `User`, `WorkingDirectory`, `EnvironmentFile`, `Requires`	Службы не запускаются; журнал — простой текстовый файл на службу
crontab, cron-next, cron-run	Пять полей, списки, диапазоны, шаги, названия месяцев и дней; правило «день месяца ИЛИ день недели»	Нет `@`-сокращений в `cron-next`; «сейчас» зафиксировано, задания не запускаются
ufw, fwtest	`status`, `enable`, `disable`, `default`, `allow`, `deny`, `insert`, `delete`; подсети; первое подходящее правило	Нет IPv6, профилей приложений, журналирования
df, lsblk, blkid, ss	Показывают учебные таблицы из `srv/var/lib/lab`	Вне полигона — настоящие программы
logrotate-check	Проверка структуры конфигурации logrotate	Не выполняет ротацию
Windows		
Get/New/Set/Enable/Disable/Remove-LocalUser, Get/New/Remove-LocalGroup, Add/Remove/Get-LocalGroupMember	Параметры и исходы; таблицы `etc\localusers.csv`, `localgroups.csv`	Нет паролей и политики; SID — условные
Get/Start/Stop/Restart/Set/New-Service, sc.exe	`config`, `binPath/obj/start`, `failure`, `qc`, `query`; проверка пути, учётной записи, зависимостей	Процессы служб не запускаются

Get-WinEvent	`-FilterHashtable` с ключами `LogName`, `Id`, `Level`, `ProviderName`, `StartTime`; `-MaxEvents`	Журнал — таблица в CSV
winget	`list`, `search`, `show`, `install`, `uninstall`, `upgrade [--all]`, `pin add/remove/list`	Локальный каталог из шести-восьми пакетов
Register/Get/Unregister-ScheduledTask, New-ScheduledTaskAction/Trigger, Get-LabNextRun	Daily, Weekly, Once, AtStartup; проверка времени и дней недели	Нет условий, повторов и запуска от имени других пользователей
Get-NetTCPConnection	Порты, протокол, адрес, блокирующее правило сильнее разрешающего	Профили ведут общее состояние; порт и адрес входят в объект правила
Get-NetFirewallRule, New/Remove/Set-NetFirewallRule, Set/Get-NetFirewallProfile, Test-LabFirewall		
Get-ComputerInfo, Get-Volume	Свойства, использованные в заданиях	Остальные свойства не заполнены

После курса стоит повторить ключевые операции на настоящей виртуальной машине: создать пользователя и изменить его группы, установить пакет и перезапустить службу, поставить задание в cron или планировщик, добавить правило брандмауэра. Команды те же, но последствия настоящие.

A.4 Как проверен полигон

Полигон проверяет не только ученика: у него есть собственная самопроверка (`selftest`). Она создаёт чистый модуль, убеждается, что ни одно задание не выполнено, применяет эталонные решения из предыдущего приложения и убеждается, что выполнены все. Так проверяется сама проверка: задание, которое проходит «до» решения, или не проходит «после», обнаруживается сразу. Ниже — честный перечень того, где и как это выполнено.

Дорожка	Среда проверки	Результат
Windows, 172 заданий в 21 модуле	Windows 11, Windows PowerShell 5.1; настоящие `icacls`, процессы, `Compress-Archive`, `robocopy`, `ssh-keygen`, `ssh -G`	Все модули: на чистом полигоне 0 выполненных заданий, после эталонных решений — все.

Linux, 203 заданий в 21 модуле	Git Bash (MSYS2) на Windows 11, Python 3; настоящие `awk`, `sed`, `grep`, `find`, `tar`, `sha256sum`, `ssh-keygen`, процессы, bash-сценарии	Модули 1, 2, 4–21 проходят (в 4 и 5 пропущены задания 4.6, 4.9 и 5.6, см. ниже).

Что не проверено на настоящем Linux. Среда Git Bash не умеет хранить права доступа Linux (`chmod 440` превращается в 444, `setuid`-биты и ACL недоступны) и не содержит `renice`. Поэтому *модуль 3 целиком* (права, umask, setgid, sticky, ACL), задания 4.6 и 4.9 (права 0440 файлов `sudoers.d`) и задание 5.6 (`renice`) при самопроверке пропускались. Их эталонные решения написаны по документации и должны быть проверены запуском самопроверки на Ubuntu 24.04 обычным пользователем:

```
bash setup.sh --force
bash selftest.sh          # ожидается: «САМОПРОВЕРКА ПРОЙДЕНА»
```

Если на вашей системе какое-либо задание проходит не так, как описано, сообщите номер: для таких случаев и существует самопроверка. Для Windows-дорожки аналогично: `.\selftest.ps1`. Результаты работы учебных команд сверены с настоящими утилитами по документации, но модель остаётся упрощением (см. предыдущее приложение).

A.5 Книга преподавателя: практикум по SQL

Решения приведены для проверки работы и разбора затруднений; допускается любой вариант, дающий тот же результат. Задания, где ответ записывается в файл `otvety`, показаны командой, которую выполняет ученик; значения зависят от `.seed`. Темы 9–15, 17 и 18 изменяют базу и опираются на результат предыдущих заданий: выполнять их нужно по порядку.

Тема 1. Первое знакомство: база данных в одном файле

1.1. Сколько клиентов. *Подсказка ученику: `SELECT COUNT(*) FROM customers;` — функция `COUNT(*)` считает строки.*

```
sqlite3 shop.db "SELECT COUNT(*) FROM customers" > otvety/1
```

1.2. Сколько поставщиков. *Подсказка ученику: `SELECT COUNT(*) FROM suppliers;`*

```
sqlite3 shop.db "SELECT COUNT(*) FROM suppliers" > otvety/2
```

1.3. Сколько товаров. *Подсказка ученику: Тот же приём, что в задании 1, но для другой таблицы: `SELECT COUNT(*) FROM products;`*

```
sqlite3 shop.db "SELECT COUNT(*) FROM products" > otvety/3
```

1.4. Сколько заказов. *Подсказка ученику: `SELECT COUNT(*) FROM orders;`*

```
sqlite3 shop.db "SELECT COUNT(*) FROM orders" > otvety/4
```

1.5. Сколько позиций в заказах. *Подсказка ученику: `sqlite3 shop.db "SELECT COUNT(*) FROM order_items;" > otvety/5`*

```
sqlite3 shop.db "SELECT COUNT(*) FROM order_items" > otvety/5
```

1.6. Клиенты и поставщики вместе. *Подсказка ученику: Два запроса `SELECT COUNT(*) FROM customers;` и `SELECT COUNT(*) FROM suppliers;`, затем сложите числа (в консоли это можно сделать и запросом `SELECT 1 + 2;` с вашими числами).*

```
sqlite3 shop.db "SELECT (SELECT COUNT(*) FROM customers) + (SELECT COUNT(*) FROM
suppliers)" > otvety/6
```

1.7. Разница товаров и поставщиков. *Подсказка ученику: Два запроса `SELECT COUNT(*) FROM products;` и `SELECT COUNT(*) FROM suppliers;`, затем вычтите числа.*

```
sqlite3 shop.db "SELECT (SELECT COUNT(*) FROM products) - (SELECT COUNT(*) FROM
suppliers)" > otvety/7
```

1.8. Строк во всей базе. *Подсказка ученику: Сложите пять чисел, полученных запросами `SELECT COUNT(*) FROM таблица;`.*

```
sqlite3 shop.db "SELECT (SELECT COUNT(*) FROM suppliers) + (SELECT COUNT(*) FROM
products) + (SELECT COUNT(*) FROM customers) + (SELECT COUNT(*) FROM orders) +
(SELECT COUNT(*) FROM order_items)" > otvety/8
```

1.9. Копия базы. *Подсказка ученику: `cp shop.db shop_copy.db` (в PowerShell `Copy-Item shop.db shop_copy.db`), затем `sqlite3 shop_copy.db "SELECT COUNT(*) FROM customers;"` с записью результата в `otvety/9`.*

```
cp shop.db shop_copy.db        # в PowerShell: Copy-Item shop.db shop_copy.db
sqlite3 shop_copy.db "SELECT COUNT(*) FROM customers;" > otvety/9
```

1.10. Вычисление без таблицы. *Подсказка ученику: `SELECT 52 * 7;` — выражение вычисляется без `FROM`. Ответ: 364.*

```
sqlite3 shop.db "SELECT 52 * 7" > otvety/10
```

1.11. Какие в базе таблицы. *Подсказка ученику: `SELECT name FROM sqlite_master WHERE type = 'table';` Служебные таблицы SQLite начинаются с `sqlite_`: в этой базе их нет.*

```
sqlite3 shop.db "SELECT name FROM sqlite_master WHERE type = 'table'" > otvety/11
```

1.12. Столбцы таблицы. *Подсказка ученику: `PRAGMA table_info(имя_таблицы);` выводит по строке на столбец: номер, имя, тип, признак NOT NULL, значение по умолчанию, признак первичного ключа.*

```
sqlite3 shop.db "PRAGMA table_info(products);" > otvety/12
```

1.13. Как создана таблица. *Подсказка ученику: `.schema orders` печатает текст команды создания таблицы. Ту же строку хранит столбец `sql` таблицы `sqlite_master`.*

```
sqlite3 shop.db ".schema orders" > otvety/13
```

1.14. Имена столбцов клиентов. *Подсказка ученику: `SELECT name FROM pragma_table_info('customers') ORDER BY cid;` либо `PRAGMA table_info(customers);` и столбец `name` результата.*

```
sqlite3 shop.db "SELECT name FROM pragma_table_info('customers') ORDER BY cid" >
otvety/14
```

1.15. Сколько столбцов. *Подсказка ученику: `SELECT COUNT(*) FROM pragma_table_info('order_items');` — одна строка описания на каждый столбец.*

```
sqlite3 shop.db "SELECT COUNT(*) FROM pragma_table_info('order_items')" >
otvety/15
```

1.16. Составной первичный ключ. *Подсказка ученику: У столбцов ключа в `table_info` значение `pk` больше нуля, и оно задаёт порядок в ключе: `SELECT name FROM pragma_table_info('order_items') WHERE pk > 0 ORDER BY pk;`*

```
sqlite3 shop.db "SELECT name FROM pragma_table_info('order_items') WHERE pk > 0
ORDER BY pk" > otvety/16
```

1.17. Столбцы, допускающие NULL. *Подсказка ученику: Выберите столбцы с `notnull = 0` и `pk = 0`: `SELECT name FROM pragma_table_info('customers') WHERE [notnull] = 0 AND pk = 0 ORDER BY cid;`*

```
sqlite3 shop.db "SELECT name FROM pragma_table_info('customers') WHERE [notnull] =
0 AND pk = 0 ORDER BY cid" > otvety/17
```

1.18. Тип цены. *Подсказка ученику: Тип — столбец `type` описания: `SELECT type FROM pragma_table_info('products') WHERE name = 'price';`*

```
sqlite3 shop.db "SELECT type FROM pragma_table_info('products') WHERE name =
'price'" > otvety/18
```

1.19. Значение по умолчанию. *Подсказка ученику: Значение по умолчанию — столбец `dflt_value`: `SELECT dflt_value FROM pragma_table_info('customers') WHERE name = 'vip';` либо найдите `DEFAULT` в `.schema customers`.*

```
sqlite3 shop.db "SELECT dflt_value FROM pragma_table_info('customers') WHERE name
= 'vip'" > otvety/19
```

1.20. Столбцов во всей базе. *Подсказка ученику: Пять запросов `SELECT COUNT(*) FROM pragma_table_info('таблица');` (по одному на таблицу), затем сложите числа.*

```
sqlite3 shop.db "SELECT COUNT(*) FROM sqlite_master m JOIN
pragma_table_info(m.name) WHERE m.type = 'table'" > otvety/20
```

1.21. Первые пять клиентов. *Подсказка ученику: `SELECT * FROM customers LIMIT 5;` — звёздочка означает «все столбцы», `LIMIT 5` ограничивает число строк.*

```
sqlite3 shop.db "SELECT * FROM customers ORDER BY id LIMIT 5" > otvety/21
```

1.22. Название и цена. *Подсказка ученику: Перечислите нужные столбцы через запятую вместо звёздочки: `SELECT name, price FROM products LIMIT 3;`*

```
sqlite3 shop.db "SELECT name, price FROM products ORDER BY id LIMIT 3" > otvety/22
```

1.23. Все поставщики. *Подсказка ученику: `SELECT * FROM suppliers;` — таблица маленькая, `LIMIT` не нужен.*

```
sqlite3 shop.db "SELECT * FROM suppliers ORDER BY id" > otvety/23
```

1.24. Столбцы в своём порядке. *Подсказка ученику: Порядок столбцов в результате задаёт список после `SELECT`: `SELECT country, name FROM suppliers;`*

```
sqlite3 shop.db "SELECT country, name FROM suppliers ORDER BY id" > otvety/24
```

1.25. Первые четыре заказа. *Подсказка ученику: `SELECT id, order_date, status FROM orders LIMIT 4;`*

```
sqlite3 shop.db "SELECT id, order_date, status FROM orders ORDER BY id LIMIT 4" >
otvety/25
```

1.26. Пропустить и взять. *Подсказка ученику: `LIMIT 5 OFFSET 5`: `SELECT id, name, stock FROM products LIMIT 5 OFFSET 5;`*

```
sqlite3 shop.db "SELECT id, name, stock FROM products ORDER BY id LIMIT 5 OFFSET
5" > otvety/26
```

1.27. Клиенты с известным городом. *Подсказка ученику: `COUNT(city)` не считает `NULL`: `SELECT COUNT(city) FROM customers;`. Ловушка: `COUNT(*)` посчитает всех клиентов.*

```
sqlite3 shop.db "SELECT COUNT(city) FROM customers" > otvety/27
```

1.28. Клиенты с email. *Подсказка ученику: `SELECT COUNT(email) FROM customers;`*

```
sqlite3 shop.db "SELECT COUNT(email) FROM customers" > otvety/28
```

1.29. Сколько разных городов. *Подсказка ученику: `SELECT COUNT(DISTINCT city) FROM customers;` — `COUNT(DISTINCT ...)` считает разные значения и пропускает `NULL`.*

```
sqlite3 shop.db "SELECT COUNT(DISTINCT city) FROM customers" > otvety/29
```

1.30. Статусы заказов. *Подсказка ученику: `SELECT DISTINCT status FROM orders;` — `DISTINCT` убирает повторы.*

```
sqlite3 shop.db "SELECT DISTINCT status FROM orders" > otvety/30
```

1.31. Отчёт с заголовками. *Подсказка ученику: Последовательность: `.headers on`, `.output report.txt`, запрос, `.output stdout`. Заголовки печатает только включённый режим `.headers on`.*

```
.headers on
.output report.txt
SELECT name, city FROM customers ORDER BY id LIMIT 5;
.output stdout
```

1.32. Поставщики с заголовками. *Подсказка ученику: `.headers on`, `.output otvety/32`, `SELECT id, name, country FROM suppliers;`, `.output stdout`.*

```
sqlite3 shop.db "SELECT id, name, country FROM suppliers ORDER BY id" > otvety/32
```

1.33. Товары в формате CSV. *Подсказка ученику: `.mode csv`, `.output otvety/33`, `SELECT id, name, price FROM products LIMIT 5;`, `.output stdout`.*

```
sqlite3 shop.db "SELECT id, name, price FROM products ORDER BY id LIMIT 5" >
otvety/33
```

1.34. Выровненные столбцы. *Подсказка ученику: `.headers on`, `.mode column`, `.output otvety/34`, запрос `SELECT id, name, category, price FROM products LIMIT 4;`, `.output stdout`.*

```
sqlite3 shop.db "SELECT id, name, category, price FROM products ORDER BY id LIMIT
4" > otvety/34
```

1.35. Файл suppliers.csv. *Подсказка ученику: Режим `.mode csv` разделяет значения запятыми, `.headers on` добавляет строку заголовков: `.headers on`, `.mode csv`, `.output suppliers.csv`, запрос, `.output stdout`.*

```
.headers on
.mode csv
.output suppliers.csv
SELECT id, name, country FROM suppliers;
.output stdout
```

1.36. Заказы без заголовков. *Подсказка ученику: .mode list, .headers off, .output otvety/36, SELECT id, status FROM orders LIMIT 6;, .output stdout.*

```
sqlite3 shop.db "SELECT id, status FROM orders ORDER BY id LIMIT 6" > otvety/36
```

1.37. VIP-признак в CSV. *Подсказка ученику: .mode csv, .output otvety/37, SELECT id, name, vip FROM customers LIMIT 6;, .output stdout.*

```
sqlite3 shop.db "SELECT id, name, vip FROM customers ORDER BY id LIMIT 6" >
otvety/37
```

1.38. Пустые значения в CSV. *Подсказка ученику: .mode csv, .output otvety/38, SELECT id, city, email FROM customers LIMIT 8;, .output stdout. NULL печатается пустым полем.*

```
sqlite3 shop.db "SELECT id, city, email FROM customers ORDER BY id LIMIT 8" >
otvety/38
```

1.39. Клиенты столбцами. *Подсказка ученику: .headers on, .mode column, .output otvety/39, SELECT id, name, registered FROM customers LIMIT 6 OFFSET 6;, .output stdout.*

```
sqlite3 shop.db "SELECT id, name, registered FROM customers ORDER BY id LIMIT 6
OFFSET 6" > otvety/39
```

1.40. Статусы заказов в CSV. *Подсказка ученику: .headers on, .mode csv, .output statuses.csv, SELECT DISTINCT status FROM orders;, .output stdout.*

```
.headers on
.mode csv
.output statuses.csv
SELECT DISTINCT status FROM orders;
.output stdout
```

1.41. Позиции заказов в файл. *Подсказка ученику: .output otvety/41, SELECT * FROM order_items LIMIT 5;, .output stdout.*

```
sqlite3 shop.db "SELECT * FROM order_items ORDER BY rowid LIMIT 5" > otvety/41
```

1.42. Клиенты и VIP. *Подсказка ученику: .output otvety/42, SELECT name, vip FROM customers LIMIT 8;, .output stdout.*

```
sqlite3 shop.db "SELECT name, vip FROM customers ORDER BY id LIMIT 8" > otvety/42
```

1.43. Сколько клиентов делали заказы. *Подсказка ученику: `.output otvety/43`, `SELECT COUNT(DISTINCT customer_id) FROM orders;`, `.output stdout`.*

```
sqlite3 shop.db "SELECT COUNT(DISTINCT customer_id) FROM orders" > otvety/43
```

1.44. Страны поставщиков. *Подсказка ученику: `SELECT COUNT(DISTINCT country) FROM suppliers;` с записью результата в `otvety/44`.*

```
sqlite3 shop.db "SELECT COUNT(DISTINCT country) FROM suppliers" > otvety/44
```

1.45. Остатки на складе. *Подсказка ученику: `.output otvety/45`, `SELECT name, stock FROM products LIMIT 6;`, `.output stdout`.*

```
sqlite3 shop.db "SELECT name, stock FROM products ORDER BY id LIMIT 6" > otvety/45
```

1.46. Заказы со смещением. *Подсказка ученику: `SELECT id, order_date FROM orders LIMIT 5 OFFSET 20;` между `.output otvety/46` и `.output stdout`.*

```
sqlite3 shop.db "SELECT id, order_date FROM orders ORDER BY id LIMIT 5 OFFSET 20"
> otvety/46
```

1.47. Два результата в одном файле. *Подсказка ученику: `.output otvety/47`, затем `SELECT COUNT(*) FROM customers;` и `SELECT COUNT(*) FROM products;`, затем `.output stdout`. Результаты одного сеанса дописываются в файл друг за другом.*

```
.output otvety/47
SELECT COUNT(*) FROM customers;
SELECT COUNT(*) FROM products;
.output stdout
```

1.48. Названия поставщиков в файл. *Подсказка ученику: `.once names.txt` перенаправляет вывод только следующего запроса: `SELECT name FROM suppliers;`. Можно и `.output names.txt` с `.output stdout` в конце.*

```
.once names.txt
SELECT name FROM suppliers;
```

1.49. Категории из оболочки. *Подсказка ученику: `sqlite3 shop.db "SELECT DISTINCT category FROM products;" > otvety/49` (в PowerShell 5.1 файл будет в UTF-16: проверка это понимает).*

```
sqlite3 shop.db "SELECT DISTINCT category FROM products;" > otvety/49
```

1.50. Товары с 11-го по 20-й. *Подсказка ученику: `.output otvety/50`, `SELECT id, name, price FROM products LIMIT 10 OFFSET 10;`, `.output stdout`.*

```
sqlite3 shop.db "SELECT id, name, price FROM products ORDER BY id LIMIT 10 OFFSET
10" > otvety/50
```

Тема 2. Выбор столбцов, выражения и сортировка

2.1. Имена и почта. *Подсказка ученику: `SELECT name, email FROM customers;` Строки идут в порядке `id`; отсутствующее значение (NULL) `sqlite3` печатает пустой строкой.*

```
sqlite3 shop.db "SELECT name, email FROM customers ORDER BY id" > otvety/1
```

2.2. Все столбцы. *Подсказка ученику: `SELECT * FROM suppliers;` — звёздочка означает «все столбцы».*

```
sqlite3 shop.db "SELECT * FROM suppliers" > otvety/2
```

2.3. Отделы и города. *Подсказка ученику: Откройте `hr.db` и выполните `SELECT name, city FROM departments;`*

```
sqlite3 hr.db "SELECT name, city FROM departments" > otvety/3
```

2.4. Столбцы в своём порядке. *Подсказка ученику: Порядок столбцов в результате задаёт список после SELECT: `SELECT price, name FROM products;`*

```
sqlite3 shop.db "SELECT price, name FROM products" > otvety/4
```

2.5. Склад. *Подсказка ученику: `SELECT id, name, stock FROM products;`*

```
sqlite3 shop.db "SELECT id, name, stock FROM products" > otvety/5
```

2.6. Клиенты и города. *Подсказка ученику: `SELECT name, city, vip FROM customers;`*

```
sqlite3 shop.db "SELECT name, city, vip FROM customers" > otvety/6
```

2.7. Состав персонала. *Подсказка ученику: `SELECT name, position, salary FROM employees;` в базе `hr.db`.*

```
sqlite3 hr.db "SELECT name, position, salary FROM employees" > otvety/7
```

2.8. Позиции заказов. *Подсказка ученику: `SELECT order_id, product_id, qty FROM order_items;`*

```
sqlite3 shop.db "SELECT order_id, product_id, qty FROM order_items" > otvety/8
```

2.9. Заказы и статусы. *Подсказка ученику: `SELECT id, customer_id, status FROM orders;`*

```
sqlite3 shop.db "SELECT id, customer_id, status FROM orders" > otvety/9
```

2.10. Вся таблица сотрудников. *Подсказка ученику: `SELECT * FROM employees;` в базе `hr.db`. Пустые поля — это `NULL`.*

```
sqlite3 hr.db "SELECT * FROM employees" > otvety/10
```

2.11. Цена в копейках. *Подсказка ученику: Выражение можно поместить в список столбцов и дать ему имя ключевым словом `AS`: `SELECT name, price * 100 AS price_kop ...`*

```
sqlite3 shop.db "SELECT name, price * 100 AS price_kop FROM products ORDER BY id"
> otvety/11
```

2.12. Цена с НДС. *Подсказка ученику: `SELECT name, price * 120 / 100 AS price_vat FROM products ORDER BY id;` Деление целых чисел даёт целое (дробная часть отбрасывается).*

```
sqlite3 shop.db "SELECT name, price * 120 / 100 AS price_vat FROM products ORDER
BY id" > otvety/12
```

2.13. Годовая зарплата. *Подсказка ученику: `SELECT name, salary * 12 AS year_salary FROM employees ORDER BY id;`*

```
sqlite3 hr.db "SELECT name, salary * 12 AS year_salary FROM employees ORDER BY id"
> otvety/13
```

2.14. Сумма позиции. *Подсказка ученику: `SELECT order_id, product_id, qty * price AS total FROM order_items ORDER BY order_id, product_id;`*

```
sqlite3 shop.db "SELECT order_id, product_id, qty * price AS total FROM
order_items ORDER BY order_id, product_id" > otvety/14
```

2.15. Склейка строк. *Подсказка ученику: В SQL строки склеивает оператор `||`: `name || ' / ' || category`. Литералы заключаются в одинарные кавычки.*

```
sqlite3 shop.db "SELECT name || ' / ' || category FROM products ORDER BY id LIMIT
5" > otvety/15
```

2.16. Остаток от деления. *Подсказка ученику: Остаток от деления — оператор `%`: `SELECT id, id % 3 AS grp FROM customers ORDER BY id;`*

```
sqlite3 shop.db "SELECT id, id % 3 AS grp FROM customers ORDER BY id" > otvety/16
```

2.17. Заглавные и длина. *Подсказка ученику: `SELECT UPPER(name) AS up, LENGTH(name) AS len FROM suppliers ORDER BY id;` Функции вызываются прямо в списке столбцов.*

```
sqlite3 shop.db "SELECT UPPER(name) AS up, LENGTH(name) AS len FROM suppliers
ORDER BY id" > otvety/17
```

2.18. Город по умолчанию. *Подсказка ученику: `COALESCE(city, 'не указан')` возвращает первое значение, не равное `NULL`.*

```
sqlite3 shop.db "SELECT id, name, COALESCE(city, 'не указан') AS city FROM
customers ORDER BY id" > otvety/18
```

2.19. NULL в склейке. *Подсказка ученику: `name || ' <' || email || '>'`: если `email` равен `NULL`, вся склейка даёт `NULL` (пустое поле).*

```
sqlite3 shop.db "SELECT id, name || ' <' || email || '>' AS contact FROM customers
ORDER BY id" > otvety/19
```

2.20. Стоимость остатков. *Подсказка ученику: Псевдоним, объявленный в списке столбцов, можно использовать в `ORDER BY`: `ORDER BY value DESC, id`.*

```
sqlite3 shop.db "SELECT name, price * stock AS value FROM products ORDER BY value
DESC, id LIMIT 5" > otvety/20
```

2.21. Названия по алфавиту. *Подсказка ученику: `SELECT name FROM products ORDER BY name;` — текст сортируется посимвольно, поэтому `Канц-10` стоит перед `Канц-2`.*

```
sqlite3 shop.db "SELECT name FROM products ORDER BY name" > otvety/21
```

2.22. От дорогих к дешёвым. *Подсказка ученику: `ORDER BY price DESC, id` — по убыванию; `id` разрешает ничьи.*

```
sqlite3 shop.db "SELECT name, price FROM products ORDER BY price DESC, id" >
otvety/22
```

2.23. Новые клиенты. *Подсказка ученику: Даты вида `ГГГГ-ММ-ДД` сортируются как обычный текст: `ORDER BY registered DESC, id`.*

```
sqlite3 shop.db "SELECT name, registered FROM customers ORDER BY registered DESC,
id" > otvety/23
```

2.24. Оклады по возрастанию. *Подсказка ученику: `ORDER BY salary, id` (по возрастанию — это порядок по умолчанию, `ASC`).*

```
sqlite3 hr.db "SELECT name, salary FROM employees ORDER BY salary, id" > otvety/24
```

2.25. Сортировка по двум ключам. *Подсказка ученику: В `ORDER BY` ключи перечисляются через запятую, у каждого свой порядок: `ORDER BY category, price DESC`.*

```
sqlite3 shop.db "SELECT category, name, price FROM products ORDER BY category,
price DESC" > otvety/25
```

2.26. Сортировка по номеру столбца. *Подсказка ученику: `ORDER BY 2 DESC, 1` — второй столбец списка по убыванию, затем первый.*

```
sqlite3 shop.db "SELECT name, stock FROM products ORDER BY 2 DESC, 1" > otvety/26
```

2.27. Сортировка по выражению. *Подсказка ученику: `SELECT id, name FROM products ORDER BY price * stock DESC, id;` — ключом сортировки может быть выражение.*

```
sqlite3 shop.db "SELECT id, name FROM products ORDER BY price * stock DESC, id" >
otvety/27
```

2.28. Пустые города. *Подсказка ученику: `NULL` в SQLite считается меньше любого значения, поэтому при `ASC` строки с пустым городом идут первыми.*

```
sqlite3 shop.db "SELECT id, name, city FROM customers ORDER BY city, id" >
otvety/28
```

2.29. Отделы и оклады. *Подсказка ученику: `ORDER BY dept_id, salary DESC, id`; пустой `dept_id` при сортировке по возрастанию идёт первым.*

```
sqlite3 hr.db "SELECT id, name, dept_id, salary FROM employees ORDER BY dept_id,
salary DESC, id" > otvety/29
```

2.30. Крупнейшие позиции. *Подсказка ученику: `... ORDER BY total DESC, order_id, product_id LIMIT 5;`*

```
sqlite3 shop.db "SELECT order_id, product_id, qty * price AS total FROM
order_items ORDER BY total DESC, order_id, product_id LIMIT 5" > otvety/30
```

2.31. Какие бывают категории. *Подсказка ученику: `SELECT DISTINCT category FROM products ORDER BY category;` — `DISTINCT` убирает повторяющиеся строки.*

```
sqlite3 shop.db "SELECT DISTINCT category FROM products ORDER BY category" >
otvety/31
```

2.32. Должности. *Подсказка ученику: `SELECT DISTINCT position FROM employees ORDER BY position;`*

```
sqlite3 hr.db "SELECT DISTINCT position FROM employees ORDER BY position" >
otvety/32
```

2.33. Статусы заказов. *Подсказка ученику: `SELECT DISTINCT status FROM orders ORDER BY status;`*

```
sqlite3 shop.db "SELECT DISTINCT status FROM orders ORDER BY status" > otvety/33
```

2.34. Страны поставщиков. *Подсказка ученику: SELECT DISTINCT country FROM suppliers ORDER BY country;*

```
sqlite3 shop.db "SELECT DISTINCT country FROM suppliers ORDER BY country" >
otvety/34
```

2.35. Уникальные пары. *Подсказка ученику: DISTINCT относится ко всей строке результата, а не к первому столбцу.*

```
sqlite3 shop.db "SELECT DISTINCT supplier_id, category FROM products ORDER BY
supplier_id, category" > otvety/35
```

2.36. Города и VIP. *Подсказка ученику: SELECT DISTINCT city, vip FROM customers ORDER BY city, vip; Все NULL считаются одинаковыми.*

```
sqlite3 shop.db "SELECT DISTINCT city, vip FROM customers ORDER BY city, vip" >
otvety/36
```

2.37. Кто заказывал. *Подсказка ученику: SELECT DISTINCT customer_id FROM orders ORDER BY customer_id;*

```
sqlite3 shop.db "SELECT DISTINCT customer_id FROM orders ORDER BY customer_id" >
otvety/37
```

2.38. Годы заказов. *Подсказка ученику: SELECT DISTINCT SUBSTR(order_date, 1, 4) AS year FROM orders ORDER BY year;*

```
sqlite3 shop.db "SELECT DISTINCT SUBSTR(order_date, 1, 4) AS year FROM orders
ORDER BY year" > otvety/38
```

2.39. Что продавалось. *Подсказка ученику: SELECT DISTINCT product_id FROM order_items ORDER BY product_id;*

```
sqlite3 shop.db "SELECT DISTINCT product_id FROM order_items ORDER BY product_id"
> otvety/39
```

2.40. Три самых больших остатка. *Подсказка ученику: SELECT DISTINCT stock FROM products ORDER BY stock DESC LIMIT 3; Сначала убираются повторы, затем берутся три строки.*

```
sqlite3 shop.db "SELECT DISTINCT stock FROM products ORDER BY stock DESC LIMIT 3"
> otvety/40
```

2.41. Пять самых дорогих. *Подсказка ученику: ORDER BY price DESC — по убыванию; LIMIT 5 оставляет первые пять строк.*

```
sqlite3 shop.db "SELECT name, price FROM products ORDER BY price DESC LIMIT 5" >
otvety/41
```

2.42. Три самых дешёвых. *Подсказка ученику: `SELECT name, price FROM products ORDER BY price, id LIMIT 3;`*

```
sqlite3 shop.db "SELECT name, price FROM products ORDER BY price, id LIMIT 3" >
otvety/42
```

2.43. Первые клиенты. *Подсказка ученику: `SELECT id, name FROM customers ORDER BY id LIMIT 6;`*

```
sqlite3 shop.db "SELECT id, name FROM customers ORDER BY id LIMIT 6" > otvety/43
```

2.44. Постраничный вывод. *Подсказка ученику: `LIMIT 10 OFFSET 10` — «взять десять строк, пропустив десять». Так устроено разбиение результата на страницы.*

```
sqlite3 shop.db "SELECT id, name FROM products ORDER BY id LIMIT 10 OFFSET 10" >
otvety/44
```

2.45. Третья страница. *Подсказка ученику: Страница 3 по 7 строк: `LIMIT 7 OFFSET (3-1)*7`, то есть `OFFSET 14`.*

```
sqlite3 shop.db "SELECT id, name FROM customers ORDER BY id LIMIT 7 OFFSET 14" >
otvety/45
```

2.46. Самый дорогой товар. *Подсказка ученику: `SELECT price FROM products ORDER BY price DESC LIMIT 1;` (позже то же даст `MAX(price)`).*

```
sqlite3 shop.db "SELECT price FROM products ORDER BY price DESC LIMIT 1" >
otvety/46
```

2.47. Хвост списка. *Подсказка ученику: `SELECT id, name FROM products ORDER BY id LIMIT -1 OFFSET 25;` — `LIMIT -1` означает «без ограничения».*

```
sqlite3 shop.db "SELECT id, name FROM products ORDER BY id LIMIT -1 OFFSET 25" >
otvety/47
```

2.48. Вторая по окладу. *Подсказка ученику: `SELECT name, salary FROM employees ORDER BY salary DESC LIMIT 1 OFFSET 1;`*

```
sqlite3 hr.db "SELECT name, salary FROM employees ORDER BY salary DESC LIMIT 1
OFFSET 1" > otvety/48
```

2.49. Вторая пятёрка. *Подсказка ученику: `ORDER BY price DESC LIMIT 5 OFFSET 5` — пропустить пять самых дорогих, взять следующие пять.*

```
sqlite3 shop.db "SELECT name, price FROM products ORDER BY price DESC LIMIT 5
OFFSET 5" > otvety/49
```

2.50. Три последних дня. *Подсказка ученику: `SELECT DISTINCT order_date FROM orders ORDER BY order_date DESC LIMIT 3;`*

```
sqlite3 shop.db "SELECT DISTINCT order_date FROM orders ORDER BY order_date DESC
LIMIT 3" > otvety/50
```

Тема 3. Условия отбора: WHERE

3.1. Дорогие товары. *Подсказка ученику: `SELECT id, name, price FROM products WHERE price > 5000 ORDER BY id;`*

```
sqlite3 shop.db "SELECT id, name, price FROM products WHERE price > 5000 ORDER BY
id" > otvety/1
```

3.2. Нет на складе. *Подсказка ученику: Равенство в SQL — один знак: `WHERE stock = 0`.*

```
sqlite3 shop.db "SELECT id, name FROM products WHERE stock = 0 ORDER BY id" >
otvety/2
```

3.3. Оплаченные заказы. *Подсказка ученику: `SELECT COUNT(*) FROM orders WHERE status = 'paid';` Текст берётся в одинарные кавычки.*

```
sqlite3 shop.db "SELECT COUNT(*) FROM orders WHERE status = 'paid'" > otvety/3
```

3.4. Не из Москвы. *Подсказка ученику: `WHERE city <> 'Москва'`. Строки, где `city` равен `NULL`, сравнение отбрасывает: «неизвестно» не есть «истина».*

```
sqlite3 shop.db "SELECT id, name, city FROM customers WHERE city <> 'Москва' ORDER
BY id" > otvety/4
```

3.5. Недорогие товары. *Подсказка ученику: `WHERE price <= 3000` — граница 3000 включается.*

```
sqlite3 shop.db "SELECT id, name, price FROM products WHERE price <= 3000 ORDER BY
id" > otvety/5
```

3.6. Заканчивается. *Подсказка ученику: `SELECT id, name, stock FROM products WHERE stock < 10 ORDER BY id;`*

```
sqlite3 shop.db "SELECT id, name, stock FROM products WHERE stock < 10 ORDER BY
id" > otvety/6
```

3.7. Старые клиенты. *Подсказка ученику: Дата — текст в формате ГГГГ-ММ-ДД, потому: `registered < '2024-01-01'`.*

```
sqlite3 shop.db "SELECT id, name, registered FROM customers WHERE registered <
'2024-01-01' ORDER BY id" > otvety/7
```

3.8. Заказы этого года. *Подсказка ученику: Даты в формате ГГГГ-ММ-ДД сравниваются как обычные строки: `order_date >= '2026-01-01'`.*

```
sqlite3 shop.db "SELECT COUNT(*) FROM orders WHERE order_date >= '2026-01-01'" >
otvety/8
```

3.9. Всё, кроме отменённых. *Подсказка ученику: `status <> 'cancelled'` (или `!=`).*

```
sqlite3 shop.db "SELECT COUNT(*) FROM orders WHERE status <> 'cancelled'" >
otvety/9
```

3.10. Три самых дорогих в наличии. *Подсказка ученику: `SELECT name, price FROM products WHERE stock > 0 ORDER BY price DESC, id LIMIT 3;` Порядок записи: `WHERE`, `ORDER BY`, `LIMIT`.*

```
sqlite3 shop.db "SELECT name, price FROM products WHERE stock > 0 ORDER BY price
DESC, id LIMIT 3" > otvety/10
```

3.11. Недорогие книги. *Подсказка ученику: Условия объединяются словом `AND`: `WHERE category = 'Книги' AND price < 8000`. Строки сравниваются с учётом регистра.*

```
sqlite3 shop.db "SELECT name, price FROM products WHERE category = 'Книги' AND
price < 8000 ORDER BY id" > otvety/11
```

3.12. Нет на складе или дорого. *Подсказка ученику: `WHERE stock = 0 OR price > 9000`. Если условий больше двух, расставляйте скобки: `AND` сильнее `OR`.*

```
sqlite3 shop.db "SELECT id FROM products WHERE stock = 0 OR price > 9000 ORDER BY
id" > otvety/12
```

3.13. Мебель на исходе. *Подсказка ученику: `WHERE category = 'Мебель' AND stock < 50`.*

```
sqlite3 shop.db "SELECT id, name, stock FROM products WHERE category = 'Мебель'
AND stock < 50 ORDER BY id" > otvety/13
```

3.14. Отгружено в 2026. *Подсказка ученику: `WHERE status = 'shipped' AND order_date >= '2026-01-01'`.*

```
sqlite3 shop.db "SELECT COUNT(*) FROM orders WHERE status = 'shipped' AND
order_date >= '2026-01-01'" > otvety/14
```

3.15. Книги или продукты. *Подсказка ученику: `WHERE category = 'Книги' OR category = 'Продукты'`.*

```
sqlite3 shop.db "SELECT id, name, category FROM products WHERE category = 'Книги'
OR category = 'Продукты' ORDER BY id" > otvety/15
```

3.16. Кроме электроники. *Подсказка ученику: `SELECT COUNT(*) FROM products WHERE NOT (category = 'Электроника');` То же, что `category <> 'Электроника'`.*

```
sqlite3 shop.db "SELECT COUNT(*) FROM products WHERE NOT (category =
'Электроника')" > otvety/16
```

3.17. Скобки решают. *Подсказка ученику: `WHERE (category = 'Канцтовары' OR category = 'Мебель') AND price < 5000`. Без скобок `AND` связался бы только с `Мебель`.*

```
sqlite3 shop.db "SELECT id, name, category, price FROM products WHERE (category =
'Канцтовары' OR category = 'Мебель') AND price < 5000 ORDER BY id" > otvety/17
```

3.18. Дорогие или недорогие книги. *Подсказка ученику: `WHERE price > 8000 OR (category = 'Книги' AND price < 2000)`. Скобки здесь для наглядности: `AND` и так сильнее `OR`.*

```
sqlite3 shop.db "SELECT id, name, category, price FROM products WHERE price > 8000
OR (category = 'Книги' AND price < 2000) ORDER BY id" > otvety/18
```

3.19. Ранние незавершённые. *Подсказка ученику: `WHERE (status = 'new' OR status = 'paid') AND order_date < '2026-01-01'` — скобки обязательны.*

```
sqlite3 shop.db "SELECT COUNT(*) FROM orders WHERE (status = 'new' OR status =
'paid') AND order_date < '2026-01-01'" > otvety/19
```

3.20. Заказы первых клиентов. *Подсказка ученику: `WHERE customer_id < 6 AND status <> 'cancelled' AND order_date >= '2026-01-01'` — три условия через `AND`.*

```
sqlite3 shop.db "SELECT id, customer_id, order_date FROM orders WHERE customer_id
< 6 AND status <> 'cancelled' AND order_date >= '2026-01-01' ORDER BY id" >
otvety/20
```

3.21. Список городов. *Подсказка ученику: `city IN ('Москва', 'Казань')` короче и надёжнее, чем два сравнения через `OR`.*

```
sqlite3 shop.db "SELECT name FROM customers WHERE city IN ('Москва', 'Казань')
ORDER BY name" > otvety/21
```

3.22. Активные статусы. *Подсказка ученику: `SELECT COUNT(*) FROM orders WHERE status IN ('new', 'paid');`*

```
sqlite3 shop.db "SELECT COUNT(*) FROM orders WHERE status IN ('new', 'paid')" >
otvety/22
```

3.23. Кроме книг и продуктов. *Подсказка ученику: `WHERE category NOT IN ('Книги', 'Продукты')`.*

```
sqlite3 shop.db "SELECT id, name FROM products WHERE category NOT IN ('Книги',
'Продукты') ORDER BY id" > otvety/23
```

3.24. Два поставщика. *Подсказка ученику: `WHERE supplier_id IN (1, 3)`. Числа в списке пишут без кавычек.*

```
sqlite3 shop.db "SELECT id, name, supplier_id FROM products WHERE supplier_id IN
(1, 3) ORDER BY id" > otvety/24
```

3.25. Заказы трёх клиентов. *Подсказка ученику: `SELECT id FROM orders WHERE customer_id IN (1, 2, 3) ORDER BY id;`*

```
sqlite3 shop.db "SELECT id FROM orders WHERE customer_id IN (1, 2, 3) ORDER BY id"
> otvety/25
```

3.26. Не из трёх городов. *Подсказка ученику: `WHERE city NOT IN ('Москва', 'Казань', 'Самара')`. Для NULL условие «неизвестно», поэтому такие клиенты отсеиваются.*

```
sqlite3 shop.db "SELECT id, name, city FROM customers WHERE city NOT IN ('Москва',
'Казань', 'Самара') ORDER BY id" > otvety/26
```

3.27. Особые остатки. *Подсказка ученику: `SELECT COUNT(*) FROM products WHERE stock IN (0, 3, 5);`*

```
sqlite3 shop.db "SELECT COUNT(*) FROM products WHERE stock IN (0, 3, 5)" >
otvety/27
```

3.28. Выборочные клиенты. *Подсказка ученику: `SELECT id, name FROM customers WHERE id IN (3, 7, 11, 15) ORDER BY name;`*

```
sqlite3 shop.db "SELECT id, name FROM customers WHERE id IN (3, 7, 11, 15) ORDER
BY name" > otvety/28
```

3.29. Оплачено и отгружено в 2026. *Подсказка ученику: `WHERE status IN ('paid', 'shipped') AND order_date >= '2026-01-01'`.*

```
sqlite3 shop.db "SELECT COUNT(*) FROM orders WHERE status IN ('paid', 'shipped')
AND order_date >= '2026-01-01'" > otvety/29
```

3.30. Пять дорогих из двух категорий. *Подсказка ученику: `SELECT name, price FROM products WHERE category IN ('Электроника', 'Мебель') ORDER BY price DESC, id LIMIT 5;`*

```
sqlite3 shop.db "SELECT name, price FROM products WHERE category IN
('Электроника', 'Мебель') ORDER BY price DESC, id LIMIT 5" > otvety/30
```

3.31. Диапазон цен. *Подсказка ученику: `BETWEEN 1000 AND 4000` включает обе границы.*

```
sqlite3 shop.db "SELECT COUNT(*) FROM products WHERE price BETWEEN 1000 AND 4000"
> otvety/31
```

3.32. Средние остатки. *Подсказка ученику: `WHERE stock BETWEEN 3 AND 12`.*

```
sqlite3 shop.db "SELECT id, name, stock FROM products WHERE stock BETWEEN 3 AND 12
ORDER BY id" > otvety/32
```

3.33. Заказы 2025 года. *Подсказка ученику: `WHERE order_date BETWEEN '2025-01-01' AND '2025-12-31'` — обе даты включаются.*

```
sqlite3 shop.db "SELECT COUNT(*) FROM orders WHERE order_date BETWEEN '2025-01-01'
AND '2025-12-31'" > otvety/33
```

3.34. Регистрация в 2024. *Подсказка ученику: `WHERE registered BETWEEN '2024-01-01' AND '2024-12-31'`.*

```
sqlite3 shop.db "SELECT id, name, registered FROM customers WHERE registered
BETWEEN '2024-01-01' AND '2024-12-31' ORDER BY id" > otvety/34
```

3.35. Крайние цены. *Подсказка ученику: `WHERE price NOT BETWEEN 500 AND 8000`.*

```
sqlite3 shop.db "SELECT id, name, price FROM products WHERE price NOT BETWEEN 500
AND 8000 ORDER BY id" > otvety/35
```

3.36. Умеренные количества. *Подсказка ученику: `SELECT COUNT(*) FROM order_items WHERE qty BETWEEN 2 AND 3;`*

```
sqlite3 shop.db "SELECT COUNT(*) FROM order_items WHERE qty BETWEEN 2 AND 3" >
otvety/36
```

3.37. Первый квартал 2026. *Подсказка ученику: `WHERE order_date BETWEEN '2026-01-01' AND '2026-03-31'`.*

```
sqlite3 shop.db "SELECT id, order_date FROM orders WHERE order_date BETWEEN
'2026-01-01' AND '2026-03-31' ORDER BY id" > otvety/37
```

3.38. Средняя цена в наличии. *Подсказка ученику: `WHERE price BETWEEN 2000 AND 8000 AND stock > 0`. Первый `AND` принадлежит `BETWEEN`, второй — логическая связка.*

```
sqlite3 shop.db "SELECT id, name, price FROM products WHERE price BETWEEN 2000 AND
8000 AND stock > 0 ORDER BY id" > otvety/38
```

3.39. Крупные позиции. *Подсказка ученику: `SELECT order_id, product_id, qty * price AS total FROM order_items WHERE qty * price BETWEEN 10000 AND 20000 ORDER BY order_id, product_id;` В `WHERE` псевдоним `total` использовать нельзя — повторите выражение.*

```
sqlite3 shop.db "SELECT order_id, product_id, qty * price AS total FROM
order_items WHERE qty * price BETWEEN 10000 AND 20000 ORDER BY order_id,
product_id" > otvety/39
```

3.40. Полугодие с оплатой. *Подсказка ученику: `WHERE order_date BETWEEN '2026-01-01' AND '2026-06-30' AND status IN ('paid', 'shipped')`.*

```
sqlite3 shop.db "SELECT id, customer_id FROM orders WHERE order_date BETWEEN
'2026-01-01' AND '2026-06-30' AND status IN ('paid', 'shipped') ORDER BY id" >
otvety/40
```

3.41. Поиск по шаблону. *Подсказка ученику: В шаблоне `LIKE` символ `%` заменяет любое число символов, `_` — ровно один. Для кириллицы `LIKE` учитывает регистр.*

```
sqlite3 shop.db "SELECT name FROM products WHERE name LIKE 'Книг%' ORDER BY id" >
otvety/41
```

3.42. Двадцатые номера. *Подсказка ученику: `WHERE name LIKE '%-2_'` — `_` заменяет ровно один символ в конце названия.*

```
sqlite3 shop.db "SELECT id, name FROM products WHERE name LIKE '%-2_' ORDER BY id"
> otvety/42
```

3.43. Почта в домене. *Подсказка ученику: `SELECT COUNT(*) FROM customers WHERE email LIKE '%@sever%';` Клиенты с `NULL` в `email` не считаются.*

```
sqlite3 shop.db "SELECT COUNT(*) FROM customers WHERE email LIKE '%@sever%'" >
otvety/43
```

3.44. Фамилия на Б. *Подсказка ученику: `WHERE name LIKE 'Б%'`. Для кириллицы регистр важен: `'б%'` ничего не найдёт.*

```
sqlite3 shop.db "SELECT id, name FROM customers WHERE name LIKE 'Б%' ORDER BY id"
> otvety/44
```

3.45. Однозначные номера. *Подсказка ученику: `WHERE name LIKE 'Книг-_'`. «Книг-14» не подойдёт: после дефиса два символа.*

```
sqlite3 shop.db "SELECT name FROM products WHERE name LIKE 'Книг-_' ORDER BY id" >
otvety/45
```

3.46. Без «-1» в названии. *Подсказка ученику: `SELECT COUNT(*) FROM products WHERE name NOT LIKE '%-1%';`*

```
sqlite3 shop.db "SELECT COUNT(*) FROM products WHERE name NOT LIKE '%-1%'" >
otvety/46
```

3.47. Регистр латиницы. *Подсказка ученику: `SELECT COUNT(*) FROM customers WHERE email LIKE '%@SEVER%';` Результат тот же, что и для `'%@sever%'`.*

```
sqlite3 shop.db "SELECT COUNT(*) FROM customers WHERE email LIKE '%@SEVER%'" >
otvety/47
```

3.48. Город на «а». *Подсказка ученику: `WHERE city LIKE '%а'`. Клиентов с `NULL` в `city` шаблон не найдёт.*

```
sqlite3 shop.db "SELECT id, name, city FROM customers WHERE city LIKE '%а' ORDER
BY id" > otvety/48
```

3.49. Поставщики по названию. *Подсказка ученику: `WHERE name LIKE '%Торг%' OR name LIKE '%Опт%'`.*

```
sqlite3 shop.db "SELECT id, name FROM suppliers WHERE name LIKE '%Торг%' OR name
LIKE '%Опт%' ORDER BY id" > otvety/49
```

3.50. Почта десятых номеров. *Подсказка ученику: `WHERE email LIKE 'user1_@%' AND city IS NOT NULL` — адреса `user10@...` – `user19@...`.*

```
sqlite3 shop.db "SELECT id, name, email FROM customers WHERE email LIKE 'user1_@%'
AND city IS NOT NULL ORDER BY id" > otvety/50
```

3.51. Клиенты без почты. *Подсказка ученику: Сравнение `email = NULL` ничего не вернёт: пустое значение проверяют только так: `email IS NULL`.*

```
sqlite3 shop.db "SELECT name FROM customers WHERE email IS NULL ORDER BY name" >
otvety/51
```

3.52. Город неизвестен. *Подсказка ученику: `SELECT COUNT(*) FROM customers WHERE city IS NULL;`*

```
sqlite3 shop.db "SELECT COUNT(*) FROM customers WHERE city IS NULL" > otvety/52
```

3.53. Почта известна. *Подсказка ученику: `SELECT COUNT(*) FROM customers WHERE email IS NOT NULL;`*

```
sqlite3 shop.db "SELECT COUNT(*) FROM customers WHERE email IS NOT NULL" >
otvety/53
```

3.54. Не из Москвы, но с неизвестными. *Подсказка ученику: `WHERE city <> 'Москва' OR city IS NULL`. В отличие от задания 4, клиенты без города здесь тоже нужны.*

```
sqlite3 shop.db "SELECT id, name, city FROM customers WHERE city <> 'Москва' OR
city IS NULL ORDER BY id" > otvety/54
```

3.55. Есть пробел в анкете. *Подсказка ученику: `WHERE city IS NULL OR email IS NULL`.*

```
sqlite3 shop.db "SELECT id, name FROM customers WHERE city IS NULL OR email IS
NULL ORDER BY id" > otvety/55
```

3.56. Вне трёх городов. *Подсказка ученику: `WHERE city NOT IN ('Москва', 'Казань') OR city IS NULL`.*

```
sqlite3 shop.db "SELECT id, name FROM customers WHERE city NOT IN ('Москва',
'Казань') OR city IS NULL ORDER BY id" > otvety/56
```

3.57. Город есть, почты нет. *Подсказка ученику: `WHERE city IS NOT NULL AND email IS NULL`.*

```
sqlite3 shop.db "SELECT id, name, city FROM customers WHERE city IS NOT NULL AND
email IS NULL ORDER BY id" > otvety/57
```

3.58. Обычные без почты. *Подсказка ученику: `WHERE vip = 0 AND email IS NULL`.*

```
sqlite3 shop.db "SELECT id, name FROM customers WHERE vip = 0 AND email IS NULL
ORDER BY id" > otvety/58
```

3.59. Полные анкеты. *Подсказка ученику: `SELECT COUNT(*) FROM customers WHERE city IS NOT NULL AND email IS NOT NULL;`*

```
sqlite3 shop.db "SELECT COUNT(*) FROM customers WHERE city IS NOT NULL AND email
IS NOT NULL" > otvety/59
```

3.60. Комбинация условий. *Подсказка ученику: `WHERE vip = 1 AND (city IS NOT NULL OR email IS NOT NULL)` — скобки обязательны.*

```
sqlite3 shop.db "SELECT name FROM customers WHERE vip = 1 AND (city IS NOT NULL OR
email IS NOT NULL) ORDER BY name" > otvety/60
```

Тема 4. Функции, выражения CASE и работа с NULL

4.1. Длинные имена. *Подсказка ученику: `SELECT name, LENGTH(name) AS len FROM customers ORDER BY len DESC, id LIMIT 5;` Для текста `LENGTH` считает символы, а не байты.*

```
sqlite3 shop.db "SELECT name, LENGTH(name) AS len FROM customers ORDER BY len
DESC, id LIMIT 5" > otvety/1
```

4.2. Длина почты. *Подсказка ученику: `SELECT id, LENGTH(email) FROM customers WHERE email IS NOT NULL ORDER BY id LIMIT 5;` Условие `email IS NOT NULL` отбирает клиентов с заданной почтой.*

```
sqlite3 shop.db "SELECT id, LENGTH(email) FROM customers WHERE email IS NOT NULL
ORDER BY id LIMIT 5" > otvety/2
```

4.3. Логин из почты. *Подсказка ученику: `SELECT id, SUBSTR(email, 1, INSTR(email, '@') - 1) FROM customers WHERE email IS NOT NULL ORDER BY id LIMIT 5;` `INSTR(email, '@')` возвращает позицию символа, `SUBSTR(email, 1, позиция - 1)` — часть строки до него.*

```
sqlite3 shop.db "SELECT id, SUBSTR(email, 1, INSTR(email, '@') - 1) FROM customers
WHERE email IS NOT NULL ORDER BY id LIMIT 5" > otvety/3
```

4.4. Домен из почты. *Подсказка ученику: SELECT id, SUBSTR(email, INSTR(email, '@') + 1) FROM customers WHERE email IS NOT NULL ORDER BY id LIMIT 5; SUBSTR без третьего аргумента берёт строку до конца.*

```
sqlite3 shop.db "SELECT id, SUBSTR(email, INSTR(email, '@') + 1) FROM customers
WHERE email IS NOT NULL ORDER BY id LIMIT 5" > otvety/4
```

4.5. Фамилия клиента. *Подсказка ученику: SELECT id, SUBSTR(name, 1, INSTR(name, ' ') - 1) FROM customers ORDER BY id LIMIT 8; Позицию пробела даёт INSTR(name, ' ').*

```
sqlite3 shop.db "SELECT id, SUBSTR(name, 1, INSTR(name, ' ') - 1) FROM customers
ORDER BY id LIMIT 8" > otvety/5
```

4.6. Имя без фамилии. *Подсказка ученику: SELECT id, SUBSTR(name, INSTR(name, ' ') + 1) FROM customers ORDER BY id LIMIT 8; Начало подстроки — позиция пробела плюс один.*

```
sqlite3 shop.db "SELECT id, SUBSTR(name, INSTR(name, ' ') + 1) FROM customers
ORDER BY id LIMIT 8" > otvety/6
```

4.7. Номер из названия. *Подсказка ученику: SELECT id, SUBSTR(name, INSTR(name, '-') + 1) FROM products ORDER BY id LIMIT 8; Ищите позицию дефиса: INSTR(name, '-').*

```
sqlite3 shop.db "SELECT id, SUBSTR(name, INSTR(name, '-') + 1) FROM products ORDER
BY id LIMIT 8" > otvety/7
```

4.8. Нет подстроки: 0 и NULL. *Подсказка ученику: SELECT id, INSTR(name, '@'), INSTR(email, '@') FROM customers ORDER BY id LIMIT 6; Если подстроки нет, INSTR даёт 0; если почта NULL, результат тоже NULL.*

```
sqlite3 shop.db "SELECT id, INSTR(name, '@'), INSTR(email, '@') FROM customers
ORDER BY id LIMIT 6" > otvety/8
```

4.9. Длинные фамилии. *Подсказка ученику: SELECT id, SUBSTR(name, 1, INSTR(name, ' ') - 1) AS surname FROM customers WHERE LENGTH(SUBSTR(name, 1, INSTR(name, ' ') - 1)) >= 7 ORDER BY id LIMIT 6; Выражение для фамилии нужно повторить в WHERE: псевдоним surname там ещё не доступен.*

```
sqlite3 shop.db "SELECT id, SUBSTR(name, 1, INSTR(name, ' ') - 1) AS surname FROM
customers WHERE LENGTH(SUBSTR(name, 1, INSTR(name, ' ') - 1)) >= 7 ORDER BY id
LIMIT 6" > otvety/9
```

4.10. Фамилия и инициал. *Подсказка ученику: SELECT id, SUBSTR(name, 1, INSTR(name, ' ') - 1) || ' ' || SUBSTR(name, INSTR(name, ' ') + 1, 1) || '.' FROM customers ORDER BY id LIMIT 6; Первая буква имени: SUBSTR(name, INSTR(name, ' ') + 1, 1).*

```
sqlite3 shop.db "SELECT id, SUBSTR(name, 1, INSTR(name, ' ') - 1) || ' ' ||
SUBSTR(name, INSTR(name, ' ') + 1, 1) || '.' FROM customers ORDER BY id LIMIT 6" >
otvety/10
```

4.11. Смена домена. *Подсказка ученику: SELECT id, REPLACE(email, 'sever.example', 'example.org') FROM customers WHERE email IS NOT NULL ORDER BY id LIMIT 5; REPLACE(строка, что_заменить, на_что).*

```
sqlite3 shop.db "SELECT id, REPLACE(email, 'sever.example', 'example.org') FROM
customers WHERE email IS NOT NULL ORDER BY id LIMIT 5" > otvety/11
```

4.12. Пробелы в подчёркивания. *Подсказка ученику: SELECT id, REPLACE(name, ' ', '_') FROM customers ORDER BY id LIMIT 5; Заменяются все вхождения подстроки.*

```
sqlite3 shop.db "SELECT id, REPLACE(name, ' ', '_') FROM customers ORDER BY id
LIMIT 5" > otvety/12
```

4.13. Почта без домена. *Подсказка ученику: SELECT id, REPLACE(email, '@sever.example', '') FROM customers WHERE email IS NOT NULL ORDER BY id LIMIT 5; Третий аргумент '' — пустая строка: текст просто исчезает.*

```
sqlite3 shop.db "SELECT id, REPLACE(email, '@sever.example', '') FROM customers
WHERE email IS NOT NULL ORDER BY id LIMIT 5" > otvety/13
```

4.14. Верхний регистр. *Подсказка ученику: SELECT id, UPPER(email) FROM customers WHERE email IS NOT NULL ORDER BY id LIMIT 5; UPPER меняет регистр латинских букв.*

```
sqlite3 shop.db "SELECT id, UPPER(email) FROM customers WHERE email IS NOT NULL
ORDER BY id LIMIT 5" > otvety/14
```

4.15. Регистр кириллицы. *Подсказка ученику: SELECT id, category, LOWER(category) FROM products ORDER BY id LIMIT 5; В SQLite LOWER и UPPER не затрагивают кириллицу.*

```
sqlite3 shop.db "SELECT id, category, LOWER(category) FROM products ORDER BY id
LIMIT 5" > otvety/15
```

4.16. Обрезка пробелов. *Подсказка ученику: SELECT id, LENGTH(TRIM(' ' || name || ' ')) FROM customers ORDER BY id LIMIT 5; После TRIM длина равна длине исходного имени.*

```
sqlite3 shop.db "SELECT id, LENGTH(TRIM('  ' || name || '  ')) FROM customers
ORDER BY id LIMIT 5" > otvety/16
```

4.17. Левая обрезка. *Подсказка ученику: `SELECT LENGTH(LTRIM('  Привет  '));` `LTRIM` убирает пробелы слева, `RTRIM` — справа; справа остаются три пробела.*

```
sqlite3 shop.db "SELECT LENGTH(LTRIM('   Привет   '))" > otvety/17
```

4.18. Сколько букв «е». *Подсказка ученику: `SELECT id, LENGTH(email) - LENGTH(REPLACE(email, 'e', '')) FROM customers WHERE email IS NOT NULL ORDER BY id LIMIT 5;` Каждое удалённое вхождение сокращает строку на один символ.*

```
sqlite3 shop.db "SELECT id, LENGTH(email) - LENGTH(REPLACE(email, 'e', '')) FROM
customers WHERE email IS NOT NULL ORDER BY id LIMIT 5" > otvety/18
```

4.19. Логин заглавными. *Подсказка ученику: `SELECT id, UPPER(SUBSTR(email, 1, INSTR(email, '@') - 1)) FROM customers WHERE email IS NOT NULL ORDER BY id LIMIT 5;` Вызовы вкладываются: `UPPER(SUBSTR(...))`.*

```
sqlite3 shop.db "SELECT id, UPPER(SUBSTR(email, 1, INSTR(email, '@') - 1)) FROM
customers WHERE email IS NOT NULL ORDER BY id LIMIT 5" > otvety/19
```

4.20. Скрытая почта. *Подсказка ученику: `SELECT id, REPLACE(REPLACE(email, '@', '[at]'), '.', '[dot]') FROM customers WHERE email IS NOT NULL ORDER BY id LIMIT 5;` Внутренний `REPLACE` выполняется первым.*

```
sqlite3 shop.db "SELECT id, REPLACE(REPLACE(email, '@', '[at]'), '.', '[dot]')
FROM customers WHERE email IS NOT NULL ORDER BY id LIMIT 5" > otvety/20
```

4.21. Округление. *Подсказка ученику: `SELECT id, ROUND(price / 3.0, 2) FROM products ORDER BY id LIMIT 5;` Число с точкой (`3.0`) делает деление вещественным: целое деление `price / 3` отбросило бы дробную часть.*

```
sqlite3 shop.db "SELECT id, ROUND(price / 3.0, 2) FROM products ORDER BY id LIMIT
5" > otvety/21
```

4.22. Целое деление и остаток. *Подсказка ученику: `SELECT id, price / 100, price % 100 FROM products ORDER BY id LIMIT 5;` Оба операнда целые, поэтому `/` даёт целое.*

```
sqlite3 shop.db "SELECT id, price / 100, price % 100 FROM products ORDER BY id
LIMIT 5" > otvety/22
```

4.23. Отклонение остатка. *Подсказка ученику: `SELECT id, stock, ABS(stock - 20) FROM products ORDER BY id LIMIT 5;` `ABS` отбрасывает знак числа.*

```
sqlite3 shop.db "SELECT id, stock, ABS(stock - 20) FROM products ORDER BY id LIMIT
5" > otvety/23
```

4.24. Цена с налогом. *Подсказка ученику: `SELECT id, ROUND(price * 1.2, 1) FROM products ORDER BY id LIMIT 5;` Множитель `1.2` — вещественный, результат дробный.*

```
sqlite3 shop.db "SELECT id, ROUND(price * 1.2, 1) FROM products ORDER BY id LIMIT
5" > otvety/24
```

4.25. Потолок цены. *Подсказка ученику: `SELECT id, MIN(price, 1000), MAX(stock, 5) FROM products ORDER BY id LIMIT 5;` С несколькими аргументами `MIN` и `MAX` сравнивают значения одной строки.*

```
sqlite3 shop.db "SELECT id, MIN(price, 1000), MAX(stock, 5) FROM products ORDER BY
id LIMIT 5" > otvety/25
```

4.26. Цена до тысяч вниз. *Подсказка ученику: `SELECT id, price / 1000 * 1000 FROM products ORDER BY id LIMIT 5;` Целое деление отбрасывает остаток, а умножение возвращает порядок величины.*

```
sqlite3 shop.db "SELECT id, price / 1000 * 1000 FROM products ORDER BY id LIMIT 5"
> otvety/26
```

4.27. Чётные товары. *Подсказка ученику: `SELECT id, name FROM products WHERE id % 2 = 0 ORDER BY id LIMIT 5;` `id % 2 = 0` — условие чётности.*

```
sqlite3 shop.db "SELECT id, name FROM products WHERE id % 2 = 0 ORDER BY id LIMIT
5" > otvety/27
```

4.28. Скидка 10 %. *Подсказка ученику: `SELECT order_id, product_id, ROUND(qty * price * 0.9, 1) FROM order_items ORDER BY order_id, product_id LIMIT 6;` Таблица `order_items` содержит количество и цену в момент покупки.*

```
sqlite3 shop.db "SELECT order_id, product_id, ROUND(qty * price * 0.9, 1) FROM
order_items ORDER BY order_id, product_id LIMIT 6" > otvety/28
```

4.29. Ловушка целого деления. *Подсказка ученику: `SELECT id, price / 7, ROUND(price / 7.0, 2) FROM products ORDER BY id LIMIT 5;` Первое выражение целочисленное и теряет дробную часть.*

```
sqlite3 shop.db "SELECT id, price / 7, ROUND(price / 7.0, 2) FROM products ORDER
BY id LIMIT 5" > otvety/29
```

4.30. Цена в тысячах. *Подсказка ученику: `SELECT name || ': ' || ROUND(price / 1000.0, 1) || ' тыс.' FROM products ORDER BY id LIMIT 5;` Склейка `||` превращает число в текст автоматически.*

```
sqlite3 shop.db "SELECT name || ': ' || ROUND(price / 1000.0, 1) || ' тыс.' FROM
products ORDER BY id LIMIT 5" > otvety/30
```

4.31. Год заказа. *Подсказка ученику: `SELECT id, STRFTIME('%Y', order_date) FROM orders ORDER BY id LIMIT 5;` `'%Y'` — год, `'%m'` — месяц, `'%d'` — день. Дата должна быть в формате ГГГГ-ММ-ДД.*

```
sqlite3 shop.db "SELECT id, STRFTIME('%Y', order_date) FROM orders ORDER BY id
LIMIT 5" > otvety/31
```

4.32. Месяц заказа. *Подсказка ученику: `SELECT id, STRFTIME('%m', order_date) FROM orders ORDER BY id LIMIT 5;` Код формата `'%m'`.*

```
sqlite3 shop.db "SELECT id, STRFTIME('%m', order_date) FROM orders ORDER BY id
LIMIT 5" > otvety/32
```

4.33. Год и месяц. *Подсказка ученику: `SELECT id, STRFTIME('%Y-%m', order_date) FROM orders ORDER BY id LIMIT 5;` В формате можно сочетать коды и обычные символы.*

```
sqlite3 shop.db "SELECT id, STRFTIME('%Y-%m', order_date) FROM orders ORDER BY id
LIMIT 5" > otvety/33
```

4.34. Срок оплаты. *Подсказка ученику: `SELECT id, order_date, DATE(order_date, '+14 days') FROM orders ORDER BY id LIMIT 5;` Допустимы также `'-1 month'`, `'+1 year'`, `'start of month'`.*

```
sqlite3 shop.db "SELECT id, order_date, DATE(order_date, '+14 days') FROM orders
ORDER BY id LIMIT 5" > otvety/34
```

4.35. День недели. *Подсказка ученику: `SELECT id, STRFTIME('%w', order_date) FROM orders ORDER BY id LIMIT 6;` Результат STRFTIME — текст.*

```
sqlite3 shop.db "SELECT id, STRFTIME('%w', order_date) FROM orders ORDER BY id
LIMIT 6" > otvety/35
```

4.36. Начало месяца. *Подсказка ученику: `SELECT id, DATE(order_date, 'start of month') FROM orders ORDER BY id LIMIT 5;` Модификатор `'start of month'`.*

```
sqlite3 shop.db "SELECT id, DATE(order_date, 'start of month') FROM orders ORDER
BY id LIMIT 5" > otvety/36
```

4.37. Конец месяца. *Подсказка ученику: `SELECT id, DATE(order_date, 'start of month', '+1 month', '-1 day') FROM orders ORDER BY id LIMIT 5;` Модификаторы применяются слева направо: начало месяца, плюс месяц, минус день.*

```
sqlite3 shop.db "SELECT id, DATE(order_date, 'start of month', '+1 month', '-1
day') FROM orders ORDER BY id LIMIT 5" > otvety/37
```

4.38. Заказы 2026 года. *Подсказка ученику: `SELECT id, order_date FROM orders WHERE STRFTIME('%Y', order_date) = '2026' ORDER BY id LIMIT 5;` Сравнивайте со строкой `'2026'`, а не с числом.*

```
sqlite3 shop.db "SELECT id, order_date FROM orders WHERE STRFTIME('%Y',
order_date) = '2026' ORDER BY id LIMIT 5" > otvety/38
```

4.39. Возраст заказа. *Подсказка ученику: SELECT id, CAST(JULIANDAY('2026-10-11') - JULIANDAY(order_date) AS INTEGER) FROM orders ORDER BY id LIMIT 5; Разность JULIANDAY — число суток; CAST AS INTEGER отбрасывает дробную часть.*

```
sqlite3 shop.db "SELECT id, CAST(JULIANDAY('2026-10-11') - JULIANDAY(order_date)
AS INTEGER) FROM orders ORDER BY id LIMIT 5" > otvety/39
```

4.40. Сколько дней в истории. *Подсказка ученику: SELECT CAST(JULIANDAY('2026-10-11') - JULIANDAY(order_date) AS INTEGER) FROM orders ORDER BY order_date, id LIMIT 1; JULIANDAY(дата) переводит дату в число суток; разность двух таких чисел — число суток между датами. Самый ранний заказ — первая строка при сортировке по order_date.*

```
sqlite3 shop.db "SELECT CAST(JULIANDAY('2026-10-11') - JULIANDAY(order_date) AS
INTEGER) FROM orders ORDER BY order_date, id LIMIT 1" > otvety/40
```

4.41. Ценовые группы. *Подсказка ученику: SELECT name, CASE WHEN price < 1000 THEN 'дёшево' WHEN price < 5000 THEN 'средне' ELSE 'дорого' END FROM products ORDER BY id LIMIT 10; Условия проверяются по порядку.*

```
sqlite3 shop.db "SELECT name, CASE WHEN price < 1000 THEN 'дёшево' WHEN price <
5000 THEN 'средне' ELSE 'дорого' END FROM products ORDER BY id LIMIT 10" >
otvety/41
```

4.42. Статусы по-русски. *Подсказка ученику: SELECT id, CASE status WHEN 'new' THEN 'Новый' WHEN 'paid' THEN 'Оплачен' WHEN 'shipped' THEN 'Отправлен' ELSE 'Отменён' END FROM orders ORDER BY id LIMIT 6; В простой форме выражение после CASE сравнивается с каждым значением WHEN.*

```
sqlite3 shop.db "SELECT id, CASE status WHEN 'new' THEN 'Новый' WHEN 'paid' THEN
'Оплачен' WHEN 'shipped' THEN 'Отправлен' ELSE 'Отменён' END FROM orders ORDER BY
id LIMIT 6" > otvety/42
```

4.43. Уровень остатка. *Подсказка ученику: SELECT id, CASE WHEN stock = 0 THEN 'нет' WHEN stock < 10 THEN 'мало' ELSE 'много' END FROM products ORDER BY id LIMIT 8; Сначала более узкое условие stock = 0.*

```
sqlite3 shop.db "SELECT id, CASE WHEN stock = 0 THEN 'нет' WHEN stock < 10 THEN
'мало' ELSE 'много' END FROM products ORDER BY id LIMIT 8" > otvety/43
```

4.44. VIP или обычный. *Подсказка ученику: SELECT id, CASE vip WHEN 1 THEN 'VIP' ELSE 'обычный' END FROM customers ORDER BY id LIMIT 6; Простая форма CASE vip WHEN 1 THEN ....*

```
sqlite3 shop.db "SELECT id, CASE vip WHEN 1 THEN 'VIP' ELSE 'обычный' END FROM
customers ORDER BY id LIMIT 6" > otvety/44
```

4.45. CASE без ELSE. *Подсказка ученику: SELECT id, CASE WHEN price < 1000 THEN 'дёшево' END FROM products ORDER BY id LIMIT 6; Без ELSE при невыполненных условиях результат — NULL.*

```
sqlite3 shop.db "SELECT id, CASE WHEN price < 1000 THEN 'дёшево' END FROM products
ORDER BY id LIMIT 6" > otvety/45
```

4.46. Прошлый и текущий год. *Подсказка ученику: SELECT id, order_date, CASE WHEN order_date < '2026-01-01' THEN '2025' ELSE '2026' END FROM orders ORDER BY id LIMIT 6; Даты ГГГГ-ММ-ДД сравниваются как строки правильно.*

```
sqlite3 shop.db "SELECT id, order_date, CASE WHEN order_date < '2026-01-01' THEN
'2025' ELSE '2026' END FROM orders ORDER BY id LIMIT 6" > otvety/46
```

4.47. CASE в ORDER BY. *Подсказка ученику: SELECT id, name, stock FROM products ORDER BY CASE WHEN stock = 0 THEN 0 ELSE 1 END, id LIMIT 6; CASE можно вставлять в ORDER BY как ключ сортировки.*

```
sqlite3 shop.db "SELECT id, name, stock FROM products ORDER BY CASE WHEN stock = 0
THEN 0 ELSE 1 END, id LIMIT 6" > otvety/47
```

4.48. Город или регион. *Подсказка ученику: SELECT id, CASE WHEN city IS NULL THEN 'не указан' WHEN city = 'Москва' THEN 'столица' ELSE 'регион' END FROM customers ORDER BY id LIMIT 8; Сравнение city = 'Москва' с NULL не истинно, поэтому NULL проверяют отдельным условием IS NULL.*

```
sqlite3 shop.db "SELECT id, CASE WHEN city IS NULL THEN 'не указан' WHEN city =
'Москва' THEN 'столица' ELSE 'регион' END FROM customers ORDER BY id LIMIT 8" >
otvety/48
```

4.49. Скидка по количеству. *Подсказка ученику: SELECT order_id, product_id, ROUND(qty * price * CASE WHEN qty >= 4 THEN 0.9 WHEN qty >= 2 THEN 0.95 ELSE 1 END, 2) FROM order_items ORDER BY order_id, product_id LIMIT 6; CASE возвращает число и участвует в арифметике.*

```
sqlite3 shop.db "SELECT order_id, product_id, ROUND(qty * price * CASE WHEN qty >=
4 THEN 0.9 WHEN qty >= 2 THEN 0.95 ELSE 1 END, 2) FROM order_items ORDER BY
order_id, product_id LIMIT 6" > otvety/49
```

4.50. Квартал заказа. *Подсказка ученику: SELECT id, order_date, CASE WHEN STRFTIME('%m', order_date) <= '03' THEN 'I' WHEN STRFTIME('%m', order_date) <= '06' THEN 'II' WHEN STRFTIME('%m', order_date) <= '09' THEN 'III' ELSE 'IV' END FROM orders ORDER BY id LIMIT 6; Месяц — текст из двух цифр, поэтому сравнивайте со строками.*

```
sqlite3 shop.db "SELECT id, order_date, CASE WHEN STRFTIME('%m', order_date) <=
'03' THEN 'I' WHEN STRFTIME('%m', order_date) <= '06' THEN 'II' WHEN
STRFTIME('%m', order_date) <= '09' THEN 'III' ELSE 'IV' END FROM orders ORDER BY
id LIMIT 6" > otvety/50
```

4.51. Почта или «нет». *Подсказка ученику: SELECT id, COALESCE(email, 'нет') FROM customers ORDER BY id LIMIT 5; COALESCE(a, b) возвращает первое значение, не равное NULL.*

```
sqlite3 shop.db "SELECT id, COALESCE(email, 'нет') FROM customers ORDER BY id
LIMIT 5" > otvety/51
```

4.52. Город или «не указан». *Подсказка ученику: SELECT id, name, COALESCE(city, 'не указан') FROM customers ORDER BY id LIMIT 8; Подставляемое значение — второй аргумент.*

```
sqlite3 shop.db "SELECT id, name, COALESCE(city, 'не указан') FROM customers ORDER
BY id LIMIT 8" > otvety/52
```

4.53. IFNULL. *Подсказка ученику: SELECT id, IFNULL(city, '-') FROM customers ORDER BY id LIMIT 8; IFNULL(a, b) — двухаргументный вариант COALESCE.*

```
sqlite3 shop.db "SELECT id, IFNULL(city, '-') FROM customers ORDER BY id LIMIT 8"
> otvety/53
```

4.54. NULL в склейке. *Подсказка ученику: SELECT id, name || ' (' || city || ')' FROM customers ORDER BY id LIMIT 8; Склейка с NULL даёт NULL.*

```
sqlite3 shop.db "SELECT id, name || ' (' || city || ')' FROM customers ORDER BY id
LIMIT 8" > otvety/54
```

4.55. Склейка без потерь. *Подсказка ученику: SELECT id, name || ' (' || COALESCE(city, 'город не указан') || ')' FROM customers ORDER BY id LIMIT 8; Оберните city в COALESCE.*

```
sqlite3 shop.db "SELECT id, name || ' (' || COALESCE(city, 'город не указан') ||
')' FROM customers ORDER BY id LIMIT 8" > otvety/55
```

4.56. Цепочка COALESCE. *Подсказка ученику: SELECT id, COALESCE(email, city, 'нет данных') FROM customers ORDER BY id LIMIT 8; COALESCE принимает любое число аргументов.*

```
sqlite3 shop.db "SELECT id, COALESCE(email, city, 'нет данных') FROM customers
ORDER BY id LIMIT 8" > otvety/56
```

4.57. Деление без ошибки. *Подсказка ученику: SELECT id, 1000 / NULLIF(stock, 0) FROM products ORDER BY id LIMIT 8; NULLIF(stock, 0) превращает ноль в NULL, а деление на NULL даёт NULL.*

```
sqlite3 shop.db "SELECT id, 1000 / NULLIF(stock, 0) FROM products ORDER BY id
LIMIT 8" > otvety/57
```

4.58. Скрыть отменённые. *Подсказка ученику: SELECT id, NULLIF(status, 'cancelled') FROM orders ORDER BY id LIMIT 8; NULLIF(a, b) возвращает NULL, если a = b.*

```
sqlite3 shop.db "SELECT id, NULLIF(status, 'cancelled') FROM orders ORDER BY id
LIMIT 8" > otvety/58
```

4.59. Не москвичи. *Подсказка ученику: SELECT id, name, city FROM customers WHERE COALESCE(city, '') <> 'Москва' ORDER BY id LIMIT 8; Условие city <> 'Москва' для NULL не истинно и такие строки потерялись бы.*

```
sqlite3 shop.db "SELECT id, name, city FROM customers WHERE COALESCE(city, '') <>
'Москва' ORDER BY id LIMIT 8" > otvety/59
```

4.60. Цена на единицу остатка. *Подсказка ученику: SELECT id, COALESCE(ROUND(price * 1.0 / NULLIF(stock, 0), 2), 0) FROM products ORDER BY id LIMIT 8; Сочетание NULLIF внутри и COALESCE снаружи.*

```
sqlite3 shop.db "SELECT id, COALESCE(ROUND(price * 1.0 / NULLIF(stock, 0), 2), 0)
FROM products ORDER BY id LIMIT 8" > otvety/60
```

4.61. Типы значений. *Подсказка ученику: SELECT TYPEOF(1), TYPEOF(1.5), TYPEOF('a'), TYPEOF(NULL); SQLite различает integer, real, text, blob и null.*

```
sqlite3 shop.db "SELECT TYPEOF(1), TYPEOF(1.5), TYPEOF('a'), TYPEOF(NULL)" >
otvety/61
```

4.62. Типы выражений. *Подсказка ученику: SELECT TYPEOF(price), TYPEOF(price / 2), TYPEOF(price / 2.0), TYPEOF(name) FROM products WHERE id = 1; Результат арифметики зависит от типов операндов.*

```
sqlite3 shop.db "SELECT TYPEOF(price), TYPEOF(price / 2), TYPEOF(price / 2.0),
TYPEOF(name) FROM products WHERE id = 1" > otvety/62
```

4.63. Текст в число. *Подсказка ученику: SELECT CAST('42' AS INTEGER) + 1; CAST превращает текст '42' в целое число.*

```
sqlite3 shop.db "SELECT CAST('42' AS INTEGER) + 1" > otvety/63
```

4.64. Отбрасывание дроби. *Подсказка ученику: SELECT CAST(3.9 AS INTEGER), CAST(-3.9 AS INTEGER); CAST не округляет, а отбрасывает дробную часть (к нулю).*

```
sqlite3 shop.db "SELECT CAST(3.9 AS INTEGER), CAST(-3.9 AS INTEGER)" > otvety/64
```

4.65. Текст не число. *Подсказка ученику: SELECT CAST('abc' AS INTEGER), CAST('12abc' AS INTEGER); Берётся числовое начало строки; если его нет — 0, без ошибки.*

```
sqlite3 shop.db "SELECT CAST('abc' AS INTEGER), CAST('12abc' AS INTEGER)" >
otvety/65
```

4.66. Число в текст. *Подсказка ученику: SELECT id, CAST(price AS TEXT) || ' руб.' FROM products ORDER BY id LIMIT 5; CAST(... AS TEXT) делает значение текстовым.*

```
sqlite3 shop.db "SELECT id, CAST(price AS TEXT) || ' руб.' FROM products ORDER BY
id LIMIT 5" > otvety/66
```

4.67. Доля остатка. *Подсказка ученику: SELECT id, ROUND(CAST(stock AS REAL) / price * 100, 2) FROM products ORDER BY id LIMIT 5; CAST AS REAL делает деление вещественным.*

```
sqlite3 shop.db "SELECT id, ROUND(CAST(stock AS REAL) / price * 100, 2) FROM
products ORDER BY id LIMIT 5" > otvety/67
```

4.68. Номер как число. *Подсказка ученику: SELECT id, TYPEOF(SUBSTR(name, INSTR(name, '-') + 1)), TYPEOF(CAST(SUBSTR(name, INSTR(name, '-') + 1) AS INTEGER)) FROM products ORDER BY id LIMIT 5; SUBSTR всегда возвращает текст.*

```
sqlite3 shop.db "SELECT id, TYPEOF(SUBSTR(name, INSTR(name, '-') + 1)),
TYPEOF(CAST(SUBSTR(name, INSTR(name, '-') + 1) AS INTEGER)) FROM products ORDER BY
id LIMIT 5" > otvety/68
```

4.69. Текст против числа. *Подсказка ученику: SELECT '10' < '9', CAST('10' AS INTEGER) < CAST('9' AS INTEGER); Строки сравниваются посимвольно: '1' меньше '9', поэтому первое сравнение истинно.*

```
sqlite3 shop.db "SELECT '10' < '9', CAST('10' AS INTEGER) < CAST('9' AS INTEGER)"
> otvety/69
```

4.70. Следующий год. *Подсказка ученику: SELECT id, TYPEOF(STRFTIME('%Y', order_date)), CAST(STRFTIME('%Y', order_date) AS INTEGER) + 1 FROM orders ORDER BY id LIMIT 5; Год из STRFTIME — текст, для арифметики его приводят к INTEGER.*

```
sqlite3 shop.db "SELECT id, TYPEOF(STRFTIME('%Y', order_date)),
CAST(STRFTIME('%Y', order_date) AS INTEGER) + 1 FROM orders ORDER BY id LIMIT 5" >
otvety/70
```

Тема 5. Агрегатные функции и группировка

5.1. Всего клиентов. *Подсказка ученику: `SELECT COUNT(*) FROM customers;`*

```
sqlite3 shop.db "SELECT COUNT(*) FROM customers" > otvety/1
```

5.2. Товары в наличии. *Подсказка ученику: `WHERE` отбирает строки до подсчёта: `SELECT COUNT(*) FROM products WHERE stock > 0;`*

```
sqlite3 shop.db "SELECT COUNT(*) FROM products WHERE stock > 0" > otvety/2
```

5.3. Клиенты с городом. *Подсказка ученику: `COUNT(city)` не считает `NULL`: `SELECT COUNT(city) FROM customers;`*

```
sqlite3 shop.db "SELECT COUNT(city) FROM customers" > otvety/3
```

5.4. Клиенты без email. *Подсказка ученику: `SELECT COUNT(*) - COUNT(email) FROM customers;` — так же можно `COUNT(*) ... WHERE email IS NULL`.*

```
sqlite3 shop.db "SELECT COUNT(*) - COUNT(email) FROM customers" > otvety/4
```

5.5. Разные города. *Подсказка ученику: `SELECT COUNT(DISTINCT city) FROM customers;` — `NULL` не считается значением.*

```
sqlite3 shop.db "SELECT COUNT(DISTINCT city) FROM customers" > otvety/5
```

5.6. Сколько покупателей. *Подсказка ученику: `COUNT(DISTINCT столбец)` считает только различные значения.*

```
sqlite3 shop.db "SELECT COUNT(DISTINCT customer_id) FROM orders" > otvety/6
```

5.7. Отправленные в 2025. *Подсказка ученику: `SELECT COUNT(*) FROM orders WHERE status = 'shipped' AND order_date LIKE '2025%';`*

```
sqlite3 shop.db "SELECT COUNT(*) FROM orders WHERE status = 'shipped' AND
order_date LIKE '2025%'" > otvety/7
```

5.8. Заказанные товары. *Подсказка ученику: `SELECT COUNT(DISTINCT product_id) FROM order_items;`*

```
sqlite3 shop.db "SELECT COUNT(DISTINCT product_id) FROM order_items" > otvety/8
```

5.9. Три способа подсчёта. *Подсказка ученику: `SELECT COUNT(*), COUNT(city), COUNT(DISTINCT city) FROM customers;` — первое число больше второго из-за `NULL`, второе больше третьего из-за повторов.*

```
sqlite3 shop.db "SELECT COUNT(*), COUNT(city), COUNT(DISTINCT city) FROM
customers" > otvety/9
```

5.10. Полностью заполненные. *Подсказка ученику: `WHERE registered >= '2024-01-01' AND city IS NOT NULL AND email IS NOT NULL`, затем `COUNT(*)`.*

```
sqlite3 shop.db "SELECT COUNT(*) FROM customers WHERE registered >= '2024-01-01'
AND city IS NOT NULL AND email IS NOT NULL" > otvety/10
```

5.11. Всего на складе. *Подсказка ученику: `SELECT SUM(stock) FROM products;`*

```
sqlite3 shop.db "SELECT SUM(stock) FROM products" > otvety/11
```

5.12. Средняя цена. *Подсказка ученику: `AVG` усредняет, `ROUND(значение, 2)` округляет до сотых.*

```
sqlite3 shop.db "SELECT ROUND(AVG(price), 2) FROM products" > otvety/12
```

5.13. Продано штук. *Подсказка ученику: `SELECT SUM(qty) FROM order_items;`*

```
sqlite3 shop.db "SELECT SUM(qty) FROM order_items" > otvety/13
```

5.14. Выручка. *Подсказка ученику: Агрегат можно считать от выражения: `SELECT SUM(qty * price) FROM order_items;`*

```
sqlite3 shop.db "SELECT SUM(qty * price) FROM order_items" > otvety/14
```

5.15. Цена электроники. *Подсказка ученику: `SELECT ROUND(AVG(price), 2) FROM products WHERE category = 'Электроника';`*

```
sqlite3 shop.db "SELECT ROUND(AVG(price), 2) FROM products WHERE category =
'Электроника'" > otvety/15
```

5.16. Остатки дорогих товаров. *Подсказка ученику: `SELECT SUM(price * stock) FROM products WHERE price > 1000;`*

```
sqlite3 shop.db "SELECT SUM(price * stock) FROM products WHERE price > 1000" >
otvety/16
```

5.17. Доля VIP. *Подсказка ученику: Флаг `vip` равен 0 или 1, поэтому `SUM(vip)` — число VIP: `SELECT SUM(vip), COUNT(*), ROUND(100.0 * SUM(vip) / COUNT(*), 1) FROM customers;`*

```
sqlite3 shop.db "SELECT SUM(vip), COUNT(*), ROUND(100.0 * SUM(vip) / COUNT(*), 1)
FROM customers" > otvety/17
```

5.18. Фонд оплаты труда. *Подсказка ученику: `sqlite3 hr.db`, затем `SELECT SUM(salary) FROM employees;`*

```
sqlite3 hr.db "SELECT SUM(salary) FROM employees" > otvety/18
```

5.19. Среднее и NULL. *Подсказка ученику: `SELECT ROUND(AVG(dept_id), 3), ROUND(AVG(COALESCE(dept_id, 0)), 3) FROM employees;` — первое среднее не учитывает NULL, второе считает их нулями.*

```
sqlite3 hr.db "SELECT ROUND(AVG(dept_id), 3), ROUND(AVG(COALESCE(dept_id, 0)), 3)
FROM employees" > otvety/19
```

5.20. Зарплата новичков. *Подсказка ученику: `SELECT ROUND(AVG(salary), 0) FROM employees WHERE hired >= '2022-01-01' AND position <> 'директор';`*

```
sqlite3 hr.db "SELECT ROUND(AVG(salary), 0) FROM employees WHERE hired >=
'2022-01-01' AND position <> 'директор'" > otvety/20
```

5.21. Минимум и максимум. *Подсказка ученику: `SELECT MIN(price), MAX(price) FROM products;`*

```
sqlite3 shop.db "SELECT MIN(price), MAX(price) FROM products" > otvety/21
```

5.22. Даты регистрации. *Подсказка ученику: `MIN` и `MAX` работают и с датами в формате ГГГГ-ММ-ДД.*

```
sqlite3 shop.db "SELECT MIN(registered), MAX(registered) FROM customers" >
otvety/22
```

5.23. Самая дорогая мебель. *Подсказка ученику: `SELECT MAX(price) FROM products WHERE category = 'Мебель';`*

```
sqlite3 shop.db "SELECT MAX(price) FROM products WHERE category = 'Мебель'" >
otvety/23
```

5.24. Последняя отправка. *Подсказка ученику: `SELECT MAX(order_date) FROM orders WHERE status = 'shipped';`*

```
sqlite3 shop.db "SELECT MAX(order_date) FROM orders WHERE status = 'shipped'" >
otvety/24
```

5.25. Первое и последнее имя. *Подсказка ученику: `MIN` и `MAX` сравнивают текст по алфавиту: `SELECT MIN(name), MAX(name) FROM customers;`*

```
sqlite3 shop.db "SELECT MIN(name), MAX(name) FROM customers" > otvety/25
```

5.26. Вилка зарплат отдела. *Подсказка ученику: `SELECT MIN(salary), MAX(salary) FROM employees WHERE dept_id = 2;`*

```
sqlite3 hr.db "SELECT MIN(salary), MAX(salary) FROM employees WHERE dept_id = 2" >
otvety/26
```

5.27. Поставщики одной строкой. *Подсказка ученику: `SELECT GROUP_CONCAT(name, ' / ') FROM suppliers;`*

```
sqlite3 shop.db "SELECT GROUP_CONCAT(name, ' / ') FROM suppliers" > otvety/27
```

5.28. Московские отделы. *Подсказка ученику: `SELECT GROUP_CONCAT(name, ' / ') FROM departments WHERE city = 'Москва';`*

```
sqlite3 hr.db "SELECT GROUP_CONCAT(name, ' / ') FROM departments WHERE city =
'Москва'" > otvety/28
```

5.29. Самый дорогой товар. *Подсказка ученику: В SQLite обычный столбец рядом с MAX берётся из той же строки: `SELECT name, MAX(price) FROM products;`*

```
sqlite3 shop.db "SELECT name, MAX(price) FROM products" > otvety/29
```

5.30. Самый дешёвый в наличии. *Подсказка ученику: `SELECT name, MIN(price) FROM products WHERE stock > 0;` — WHERE отбирает строки до поиска минимума.*

```
sqlite3 shop.db "SELECT name, MIN(price) FROM products WHERE stock > 0" >
otvety/30
```

5.31. Товары по категориям. *Подсказка ученику: `SELECT category, COUNT(*) FROM products GROUP BY category ORDER BY category;`*

```
sqlite3 shop.db "SELECT category, COUNT(*) FROM products GROUP BY category ORDER
BY category" > otvety/31
```

5.32. Заказы по статусам. *Подсказка ученику: `ORDER BY n DESC, status` — псевдоним агрегата можно использовать в `ORDER BY`.*

```
sqlite3 shop.db "SELECT status, COUNT(*) AS n FROM orders GROUP BY status ORDER BY
n DESC, status" > otvety/32
```

5.33. Клиенты по городам. *Подсказка ученику: `SELECT city, COUNT(*) FROM customers GROUP BY city ORDER BY city;` — NULL образует свою группу и при сортировке идёт первым.*

```
sqlite3 shop.db "SELECT city, COUNT(*) FROM customers GROUP BY city ORDER BY city"
> otvety/33
```

5.34. Стоимость остатков по категориям. *Подсказка ученику: Агрегат можно считать от выражения: `SUM(price * stock)`.*

```
sqlite3 shop.db "SELECT category, SUM(price * stock) AS v FROM products GROUP BY
category ORDER BY v DESC, category" > otvety/34
```

5.35. Товары по поставщикам. *Подсказка ученику: `SELECT supplier_id, COUNT(*), ROUND(AVG(price), 2) FROM products GROUP BY supplier_id ORDER BY supplier_id;`*

```
sqlite3 shop.db "SELECT supplier_id, COUNT(*), ROUND(AVG(price), 2) FROM products
GROUP BY supplier_id ORDER BY supplier_id" > otvety/35
```

5.36. Заказы по месяцам. *Подсказка ученику: Группировать можно по выражению или по псевдониму: `GROUP BY ym`.*

```
sqlite3 shop.db "SELECT STRFTIME('%Y-%m', order_date) AS ym, COUNT(*) FROM orders
GROUP BY ym ORDER BY ym" > otvety/36
```

5.37. Поставщик и категория. *Подсказка ученику: `GROUP BY supplier_id, category` — группа определяется парой значений.*

```
sqlite3 shop.db "SELECT supplier_id, category, COUNT(*) FROM products GROUP BY
supplier_id, category ORDER BY supplier_id, category" > otvety/37
```

5.38. Заказы 2026 без отмен. *Подсказка ученику: Условие на строки — в `WHERE`, до `GROUP BY`: `WHERE order_date >= '2026-01-01' AND status <> 'cancelled' GROUP BY status`.*

```
sqlite3 shop.db "SELECT status, COUNT(*) FROM orders WHERE order_date >=
'2026-01-01' AND status <> 'cancelled' GROUP BY status ORDER BY status" >
otvety/38
```

5.39. Регистрации по годам. *Подсказка ученику: `SELECT STRFTIME('%Y', registered) AS y, COUNT(*), SUM(vip) FROM customers GROUP BY y ORDER BY y;`*

```
sqlite3 shop.db "SELECT STRFTIME('%Y', registered) AS y, COUNT(*), SUM(vip) FROM
customers GROUP BY y ORDER BY y" > otvety/39
```

5.40. Пять самых ходовых товаров. *Подсказка ученику: `GROUP BY product_id ORDER BY total DESC, product_id LIMIT 5` — `LIMIT` выполняется последним.*

```
sqlite3 shop.db "SELECT product_id, SUM(qty) AS total FROM order_items GROUP BY
product_id ORDER BY total DESC, product_id LIMIT 5" > otvety/40
```

5.41. Активные клиенты. *Подсказка ученику: Условие на результат агрегата пишут в `HAVING`: `GROUP BY customer_id HAVING COUNT(*) >= 3`.*

```
sqlite3 shop.db "SELECT customer_id, COUNT(*) FROM orders GROUP BY customer_id
HAVING COUNT(*) >= 3 ORDER BY customer_id" > otvety/41
```

5.42. Дорогие категории. *Подсказка ученику: `GROUP BY category HAVING AVG(price) > 3000`; `ROUND` применяется только к выводимому значению.*

```
sqlite3 shop.db "SELECT category, ROUND(AVG(price), 2) FROM products GROUP BY
category HAVING AVG(price) > 3000 ORDER BY category" > otvety/42
```

5.43. Разброс цен. *Подсказка ученику: HAVING может использовать псевдоним агрегата из списка столбцов: HAVING spread > 3000.*

```
sqlite3 shop.db "SELECT category, MAX(price) - MIN(price) AS spread FROM products
GROUP BY category HAVING spread > 3000 ORDER BY category" > otvety/43
```

5.44. Частые статусы. *Подсказка ученику: SELECT status, COUNT(*) FROM orders GROUP BY status HAVING COUNT(*) > 20 ORDER BY status;*

```
sqlite3 shop.db "SELECT status, COUNT(*) FROM orders GROUP BY status HAVING
COUNT(*) > 20 ORDER BY status" > otvety/44
```

5.45. Многолюдные города. *Подсказка ученику: Условие на столбец — в WHERE (city IS NOT NULL), условие на число — в HAVING COUNT(*) >= 4.*

```
sqlite3 shop.db "SELECT city, COUNT(*) FROM customers WHERE city IS NOT NULL GROUP
BY city HAVING COUNT(*) >= 4 ORDER BY city" > otvety/45
```

5.46. Поставщики с запасом. *Подсказка ученику: WHERE stock > 0 отбирает строки, HAVING COUNT(*) >= 3 — группы.*

```
sqlite3 shop.db "SELECT supplier_id, COUNT(*) FROM products WHERE stock > 0 GROUP
BY supplier_id HAVING COUNT(*) >= 3 ORDER BY supplier_id" > otvety/46
```

5.47. Ходовые товары. *Подсказка ученику: GROUP BY product_id HAVING SUM(qty) >= 30.*

```
sqlite3 shop.db "SELECT product_id, SUM(qty) AS total FROM order_items GROUP BY
product_id HAVING total >= 30 ORDER BY product_id" > otvety/47
```

5.48. Постоянные получатели. *Подсказка ученику: WHERE status = 'shipped' GROUP BY customer_id HAVING COUNT(*) >= 2.*

```
sqlite3 shop.db "SELECT customer_id, COUNT(*) FROM orders WHERE status = 'shipped'
GROUP BY customer_id HAVING COUNT(*) >= 2 ORDER BY customer_id" > otvety/48
```

5.49. Оживлённые месяцы 2025. *Подсказка ученику: WHERE order_date < '2026-01-01' GROUP BY ym HAVING COUNT(*) >= 5 ORDER BY ym.*

```
sqlite3 shop.db "SELECT STRFTIME('%Y-%m', order_date) AS ym, COUNT(*) FROM orders
WHERE order_date < '2026-01-01' GROUP BY ym HAVING COUNT(*) >= 5 ORDER BY ym" >
otvety/49
```

5.50. Крупные отделы. *Подсказка ученику: WHERE dept_id IS NOT NULL GROUP BY dept_id HAVING COUNT(*) >= 3 ORDER BY dept_id на базе hr.db.*

```
sqlite3 hr.db "SELECT dept_id, COUNT(*), ROUND(AVG(salary), 0) FROM employees
WHERE dept_id IS NOT NULL GROUP BY dept_id HAVING COUNT(*) >= 3 ORDER BY dept_id"
> otvety/50
```

Тема 6. Соединение таблиц: JOIN

6.1. Сколько таблиц. *Подсказка ученику: SELECT COUNT(*) FROM sqlite_master WHERE type = 'table';*

```
sqlite3 shop.db "SELECT COUNT(*) FROM sqlite_master WHERE type = 'table'" >
otvety/1
```

6.2. Список таблиц. *Подсказка ученику: SELECT name FROM sqlite_master WHERE type = 'table' ORDER BY name;*

```
sqlite3 shop.db "SELECT name FROM sqlite_master WHERE type = 'table' ORDER BY
name" > otvety/2
```

6.3. Столбцы заказов. *Подсказка ученику: SELECT name FROM pragma_table_info('orders') ORDER BY cid;*

```
sqlite3 shop.db "SELECT name FROM pragma_table_info('orders') ORDER BY cid" >
otvety/3
```

6.4. Составной ключ. *Подсказка ученику: ... FROM pragma_table_info('order_items') WHERE pk > 0 ORDER BY pk;*

```
sqlite3 shop.db "SELECT name FROM pragma_table_info('order_items') WHERE pk > 0
ORDER BY pk" > otvety/4
```

6.5. Ссылки позиций заказа. *Подсказка ученику: SELECT "from", "table", "to" FROM pragma_foreign_key_list('order_items') ORDER BY "from"; — имена полей берут в кавычки (или квадратные скобки).*

```
sqlite3 shop.db "SELECT [from], [table], [to] FROM
pragma_foreign_key_list('order_items') ORDER BY [from]" > otvety/5
```

6.6. Ссылки сотрудников. *Подсказка ученику: SELECT "from", "table" FROM pragma_foreign_key_list('employees') ORDER BY "from"; — одна из ссылок указывает на саму таблицу employees.*

```
sqlite3 hr.db "SELECT [from], [table] FROM pragma_foreign_key_list('employees')
ORDER BY [from]" > otvety/6
```

6.7. Сотрудники с отделом. *Подсказка ученику: SELECT COUNT(dept_id) FROM employees; — COUNT(*) посчитал бы и сотрудников без отдела.*

```
sqlite3 hr.db "SELECT COUNT(dept_id) FROM employees" > otvety/7
```

6.8. Сколько столбцов. *Подсказка ученику: SELECT COUNT(*) FROM pragma_table_info('products'); — функция используется в FROM как таблица.*

```
sqlite3 shop.db "SELECT COUNT(*) FROM pragma_table_info('products')" > otvety/8
```

6.9. Обязательные столбцы. *Подсказка ученику: SELECT name, type FROM pragma_table_info('customers') WHERE [notnull] = 1 ORDER BY cid;*

```
sqlite3 shop.db "SELECT name, type FROM pragma_table_info('customers') WHERE
[notnull] = 1 ORDER BY cid" > otvety/9
```

6.10. Директор. *Подсказка ученику: SELECT name, position FROM employees WHERE manager_id IS NULL; — у директора ссылка пуста.*

```
sqlite3 hr.db "SELECT name, position FROM employees WHERE manager_id IS NULL" >
otvety/10
```

6.11. Заказ и клиент. *Подсказка ученику: FROM orders o JOIN customers c ON c.id = o.customer_id — после ON записывают условие связи; o и c — короткие псевдонимы таблиц.*

```
sqlite3 shop.db "SELECT o.id, c.name FROM orders o JOIN customers c ON c.id =
o.customer_id ORDER BY o.id LIMIT 10" > otvety/11
```

6.12. Товар и поставщик. *Подсказка ученику: Связь: products.supplier_id ссылается на suppliers.id.*

```
sqlite3 shop.db "SELECT p.name, s.name FROM products p JOIN suppliers s ON s.id =
p.supplier_id ORDER BY p.id LIMIT 10" > otvety/12
```

6.13. Сотрудник и отдел. *Подсказка ученику: FROM employees e JOIN departments d ON d.id = e.dept_id ORDER BY e.id LIMIT 10*

```
sqlite3 hr.db "SELECT e.name, d.name FROM employees e JOIN departments d ON d.id =
e.dept_id ORDER BY e.id LIMIT 10" > otvety/13
```

6.14. Состав первого заказа. *Подсказка ученику: FROM order_items i JOIN products p ON p.id = i.product_id WHERE i.order_id = 1 ORDER BY p.id*

```
sqlite3 shop.db "SELECT p.name, i.qty FROM order_items i JOIN products p ON p.id =
i.product_id WHERE i.order_id = 1 ORDER BY p.id" > otvety/14
```

6.15. Оплаченные заказы. *Подсказка ученику: Условие по статусу — в WHERE после соединения: WHERE o.status = 'paid' ORDER BY o.id LIMIT 8.*

```
sqlite3 shop.db "SELECT o.id, c.name, o.order_date FROM orders o JOIN customers c
ON c.id = o.customer_id WHERE o.status = 'paid' ORDER BY o.id LIMIT 8" > otvety/15
```

6.16. Российские товары. *Подсказка ученику: Страна лежит в таблице поставщиков: WHERE s.country = 'Россия'.*

```
sqlite3 shop.db "SELECT p.name, s.name FROM products p JOIN suppliers s ON s.id =
p.supplier_id WHERE s.country = 'Россия' ORDER BY p.id LIMIT 10" > otvety/16
```

6.17. Московские отделы. *Подсказка ученику: `WHERE d.city = 'Москва' ORDER BY e.id LIMIT 12` — город берётся из `departments`, а не из `employees`.*

```
sqlite3 hr.db "SELECT e.name, d.name FROM employees e JOIN departments d ON d.id =
e.dept_id WHERE d.city = 'Москва' ORDER BY e.id LIMIT 12" > otvety/17
```

6.18. Дорогие позиции. *Подсказка ученику: `SELECT i.order_id, p.name, i.qty * i.price AS line ... ORDER BY line DESC, i.order_id, p.id LIMIT 5`*

```
sqlite3 shop.db "SELECT i.order_id, p.name, i.qty * i.price AS line FROM
order_items i JOIN products p ON p.id = i.product_id ORDER BY line DESC,
i.order_id, p.id LIMIT 5" > otvety/18
```

6.19. Позиции отменённых. *Подсказка ученику: `SELECT COUNT(*) FROM orders o JOIN order_items i ON i.order_id = o.id WHERE o.status = 'cancelled';` — после соединения считаются строки позиций.*

```
sqlite3 shop.db "SELECT COUNT(*) FROM orders o JOIN order_items i ON i.order_id =
o.id WHERE o.status = 'cancelled'" > otvety/19
```

6.20. Зарплата по отделам. *Подсказка ученику: `SELECT d.name, ROUND(AVG(e.salary), 0) FROM employees e JOIN departments d ON d.id = e.dept_id GROUP BY d.id ORDER BY d.id`*

```
sqlite3 hr.db "SELECT d.name, ROUND(AVG(e.salary), 0) FROM employees e JOIN
departments d ON d.id = e.dept_id GROUP BY d.id ORDER BY d.id" > otvety/20
```

6.21. Состав первых заказов. *Подсказка ученику: `FROM orders o JOIN order_items i ON i.order_id = o.id JOIN products p ON p.id = i.product_id WHERE o.id <= 3`*

```
sqlite3 shop.db "SELECT o.id, p.name, i.qty FROM orders o JOIN order_items i ON
i.order_id = o.id JOIN products p ON p.id = i.product_id WHERE o.id <= 3 ORDER BY
o.id, p.id" > otvety/21
```

6.22. Товары и поставщики заказа. *Подсказка ученику: `FROM order_items i JOIN products p ON p.id = i.product_id JOIN suppliers s ON s.id = p.supplier_id WHERE i.order_id = 1`*

```
sqlite3 shop.db "SELECT p.name, s.name FROM order_items i JOIN products p ON p.id
= i.product_id JOIN suppliers s ON s.id = p.supplier_id WHERE i.order_id = 1 ORDER
BY p.id" > otvety/22
```

6.23. Кто что купил. *Подсказка ученику: Цепочка из четырёх таблиц, каждая со своим `ON`; условие `WHERE o.id = 5`.*

```
sqlite3 shop.db "SELECT c.name, p.name, i.qty FROM customers c JOIN orders o ON
o.customer_id = c.id JOIN order_items i ON i.order_id = o.id JOIN products p ON
p.id = i.product_id WHERE o.id = 5 ORDER BY p.id" > otvety/23
```

6.24. Лучшие клиенты. *Подсказка ученику: Цепочка соединений `customers → orders → order_items`, условие `WHERE o.status <> 'cancelled'`, группировка по `c.id`.*

```
sqlite3 shop.db "SELECT c.name, SUM(i.qty * i.price) AS total FROM customers c
JOIN orders o ON o.customer_id = c.id JOIN order_items i ON i.order_id = o.id
WHERE o.status <> 'cancelled' GROUP BY c.id ORDER BY total DESC, c.id LIMIT 5" >
otvety/24
```

6.25. Покупатели электроники. *Подсказка ученику: Без `DISTINCT` клиент повторится столько раз, сколько он покупал: `SELECT DISTINCT c.name ... ORDER BY c.name LIMIT 10`.*

```
sqlite3 shop.db "SELECT DISTINCT c.name FROM customers c JOIN orders o ON
o.customer_id = c.id JOIN order_items i ON i.order_id = o.id JOIN products p ON
p.id = i.product_id WHERE p.category = 'Электроника' ORDER BY c.name LIMIT 10" >
otvety/25
```

6.26. Книги в доставке. *Подсказка ученику: `WHERE o.status = 'shipped' AND p.category = 'Книги' ORDER BY o.id, p.id LIMIT 10` — условия по разным таблицам цепочки в одном `WHERE`.*

```
sqlite3 shop.db "SELECT o.id, c.name, p.name FROM orders o JOIN customers c ON
c.id = o.customer_id JOIN order_items i ON i.order_id = o.id JOIN products p ON
p.id = i.product_id WHERE o.status = 'shipped' AND p.category = 'Книги' ORDER BY
o.id, p.id LIMIT 10" > otvety/26
```

6.27. Китайские поставщики. *Подсказка ученику: `... JOIN suppliers s ON s.id = p.supplier_id WHERE s.country = 'Китай' ORDER BY i.order_id, p.id LIMIT 10`*

```
sqlite3 shop.db "SELECT i.order_id, p.name, s.name FROM order_items i JOIN
products p ON p.id = i.product_id JOIN suppliers s ON s.id = p.supplier_id WHERE
s.country = 'Китай' ORDER BY i.order_id, p.id LIMIT 10" > otvety/27
```

6.28. Заказы VIP-клиентов. *Подсказка ученику: Цепочка из четырёх таблиц, условие `WHERE c.vip = 1 ORDER BY o.id, p.id LIMIT 10`.*

```
sqlite3 shop.db "SELECT o.id, c.name, p.name FROM customers c JOIN orders o ON
o.customer_id = c.id JOIN order_items i ON i.order_id = o.id JOIN products p ON
p.id = i.product_id WHERE c.vip = 1 ORDER BY o.id, p.id LIMIT 10" > otvety/28
```

6.29. Выручка по категориям. *Подсказка ученику: `... JOIN products p ON p.id = i.product_id WHERE o.status <> 'cancelled' GROUP BY p.category ORDER BY p.category`*

```
sqlite3 shop.db "SELECT p.category, SUM(i.qty * i.price) AS total FROM orders o
JOIN order_items i ON i.order_id = o.id JOIN products p ON p.id = i.product_id
WHERE o.status <> 'cancelled' GROUP BY p.category ORDER BY p.category" > otvety/29
```

6.30. Выручка по странам. *Подсказка ученику: Группировка по `s.country`: `GROUP BY s.country ORDER BY total DESC, s.country` — два поставщика из одной страны сольются в одну строку.*

```
sqlite3 shop.db "SELECT s.country, SUM(i.qty * i.price) AS total FROM orders o
JOIN order_items i ON i.order_id = o.id JOIN products p ON p.id = i.product_id
JOIN suppliers s ON s.id = p.supplier_id WHERE o.status <> 'cancelled' GROUP BY
s.country ORDER BY total DESC, s.country" > otvety/30
```

6.31. Поставщики и товары. *Подсказка ученику: `SELECT s.name, COALESCE(p.name, 'нет товаров') FROM suppliers s LEFT JOIN products p ON p.supplier_id = s.id WHERE s.id <= 2 ORDER BY s.id, p.id LIMIT 12`*

```
sqlite3 shop.db "SELECT s.name, COALESCE(p.name, 'нет товаров') FROM suppliers s
LEFT JOIN products p ON p.supplier_id = s.id WHERE s.id <= 2 ORDER BY s.id, p.id
LIMIT 12" > otvety/31
```

6.32. Клиенты и их заказы. *Подсказка ученику: `FROM customers c LEFT JOIN orders o ON o.customer_id = c.id WHERE c.id BETWEEN 31 AND 33 ORDER BY c.id, o.id` — для клиентов без заказов `o.id` пуст.*

```
sqlite3 shop.db "SELECT c.name, o.id FROM customers c LEFT JOIN orders o ON
o.customer_id = c.id WHERE c.id BETWEEN 31 AND 33 ORDER BY c.id, o.id" > otvety/32
```

6.33. Сотрудники и отделы. *Подсказка ученику: `SELECT e.name, COALESCE(d.name, 'нет отдела') FROM employees e LEFT JOIN departments d ON d.id = e.dept_id WHERE e.id BETWEEN 22 AND 25 ORDER BY e.id`*

```
sqlite3 hr.db "SELECT e.name, COALESCE(d.name, 'нет отдела') FROM employees e LEFT
JOIN departments d ON d.id = e.dept_id WHERE e.id BETWEEN 22 AND 25 ORDER BY e.id"
> otvety/33
```

6.34. Число заказов клиента. *Подсказка ученику: `COUNT(o.id)` не считает `NULL`: `... LEFT JOIN orders o ON o.customer_id = c.id WHERE c.id BETWEEN 30 AND 36 GROUP BY c.id ORDER BY c.id`.*

```
sqlite3 shop.db "SELECT c.name, COUNT(o.id) FROM customers c LEFT JOIN orders o ON
o.customer_id = c.id WHERE c.id BETWEEN 30 AND 36 GROUP BY c.id ORDER BY c.id" >
otvety/34
```

6.35. Численность отделов. *Подсказка ученику: `COUNT(e.id)` не считает `NULL`, поэтому для пустого отдела даст 0 (в отличие от `COUNT(*)`).*

```
sqlite3 hr.db "SELECT d.name, COUNT(e.id) FROM departments d LEFT JOIN employees e
ON e.dept_id = d.id GROUP BY d.id ORDER BY d.id" > otvety/35
```

6.36. Дата последнего заказа. *Подсказка ученику: `SELECT c.name, COALESCE(MAX(o.order_date), 'нет') ... GROUP BY c.id ORDER BY c.id` — для пустой группы `MAX` вернёт `NULL`.*

```
sqlite3 shop.db "SELECT c.name, COALESCE(MAX(o.order_date), 'нет') FROM customers
c LEFT JOIN orders o ON o.customer_id = c.id WHERE c.id BETWEEN 30 AND 34 GROUP BY
c.id ORDER BY c.id" > otvety/36
```

6.37. Оплаченные заказы клиентов. *Подсказка ученику: `LEFT JOIN orders o ON o.customer_id = c.id AND o.status = 'paid'` `WHERE c.id <= 8` — в `WHERE` условие по `o.status` убрало бы клиентов без таких заказов.*

```
sqlite3 shop.db "SELECT c.name, COUNT(o.id) FROM customers c LEFT JOIN orders o ON
o.customer_id = c.id AND o.status = 'paid' WHERE c.id <= 8 GROUP BY c.id ORDER BY
c.id" > otvety/37
```

6.38. Товары в наличии. *Подсказка ученику: `LEFT JOIN products p ON p.supplier_id = s.id AND p.stock > 0 GROUP BY s.id ORDER BY s.id`*

```
sqlite3 shop.db "SELECT s.name, COUNT(p.id) FROM suppliers s LEFT JOIN products p
ON p.supplier_id = s.id AND p.stock > 0 GROUP BY s.id ORDER BY s.id" > otvety/38
```

6.39. Продажи первых товаров. *Подсказка ученику: `SELECT p.name, COALESCE(SUM(i.qty), 0) FROM products p LEFT JOIN order_items i ON i.product_id = p.id WHERE p.id <= 10 GROUP BY p.id ORDER BY p.id`*

```
sqlite3 shop.db "SELECT p.name, COALESCE(SUM(i.qty), 0) FROM products p LEFT JOIN
order_items i ON i.product_id = p.id WHERE p.id <= 10 GROUP BY p.id ORDER BY p.id"
> otvety/39
```

6.40. Заказы и выручка клиентов. *Подсказка ученику: Два `LEFT JOIN` подряд; `COUNT(DISTINCT o.id)` — иначе заказ посчитается по числу позиций; `COALESCE(SUM(i.qty * i.price), 0)`.*

```
sqlite3 shop.db "SELECT c.name, COUNT(DISTINCT o.id), COALESCE(SUM(i.qty *
i.price), 0) FROM customers c LEFT JOIN orders o ON o.customer_id = c.id LEFT JOIN
order_items i ON i.order_id = o.id WHERE c.id BETWEEN 30 AND 34 GROUP BY c.id
ORDER BY c.id" > otvety/40
```

6.41. Клиенты без заказов. *Подсказка ученику: После `LEFT JOIN` у клиентов без заказов столбцы заказа равны `NULL`: `WHERE o.id IS NULL`.*

```
sqlite3 shop.db "SELECT c.name FROM customers c LEFT JOIN orders o ON
o.customer_id = c.id WHERE o.id IS NULL ORDER BY c.name" > otvety/41
```

6.42. Сотрудники без отдела. *Подсказка ученику: `LEFT JOIN departments d ON d.id = e.dept_id ... WHERE d.id IS NULL`*

```
sqlite3 hr.db "SELECT e.name FROM employees e LEFT JOIN departments d ON d.id =
e.dept_id WHERE d.id IS NULL ORDER BY e.id" > otvety/42
```

6.43. Без оплаченных заказов. *Подсказка ученику: `LEFT JOIN orders o ON o.customer_id = c.id AND o.status = 'paid' WHERE o.id IS NULL ORDER BY c.id LIMIT 12`*

```
sqlite3 shop.db "SELECT c.id, c.name FROM customers c LEFT JOIN orders o ON
o.customer_id = c.id AND o.status = 'paid' WHERE o.id IS NULL ORDER BY c.id LIMIT
12" > otvety/43
```

6.44. Нет в первом заказе. *Подсказка ученику: LEFT JOIN order_items i ON i.product_id = p.id AND i.order_id = 1 WHERE i.order_id IS NULL ORDER BY p.id LIMIT 10*

```
sqlite3 shop.db "SELECT p.name FROM products p LEFT JOIN order_items i ON
i.product_id = p.id AND i.order_id = 1 WHERE i.order_id IS NULL ORDER BY p.id
LIMIT 10" > otvety/44
```

6.45. Сколько без заказов. *Подсказка ученику: SELECT COUNT(*) FROM customers c LEFT JOIN orders o ON o.customer_id = c.id WHERE o.id IS NULL;*

```
sqlite3 shop.db "SELECT COUNT(*) FROM customers c LEFT JOIN orders o ON
o.customer_id = c.id WHERE o.id IS NULL" > otvety/45
```

6.46. Без отменённых. *Подсказка ученику: LEFT JOIN orders o ON o.customer_id = c.id AND o.status = 'cancelled' WHERE o.id IS NULL ORDER BY c.id LIMIT 12*

```
sqlite3 shop.db "SELECT c.id, c.name FROM customers c LEFT JOIN orders o ON
o.customer_id = c.id AND o.status = 'cancelled' WHERE o.id IS NULL ORDER BY c.id
LIMIT 12" > otvety/46
```

6.47. Отделы без тестировщиков. *Подсказка ученику: LEFT JOIN employees e ON e.dept_id = d.id AND e.position = 'тестировщик' WHERE e.id IS NULL ORDER BY d.id*

```
sqlite3 hr.db "SELECT d.name FROM departments d LEFT JOIN employees e ON e.dept_id
= d.id AND e.position = 'тестировщик' WHERE e.id IS NULL ORDER BY d.id" >
otvety/47
```

6.48. Нет дорогих товаров. *Подсказка ученику: LEFT JOIN products p ON p.supplier_id = s.id AND p.price > 9500 WHERE p.id IS NULL ORDER BY s.id*

```
sqlite3 shop.db "SELECT s.name FROM suppliers s LEFT JOIN products p ON
p.supplier_id = s.id AND p.price > 9500 WHERE p.id IS NULL ORDER BY s.id" >
otvety/48
```

6.49. Сколько с заказами. *Подсказка ученику: SELECT COUNT(DISTINCT c.id) FROM customers c JOIN orders o ON o.customer_id = c.id;*

```
sqlite3 shop.db "SELECT COUNT(DISTINCT c.id) FROM customers c JOIN orders o ON
o.customer_id = c.id" > otvety/49
```

6.50. Не заказывали в 2026. *Подсказка ученику: LEFT JOIN orders o ON o.customer_id = c.id AND o.order_date >= '2026-01-01' WHERE o.id IS NULL ORDER BY c.id LIMIT 12*

```
sqlite3 shop.db "SELECT c.id, c.name FROM customers c LEFT JOIN orders o ON
o.customer_id = c.id AND o.order_date >= '2026-01-01' WHERE o.id IS NULL ORDER BY
c.id LIMIT 12" > otvety/50
```

6.51. Сотрудник и руководитель. *Подсказка ученику: Одну таблицу подключают дважды под разными псевдонимами: `FROM employees e LEFT JOIN employees m ON m.id = e.manager_id`.*

```
sqlite3 hr.db "SELECT e.name, m.name FROM employees e LEFT JOIN employees m ON
m.id = e.manager_id ORDER BY e.id" > otvety/51
```

6.52. Только с руководителем. *Подсказка ученику: `FROM employees e JOIN employees m ON m.id = e.manager_id ORDER BY e.id LIMIT 10` — строка с `manager_id IS NULL` пары не имеет.*

```
sqlite3 hr.db "SELECT e.name, m.name FROM employees e JOIN employees m ON m.id =
e.manager_id ORDER BY e.id LIMIT 10" > otvety/52
```

6.53. Подчинённые директора. *Подсказка ученику: `WHERE m.position = 'директор'` — условие по «копии» `m`, то есть по руководителю.*

```
sqlite3 hr.db "SELECT e.name, e.position FROM employees e JOIN employees m ON m.id
= e.manager_id WHERE m.position = 'директор' ORDER BY e.id" > otvety/53
```

6.54. Разрыв в зарплате. *Подсказка ученику: `SELECT e.name, m.salary - e.salary AS gap ... ORDER BY gap DESC, e.id LIMIT 5`*

```
sqlite3 hr.db "SELECT e.name, m.salary - e.salary AS gap FROM employees e JOIN
employees m ON m.id = e.manager_id ORDER BY gap DESC, e.id LIMIT 5" > otvety/54
```

6.55. Число подчинённых. *Подсказка ученику: `SELECT m.name, COUNT(*) AS n FROM employees e JOIN employees m ON m.id = e.manager_id GROUP BY m.id ORDER BY n DESC, m.id`*

```
sqlite3 hr.db "SELECT m.name, COUNT(*) AS n FROM employees e JOIN employees m ON
m.id = e.manager_id GROUP BY m.id ORDER BY n DESC, m.id" > otvety/55
```

6.56. Крупные руководители. *Подсказка ученику: `... GROUP BY m.id HAVING COUNT(*) > 3 ORDER BY m.id`*

```
sqlite3 hr.db "SELECT m.name FROM employees e JOIN employees m ON m.id =
e.manager_id GROUP BY m.id HAVING COUNT(*) > 3 ORDER BY m.id" > otvety/56
```

6.57. Начальники отделов. *Подсказка ученику: `WHERE e.position = 'начальник отдела'` — условие по «копии» `e` (подчинённый).*

```
sqlite3 hr.db "SELECT e.name, m.name FROM employees e JOIN employees m ON m.id =
e.manager_id WHERE e.position = 'начальник отдела' ORDER BY e.id" > otvety/57
```

6.58. Пары коллег. *Подсказка ученику: `FROM employees a JOIN employees b ON b.manager_id = a.manager_id AND a.id < b.id WHERE a.manager_id = 1 ORDER BY a.id, b.id LIMIT 10` — `a.id < b.id` убирает пару с самим собой и зеркальные дубли.*

```
sqlite3 hr.db "SELECT a.name, b.name FROM employees a JOIN employees b ON
b.manager_id = a.manager_id AND a.id < b.id WHERE a.manager_id = 1 ORDER BY a.id,
b.id LIMIT 10" > otvety/58
```

6.59. Руководитель руководителя. *Подсказка ученику: FROM employees e JOIN employees m ON m.id = e.manager_id JOIN employees b ON b.id = m.manager_id ORDER BY e.id LIMIT 10*

```
sqlite3 hr.db "SELECT e.name, m.name, b.name FROM employees e JOIN employees m ON
m.id = e.manager_id JOIN employees b ON b.id = m.manager_id ORDER BY e.id LIMIT
10" > otvety/59
```

6.60. Зарплата подчинённых. *Подсказка ученику: SELECT m.name, ROUND(AVG(e.salary), 0) FROM employees e JOIN employees m ON m.id = e.manager_id GROUP BY m.id ORDER BY m.id*

```
sqlite3 hr.db "SELECT m.name, ROUND(AVG(e.salary), 0) FROM employees e JOIN
employees m ON m.id = e.manager_id GROUP BY m.id ORDER BY m.id" > otvety/60
```

6.61. Позиции в заказах. *Подсказка ученику: ... GROUP BY o.id ORDER BY o.id LIMIT 10*

```
sqlite3 shop.db "SELECT o.id, COUNT(*) FROM orders o JOIN order_items i ON
i.order_id = o.id GROUP BY o.id ORDER BY o.id LIMIT 10" > otvety/61
```

6.62. Штуки по категориям. *Подсказка ученику: SELECT p.category, SUM(i.qty) AS pieces FROM order_items i JOIN products p ON p.id = i.product_id GROUP BY p.category ORDER BY p.category*

```
sqlite3 shop.db "SELECT p.category, SUM(i.qty) AS pieces FROM order_items i JOIN
products p ON p.id = i.product_id GROUP BY p.category ORDER BY p.category" >
otvety/62
```

6.63. Фонд зарплат. *Подсказка ученику: SELECT d.name, SUM(e.salary) AS fund FROM departments d JOIN employees e ON e.dept_id = d.id GROUP BY d.id ORDER BY d.id*

```
sqlite3 hr.db "SELECT d.name, SUM(e.salary) AS fund FROM departments d JOIN
employees e ON e.dept_id = d.id GROUP BY d.id ORDER BY d.id" > otvety/63
```

6.64. Самые крупные заказы. *Подсказка ученику: SUM(i.qty * i.price) AS total ... ORDER BY total DESC, o.id LIMIT 5*

```
sqlite3 shop.db "SELECT o.id, SUM(i.qty * i.price) AS total FROM orders o JOIN
order_items i ON i.order_id = o.id GROUP BY o.id ORDER BY total DESC, o.id LIMIT
5" > otvety/64
```

6.65. Самые продаваемые. *Подсказка ученику: Соедините `products` с `order_items` и сгруппируйте по товару.*

```
sqlite3 shop.db "SELECT p.name, SUM(i.qty) AS q FROM products p JOIN order_items i
ON i.product_id = p.id GROUP BY p.id ORDER BY q DESC, p.id LIMIT 3" > otvety/65
```

6.66. Заказы и строки. *Подсказка ученику: Два числа различаются: заказ размножается по числу позиций. `COUNT(DISTINCT o.id)` — заказы, `COUNT(*)` — строки после `JOIN`.*

```
sqlite3 shop.db "SELECT o.status, COUNT(DISTINCT o.id) AS orders, COUNT(*) AS
rows_joined FROM orders o JOIN order_items i ON i.order_id = o.id GROUP BY
o.status ORDER BY o.status" > otvety/66
```

6.67. Заказы из многих позиций. *Подсказка ученику: `... GROUP BY o.id HAVING COUNT(*) > 2 ORDER BY o.id LIMIT 10`*

```
sqlite3 shop.db "SELECT o.id FROM orders o JOIN order_items i ON i.order_id = o.id
GROUP BY o.id HAVING COUNT(*) > 2 ORDER BY o.id LIMIT 10" > otvety/67
```

6.68. Выручка по месяцам. *Подсказка ученику: `SELECT strftime('%Y-%m', o.order_date) AS ym, SUM(i.qty * i.price) AS total ... WHERE o.status <> 'cancelled' GROUP BY ym ORDER BY ym LIMIT 6`*

```
sqlite3 shop.db "SELECT strftime('%Y-%m', o.order_date) AS ym, SUM(i.qty *
i.price) AS total FROM orders o JOIN order_items i ON i.order_id = o.id WHERE
o.status <> 'cancelled' GROUP BY ym ORDER BY ym LIMIT 6" > otvety/68
```

6.69. Выручка по городам. *Подсказка ученику: `WHERE o.status <> 'cancelled' AND c.city IS NOT NULL GROUP BY c.city ORDER BY total DESC, c.city LIMIT 3`*

```
sqlite3 shop.db "SELECT c.city, SUM(i.qty * i.price) AS total FROM customers c
JOIN orders o ON o.customer_id = c.id JOIN order_items i ON i.order_id = o.id
WHERE o.status <> 'cancelled' AND c.city IS NOT NULL GROUP BY c.city ORDER BY
total DESC, c.city LIMIT 3" > otvety/69
```

6.70. Средняя строка заказа. *Подсказка ученику: `SELECT ROUND(AVG(i.qty * i.price), 2) FROM orders o JOIN order_items i ON i.order_id = o.id WHERE o.status <> 'cancelled';`*

```
sqlite3 shop.db "SELECT ROUND(AVG(i.qty * i.price), 2) FROM orders o JOIN
order_items i ON i.order_id = o.id WHERE o.status <> 'cancelled'" > otvety/70
```

Тема 7. Подзапросы и операции над множествами

7.1. Дороже среднего. *Подсказка ученику: `WHERE price > (SELECT AVG(price) FROM products)` — подзапрос в скобках возвращает одно число.*

```
sqlite3 shop.db "SELECT name, price FROM products WHERE price > (SELECT AVG(price)
FROM products) ORDER BY price DESC, id" > otvety/1
```

7.2. Дешевле среднего. *Подсказка ученику: `WHERE price < (SELECT AVG(price) FROM products) ORDER BY price`*

```
sqlite3 shop.db "SELECT name, price FROM products WHERE price < (SELECT AVG(price)
FROM products) ORDER BY price, id" > otvety/2
```

7.3. Самый дорогой. *Подсказка ученику: `WHERE price = (SELECT MAX(price) FROM products)` вернёт всех «самых дорогих», если их несколько.*

```
sqlite3 shop.db "SELECT name FROM products WHERE price = (SELECT MAX(price) FROM
products) ORDER BY id" > otvety/3
```

7.4. Самый дешёвый. *Подсказка ученику: `WHERE price = (SELECT MIN(price) FROM products)`*

```
sqlite3 shop.db "SELECT name, price FROM products WHERE price = (SELECT MIN(price)
FROM products) ORDER BY id" > otvety/4
```

7.5. Много на складе. *Подсказка ученику: `WHERE stock > (SELECT AVG(stock) FROM products) ORDER BY stock DESC, id`*

```
sqlite3 shop.db "SELECT name, stock FROM products WHERE stock > (SELECT AVG(stock)
FROM products) ORDER BY stock DESC, id" > otvety/5
```

7.6. Первый заказ магазина. *Подсказка ученику: `WHERE order_date = (SELECT MIN(order_date) FROM orders)` — даты в формате ГГГГ-ММ-ДД сравниваются как текст правильно.*

```
sqlite3 shop.db "SELECT id, order_date FROM orders WHERE order_date = (SELECT
MIN(order_date) FROM orders) ORDER BY id" > otvety/6
```

7.7. Сколько дорогих. *Подсказка ученику: `SELECT COUNT(*) FROM products WHERE price > (SELECT AVG(price) FROM products);`*

```
sqlite3 shop.db "SELECT COUNT(*) FROM products WHERE price > (SELECT AVG(price)
FROM products)" > otvety/7
```

7.8. Клиент последнего заказа. *Подсказка ученику: `WHERE id = (SELECT customer_id FROM orders ORDER BY order_date DESC, id DESC LIMIT 1)` — без `LIMIT 1` SQLite взял бы первую строку, но результат зависел бы от порядка.*

```
sqlite3 shop.db "SELECT name FROM customers WHERE id = (SELECT customer_id FROM
orders ORDER BY order_date DESC, id DESC LIMIT 1)" > otvety/8
```

7.9. Техника и мебель дороже средней. *Подсказка ученику: Подзапрос не зависит от категории: `AND price > (SELECT AVG(price) FROM products)`.*

```
sqlite3 shop.db "SELECT name, price FROM products WHERE category IN
('Электроника', 'Мебель') AND price > (SELECT AVG(price) FROM products) ORDER BY
name" > otvety/9
```

7.10. Крупные позиции. *Подсказка ученику: ... JOIN products p ON p.id = i.product_id WHERE i.qty * i.price > (SELECT AVG(qty * price) FROM order_items) ORDER BY s DESC, i.order_id, i.product_id LIMIT 5*

```
sqlite3 shop.db "SELECT p.name, i.qty * i.price AS s FROM order_items i JOIN
products p ON p.id = i.product_id WHERE i.qty * i.price > (SELECT AVG(qty * price)
FROM order_items) ORDER BY s DESC, i.order_id, i.product_id LIMIT 5" > otvety/10
```

7.11. Были отмены. *Подсказка ученику: WHERE id IN (SELECT customer_id FROM orders WHERE status = 'cancelled')*

```
sqlite3 shop.db "SELECT name FROM customers WHERE id IN (SELECT customer_id FROM
orders WHERE status = 'cancelled') ORDER BY name" > otvety/11
```

7.12. Не заказывали. *Подсказка ученику: Подзапрос возвращает идентификаторы клиентов с заказами; NOT IN отбирает остальных. Если бы в списке был NULL, NOT IN ничего не вернул бы.*

```
sqlite3 shop.db "SELECT name FROM customers WHERE id NOT IN (SELECT customer_id
FROM orders) ORDER BY name" > otvety/12
```

7.13. Поставщики из Германии. *Подсказка ученику: WHERE supplier_id IN (SELECT id FROM suppliers WHERE country = 'Германия')*

```
sqlite3 shop.db "SELECT name FROM products WHERE supplier_id IN (SELECT id FROM
suppliers WHERE country = 'Германия') ORDER BY id" > otvety/13
```

7.14. Брали по пять. *Подсказка ученику: WHERE id IN (SELECT product_id FROM order_items WHERE qty = 5)*

```
sqlite3 shop.db "SELECT name FROM products WHERE id IN (SELECT product_id FROM
order_items WHERE qty = 5) ORDER BY id" > otvety/14
```

7.15. Ни разу не платили. *Подсказка ученику: WHERE id NOT IN (SELECT customer_id FROM orders WHERE status = 'paid') — в orders.customer_id нет NULL, поэтому ловушки нет.*

```
sqlite3 shop.db "SELECT name FROM customers WHERE id NOT IN (SELECT customer_id
FROM orders WHERE status = 'paid') ORDER BY name" > otvety/15
```

7.16. Заказывали мебель. *Подсказка ученику: WHERE id IN (SELECT customer_id FROM orders WHERE id IN (SELECT order_id FROM order_items WHERE product_id IN (SELECT id FROM products WHERE category = 'Мебель')))*

```
sqlite3 shop.db "SELECT name FROM customers WHERE id IN (SELECT customer_id FROM
orders WHERE id IN (SELECT order_id FROM order_items WHERE product_id IN (SELECT
id FROM products WHERE category = 'Мебель'))) ORDER BY name" > otvety/16
```

7.17. Ловушка с NULL. *Подсказка ученику: `SELECT COUNT(*) FROM customers WHERE city NOT IN ('Москва', 'Казань');` — у клиентов с `city IS NULL` сравнение даёт NULL, они в результат не попадают.*

```
sqlite3 shop.db "SELECT COUNT(*) FROM customers WHERE city NOT IN ('Москва',
'Казань')" > otvety/17
```

7.18. Заказывали, но не платили. *Подсказка ученику: `WHERE id IN (SELECT customer_id FROM orders) AND id NOT IN (SELECT customer_id FROM orders WHERE status = 'paid')`*

```
sqlite3 shop.db "SELECT name FROM customers WHERE id IN (SELECT customer_id FROM
orders) AND id NOT IN (SELECT customer_id FROM orders WHERE status = 'paid') ORDER
BY name" > otvety/18
```

7.19. Ещё не отправлены. *Подсказка ученику: `SELECT COUNT(*) FROM orders WHERE status IN ('new', 'paid');` — список значений можно писать прямо.*

```
sqlite3 shop.db "SELECT COUNT(*) FROM orders WHERE status IN ('new', 'paid')" >
otvety/19
```

7.20. Постоянные клиенты. *Подсказка ученику: `WHERE id IN (SELECT customer_id FROM orders GROUP BY customer_id HAVING COUNT(*) >= 5)`*

```
sqlite3 shop.db "SELECT name FROM customers WHERE id IN (SELECT customer_id FROM
orders GROUP BY customer_id HAVING COUNT(*) >= 5) ORDER BY name" > otvety/20
```

7.21. Покупали в 2026 году. *Подсказка ученику: Внутри `EXISTS` подзапрос ссылается на внешнюю таблицу: `o.customer_id = c.id`. Что именно выбирается (`1`), неважно.*

```
sqlite3 shop.db "SELECT c.name FROM customers c WHERE EXISTS (SELECT 1 FROM orders
o WHERE o.customer_id = c.id AND o.order_date >= '2026-01-01') ORDER BY c.name" >
otvety/21
```

7.22. Товар закончился. *Подсказка ученику: `WHERE EXISTS (SELECT 1 FROM products p WHERE p.supplier_id = s.id AND p.stock = 0)`*

```
sqlite3 shop.db "SELECT s.name FROM suppliers s WHERE EXISTS (SELECT 1 FROM
products p WHERE p.supplier_id = s.id AND p.stock = 0) ORDER BY s.name" >
otvety/22
```

7.23. Нет отгруженных. *Подсказка ученику: `WHERE NOT EXISTS (SELECT 1 FROM orders o WHERE o.customer_id = c.id AND o.status = 'shipped')`*

```
sqlite3 shop.db "SELECT c.name FROM customers c WHERE NOT EXISTS (SELECT 1 FROM
orders o WHERE o.customer_id = c.id AND o.status = 'shipped') ORDER BY c.name" >
otvety/23
```

7.24. Товары из отмен. *Подсказка ученику: WHERE EXISTS (SELECT 1 FROM order_items i JOIN orders o ON o.id = i.order_id WHERE i.product_id = p.id AND o.status = 'cancelled')*

```
sqlite3 shop.db "SELECT p.name FROM products p WHERE EXISTS (SELECT 1 FROM
order_items i JOIN orders o ON o.id = i.order_id WHERE i.product_id = p.id AND
o.status = 'cancelled') ORDER BY p.id" > otvety/24
```

7.25. VIP с заказами. *Подсказка ученику: WHERE c.vip = 1 AND EXISTS (SELECT 1 FROM orders o WHERE o.customer_id = c.id)*

```
sqlite3 shop.db "SELECT c.name FROM customers c WHERE c.vip = 1 AND EXISTS (SELECT
1 FROM orders o WHERE o.customer_id = c.id) ORDER BY c.name" > otvety/25
```

7.26. Заказы с пятёркой. *Подсказка ученику: WHERE EXISTS (SELECT 1 FROM order_items i WHERE i.order_id = o.id AND i.qty = 5) ORDER BY o.id LIMIT 10*

```
sqlite3 shop.db "SELECT o.id FROM orders o WHERE EXISTS (SELECT 1 FROM order_items
i WHERE i.order_id = o.id AND i.qty = 5) ORDER BY o.id LIMIT 10" > otvety/26
```

7.27. Платили и отменяли. *Подсказка ученику: Два независимых EXISTS, соединённых через AND; псевдоним o в каждом подзапросе свой.*

```
sqlite3 shop.db "SELECT c.name FROM customers c WHERE EXISTS (SELECT 1 FROM orders
o WHERE o.customer_id = c.id AND o.status = 'paid') AND EXISTS (SELECT 1 FROM
orders o WHERE o.customer_id = c.id AND o.status = 'cancelled') ORDER BY c.name" >
otvety/27
```

7.28. В 2025 и без отмен. *Подсказка ученику: WHERE EXISTS (... AND o.order_date LIKE '2025%') AND NOT EXISTS (... AND o.status = 'cancelled')*

```
sqlite3 shop.db "SELECT c.name FROM customers c WHERE EXISTS (SELECT 1 FROM orders
o WHERE o.customer_id = c.id AND o.order_date LIKE '2025%') AND NOT EXISTS (SELECT
1 FROM orders o WHERE o.customer_id = c.id AND o.status = 'cancelled') ORDER BY
c.name" > otvety/28
```

7.29. Без дорогих товаров. *Подсказка ученику: WHERE NOT EXISTS (SELECT 1 FROM products p WHERE p.supplier_id = s.id AND p.price > 9500)*

```
sqlite3 shop.db "SELECT s.name FROM suppliers s WHERE NOT EXISTS (SELECT 1 FROM
products p WHERE p.supplier_id = s.id AND p.price > 9500) ORDER BY s.name" >
otvety/29
```

7.30. Был крупный заказ. *Подсказка ученику: EXISTS (SELECT 1 FROM orders o JOIN order_items i ON i.order_id = o.id WHERE o.customer_id = c.id GROUP BY o.id HAVING SUM(i.qty * i.price) > 30000)*

```
sqlite3 shop.db "SELECT c.name FROM customers c WHERE EXISTS (SELECT 1 FROM orders
o JOIN order_items i ON i.order_id = o.id WHERE o.customer_id = c.id GROUP BY o.id
HAVING SUM(i.qty * i.price) > 30000) ORDER BY c.name" > otvety/30
```

7.31. Проданное количество. *Подсказка ученику: SELECT p.name, (SELECT COALESCE(SUM(i.qty), 0) FROM order_items i WHERE i.product_id = p.id) AS q FROM products p ...*

```
sqlite3 shop.db "SELECT p.name, (SELECT COALESCE(SUM(i.qty), 0) FROM order_items i
WHERE i.product_id = p.id) AS q FROM products p ORDER BY q DESC, p.id LIMIT 5" >
otvety/31
```

7.32. Число заказов. *Подсказка ученику: SELECT c.name, (SELECT COUNT(*) FROM orders o WHERE o.customer_id = c.id) AS n FROM customers c ORDER BY n DESC, c.id LIMIT 5*

```
sqlite3 shop.db "SELECT c.name, (SELECT COUNT(*) FROM orders o WHERE o.customer_id
= c.id) AS n FROM customers c ORDER BY n DESC, c.id LIMIT 5" > otvety/32
```

7.33. Максимальная цена у поставщика. *Подсказка ученику: (SELECT COALESCE(MAX(p.price), 0) FROM products p WHERE p.supplier_id = s.id) AS mx*

```
sqlite3 shop.db "SELECT s.name, (SELECT COALESCE(MAX(p.price), 0) FROM products p
WHERE p.supplier_id = s.id) AS mx FROM suppliers s ORDER BY s.id" > otvety/33
```

7.34. Последний заказ клиента. *Подсказка ученику: (SELECT MAX(o.order_date) FROM orders o WHERE o.customer_id = c.id) AS last_date*

```
sqlite3 shop.db "SELECT c.name, (SELECT MAX(o.order_date) FROM orders o WHERE
o.customer_id = c.id) AS last_date FROM customers c ORDER BY last_date DESC, c.id
LIMIT 5" > otvety/34
```

7.35. Сумма покупок. *Подсказка ученику: Подзапрос может содержать JOIN: (SELECT COALESCE(SUM(i.qty * i.price), 0) FROM orders o JOIN order_items i ON i.order_id = o.id WHERE o.customer_id = c.id) AS total*

```
sqlite3 shop.db "SELECT c.name, (SELECT COALESCE(SUM(i.qty * i.price), 0) FROM
orders o JOIN order_items i ON i.order_id = o.id WHERE o.customer_id = c.id) AS
total FROM customers c ORDER BY total DESC, c.id LIMIT 3" > otvety/35
```

7.36. Отклонение от средней цены. *Подсказка ученику: Подзапрос без связи с внешней строкой тоже можно писать в списке столбцов: ROUND(price - (SELECT AVG(price) FROM products), 1) AS diff*

```
sqlite3 shop.db "SELECT name, ROUND(price - (SELECT AVG(price) FROM products), 1)
AS diff FROM products ORDER BY price DESC, id LIMIT 5" > otvety/36
```

7.37. Товары и склад по поставщикам. *Подсказка ученику: В списке столбцов может быть сколько угодно подзапросов, у каждого свой AS.*

```
sqlite3 shop.db "SELECT s.name, (SELECT COUNT(*) FROM products p WHERE
p.supplier_id = s.id) AS n, (SELECT COALESCE(SUM(p.stock), 0) FROM products p
WHERE p.supplier_id = s.id) AS total_stock FROM suppliers s ORDER BY s.id" >
otvety/37
```

7.38. Заказы с итогом. *Подсказка ученику: Два подзапроса по `order_items` с условием `i.order_id = o.id`; внешний запрос — `FROM orders o WHERE o.id <= 10`.*

```
sqlite3 shop.db "SELECT o.id, (SELECT COUNT(*) FROM order_items i WHERE i.order_id
= o.id) AS positions, (SELECT SUM(i.qty * i.price) FROM order_items i WHERE
i.order_id = o.id) AS total FROM orders o WHERE o.id <= 10 ORDER BY o.id" >
otvety/38
```

7.39. Самый дорогой товар поставщика. *Подсказка ученику: `(SELECT p.name FROM products p WHERE p.supplier_id = s.id ORDER BY p.price DESC LIMIT 1) AS top_item` — подзапрос возвращает текст, не только число.*

```
sqlite3 shop.db "SELECT s.name, (SELECT p.name FROM products p WHERE p.supplier_id
= s.id ORDER BY p.price DESC LIMIT 1) AS top_item FROM suppliers s ORDER BY s.id"
> otvety/39
```

7.40. Доля заказов клиента. *Подсказка ученику: Подзапрос может входить в выражение: `ROUND(100.0 * (SELECT COUNT(*) FROM orders o WHERE o.customer_id = c.id) / (SELECT COUNT(*) FROM orders), 1) AS pct`; первый связан с `c.id`, второй — нет.*

```
sqlite3 shop.db "SELECT c.name, (SELECT COUNT(*) FROM orders o WHERE o.customer_id
= c.id) AS n, ROUND(100.0 * (SELECT COUNT(*) FROM orders o WHERE o.customer_id =
c.id) / (SELECT COUNT(*) FROM orders), 1) AS pct FROM customers c ORDER BY n DESC,
c.id LIMIT 5" > otvety/40
```

7.41. Средний чек. *Подсказка ученику: `SELECT ROUND(AVG(t), 2) FROM (SELECT SUM(qty * price) AS t FROM order_items GROUP BY order_id);` — у подзапроса в FROM в SQLite псевдоним не обязателен.*

```
sqlite3 shop.db "SELECT ROUND(AVG(t), 2) FROM (SELECT SUM(qty * price) AS t FROM
order_items GROUP BY order_id)" > otvety/41
```

7.42. Рекорд по заказам. *Подсказка ученику: `SELECT MAX(n) FROM (SELECT COUNT(*) AS n FROM orders GROUP BY customer_id);`*

```
sqlite3 shop.db "SELECT MAX(n) FROM (SELECT COUNT(*) AS n FROM orders GROUP BY
customer_id)" > otvety/42
```

7.43. Позиций в заказе. *Подсказка ученику: SELECT ROUND(AVG(cnt), 2) FROM (SELECT COUNT(*) AS cnt FROM order_items GROUP BY order_id);*

```
sqlite3 shop.db "SELECT ROUND(AVG(cnt), 2) FROM (SELECT COUNT(*) AS cnt FROM
order_items GROUP BY order_id)" > otvety/43
```

7.44. Самый крупный заказ. *Подсказка ученику: SELECT MAX(t) FROM (SELECT SUM(qty * price) AS t FROM order_items GROUP BY order_id);*

```
sqlite3 shop.db "SELECT MAX(t) FROM (SELECT SUM(qty * price) AS t FROM order_items
GROUP BY order_id)" > otvety/44
```

7.45. Больше трёх заказов. *Подсказка ученику: SELECT COUNT(*) FROM (SELECT customer_id FROM orders GROUP BY customer_id HAVING COUNT(*) > 3);*

```
sqlite3 shop.db "SELECT COUNT(*) FROM (SELECT customer_id FROM orders GROUP BY
customer_id HAVING COUNT(*) > 3)" > otvety/45
```

7.46. Средний чек по статусам. *Подсказка ученику: Внутри — сумма по каждому заказу (GROUP BY o.id), снаружи — среднее по статусам (GROUP BY status).*

```
sqlite3 shop.db "SELECT status, ROUND(AVG(t), 2) AS avg_total FROM (SELECT
o.status AS status, SUM(i.qty * i.price) AS t FROM orders o JOIN order_items i ON
i.order_id = o.id GROUP BY o.id) GROUP BY status ORDER BY status" > otvety/46
```

7.47. Пять недорогих среди крупных. *Подсказка ученику: SELECT order_id, t FROM (SELECT order_id, SUM(qty * price) AS t FROM order_items GROUP BY order_id) WHERE t > 20000 ORDER BY t, order_id LIMIT 5*

```
sqlite3 shop.db "SELECT order_id, t FROM (SELECT order_id, SUM(qty * price) AS t
FROM order_items GROUP BY order_id) WHERE t > 20000 ORDER BY t, order_id LIMIT 5"
> otvety/47
```

7.48. Лучший средний чек. *Подсказка ученику: Подзапрос в FROM можно соединять с таблицей: FROM customers c JOIN (SELECT o.customer_id AS customer_id, SUM(...) AS total ... GROUP BY o.id) t ON t.customer_id = c.id GROUP BY c.id.*

```
sqlite3 shop.db "SELECT c.name, ROUND(AVG(t.total), 2) AS avg_check FROM customers
c JOIN (SELECT o.customer_id AS customer_id, SUM(i.qty * i.price) AS total FROM
orders o JOIN order_items i ON i.order_id = o.id GROUP BY o.id) t ON t.customer_id
= c.id GROUP BY c.id ORDER BY avg_check DESC, c.id LIMIT 3" > otvety/48
```

7.49. Дороже своей категории. *Подсказка ученику: ... JOIN (SELECT category, AVG(price) AS ap FROM products GROUP BY category) a ON a.category = p.category WHERE p.price > a.ap*

```
sqlite3 shop.db "SELECT p.name, p.category, p.price FROM products p JOIN (SELECT
category, AVG(price) AS ap FROM products GROUP BY category) a ON a.category =
p.category WHERE p.price > a.ap ORDER BY p.id" > otvety/49
```

7.50. Распределение по числу заказов. *Подсказка ученику: Группировка над группировкой: SELECT n, COUNT(*) AS clients FROM (SELECT customer_id, COUNT(*) AS n FROM orders GROUP BY customer_id) GROUP BY n.*

```
sqlite3 shop.db "SELECT n, COUNT(*) AS clients FROM (SELECT customer_id, COUNT(*)
AS n FROM orders GROUP BY customer_id) GROUP BY n ORDER BY n" > otvety/50
```

7.51. Объединение. *Подсказка ученику: SELECT ... UNION SELECT ... ORDER BY name; — ORDER BY пишется один раз в конце всего составного запроса.*

```
sqlite3 shop.db "SELECT name FROM products WHERE price > 9000 UNION SELECT name
FROM products WHERE stock = 0 ORDER BY name" > otvety/51
```

7.52. Клиенты и поставщики. *Подсказка ученику: SELECT name, 'клиент' AS kind FROM customers WHERE vip = 1 UNION ALL SELECT name, 'поставщик' FROM suppliers ORDER BY kind, name;*

```
sqlite3 shop.db "SELECT name, 'клиент' AS kind FROM customers WHERE vip = 1 UNION
ALL SELECT name, 'поставщик' FROM suppliers ORDER BY kind, name" > otvety/52
```

7.53. Платили или получали. *Подсказка ученику: UNION убирает повторы, поэтому клиент, у которого есть заказы обоих статусов, будет посчитан один раз.*

```
sqlite3 shop.db "SELECT COUNT(*) FROM (SELECT customer_id FROM orders WHERE status
= 'paid' UNION SELECT customer_id FROM orders WHERE status = 'shipped')" >
otvety/53
```

7.54. Три дешёвых и три дорогих. *Подсказка ученику: SELECT * FROM (SELECT name, price FROM products ORDER BY price LIMIT 3) UNION ALL SELECT * FROM (SELECT name, price FROM products ORDER BY price DESC LIMIT 3) ORDER BY price;*

```
sqlite3 shop.db "SELECT name, price FROM (SELECT name, price FROM products ORDER
BY price LIMIT 3) UNION ALL SELECT name, price FROM (SELECT name, price FROM
products ORDER BY price DESC LIMIT 3) ORDER BY price" > otvety/54
```

7.55. Три условия. *Подсказка ученику: UNION можно ставить несколько раз подряд; ORDER BY name — в самом конце.*

```
sqlite3 shop.db "SELECT name FROM products WHERE category = 'Книги' UNION SELECT
name FROM products WHERE category = 'Мебель' UNION SELECT name FROM products WHERE
price < 1000 ORDER BY name" > otvety/55
```

7.56. Заказы по статусам с итогом. *Подсказка ученику: SELECT status, COUNT(*) AS n FROM orders GROUP BY status UNION ALL SELECT 'ИТОГО', COUNT(*) FROM orders ORDER BY n DESC, status;*

```
sqlite3 shop.db "SELECT status, COUNT(*) AS n FROM orders GROUP BY status UNION
ALL SELECT 'ИТОГО', COUNT(*) FROM orders ORDER BY n DESC, status" > otvety/56
```

7.57. Мин, макс и среднее. *Подсказка ученику: SELECT 'мин' AS what, MIN(price) AS v FROM products UNION ALL SELECT 'макс', MAX(price) FROM products UNION ALL SELECT 'средняя', ROUND(AVG(price)) FROM products ORDER BY v;*

```
sqlite3 shop.db "SELECT 'мин' AS what, MIN(price) AS v FROM products UNION ALL
SELECT 'макс', MAX(price) FROM products UNION ALL SELECT 'средняя',
ROUND(AVG(price)) FROM products ORDER BY v" > otvety/57
```

7.58. Отмена или самый ранний заказ. *Подсказка ученику: В ORDER BY пишите имя столбца результата (name), а не c.name.*

```
sqlite3 shop.db "SELECT c.name FROM customers c JOIN orders o ON o.customer_id =
c.id WHERE o.status = 'cancelled' UNION SELECT c.name FROM customers c JOIN orders
o ON o.customer_id = c.id WHERE o.order_date < '2025-03-01' ORDER BY name" >
otvety/58
```

7.59. Два столбца и сортировка. *Подсказка ученику: Столбцы обеих частей должны совпадать по числу и смыслу; ORDER BY price DESC, name — один раз в конце.*

```
sqlite3 shop.db "SELECT name, price FROM products WHERE stock = 0 UNION SELECT
name, price FROM products WHERE price > 9000 ORDER BY price DESC, name" >
otvety/59
```

7.60. Первый и последний. *Подсказка ученику: ... WHERE order_date = (SELECT MIN(order_date) FROM orders) UNION ... WHERE order_date = (SELECT MAX(order_date) FROM orders)*

```
sqlite3 shop.db "SELECT customer_id FROM orders WHERE order_date = (SELECT
MIN(order_date) FROM orders) UNION SELECT customer_id FROM orders WHERE order_date
= (SELECT MAX(order_date) FROM orders) ORDER BY customer_id" > otvety/60
```

7.61. Покупали и в 2025, и в 2026. *Подсказка ученику: ... WHERE order_date LIKE '2025%' INTERSECT ... WHERE order_date LIKE '2026%'*

```
sqlite3 shop.db "SELECT customer_id FROM orders WHERE order_date LIKE '2025%'
INTERSECT SELECT customer_id FROM orders WHERE order_date LIKE '2026%' ORDER BY
customer_id" > otvety/61
```

7.62. Разность множеств. *Подсказка ученику: SELECT id FROM customers EXCEPT SELECT customer_id FROM orders;*

```
sqlite3 shop.db "SELECT id FROM customers EXCEPT SELECT customer_id FROM orders
ORDER BY id" > otvety/62
```

7.63. Платили и отменяли. *Подсказка ученику: SELECT customer_id FROM orders WHERE status = 'paid' INTERSECT SELECT customer_id FROM orders WHERE status = 'cancelled';*

```
sqlite3 shop.db "SELECT customer_id FROM orders WHERE status = 'paid' INTERSECT
SELECT customer_id FROM orders WHERE status = 'cancelled' ORDER BY customer_id" >
otvety/63
```

7.64. Ни разу не отменяли. *Подсказка ученику: SELECT customer_id FROM orders EXCEPT SELECT customer_id FROM orders WHERE status = 'cancelled';*

```
sqlite3 shop.db "SELECT customer_id FROM orders EXCEPT SELECT customer_id FROM
orders WHERE status = 'cancelled' ORDER BY customer_id" > otvety/64
```

7.65. Оплачены, но без пятёрок. *Подсказка ученику: SELECT id FROM orders WHERE status = 'paid' EXCEPT SELECT order_id FROM order_items WHERE qty = 5 ORDER BY id; — имя в ORDER BY берётся из первого запроса.*

```
sqlite3 shop.db "SELECT id FROM orders WHERE status = 'paid' EXCEPT SELECT
order_id FROM order_items WHERE qty = 5 ORDER BY id" > otvety/65
```

7.66. Электроника и книги. *Подсказка ученику: Обе части — один и тот же запрос с четырьмя таблицами, различается лишь категория в WHERE.*

```
sqlite3 shop.db "SELECT c.name AS name FROM customers c JOIN orders o ON
o.customer_id = c.id JOIN order_items i ON i.order_id = o.id JOIN products p ON
p.id = i.product_id WHERE p.category = 'Электроника' INTERSECT SELECT c.name AS
name FROM customers c JOIN orders o ON o.customer_id = c.id JOIN order_items i ON
i.order_id = o.id JOIN products p ON p.id = i.product_id WHERE p.category =
'Книги' ORDER BY name" > otvety/66
```

7.67. Не покупали в 2026. *Подсказка ученику: SELECT COUNT(*) FROM (SELECT id FROM customers EXCEPT SELECT customer_id FROM orders WHERE order_date LIKE '2026%');*

```
sqlite3 shop.db "SELECT COUNT(*) FROM (SELECT id FROM customers EXCEPT SELECT
customer_id FROM orders WHERE order_date LIKE '2026%')" > otvety/67
```

7.68. Дорогие и в наличии. *Подсказка ученику: SELECT name FROM products WHERE price > (SELECT AVG(price) FROM products) INTERSECT SELECT name FROM products WHERE stock > 0 ORDER BY name;*

```
sqlite3 shop.db "SELECT name FROM products WHERE price > (SELECT AVG(price) FROM
products) INTERSECT SELECT name FROM products WHERE stock > 0 ORDER BY name" >
otvety/68
```

7.69. Товары для всех. *Подсказка ученику: Части различаются только условием c.vip = 1 / c.vip = 0; в ORDER BY — name.*

```
sqlite3 shop.db "SELECT p.name AS name FROM products p JOIN order_items i ON
i.product_id = p.id JOIN orders o ON o.id = i.order_id JOIN customers c ON c.id =
o.customer_id WHERE c.vip = 1 INTERSECT SELECT p.name AS name FROM products p JOIN
order_items i ON i.product_id = p.id JOIN orders o ON o.id = i.order_id JOIN
customers c ON c.id = o.customer_id WHERE c.vip = 0 ORDER BY name" > otvety/69
```

7.70. Слева направо. *Подсказка ученику: Сначала выполняется UNION (new, paid), затем из результата вычитается cancelled; приоритета у EXCEPT нет.*

```
sqlite3 shop.db "SELECT customer_id FROM orders WHERE status = 'new' UNION SELECT
customer_id FROM orders WHERE status = 'paid' EXCEPT SELECT customer_id FROM
orders WHERE status = 'cancelled' ORDER BY customer_id" > otvety/70
```

Тема 8. CTE, рекурсия и оконные функции

8.1. Средний и наибольший заказ. *Подсказка ученику: WITH t AS (SELECT order_id, SUM(qty * price) AS total FROM order_items GROUP BY order_id) SELECT ROUND(AVG(total), 2), MAX(total) FROM t;*

```
sqlite3 shop.db "WITH t AS (SELECT order_id, SUM(qty * price) AS total FROM
order_items GROUP BY order_id) SELECT ROUND(AVG(total), 2), MAX(total) FROM t" >
otvety/1
```

8.2. Клиенты выше среднего. *Подсказка ученику: CTE можно использовать несколько раз: и в JOIN, и в подзапросе (SELECT AVG(total) FROM rev). WITH rev AS (SELECT o.customer_id AS cid, SUM(i.qty * i.price) AS total FROM orders o JOIN order_items i ON i.order_id = o.id GROUP BY o.customer_id) SELECT c.name, rev.total FROM rev JOIN customers c ON c.id = rev.cid WHERE rev.total > (SELECT AVG(total) FROM rev) ORDER BY rev.total DESC, c.id;*

```
sqlite3 shop.db "WITH rev AS (SELECT o.customer_id AS cid, SUM(i.qty * i.price) AS
total FROM orders o JOIN order_items i ON i.order_id = o.id GROUP BY
o.customer_id) SELECT c.name, rev.total FROM rev JOIN customers c ON c.id =
rev.cid WHERE rev.total > (SELECT AVG(total) FROM rev) ORDER BY rev.total DESC,
c.id" > otvety/2
```

8.3. Активные клиенты. *Подсказка ученику: WITH cnt AS (SELECT customer_id, COUNT(*) AS n FROM orders GROUP BY customer_id) SELECT COUNT(*) FROM cnt WHERE n > 3;*

```
sqlite3 shop.db "WITH cnt AS (SELECT customer_id, COUNT(*) AS n FROM orders GROUP
BY customer_id) SELECT COUNT(*) FROM cnt WHERE n > 3" > otvety/3
```

8.4. Остатки по категориям. *Подсказка ученику: Список имён после имени CTE заменяет AS у столбцов внутри. WITH s(cat, v) AS (SELECT category, SUM(stock) FROM products GROUP BY category) SELECT cat, v FROM s ORDER BY v DESC, cat;*

```
sqlite3 shop.db "WITH s(cat, v) AS (SELECT category, SUM(stock) FROM products
GROUP BY category) SELECT cat, v FROM s ORDER BY v DESC, cat" > otvety/4
```

8.5. Заказы выше среднего. *Подсказка ученику: Слово WITH пишут один раз, второе CTE может ссылаться на первое. WITH t AS (SELECT order_id, SUM(qty * price) AS total FROM order_items GROUP BY order_id), m AS (SELECT AVG(total) AS a FROM t) SELECT COUNT(*) FROM t, m WHERE t.total > m.a;*

```
sqlite3 shop.db "WITH t AS (SELECT order_id, SUM(qty * price) AS total FROM
order_items GROUP BY order_id), m AS (SELECT AVG(total) AS a FROM t) SELECT
COUNT(*) FROM t, m WHERE t.total > m.a" > otvety/5
```

8.6. Дорогие для своего поставщика. *Подсказка ученику: WITH a AS (SELECT supplier_id, AVG(price) AS p FROM products GROUP BY supplier_id) SELECT pr.id, pr.name, pr.price FROM products pr JOIN a ON a.supplier_id = pr.supplier_id WHERE pr.price > a.p ORDER BY pr.id;*

```
sqlite3 shop.db "WITH a AS (SELECT supplier_id, AVG(price) AS p FROM products
GROUP BY supplier_id) SELECT pr.id, pr.name, pr.price FROM products pr JOIN a ON
a.supplier_id = pr.supplier_id WHERE pr.price > a.p ORDER BY pr.id" > otvety/6
```

8.7. Самые загруженные месяцы. *Подсказка ученику: WITH m AS (SELECT substr(order_date, 1, 7) AS ym, COUNT(*) AS n FROM orders GROUP BY substr(order_date, 1, 7)) SELECT ym, n FROM m WHERE n = (SELECT MAX(n) FROM m) ORDER BY ym;*

```
sqlite3 shop.db "WITH m AS (SELECT substr(order_date, 1, 7) AS ym, COUNT(*) AS n
FROM orders GROUP BY substr(order_date, 1, 7)) SELECT ym, n FROM m WHERE n =
(SELECT MAX(n) FROM m) ORDER BY ym" > otvety/7
```

8.8. Отмены у клиентов. *Подсказка ученику: WITH c AS (SELECT customer_id, COUNT(*) AS k FROM orders WHERE status = 'cancelled' GROUP BY customer_id) SELECT cu.name, c.k FROM c JOIN customers cu ON cu.id = c.customer_id ORDER BY c.k DESC, cu.id LIMIT 3;*

```
sqlite3 shop.db "WITH c AS (SELECT customer_id, COUNT(*) AS k FROM orders WHERE
status = 'cancelled' GROUP BY customer_id) SELECT cu.name, c.k FROM c JOIN
customers cu ON cu.id = c.customer_id ORDER BY c.k DESC, cu.id LIMIT 3" > otvety/8
```

8.9. Клиенты без заказов. *Подсказка ученику: WITH has AS (SELECT DISTINCT customer_id FROM orders) SELECT name FROM customers WHERE id NOT IN (SELECT customer_id FROM has) ORDER BY id;*

```
sqlite3 shop.db "WITH has AS (SELECT DISTINCT customer_id FROM orders) SELECT name
FROM customers WHERE id NOT IN (SELECT customer_id FROM has) ORDER BY id" >
otvety/9
```

8.10. Выручка клиента по городам. *Подсказка ученику: WITH rev AS (SELECT o.customer_id AS cid, SUM(i.qty * i.price) AS total FROM orders o JOIN order_items i ON i.order_id = o.id GROUP BY o.customer_id) SELECT c.city, ROUND(AVG(rev.total), 2), COUNT(*) FROM rev JOIN customers c ON c.id = rev.cid WHERE c.city IS NOT NULL GROUP BY c.city ORDER BY c.city;*

```
sqlite3 shop.db "WITH rev AS (SELECT o.customer_id AS cid, SUM(i.qty * i.price) AS
total FROM orders o JOIN order_items i ON i.order_id = o.id GROUP BY
o.customer_id) SELECT c.city, ROUND(AVG(rev.total), 2), COUNT(*) FROM rev JOIN
customers c ON c.id = rev.cid WHERE c.city IS NOT NULL GROUP BY c.city ORDER BY
c.city" > otvety/10
```

8.11. Сумма чисел рекурсией. *Подсказка ученику: Условие WHERE x < 100 останавливает рекурсию. WITH RECURSIVE n(x) AS (SELECT 1 UNION ALL SELECT x + 1 FROM n WHERE x < 100) SELECT SUM(x) FROM n;*

```
sqlite3 shop.db "WITH RECURSIVE n(x) AS (SELECT 1 UNION ALL SELECT x + 1 FROM n
WHERE x < 100) SELECT SUM(x) FROM n" > otvety/11
```

8.12. Степени двойки. *Подсказка ученику: WITH RECURSIVE p(n, v) AS (SELECT 0, 1 UNION ALL SELECT n + 1, v * 2 FROM p WHERE n < 20) SELECT n, v FROM p ORDER BY n;*

```
sqlite3 shop.db "WITH RECURSIVE p(n, v) AS (SELECT 0, 1 UNION ALL SELECT n + 1, v
* 2 FROM p WHERE n < 20) SELECT n, v FROM p ORDER BY n" > otvety/12
```

8.13. Факториал. *Подсказка ученику: WITH RECURSIVE f(n, v) AS (SELECT 1, 1 UNION ALL SELECT n + 1, v * (n + 1) FROM f WHERE n < 10) SELECT v FROM f WHERE n = 10;*

```
sqlite3 shop.db "WITH RECURSIVE f(n, v) AS (SELECT 1, 1 UNION ALL SELECT n + 1, v
* (n + 1) FROM f WHERE n < 10) SELECT v FROM f WHERE n = 10" > otvety/13
```

8.14. Число Фибоначчи. *Подсказка ученику: WITH RECURSIVE fib(a, b, n) AS (SELECT 0, 1, 1 UNION ALL SELECT b, a + b, n + 1 FROM fib WHERE n < 15) SELECT a FROM fib WHERE n = 15;*

```
sqlite3 shop.db "WITH RECURSIVE fib(a, b, n) AS (SELECT 0, 1, 1 UNION ALL SELECT
b, a + b, n + 1 FROM fib WHERE n < 15) SELECT a FROM fib WHERE n = 15" > otvety/14
```

8.15. Воскресенья января. *Подсказка ученику: WITH RECURSIVE d(x) AS (SELECT '2026-01-01' UNION ALL SELECT date(x, '+1 day') FROM d WHERE x < '2026-01-31') SELECT COUNT(*) FROM d WHERE strftime('%w', x) = '0';*

```
sqlite3 shop.db "WITH RECURSIVE d(x) AS (SELECT '2026-01-01' UNION ALL SELECT
date(x, '+1 day') FROM d WHERE x < '2026-01-31') SELECT COUNT(*) FROM d WHERE
strftime('%w', x) = '0'" > otvety/15
```

8.16. Уровни подчинения. *Подсказка ученику: Рекурсивная часть соединяет employees с уже найденными строками tree по manager_id. WITH RECURSIVE tree(id, lvl) AS (SELECT id, 0 FROM employees WHERE manager_id IS NULL UNION ALL SELECT e.id, tree.lvl + 1 FROM employees e JOIN tree ON e.manager_id = tree.id) SELECT lvl, COUNT(*) FROM tree GROUP BY lvl ORDER BY lvl;*

```
sqlite3 hr.db "WITH RECURSIVE tree(id, lvl) AS (SELECT id, 0 FROM employees WHERE
manager_id IS NULL UNION ALL SELECT e.id, tree.lvl + 1 FROM employees e JOIN tree
ON e.manager_id = tree.id) SELECT lvl, COUNT(*) FROM tree GROUP BY lvl ORDER BY
lvl" > otvety/16
```

8.17. Подчинённые начальника разработки. *Подсказка ученику: WITH RECURSIVE sub(id, lvl) AS (SELECT id, 1 FROM employees WHERE manager_id = 2 UNION ALL SELECT e.id, sub.lvl + 1 FROM employees e JOIN sub ON e.manager_id = sub.id) SELECT e.name, sub.lvl FROM sub JOIN employees e ON e.id = sub.id ORDER BY sub.lvl, e.id;*

```
sqlite3 hr.db "WITH RECURSIVE sub(id, lvl) AS (SELECT id, 1 FROM employees WHERE
manager_id = 2 UNION ALL SELECT e.id, sub.lvl + 1 FROM employees e JOIN sub ON
e.manager_id = sub.id) SELECT e.name, sub.lvl FROM sub JOIN employees e ON e.id =
sub.id ORDER BY sub.lvl, e.id" > otvety/17
```

8.18. Цепочка начальников. *Подсказка ученику: Для подъёма вверх соединяют по e.id = up.mgr. WITH RECURSIVE up(id, mgr, depth) AS (SELECT id, manager_id, 0 FROM employees WHERE id = 22 UNION ALL SELECT e.id, e.manager_id, up.depth + 1 FROM employees e JOIN up ON e.id = up.mgr) SELECT id, depth FROM up ORDER BY depth;*

```
sqlite3 hr.db "WITH RECURSIVE up(id, mgr, depth) AS (SELECT id, manager_id, 0 FROM
employees WHERE id = 22 UNION ALL SELECT e.id, e.manager_id, up.depth + 1 FROM
employees e JOIN up ON e.id = up.mgr) SELECT id, depth FROM up ORDER BY depth" >
otvety/18
```

8.19. Фонд отдела 3. *Подсказка ученику: WITH RECURSIVE t(id) AS (SELECT 3 UNION ALL SELECT e.id FROM employees e JOIN t ON e.manager_id = t.id) SELECT SUM(salary) FROM employees WHERE id IN (SELECT id FROM t);*

```
sqlite3 hr.db "WITH RECURSIVE t(id) AS (SELECT 3 UNION ALL SELECT e.id FROM
employees e JOIN t ON e.manager_id = t.id) SELECT SUM(salary) FROM employees WHERE
id IN (SELECT id FROM t)" > otvety/19
```

8.20. Зарплата по уровням. *Подсказка ученику: WITH RECURSIVE tree(id, lvl) AS (SELECT id, 0 FROM employees WHERE manager_id IS NULL UNION ALL SELECT e.id, tree.lvl + 1 FROM employees e JOIN tree ON e.manager_id = tree.id) SELECT tree.lvl, COUNT(*), ROUND(AVG(e.salary)) FROM tree JOIN employees e ON e.id = tree.id GROUP BY tree.lvl ORDER BY tree.lvl;*

```
sqlite3 hr.db "WITH RECURSIVE tree(id, lvl) AS (SELECT id, 0 FROM employees WHERE
manager_id IS NULL UNION ALL SELECT e.id, tree.lvl + 1 FROM employees e JOIN tree
ON e.manager_id = tree.id) SELECT tree.lvl, COUNT(*), ROUND(AVG(e.salary)) FROM
tree JOIN employees e ON e.id = tree.id GROUP BY tree.lvl ORDER BY tree.lvl" >
otvety/20
```

8.21. Цена и средняя цена. *Подсказка ученику: Окно считается по всей таблице, LIMIT действует уже после него. SELECT id, price, ROUND(AVG(price) OVER (), 2) FROM products ORDER BY id LIMIT 5;*

```
sqlite3 shop.db "SELECT id, price, ROUND(AVG(price) OVER (), 2) FROM products
ORDER BY id LIMIT 5" > otvety/21
```

8.22. Размер категории. *Подсказка ученику: SELECT id, category, COUNT(*) OVER (PARTITION BY category) FROM products ORDER BY id LIMIT 8;*

```
sqlite3 shop.db "SELECT id, category, COUNT(*) OVER (PARTITION BY category) FROM
products ORDER BY id LIMIT 8" > otvety/22
```

8.23. Отклонение от средней цены категории. *Подсказка ученику: SELECT id, name, ROUND(price - AVG(price) OVER (PARTITION BY category), 1) FROM products ORDER BY id LIMIT 8;*

```
sqlite3 shop.db "SELECT id, name, ROUND(price - AVG(price) OVER (PARTITION BY
category), 1) FROM products ORDER BY id LIMIT 8" > otvety/23
```

8.24. Зарплата и средняя по отделу. *Подсказка ученику: SELECT id, salary, ROUND(AVG(salary) OVER (PARTITION BY dept_id)) FROM employees WHERE dept_id IS NOT NULL ORDER BY id LIMIT 10;*

```
sqlite3 hr.db "SELECT id, salary, ROUND(AVG(salary) OVER (PARTITION BY dept_id))
FROM employees WHERE dept_id IS NOT NULL ORDER BY id LIMIT 10" > otvety/24
```

8.25. Доля цены в категории. *Подсказка ученику: SELECT id, category, ROUND(100.0 * price / SUM(price) OVER (PARTITION BY category), 1) FROM products ORDER BY id LIMIT 10;*

```
sqlite3 shop.db "SELECT id, category, ROUND(100.0 * price / SUM(price) OVER
(PARTITION BY category), 1) FROM products ORDER BY id LIMIT 10" > otvety/25
```

8.26. Окно после WHERE. *Подсказка ученику: WHERE выполняется раньше окна, поэтому счётчик учитывает только отобранные строки. SELECT id, salary, COUNT(*) OVER () FROM employees WHERE salary > 100000 ORDER BY id LIMIT 5;*

```
sqlite3 hr.db "SELECT id, salary, COUNT(*) OVER () FROM employees WHERE salary >
100000 ORDER BY id LIMIT 5" > otvety/26
```

8.27. Доля категории. *Подсказка ученику: Пустое окно OVER () означает «вся таблица»: SUM(v) OVER () — общий итог в каждой строке. WITH c AS (SELECT category, SUM(price * stock) AS v FROM products GROUP BY category) SELECT category, v, ROUND(100.0 * v / SUM(v) OVER (), 1) FROM c ORDER BY category;*

```
sqlite3 shop.db "WITH c AS (SELECT category, SUM(price * stock) AS v FROM products
GROUP BY category) SELECT category, v, ROUND(100.0 * v / SUM(v) OVER (), 1) FROM c
ORDER BY category" > otvety/27
```

8.28. Заказы клиента рядом с заказом. *Подсказка ученику: SELECT id, customer_id, COUNT(*) OVER (PARTITION BY customer_id) FROM orders ORDER BY id LIMIT 10;*

```
sqlite3 shop.db "SELECT id, customer_id, COUNT(*) OVER (PARTITION BY customer_id)
FROM orders ORDER BY id LIMIT 10" > otvety/28
```

8.29. Размах цен категории. *Подсказка ученику: SELECT id, price, MAX(price) OVER (PARTITION BY category) - MIN(price) OVER (PARTITION BY category) FROM products ORDER BY id LIMIT 6;*

```
sqlite3 shop.db "SELECT id, price, MAX(price) OVER (PARTITION BY category) -
MIN(price) OVER (PARTITION BY category) FROM products ORDER BY id LIMIT 6" >
otvety/29
```

8.30. Выручка городов и доли. *Подсказка ученику: WITH r AS (SELECT c.city AS city, SUM(i.qty * i.price) AS v FROM customers c JOIN orders o ON o.customer_id = c.id JOIN order_items i ON i.order_id = o.id WHERE c.city IS NOT NULL GROUP BY c.city) SELECT city, v, ROUND(100.0 * v / SUM(v) OVER (), 1) FROM r ORDER BY v DESC, city;*

```
sqlite3 shop.db "WITH r AS (SELECT c.city AS city, SUM(i.qty * i.price) AS v FROM
customers c JOIN orders o ON o.customer_id = c.id JOIN order_items i ON i.order_id
= o.id WHERE c.city IS NOT NULL GROUP BY c.city) SELECT city, v, ROUND(100.0 * v /
SUM(v) OVER (), 1) FROM r ORDER BY v DESC, city" > otvety/30
```

8.31. Рейтинг цен. *Подсказка ученику: ROW_NUMBER() OVER (ORDER BY price DESC, id) AS rn — номер строки в заданном порядке. SELECT id, name, price, ROW_NUMBER() OVER (ORDER BY price DESC, id) AS rn FROM products ORDER BY rn LIMIT 5;*

```
sqlite3 shop.db "SELECT id, name, price, ROW_NUMBER() OVER (ORDER BY price DESC,
id) AS rn FROM products ORDER BY rn LIMIT 5" > otvety/31
```

8.32. Новые клиенты. *Подсказка ученику: SELECT ROW_NUMBER() OVER (ORDER BY registered DESC, id) AS rn, id, name, registered FROM customers ORDER BY rn LIMIT 5;*

```
sqlite3 shop.db "SELECT ROW_NUMBER() OVER (ORDER BY registered DESC, id) AS rn,
id, name, registered FROM customers ORDER BY rn LIMIT 5" > otvety/32
```

8.33. Место в категории. *Подсказка ученику: PARTITION BY category делит строки на «окна» по категориям; ранг считается внутри каждого окна. SELECT category, name, price, RANK() OVER (PARTITION BY category ORDER BY price DESC) AS r FROM products ORDER BY category, r;*

```
sqlite3 shop.db "SELECT category, name, price, RANK() OVER (PARTITION BY category
ORDER BY price DESC) AS r FROM products ORDER BY category, r" > otvety/33
```

8.34. Ранги остатков. *Подсказка ученику: При равных остатках RANK выдаёт одинаковое место и пропускает следующие. SELECT id, stock, RANK() OVER (ORDER BY stock DESC) AS rk FROM products ORDER BY rk, id LIMIT 10;*

```
sqlite3 shop.db "SELECT id, stock, RANK() OVER (ORDER BY stock DESC) AS rk FROM
products ORDER BY rk, id LIMIT 10" > otvety/34
```

8.35. Плотные ранги остатков. *Подсказка ученику: SELECT id, stock, DENSE_RANK() OVER (ORDER BY stock DESC) AS dr FROM products ORDER BY dr, id LIMIT 10;*

```
sqlite3 shop.db "SELECT id, stock, DENSE_RANK() OVER (ORDER BY stock DESC) AS dr
FROM products ORDER BY dr, id LIMIT 10" > otvety/35
```

8.36. Сколько разных остатков. *Подсказка ученику: SELECT MAX(d) FROM (SELECT DENSE_RANK() OVER (ORDER BY stock) AS d FROM products);*

```
sqlite3 shop.db "SELECT MAX(d) FROM (SELECT DENSE_RANK() OVER (ORDER BY stock) AS
d FROM products)" > otvety/36
```

8.37. Три функции рядом. *Подсказка ученику: SELECT id, stock, ROW_NUMBER() OVER (ORDER BY stock DESC, id) AS rn, RANK() OVER (ORDER BY stock DESC) AS rk, DENSE_RANK() OVER (ORDER BY stock DESC) AS dr FROM products ORDER BY rn LIMIT 8;*

```
sqlite3 shop.db "SELECT id, stock, ROW_NUMBER() OVER (ORDER BY stock DESC, id) AS
rn, RANK() OVER (ORDER BY stock DESC) AS rk, DENSE_RANK() OVER (ORDER BY stock
DESC) AS dr FROM products ORDER BY rn LIMIT 8" > otvety/37
```

8.38. Рейтинг зарплат в отделе. *Подсказка ученику: SELECT dept_id, name, salary, ROW_NUMBER() OVER (PARTITION BY dept_id ORDER BY salary DESC, id) AS rn FROM employees WHERE dept_id IS NOT NULL ORDER BY dept_id, rn LIMIT 10;*

```
sqlite3 hr.db "SELECT dept_id, name, salary, ROW_NUMBER() OVER (PARTITION BY
dept_id ORDER BY salary DESC, id) AS rn FROM employees WHERE dept_id IS NOT NULL
ORDER BY dept_id, rn LIMIT 10" > otvety/38
```

8.39. Рейтинг клиентов по выручке. *Подсказка ученику: WITH rev AS (SELECT o.customer_id AS cid, SUM(i.qty * i.price) AS total FROM orders o JOIN order_items i ON i.order_id = o.id GROUP BY o.customer_id) SELECT RANK() OVER (ORDER BY rev.total DESC) AS rk, c.name, rev.total FROM rev JOIN customers c ON c.id = rev.cid ORDER BY rk, c.id LIMIT 5;*

```
sqlite3 shop.db "WITH rev AS (SELECT o.customer_id AS cid, SUM(i.qty * i.price) AS
total FROM orders o JOIN order_items i ON i.order_id = o.id GROUP BY
o.customer_id) SELECT RANK() OVER (ORDER BY rev.total DESC) AS rk, c.name,
rev.total FROM rev JOIN customers c ON c.id = rev.cid ORDER BY rk, c.id LIMIT 5" >
otvety/39
```

8.40. Порядковый номер заказа клиента. *Подсказка ученику: SELECT id, customer_id, order_date, ROW_NUMBER() OVER (PARTITION BY customer_id ORDER BY order_date, id) FROM orders ORDER BY id LIMIT 10;*

```
sqlite3 shop.db "SELECT id, customer_id, order_date, ROW_NUMBER() OVER (PARTITION
BY customer_id ORDER BY order_date, id) FROM orders ORDER BY id LIMIT 10" >
otvety/40
```

8.41. Нарастающий итог. *Подсказка ученику: SUM(total) OVER (ORDER BY order_id) суммирует от первой строки до текущей. WITH t AS (SELECT order_id, SUM(qty * price) AS total FROM order_items GROUP BY order_id) SELECT order_id, total, SUM(total) OVER (ORDER BY order_id) FROM t ORDER BY order_id LIMIT 5;*

```
sqlite3 shop.db "WITH t AS (SELECT order_id, SUM(qty * price) AS total FROM
order_items GROUP BY order_id) SELECT order_id, total, SUM(total) OVER (ORDER BY
order_id) FROM t ORDER BY order_id LIMIT 5" > otvety/41
```

8.42. Накопление остатков. *Подсказка ученику: SELECT id, stock, SUM(stock) OVER (ORDER BY id) FROM products ORDER BY id LIMIT 6;*

```
sqlite3 shop.db "SELECT id, stock, SUM(stock) OVER (ORDER BY id) FROM products
ORDER BY id LIMIT 6" > otvety/42
```

8.43. Рост фонда зарплат. *Подсказка ученику: SELECT id, hired, salary, SUM(salary) OVER (ORDER BY hired, id) FROM employees ORDER BY hired, id LIMIT 5;*

```
sqlite3 hr.db "SELECT id, hired, salary, SUM(salary) OVER (ORDER BY hired, id)
FROM employees ORDER BY hired, id LIMIT 5" > otvety/43
```

8.44. Нарастающий итог по отделу. *Подсказка ученику: SELECT dept_id, id, salary, SUM(salary) OVER (PARTITION BY dept_id ORDER BY hired, id) FROM employees WHERE dept_id IS NOT NULL ORDER BY dept_id, hired, id LIMIT 6;*

```
sqlite3 hr.db "SELECT dept_id, id, salary, SUM(salary) OVER (PARTITION BY dept_id
ORDER BY hired, id) FROM employees WHERE dept_id IS NOT NULL ORDER BY dept_id,
hired, id LIMIT 6" > otvety/44
```

8.45. Ловушка неуникального ключа. *Подсказка ученику: Рамка по умолчанию включает все строки с тем же значением ключа. SELECT id, stock, SUM(stock) OVER (ORDER BY stock DESC) FROM products ORDER BY stock DESC, id LIMIT 8;*

```
sqlite3 shop.db "SELECT id, stock, SUM(stock) OVER (ORDER BY stock DESC) FROM
products ORDER BY stock DESC, id LIMIT 8" > otvety/45
```

8.46. Честный итог через ROWS. *Подсказка ученику: SELECT id, stock, SUM(stock) OVER (ORDER BY stock DESC, id ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) FROM products ORDER BY stock DESC, id LIMIT 8;*

```
sqlite3 shop.db "SELECT id, stock, SUM(stock) OVER (ORDER BY stock DESC, id ROWS
BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) FROM products ORDER BY stock DESC, id
LIMIT 8" > otvety/46
```

8.47. Скользящее среднее цен. *Подсказка ученику: SELECT id, price, ROUND(AVG(price) OVER (ORDER BY id ROWS BETWEEN 2 PRECEDING AND CURRENT ROW), 1) FROM products ORDER BY id LIMIT 8;*

```
sqlite3 shop.db "SELECT id, price, ROUND(AVG(price) OVER (ORDER BY id ROWS BETWEEN
2 PRECEDING AND CURRENT ROW), 1) FROM products ORDER BY id LIMIT 8" > otvety/47
```

8.48. Выручка нарастающим итогом. *Подсказка ученику: WITH m AS (SELECT substr(o.order_date, 1, 7) AS ym, SUM(i.qty * i.price) AS v FROM orders o JOIN order_items i ON i.order_id = o.id GROUP BY substr(o.order_date, 1, 7)) SELECT ym, v, SUM(v) OVER (ORDER BY ym) FROM m ORDER BY ym LIMIT 6;*

```
sqlite3 shop.db "WITH m AS (SELECT substr(o.order_date, 1, 7) AS ym, SUM(i.qty *
i.price) AS v FROM orders o JOIN order_items i ON i.order_id = o.id GROUP BY
substr(o.order_date, 1, 7)) SELECT ym, v, SUM(v) OVER (ORDER BY ym) FROM m ORDER
BY ym LIMIT 6" > otvety/48
```

8.49. Остаток заказов до конца. *Подсказка ученику: WITH t AS (SELECT order_id, SUM(qty * price) AS total FROM order_items GROUP BY order_id) SELECT order_id, total, SUM(total) OVER (ORDER BY order_id ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING) FROM t ORDER BY order_id LIMIT 5;*

```
sqlite3 shop.db "WITH t AS (SELECT order_id, SUM(qty * price) AS total FROM
order_items GROUP BY order_id) SELECT order_id, total, SUM(total) OVER (ORDER BY
order_id ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING) FROM t ORDER BY
order_id LIMIT 5" > otvety/49
```

8.50. Месяц половины выручки. *Подсказка ученику: WITH m AS (SELECT substr(o.order_date, 1, 7) AS ym, SUM(i.qty * i.price) AS v FROM orders o JOIN order_items i ON i.order_id = o.id GROUP BY substr(o.order_date, 1, 7)), x AS (SELECT ym, SUM(v) OVER (ORDER BY ym) AS cum, SUM(v) OVER () AS tot FROM m) SELECT ym, cum FROM x WHERE cum * 2 >= tot ORDER BY ym LIMIT 1;*

```
sqlite3 shop.db "WITH m AS (SELECT substr(o.order_date, 1, 7) AS ym, SUM(i.qty *
i.price) AS v FROM orders o JOIN order_items i ON i.order_id = o.id GROUP BY
substr(o.order_date, 1, 7)), x AS (SELECT ym, SUM(v) OVER (ORDER BY ym) AS cum,
SUM(v) OVER () AS tot FROM m) SELECT ym, cum FROM x WHERE cum * 2 >= tot ORDER BY
ym LIMIT 1" > otvety/50
```

8.51. Предыдущий заказ. *Подсказка ученику: Окно упорядочено по дате: LAG берёт значение из предыдущей строки окна; у первой строки окна оно NULL. SELECT id, customer_id, order_date, LAG(order_date) OVER (PARTITION BY customer_id ORDER BY order_date, id) FROM orders ORDER BY id LIMIT 10;*

```
sqlite3 shop.db "SELECT id, customer_id, order_date, LAG(order_date) OVER
(PARTITION BY customer_id ORDER BY order_date, id) FROM orders ORDER BY id LIMIT
10" > otvety/51
```

8.52. Предыдущая цена. *Подсказка ученику: SELECT id, price, LAG(price) OVER (ORDER BY id) FROM products ORDER BY id LIMIT 6;*

```
sqlite3 shop.db "SELECT id, price, LAG(price) OVER (ORDER BY id) FROM products
ORDER BY id LIMIT 6" > otvety/52
```

8.53. Кого приняли следом. *Подсказка ученику: SELECT name, LEAD(name) OVER (ORDER BY hired, id) FROM employees ORDER BY hired, id LIMIT 6;*

```
sqlite3 hr.db "SELECT name, LEAD(name) OVER (ORDER BY hired, id) FROM employees
ORDER BY hired, id LIMIT 6" > otvety/53
```

8.54. Разность с соседом. *Подсказка ученику: SELECT id, price, price - LAG(price) OVER (ORDER BY id) FROM products ORDER BY id LIMIT 6;*

```
sqlite3 shop.db "SELECT id, price, price - LAG(price) OVER (ORDER BY id) FROM
products ORDER BY id LIMIT 6" > otvety/54
```

8.55. Значение по умолчанию. *Подсказка ученику: SELECT id, stock, LAG(stock, 1, 0) OVER (ORDER BY id) FROM products ORDER BY id LIMIT 6;*

```
sqlite3 shop.db "SELECT id, stock, LAG(stock, 1, 0) OVER (ORDER BY id) FROM
products ORDER BY id LIMIT 6" > otvety/55
```

8.56. Смещение на два. *Подсказка ученику: SELECT id, price, LAG(price, 2) OVER (ORDER BY id), LEAD(price, 2) OVER (ORDER BY id) FROM products ORDER BY id LIMIT 6;*

```
sqlite3 shop.db "SELECT id, price, LAG(price, 2) OVER (ORDER BY id), LEAD(price,
2) OVER (ORDER BY id) FROM products ORDER BY id LIMIT 6" > otvety/56
```

8.57. Паузы между заказами. *Подсказка ученику: Условие по результату LAG пишут во внешнем запросе. WITH x AS (SELECT id, customer_id, order_date, LAG(order_date) OVER (PARTITION BY customer_id ORDER BY order_date, id) AS prev FROM orders) SELECT id, customer_id, CAST(julianday(order_date) - julianday(prev) AS INTEGER) FROM x WHERE prev IS NOT NULL ORDER BY id LIMIT 8;*

```
sqlite3 shop.db "WITH x AS (SELECT id, customer_id, order_date, LAG(order_date)
OVER (PARTITION BY customer_id ORDER BY order_date, id) AS prev FROM orders)
SELECT id, customer_id, CAST(julianday(order_date) - julianday(prev) AS INTEGER)
FROM x WHERE prev IS NOT NULL ORDER BY id LIMIT 8" > otvety/57
```

8.58. Самая долгая пауза. *Подсказка ученику: WITH x AS (SELECT order_date, LAG(order_date) OVER (PARTITION BY customer_id ORDER BY order_date, id) AS prev FROM orders) SELECT MAX(CAST(julianday(order_date) - julianday(prev) AS INTEGER)) FROM x WHERE prev IS NOT NULL;*

```
sqlite3 shop.db "WITH x AS (SELECT order_date, LAG(order_date) OVER (PARTITION BY
customer_id ORDER BY order_date, id) AS prev FROM orders) SELECT
MAX(CAST(julianday(order_date) - julianday(prev) AS INTEGER)) FROM x WHERE prev IS
NOT NULL" > otvety/58
```

8.59. Зарплата выше предыдущего. *Подсказка ученику: WITH x AS (SELECT salary, LAG(salary) OVER (ORDER BY hired, id) AS p FROM employees) SELECT COUNT(*) FROM x WHERE salary > p;*

```
sqlite3 hr.db "WITH x AS (SELECT salary, LAG(salary) OVER (ORDER BY hired, id) AS
p FROM employees) SELECT COUNT(*) FROM x WHERE salary > p" > otvety/59
```

8.60. Рост выручки по месяцам. *Подсказка ученику: WITH m AS (SELECT substr(o.order_date, 1, 7) AS ym, SUM(i.qty * i.price) AS v FROM orders o JOIN order_items i ON i.order_id = o.id GROUP BY substr(o.order_date, 1, 7)), d AS (SELECT ym, v, LAG(v) OVER (ORDER BY ym) AS p FROM m) SELECT ym, v, ROUND(100.0 * (v - p) / p, 1) FROM d WHERE p IS NOT NULL ORDER BY ym LIMIT 6;*

```
sqlite3 shop.db "WITH m AS (SELECT substr(o.order_date, 1, 7) AS ym, SUM(i.qty *
i.price) AS v FROM orders o JOIN order_items i ON i.order_id = o.id GROUP BY
substr(o.order_date, 1, 7)), d AS (SELECT ym, v, LAG(v) OVER (ORDER BY ym) AS p
FROM m) SELECT ym, v, ROUND(100.0 * (v - p) / p, 1) FROM d WHERE p IS NOT NULL
ORDER BY ym LIMIT 6" > otvety/60
```

8.61. Лидер категории. *Подсказка ученику: Оконную функцию нельзя поставить в WHERE; поэтому сначала считают номер в CTE, а затем отбирают rn = 1. WITH r AS (SELECT category, name, price, ROW_NUMBER() OVER (PARTITION BY category ORDER BY price DESC, id) AS rn FROM products) SELECT category, name, price FROM r WHERE rn = 1 ORDER BY category;*

```
sqlite3 shop.db "WITH r AS (SELECT category, name, price, ROW_NUMBER() OVER
(PARTITION BY category ORDER BY price DESC, id) AS rn FROM products) SELECT
category, name, price FROM r WHERE rn = 1 ORDER BY category" > otvety/61
```

8.62. Два самых дорогих. *Подсказка ученику: WITH r AS (SELECT category, name, price, ROW_NUMBER() OVER (PARTITION BY category ORDER BY price DESC, id) AS rn FROM products) SELECT category, name, price FROM r WHERE rn <= 2 ORDER BY category, rn;*

```
sqlite3 shop.db "WITH r AS (SELECT category, name, price, ROW_NUMBER() OVER
(PARTITION BY category ORDER BY price DESC, id) AS rn FROM products) SELECT
category, name, price FROM r WHERE rn <= 2 ORDER BY category, rn" > otvety/62
```

8.63. Самый дешёвый. *Подсказка ученику: WITH r AS (SELECT category, name, price, ROW_NUMBER() OVER (PARTITION BY category ORDER BY price, id) AS rn FROM products) SELECT category, name, price FROM r WHERE rn = 1 ORDER BY category;*

```
sqlite3 shop.db "WITH r AS (SELECT category, name, price, ROW_NUMBER() OVER
(PARTITION BY category ORDER BY price, id) AS rn FROM products) SELECT category,
name, price FROM r WHERE rn = 1 ORDER BY category" > otvety/63
```

8.64. Старейший клиент города. *Подсказка ученику: WITH r AS (SELECT city, name, registered, ROW_NUMBER() OVER (PARTITION BY city ORDER BY registered, id) AS rn FROM customers WHERE city IS NOT NULL) SELECT city, name, registered FROM r WHERE rn = 1 ORDER BY city;*

```
sqlite3 shop.db "WITH r AS (SELECT city, name, registered, ROW_NUMBER() OVER
(PARTITION BY city ORDER BY registered, id) AS rn FROM customers WHERE city IS NOT
NULL) SELECT city, name, registered FROM r WHERE rn = 1 ORDER BY city" > otvety/64
```

8.65. Последний заказ клиента. *Подсказка ученику: WITH r AS (SELECT customer_id, id, order_date, ROW_NUMBER() OVER (PARTITION BY customer_id ORDER BY order_date DESC, id DESC) AS rn FROM orders) SELECT customer_id, id, order_date FROM r WHERE rn = 1 ORDER BY customer_id;*

```
sqlite3 shop.db "WITH r AS (SELECT customer_id, id, order_date, ROW_NUMBER() OVER
(PARTITION BY customer_id ORDER BY order_date DESC, id DESC) AS rn FROM orders)
SELECT customer_id, id, order_date FROM r WHERE rn = 1 ORDER BY customer_id" >
otvety/65
```

8.66. Лучший оклад отдела. *Подсказка ученику: WITH r AS (SELECT dept_id, name, salary, ROW_NUMBER() OVER (PARTITION BY dept_id ORDER BY salary DESC, id) AS rn FROM employees WHERE dept_id IS NOT NULL) SELECT d.name, r.name, r.salary FROM r JOIN departments d ON d.id = r.dept_id WHERE r.rn = 1 ORDER BY d.id;*

```
sqlite3 hr.db "WITH r AS (SELECT dept_id, name, salary, ROW_NUMBER() OVER
(PARTITION BY dept_id ORDER BY salary DESC, id) AS rn FROM employees WHERE dept_id
IS NOT NULL) SELECT d.name, r.name, r.salary FROM r JOIN departments d ON d.id =
r.dept_id WHERE r.rn = 1 ORDER BY d.id" > otvety/66
```

8.67. Все лидеры по остатку. *Подсказка ученику: WITH r AS (SELECT id, category, name, stock, RANK() OVER (PARTITION BY category ORDER BY stock DESC) AS rk FROM products) SELECT category, name, stock FROM r WHERE rk = 1 ORDER BY category, id;*

```
sqlite3 shop.db "WITH r AS (SELECT id, category, name, stock, RANK() OVER
(PARTITION BY category ORDER BY stock DESC) AS rk FROM products) SELECT category,
name, stock FROM r WHERE rk = 1 ORDER BY category, id" > otvety/67
```

8.68. Топ-2 клиента города. *Подсказка ученику: WITH rev AS (SELECT o.customer_id AS cid, SUM(i.qty * i.price) AS total FROM orders o JOIN order_items i ON i.order_id = o.id GROUP BY o.customer_id), r AS (SELECT c.city AS city, c.name AS name, rev.total AS total, ROW_NUMBER() OVER (PARTITION BY c.city ORDER BY rev.total DESC, c.id) AS rn FROM rev JOIN customers c ON c.id = rev.cid WHERE c.city IS NOT NULL) SELECT city, name, total FROM r WHERE rn <= 2 ORDER BY city, rn;*

```
sqlite3 shop.db "WITH rev AS (SELECT o.customer_id AS cid, SUM(i.qty * i.price) AS
total FROM orders o JOIN order_items i ON i.order_id = o.id GROUP BY
o.customer_id), r AS (SELECT c.city AS city, c.name AS name, rev.total AS total,
ROW_NUMBER() OVER (PARTITION BY c.city ORDER BY rev.total DESC, c.id) AS rn FROM
rev JOIN customers c ON c.id = rev.cid WHERE c.city IS NOT NULL) SELECT city,
name, total FROM r WHERE rn <= 2 ORDER BY city, rn" > otvety/68
```

8.69. Третий заказ клиента. *Подсказка ученику: WITH r AS (SELECT customer_id, id, order_date, ROW_NUMBER() OVER (PARTITION BY customer_id ORDER BY order_date, id) AS rn FROM orders) SELECT customer_id, id, order_date FROM r WHERE rn = 3 ORDER BY customer_id;*

```
sqlite3 shop.db "WITH r AS (SELECT customer_id, id, order_date, ROW_NUMBER() OVER
(PARTITION BY customer_id ORDER BY order_date, id) AS rn FROM orders) SELECT
customer_id, id, order_date FROM r WHERE rn = 3 ORDER BY customer_id" > otvety/69
```

8.70. Вторая зарплата отдела. *Подсказка ученику: WITH r AS (SELECT dept_id, name, salary, DENSE_RANK() OVER (PARTITION BY dept_id ORDER BY salary DESC) AS dr FROM employees WHERE dept_id IS NOT NULL) SELECT d.name, r.name, r.salary FROM r JOIN departments d ON d.id = r.dept_id WHERE r.dr = 2 ORDER BY d.id;*

```
sqlite3 hr.db "WITH r AS (SELECT dept_id, name, salary, DENSE_RANK() OVER
(PARTITION BY dept_id ORDER BY salary DESC) AS dr FROM employees WHERE dept_id IS
NOT NULL) SELECT d.name, r.name, r.salary FROM r JOIN departments d ON d.id =
r.dept_id WHERE r.dr = 2 ORDER BY d.id" > otvety/70
```

Тема 9. Изменение данных: INSERT, UPDATE, DELETE

9.1. Новый клиент. *Подсказка ученику: INSERT INTO customers (id, name, city, email, registered, vip) VALUES (1000, 'Тестов Тест', 'Тула', ...); — перечислите столбцы явно: порядок значений должен им соответствовать.*

```
INSERT INTO customers (id, name, city, email, registered, vip) VALUES (1000,
'Тестов Тест', 'Тула', 'test@sever.example', '2026-10-11', 0);
```

9.2. Три товара одной командой. *Подсказка ученику: После VALUES можно перечислить несколько строк в скобках через запятую.*

```
INSERT INTO products (id, name, category, price, stock, supplier_id) VALUES (101,
'Тест-1', 'Тест', 100, 10, 1), (102, 'Тест-2', 'Тест', 200, 10, 1), (103,
'Тест-3', 'Тест', 300, 10, 1);
```

9.3. Книги дорожают. *Подсказка ученику: Сначала посмотрите, что заденет условие: SELECT id, price FROM products WHERE category = 'Книги';, затем UPDATE products SET price = price * 11 / 10 WHERE category = 'Книги';*

```
UPDATE products SET price = price * 11 / 10 WHERE category = 'Книги';
```

9.4. Обнулить остаток. *Подсказка ученику: UPDATE products SET stock = 0 WHERE price > 9000; Без WHERE изменились бы все товары.*

```
UPDATE products SET stock = 0 WHERE price > 9000;
```

9.5. Добавить или обновить. *Подсказка ученику: Конструкция ON CONFLICT(id) DO UPDATE SET stock = stock + excluded.stock срабатывает, если строка с таким id уже есть; excluded.stock — значение из вставляемой строки.*

```
INSERT INTO products (id, name, category, price, stock, supplier_id) VALUES (5,
'x', 'x', 0, 10, 1)
ON CONFLICT(id) DO UPDATE SET stock = stock + excluded.stock;
```

9.6. Постоянные клиенты. *Подсказка ученику: UPDATE customers SET vip = 1 WHERE id IN (SELECT customer_id FROM orders GROUP BY customer_id HAVING COUNT(*) >= 4);*

```
UPDATE customers SET vip = 1 WHERE id IN (SELECT customer_id FROM orders GROUP BY
customer_id HAVING COUNT(*) >= 4);
```

9.7. Архив заказов. *Подсказка ученику: `CREATE TABLE orders_archive AS SELECT * FROM orders WHERE order_date < '2026-01-01';` — таблица создаётся и заполняется одной командой.*

```
CREATE TABLE orders_archive AS SELECT * FROM orders WHERE order_date <
'2026-01-01';
```

9.8. Новый поставщик и его id. *Подсказка ученику: `INSERT ... RETURNING id;` возвращает значения только что вставленной строки; `id` назначится автоматически.*

```
INSERT INTO suppliers (name, country) VALUES ('Новый поставщик', 'Россия')
RETURNING id;
```

9.9. Исправить опечатки. *Подсказка ученику: `UPDATE customers SET city = 'Тверь' WHERE city = 'Твер';` Повторите `SELECT`, чтобы убедиться, что опечаток не осталось.*

```
UPDATE customers SET city = 'Тверь' WHERE city = 'Твер';
```

9.10. Удалить отменённые заказы. *Подсказка ученику: Порядок важен: сначала дочерние строки (`order_items`) через подзапрос `IN (SELECT id FROM orders WHERE status = 'cancelled')`, затем родительские.*

```
DELETE FROM order_items WHERE order_id IN (SELECT id FROM orders WHERE status =
'cancelled');
DELETE FROM orders WHERE status = 'cancelled';
```

Тема 10. Создание таблиц и ограничения

10.1. Таблица товаров. *Подсказка ученику: `CREATE TABLE items (id INTEGER PRIMARY KEY, name TEXT NOT NULL, qty INTEGER NOT NULL DEFAULT 0, price REAL);` Столбец `INTEGER PRIMARY KEY` в SQLite нумеруется автоматически.*

```
CREATE TABLE items (id INTEGER PRIMARY KEY, name TEXT NOT NULL, qty INTEGER NOT
NULL DEFAULT 0, price REAL);
```

10.2. Названия не повторяются. *Подсказка ученику: `CREATE UNIQUE INDEX items_name_uq ON items(name);` Попытка вставить второй `Болт` теперь даст ошибку `UNIQUE constraint failed`.*

```
CREATE UNIQUE INDEX items_name_uq ON items(name);
INSERT INTO items (name, qty, price) VALUES ('Болт', 100, 1.5), ('Гайка', 200,
0.5);
```

10.3. Остаток не уходит в минус. *Подсказка ученику: Порядок: `CREATE TABLE items_new (...CHECK (qty >= 0)...)`, `INSERT INTO items_new SELECT * FROM items`, `DROP TABLE items`, `ALTER TABLE items_new RENAME TO items`, затем снова `CREATE UNIQUE INDEX`. Индекс удаляется вместе со старой таблицей.*

```
BEGIN;
CREATE TABLE items_new (id INTEGER PRIMARY KEY, name TEXT NOT NULL, qty INTEGER
NOT NULL DEFAULT 0 CHECK (qty >= 0), price REAL);
INSERT INTO items_new SELECT id, name, qty, price FROM items;
DROP TABLE items;
ALTER TABLE items_new RENAME TO items;
CREATE UNIQUE INDEX items_name_uq ON items(name);
COMMIT;
```

10.4. Места хранения. *Подсказка ученику: `ALTER TABLE items ADD COLUMN place_id INTEGER REFERENCES places(id) ON DELETE SET NULL;` — новый столбец с внешним ключом должен допускать `NULL`.*

```
CREATE TABLE places (id INTEGER PRIMARY KEY, name TEXT NOT NULL UNIQUE);
ALTER TABLE items ADD COLUMN place_id INTEGER REFERENCES places(id) ON DELETE SET
NULL;
```

10.5. Примечание. *Подсказка ученику: `ALTER TABLE items ADD COLUMN note TEXT;`*

```
ALTER TABLE items ADD COLUMN note TEXT;
```

10.6. Переименование столбца. *Подсказка ученику: `ALTER TABLE items RENAME COLUMN price TO unit_price;`*

```
ALTER TABLE items RENAME COLUMN price TO unit_price;
```

10.7. Журнал движений. *Подсказка ученику: `CHECK (delta <> 0)` запретит нулевое движение; `DEFAULT CURRENT_TIMESTAMP` подставит текущие дату и время UTC.*

```
CREATE TABLE moves (id INTEGER PRIMARY KEY, item_id INTEGER NOT NULL REFERENCES
items(id), delta INTEGER NOT NULL CHECK (delta <> 0), moved_at TEXT NOT NULL
DEFAULT CURRENT_TIMESTAMP);
```

10.8. Удалить временное. *Подсказка ученику: `DROP TABLE tmp_import;` — действие необратимо, поэтому сначала убедитесь, что таблица никому не нужна. `DROP TABLE IF EXISTS` не даёт ошибки, если таблицы нет.*

```
DROP TABLE tmp_import;
```

10.9. Метки товаров. *Подсказка ученику: Составной ключ записывают отдельным предложением: `PRIMARY KEY (item_id, tag)`.*

```
CREATE TABLE item_tags (item_id INTEGER NOT NULL REFERENCES items(id), tag TEXT
NOT NULL, PRIMARY KEY (item_id, tag));
```

10.10. Схема в файл. *Подсказка ученику: Команда `.schema` печатает все команды CREATE. Проверьте файл: `sqlite3 test.db < schema.sql` создаст новую базу с той же структурой.*

```
.output schema.sql
.schema
.output stdout
```

Тема 11. Проектирование схемы и нормализация

11.1. Противоречия в таблице. *Подсказка ученику: `SELECT dept, COUNT(DISTINCT dept_city) FROM sheet GROUP BY dept HAVING COUNT(DISTINCT dept_city) > 1;` Это и есть аномалия избыточных данных.*

```
sqlite3 uchet.db "SELECT dept, COUNT(DISTINCT dept_city) FROM sheet GROUP BY dept
HAVING COUNT(DISTINCT dept_city) > 1" > otvety/1
```

11.2. Таблица отделов. *Подсказка ученику: Сначала `SELECT dept, dept_city, COUNT(*) FROM sheet GROUP BY dept, dept_city ORDER BY dept, 3 DESC;` покажет, какой город встречается чаще. Затем создайте таблицу и вставьте строки: `INSERT INTO departments (name, city) SELECT ... FROM (подзапрос) t WHERE n = (подзапрос с MAX);`*

```
CREATE TABLE departments (id INTEGER PRIMARY KEY, name TEXT NOT NULL UNIQUE, city
TEXT NOT NULL);
INSERT INTO departments (name, city)
SELECT dept, dept_city FROM (SELECT dept, dept_city, COUNT(*) AS n FROM sheet
GROUP BY dept, dept_city) t
WHERE n = (SELECT MAX(n2) FROM (SELECT COUNT(*) AS n2 FROM sheet WHERE dept =
t.dept GROUP BY dept_city));
```

11.3. Таблица сотрудников. *Подсказка ученику: Идентификатор отдела получают соединением: `FROM sheet s JOIN departments d ON d.name = s.dept`. `DISTINCT` оставит по одной строке на сотрудника.*

```
CREATE TABLE employees (id INTEGER PRIMARY KEY, name TEXT NOT NULL UNIQUE, dept_id
INTEGER NOT NULL REFERENCES departments(id));
INSERT INTO employees (name, dept_id) SELECT DISTINCT s.employee, d.id FROM sheet
s JOIN departments d ON d.name = s.dept ORDER BY s.employee;
```

11.4. Таблица устройств. *Подсказка ученику: `INSERT INTO devices (serial, model) SELECT DISTINCT serial, device FROM sheet;` Столбец `device` в `sheet` содержит модель; он попадает в `devices.model`.*

```
CREATE TABLE devices (id INTEGER PRIMARY KEY, serial TEXT NOT NULL UNIQUE, model
TEXT NOT NULL);
INSERT INTO devices (serial, model) SELECT DISTINCT serial, device FROM sheet
ORDER BY serial;
```

11.5. Выдачи. *Подсказка ученику: `INSERT INTO assignments SELECT e.id, v.id, s.issued FROM sheet s JOIN employees e ON ... JOIN devices v ON ...;` Столбцы SELECT должны идти в порядке столбцов таблицы.*

```
CREATE TABLE assignments (employee_id INTEGER NOT NULL REFERENCES employees(id),
device_id INTEGER NOT NULL REFERENCES devices(id), issued_on TEXT NOT NULL,
PRIMARY KEY (device_id, issued_on));
INSERT INTO assignments SELECT e.id, v.id, s.issued FROM sheet s JOIN employees e
ON e.name = s.employee JOIN devices v ON v.serial = s.serial;
```

11.6. Контрольная выборка. *Подсказка ученику: SELECT e.name, v.model, v.serial, a.issued_on FROM assignments a JOIN employees e ON ... JOIN devices v ON ... ORDER BY v.serial LIMIT 5;*

```
sqlite3 uchet.db "SELECT employee, device, serial, issued FROM sheet ORDER BY
serial LIMIT 5" > otvety/6
```

11.7. Внешние ключи работают. *Подсказка ученику: Если проверка не проходит, в заданиях 3 и 5 не заданы REFERENCES. В SQLite внешние ключи нельзя добавить к существующей таблице командой ALTER TABLE: таблицу пересоздают.*

```
PRAGMA foreign_keys = ON;
INSERT INTO employees (name, dept_id) VALUES ('Тест', 999);   -- ошибка FOREIGN
KEY constraint failed
DELETE FROM departments WHERE id = 1;                          -- ошибка
```

11.8. Навыки сотрудников. *Подсказка ученику: Связь «многие ко многим» всегда оформляют отдельной таблицей из двух внешних ключей; пара внешних ключей и есть её первичный ключ.*

```
CREATE TABLE skills (id INTEGER PRIMARY KEY, name TEXT NOT NULL UNIQUE);
CREATE TABLE employee_skills (employee_id INTEGER NOT NULL REFERENCES
employees(id), skill_id INTEGER NOT NULL REFERENCES skills(id), PRIMARY KEY
(employee_id, skill_id));
INSERT INTO skills (name) VALUES ('SQL'), ('Linux'), ('Python');
INSERT INTO employee_skills (employee_id, skill_id) SELECT 1, id FROM skills WHERE
name IN ('SQL', 'Linux');
INSERT INTO employee_skills (employee_id, skill_id) SELECT 2, id FROM skills WHERE
name = 'SQL';
INSERT INTO employee_skills (employee_id, skill_id) SELECT 3, id FROM skills WHERE
name IN ('Linux', 'Python');
```

11.9. Представление для кадровиков. *Подсказка ученику: CREATE VIEW v_assignments AS SELECT ... AS employee, ... AS dept ...; Имена столбцов задают псевдонимами.*

```
CREATE VIEW v_assignments AS
SELECT e.name AS employee, d.name AS dept, d.city AS city, v.model AS model,
v.serial AS serial, a.issued_on AS issued
FROM assignments a JOIN employees e ON e.id = a.employee_id JOIN departments d ON
d.id = e.dept_id JOIN devices v ON v.id = a.device_id;
```

11.10. Удалить исходную таблицу. *Подсказка ученику: Таблица в 1НФ (значения атомарны), во 2НФ (ключ из одного столбца — частичных зависимостей нет), но не в 3НФ: dept_city зависит от dept, а не от ключа.*

```
DROP TABLE sheet;
-- ответ: 2 (нарушена 3НФ: город зависит от отдела, а не от ключа row_id)
```

Тема 12. Индексы и планы запросов

12.1. План без индекса. *Подсказка ученику: `sqlite3 logs.db "EXPLAIN QUERY PLAN SELECT * FROM access WHERE ip = '192.168.1.14';" > otvety/1`*

```
EXPLAIN QUERY PLAN SELECT * FROM access WHERE ip = '192.168.1.14';
```

12.2. Индекс по адресу. *Подсказка ученику: `CREATE INDEX idx_access_ip ON access(ip);` Имя индекса выберите сами.*

```
CREATE INDEX idx_access_ip ON access(ip);
```

12.3. План с индексом. *Подсказка ученику: Тот же `EXPLAIN QUERY PLAN`, но после выполнения задания 2. `SEARCH access USING INDEX ... (ip=?)` означает поиск по индексу.*

```
EXPLAIN QUERY PLAN SELECT * FROM access WHERE ip = '192.168.1.14';
```

12.4. Фильтр и сортировка. *Подсказка ученику: В составном индексе строки упорядочены сначала по `status`, затем по `ts`: поэтому при `status = 404` они уже идут в порядке `ts`.*

```
CREATE INDEX idx_access_status_ts ON access(status, ts);
```

12.5. Покрывающий индекс. *Подсказка ученику: Если все нужные столбцы есть в индексе, SQLite берёт значения прямо из него. Проверьте `EXPLAIN QUERY PLAN SELECT path FROM access WHERE ip = '192.168.1.14';`*

```
CREATE INDEX idx_access_ip_path ON access(ip, path);
```

12.6. Частичный индекс. *Подсказка ученику: Частичный индекс содержит только строки, подходящие под `WHERE`; запрос должен содержать такое же условие, чтобы SQLite им воспользовался.*

```
CREATE INDEX idx_access_err ON access(ip) WHERE status >= 400;
```

12.7. Индекс по выражению. *Подсказка ученику: `CREATE INDEX idx_access_hour ON access(substr(ts, 1, 13));` Запрос должен использовать то же выражение в точно такой же записи.*

```
CREATE INDEX idx_access_hour ON access(substr(ts, 1, 13));
```

12.8. Статистика. *Подсказка ученику: `ANALYZE;` без параметров анализирует всю базу.*

```
ANALYZE;
```

12.9. Лишние индексы. *Подсказка ученику: Сначала посмотрите все индексы таблицы: `.indexes access` или `PRAGMA index_list(access);`. Удалять нужно один из двух одинаковых.*

```
DROP INDEX idx_status_b;
```

12.10. Диапазон по времени. *Подсказка ученику: Для диапазона нужен индекс по `ts`; в плане будет `SEARCH access USING COVERING INDEX ... (ts>? AND ts<?)`.*

```
CREATE INDEX idx_access_ts ON access(ts);
EXPLAIN QUERY PLAN SELECT COUNT(*) FROM access WHERE ts >= '2026-10-11 05' AND ts
< '2026-10-11 06';
```

Тема 13. Представления и триггеры

13.1. Суммы заказов. *Подсказка ученику: Запрос с `JOIN` и `GROUP BY o.id` оформляется как `CREATE VIEW v_order_totals AS SELECT ...;` названия столбцов задайте псевдонимами.*

```
CREATE VIEW v_order_totals AS
SELECT o.id AS order_id, c.name AS customer, o.order_date AS order_date, o.status
AS status, SUM(i.qty * i.price) AS total
FROM orders o JOIN customers c ON c.id = o.customer_id JOIN order_items i ON
i.order_id = o.id GROUP BY o.id;
```

13.2. Выручка по клиентам. *Подсказка ученику: Условие `o.status <> 'cancelled'` ставится в `WHERE` внутри представления.*

```
CREATE VIEW v_customer_revenue AS
SELECT c.id AS customer_id, c.name AS name, SUM(i.qty * i.price) AS revenue FROM
customers c
JOIN orders o ON o.customer_id = c.id JOIN order_items i ON i.order_id = o.id
WHERE o.status <> 'cancelled' GROUP BY c.id;
```

13.3. Запрос к представлению. *Подсказка ученику: `SELECT name, revenue FROM v_customer_revenue ORDER BY revenue DESC, customer_id LIMIT 3;` — к представлению обращаются как к таблице.*

```
sqlite3 shop.db "SELECT name, revenue FROM v_customer_revenue ORDER BY revenue
DESC, customer_id LIMIT 3;" > otvety/3
```

13.4. Журнал цен. *Подсказка ученику: Тело триггера пишут между `BEGIN` и `END;`. Условие `WHEN OLD.price <> NEW.price` отсекает «пустые» обновления. Не забудьте точку с запятой после `END`.*

```
CREATE TABLE price_log (id INTEGER PRIMARY KEY, product_id INTEGER NOT NULL,
old_price INTEGER, new_price INTEGER, changed_at TEXT NOT NULL DEFAULT
CURRENT_TIMESTAMP);
CREATE TRIGGER trg_price_log AFTER UPDATE OF price ON products WHEN OLD.price <>
NEW.price
BEGIN INSERT INTO price_log (product_id, old_price, new_price) VALUES (OLD.id,
OLD.price, NEW.price); END;
```

13.5. Нельзя удалить клиента с заказами. *Подсказка ученику: `WHEN EXISTS (SELECT 1 FROM orders WHERE customer_id = OLD.id)` — условие срабатывания; `RAISE(ABORT, ...)` отменяет команду с сообщением.*

```
CREATE TRIGGER trg_no_delete_customer BEFORE DELETE ON customers WHEN EXISTS
(SELECT 1 FROM orders WHERE customer_id = OLD.id)
BEGIN SELECT RAISE(ABORT, 'у клиента есть заказы'); END;
```

13.6. Списание при продаже. *Подсказка ученику: В триггере `AFTER INSERT` доступна только строка `NEW`: `UPDATE products SET stock = stock - NEW.qty WHERE id = NEW.product_id;`*

```
CREATE TRIGGER trg_sell AFTER INSERT ON order_items
BEGIN UPDATE products SET stock = stock - NEW.qty WHERE id = NEW.product_id; END;
```

13.7. Остаток не ниже нуля. *Подсказка ученику: Триггер `BEFORE` срабатывает до изменения и может его отменить функцией `RAISE(ABORT, 'сообщение')`.*

```
CREATE TRIGGER trg_stock_nonneg BEFORE UPDATE OF stock ON products WHEN NEW.stock
< 0
BEGIN SELECT RAISE(ABORT, 'остаток не может быть отрицательным'); END;
```

13.8. Вставка через представление. *Подсказка ученику: Представления в SQLite только для чтения; триггер `INSTEAD OF` перехватывает попытку записи и выполняет действия над настоящей таблицей.*

```
CREATE VIEW v_customers_simple AS SELECT id, name, city FROM customers;
CREATE TRIGGER trg_ins_customer INSTEAD OF INSERT ON v_customers_simple
BEGIN INSERT INTO customers (name, city, registered, vip) VALUES (NEW.name,
NEW.city, date('now'), 0); END;
```

13.9. Дата регистрации неизменна. *Подсказка ученику: Конструкция `UPDATE OF registered` ограничивает срабатывание триггера изменением конкретного столбца.*

```
CREATE TRIGGER trg_registered_ro BEFORE UPDATE OF registered ON customers
BEGIN SELECT RAISE(ABORT, 'дату регистрации менять нельзя'); END;
```

13.10. Что создано. *Подсказка ученику: В `sqlite_master` каждый объект базы описан строкой: тип (`table`, `index`, `view`, `trigger`), имя и текст создания.*

```
SELECT type, name FROM sqlite_master WHERE type IN ('view', 'trigger') ORDER BY
type, name;
```

Тема 14. Транзакции, целостность и PRAGMA

14.1. Перевод в транзакции. *Подсказка ученику: Всё между `BEGIN;` и `COMMIT;` либо выполнится целиком, либо не выполнится вообще. Проверьте результат запросом `SELECT * FROM accounts;`.*

```
BEGIN;
UPDATE accounts SET balance = balance - 500 WHERE id = 1;
UPDATE accounts SET balance = balance + 500 WHERE id = 2;
COMMIT;
```

14.2. Откат. *Подсказка ученику: `ROLLBACK;` возвращает базу к состоянию на момент `BEGIN;`. Запрос `SELECT balance FROM accounts WHERE id = 3;` после отката даст исходное значение.*

```
BEGIN;
UPDATE accounts SET balance = balance - 300 WHERE id = 3;
SELECT balance FROM accounts WHERE id = 3;   -- 1200
ROLLBACK;
SELECT balance FROM accounts WHERE id = 3;   -- 1500
```

14.3. Ошибка ограничения. *Подсказка ученику: Сообщения об ошибках `sqlite3` выводит в поток ошибок (stderr); в bash его перенаправляют `2>`, в PowerShell — `2>`, но тогда файл будет в UTF-16: проверка это понимает.*

```
UPDATE accounts SET balance = balance - 9999 WHERE id = 4;   -- Runtime error:
CHECK constraint failed: balance >= 0
```

14.4. Внешние ключи по умолчанию. *Подсказка ученику: В SQLite внешние ключи по умолчанию **не проверяются**; их включают в каждом новом соединении.*

```
PRAGMA foreign_keys;                -- 0
PRAGMA foreign_keys = ON;
PRAGMA foreign_keys;                -- 1
```

14.5. Режим журнала WAL. *Подсказка ученику: `PRAGMA journal_mode = WAL;` допускает чтение во время записи и обычно ускоряет запись. Рядом с базой появятся файлы `bank.db-wal` и `bank.db-shm`.*

```
PRAGMA journal_mode = WAL;
```

14.6. Версия схемы. *Подсказка ученику: `PRAGMA user_version = 3;` хранит целое число в заголовке файла; читают его `PRAGMA user_version;`.*

```
PRAGMA user_version = 3;
```

14.7. Проверка целостности. *Подсказка ученику: Команда обходит всю базу и сообщает о найденных повреждениях; для исправной выводит одну строку `ok`.*

```
PRAGMA integrity_check;
```

14.8. Размер базы. *Подсказка ученику: Размер файла базы равен `page_size * page_count`. Сравните с размером файла в `ls -l bank.db`.*

```
PRAGMA page_size;
PRAGMA page_count;
```

14.9. Точки сохранения. *Подсказка ученику: `ROLLBACK TO имя` откатывает только изменения после точки сохранения, не завершая транзакцию. `RELEASE` убирает точку, `COMMIT` фиксирует остальное.*

```
BEGIN;
INSERT INTO accounts (owner, balance) VALUES ('Тест A', 100);
SAVEPOINT s1;
INSERT INTO accounts (owner, balance) VALUES ('Тест B', 200);
SAVEPOINT s2;
INSERT INTO accounts (owner, balance) VALUES ('Тест C', 300);
ROLLBACK TO s2;
RELEASE s1;
COMMIT;
```

14.10. Итоговая сумма. *Подсказка ученику: Должно получиться 5000 (исходная сумма) + 100 + 200 (счета, добавленные в задании 9).*

```
SELECT SUM(balance) FROM accounts;   -- 5300
```

Тема 15. Резервные копии, обмен данными и обслуживание базы

15.1. Горячая копия. *Подсказка ученику: `.backup` копирует базу консистентно, даже если с ней работают другие программы; простое копирование файла такой гарантии не даёт.*

```
sqlite3 shop.db ".backup backup-2026-10-11.db"
```

15.2. Текстовая копия. *Подсказка ученику: `.dump` печатает команды `CREATE` и `INSERT`, из которых можно собрать базу заново. В PowerShell 5.1 лучше `.output dump.sql` внутри `sqlite3`, чтобы не получить UTF-16.*

```
sqlite3 shop.db .dump > dump.sql
```

15.3. Восстановление. *Подсказка ученику: Восстановление «в другое место» — основной способ убедиться, что резервная копия пригодна. Рабочую базу при этом не трогают.*

```
sqlite3 restored.db < dump.sql
```

15.4. Экспорт в CSV. *Подсказка ученику: Режим `.mode csv` экранирует запятые и кавычки по правилам CSV. Не забудьте вернуть вывод на экран: `.output stdout`.*

```
.headers on
.mode csv
.output customers.csv
SELECT * FROM customers;
.output stdout
```

15.5. Импорт из CSV. *Подсказка ученику: Если `.import` не знает ключа `--csv`, используйте `.mode csv` и затем `.import new_products.csv products_new`. При создании таблицы по заголовку первая строка файла становится именами столбцов.*

```
.import --csv new_products.csv products_new
```

15.6. Сжать файл. *Подсказка ученику: `DROP TABLE` лишь помечает страницы свободными, а `VACUUM` перестраивает файл без пустот. Сравните размеры файла до и после: `ls -l shop.db` в Linux, `dir shop.db` в Windows.*

```
DROP TABLE junk;
VACUUM;
```

15.7. Сжатая копия. *Подсказка ученику: `VACUUM INTO` записывает перестроенную копию в новый файл, не изменяя исходную базу.*

```
VACUUM INTO 'compact.db';
```

15.8. Применить миграцию. *Подсказка ученику: Прочитайте сценарий, затем выполните его командой `.read`. Результат: `PRAGMA user_version;` вернёт 1.*

```
.read migrate_001.sql
```

15.9. Написать миграцию. *Подсказка ученику: Миграция — это обычный SQL-файл, который можно хранить в репозитории и применять по порядку номеров. Версия схемы хранится в `user_version`.*

```
-- migrate_002.sql
CREATE TABLE notes (id INTEGER PRIMARY KEY, customer_id INTEGER NOT NULL
REFERENCES customers(id), text TEXT NOT NULL, created_at TEXT NOT NULL DEFAULT
CURRENT_TIMESTAMP);
PRAGMA user_version = 2;
```

15.10. Сверка копии. *Подсказка ученику: Склейте четыре запроса через `UNION ALL`, каждый возвращает пару `('имя', COUNT(*))`. Выполните запрос именно на файле копии.*

```
sqlite3 backup-2026-10-11.db "SELECT 'customers', COUNT(*) FROM customers UNION
ALL SELECT 'order_items', COUNT(*) FROM order_items UNION ALL SELECT 'orders',
COUNT(*) FROM orders UNION ALL SELECT 'products', COUNT(*) FROM products;" >
otvety/10
```

Тема 16. SQL в сценариях, безопасность и анализ журналов

16.1. Сценарий: самые активные адреса. *Подсказка ученику: `sqlite3.connect('logs.db')`, затем `for ip, n in c.execute('SELECT ...'): print(ip, n)`. Запрос такой же, как при работе в консоли.*

```
import sqlite3
c = sqlite3.connect('logs.db')
for ip, n in c.execute("SELECT ip, COUNT(*) AS n FROM access GROUP BY ip ORDER BY
n DESC, ip LIMIT 5"):
    print(ip, n)
```

16.2. Параметр запроса. *Подсказка ученику: Второй аргумент `execute` — кортеж значений. Не забудьте запятую в кортеже из одного элемента: `(sys.argv[1],)`.*

```
import sqlite3, sys
c = sqlite3.connect('logs.db')
print(c.execute('SELECT COUNT(*) FROM access WHERE ip = ?',
(sys.argv[1],)).fetchone()[0])
```

16.3. Инъекция. *Подсказка ученику: Нужно закрыть кавычку, добавить условие, которое всегда истинно, и «съесть» оставшуюся кавычку: `x' OR '1'='1`. Получится `WHERE name = 'x' OR '1'='1'`.*

```
x' OR '1'='1
```

16.4. Исправить уязвимость. *Подсказка ученику: С параметром весь ввод считается значением, а не частью запроса: специальные символы теряют силу.*

```
import sqlite3, sys
c = sqlite3.connect('security.db')
for (s,) in c.execute('SELECT secret FROM users WHERE name = ?', (sys.argv[1],)):
    print(s)
```

16.5. Отчёт в CSV. *Подсказка ученику: `csv.writer(f).writerow([...])` для заголовка и `writerows(rows)` для данных. Файл открывают с `newline=''`.*

```
import csv, sqlite3
c = sqlite3.connect('logs.db')
rows = c.execute('SELECT ip, COUNT(*) AS hits FROM access GROUP BY ip HAVING hits
> 20 ORDER BY hits DESC, ip').fetchall()
with open('report.csv', 'w', newline='', encoding='utf-8') as f:
    w = csv.writer(f)
    w.writerow(['ip', 'hits'])
    w.writerows(rows)
```

16.6. Всё или ничего. *Подсказка ученику: Блок `with c:` (где `c` — соединение) фиксирует изменения при успехе и откатывает при исключении. Исключение перехватите и завершите сценарий `sys.exit(1)`.*

```
import csv, sqlite3, sys
c = sqlite3.connect('security.db')
rows = list(csv.DictReader(open(sys.argv[1], encoding='utf-8')))
try:
    with c:
        for r in rows:
```

```
            c.execute('INSERT INTO users (name, secret) VALUES (?, ?)',
            (r['name'], r['secret']))
except sqlite3.IntegrityError as e:
    print('Ошибка, всё отменено:', e)
    sys.exit(1)
```

16.7. Подбор пароля. *Подсказка ученику: `WHERE method = 'POST' AND path = '/login' AND status = 401 GROUP BY ip HAVING COUNT(*) >= 20`*

```
sqlite3 logs.db "SELECT ip, COUNT(*) FROM access WHERE method = 'POST' AND path =
'/login' AND status = 401 GROUP BY ip HAVING COUNT(*) >= 20" > otvety/7
```

16.8. Сканер. *Подсказка ученику: `WHERE status = 404 GROUP BY ip HAVING COUNT(*) > 50`*

```
sqlite3 logs.db "SELECT ip, COUNT(*) FROM access WHERE status = 404 GROUP BY ip
HAVING COUNT(*) > 50" > otvety/8
```

16.9. Самый медленный запрос. *Подсказка ученику: `SELECT path FROM access ORDER BY rt DESC LIMIT 1;`*

```
sqlite3 logs.db "SELECT path FROM access ORDER BY rt DESC, id LIMIT 1" > otvety/9
```

16.10. Трафик по часам. *Подсказка ученику: `SELECT SUBSTR(ts, 1, 13) AS h, SUM(bytes) FROM access GROUP BY h ORDER BY h;`*

```
sqlite3 logs.db "SELECT SUBSTR(ts, 1, 13) AS h, SUM(bytes) FROM access GROUP BY h
ORDER BY h" > otvety/10
```

Тема 17. Итоговый проект: заявки службы поддержки

17.1. Открытые по приоритетам. *Подсказка ученику: `SELECT priority, COUNT(*) FROM tickets WHERE closed IS NULL GROUP BY priority ORDER BY priority;`*

```
sqlite3 helpdesk.db "SELECT priority, COUNT(*) FROM tickets WHERE closed IS NULL
GROUP BY priority ORDER BY priority" > otvety/1
```

17.2. Проблемные серверы. *Подсказка ученику: Соедините `tickets` с `servers`, сгруппируйте по серверу и отберите группы через `HAVING`.*

```
sqlite3 helpdesk.db "SELECT s.name, COUNT(*) AS n FROM tickets t JOIN servers s ON
s.id = t.server_id GROUP BY s.id HAVING n >= 5 ORDER BY n DESC, s.name" > otvety/2
```

17.3. Скорость инженеров. *Подсказка ученику: `julianday(closed) - julianday(opened)` — разность в сутках; умножьте на 24, чтобы получить часы.*

```
sqlite3 helpdesk.db "SELECT e.name, ROUND(AVG((julianday(t.closed) -
julianday(t.opened)) * 24), 1) FROM tickets t JOIN engineers e ON e.id =
t.engineer_id WHERE t.closed IS NOT NULL GROUP BY e.id ORDER BY e.id" > otvety/3
```

17.4. Справочник сроков. *Подсказка ученику: Справочник выносят в отдельную таблицу, чтобы срок хранился в одном месте.*

```
CREATE TABLE priorities (code TEXT PRIMARY KEY, sla_hours INTEGER NOT NULL);
INSERT INTO priorities VALUES ('P1', 4), ('P2', 24), ('P3', 72);
```

17.5. Просроченные заявки. *Подсказка ученику: Нужны три таблицы: `tickets`, `servers` (название) и `priorities` (срок). Условие: `closed IS NULL` и часы с открытия `> sla_hours`.*

```
CREATE VIEW v_overdue AS
SELECT t.id AS ticket_id, t.title AS title, s.name AS server, t.priority AS
priority, ROUND((julianday('2026-10-11 12:00') - julianday(t.opened)) * 24, 1) AS
hours_open
FROM tickets t JOIN servers s ON s.id = t.server_id JOIN priorities p ON p.code =
t.priority
WHERE t.closed IS NULL AND (julianday('2026-10-11 12:00') - julianday(t.opened)) *
24 > p.sla_hours;
```

17.6. Индекс для открытых заявок. *Подсказка ученику: Частичный индекс не содержит закрытых заявок, поэтому он маленький и быстрый.*

```
CREATE INDEX idx_tickets_open ON tickets(priority) WHERE closed IS NULL;
```

17.7. Закрыть без инженера нельзя. *Подсказка ученику: `WHEN NEW.closed IS NOT NULL AND NEW.engineer_id IS NULL` и `SELECT RAISE(ABORT, 'сообщение');` в теле триггера.*

```
CREATE TRIGGER trg_close_needs_engineer BEFORE UPDATE OF closed ON tickets
WHEN NEW.closed IS NOT NULL AND NEW.engineer_id IS NULL
BEGIN SELECT RAISE(ABORT, 'заявку без инженера закрыть нельзя'); END;
```

17.8. Срочные заявки — инженеру 1. *Подсказка ученику: Условие `WHERE priority = 'P1' AND closed IS NULL` защищает остальные строки; проверьте его `SELECT`-ом до `UPDATE`.*

```
BEGIN;
UPDATE tickets SET engineer_id = 1 WHERE priority = 'P1' AND closed IS NULL;
COMMIT;
```

17.9. Резервная копия проекта. *Подсказка ученику: `.backup` из консоли `sqlite3` — как в теме о копиях.*

```
sqlite3 helpdesk.db ".backup helpdesk-backup.db"
```

17.10. Отчёт руководителю. *Подсказка ученику: Файл можно выполнить целиком: `sqlite3 helpdesk.db < queries.sql`.*

```
-- queries.sql
SELECT priority, COUNT(*) FROM tickets WHERE closed IS NULL GROUP BY priority;
SELECT s.name, COUNT(*) FROM tickets t JOIN servers s ON s.id = t.server_id GROUP
BY s.id;
SELECT * FROM v_overdue;
-- ответ: SELECT COUNT(*) FROM v_overdue;
```

Тема 18. JSON, полнотекстовый поиск и подключение баз

18.1. Сколько ошибок. *Подсказка ученику: `json_extract(payload, '$.level')` достаёт поле `level` из JSON. Путь начинается со знака `$`.*

```
sqlite3 events.db "SELECT COUNT(*) FROM events WHERE json_extract(payload,
'\$.level') = 'error'" > otvety/1
```

18.2. Какие серверы пишут события. *Подсказка ученику: `SELECT DISTINCT json_extract(payload, '$.host') FROM events ORDER BY 1;`*

```
sqlite3 events.db "SELECT DISTINCT json_extract(payload, '\$.host') FROM events
ORDER BY 1" > otvety/2
```

18.3. Среднее время ответа. *Подсказка ученику: Функцию JSON можно использовать в `GROUP BY` так же, как обычный столбец; удобно задать псевдоним.*

```
sqlite3 events.db "SELECT json_extract(payload, '\$.host') AS h,
ROUND(AVG(json_extract(payload, '\$.ms')), 1) FROM events GROUP BY h ORDER BY h" >
otvety/3
```

18.4. События с тегом db. *Подсказка ученику: `json_each(payload, '$.tags')` разворачивает массив в таблицу из строк; значение — в столбце `value`. Соедините её с `events`.*

```
sqlite3 events.db "SELECT e.id FROM events e, json_each(e.payload, '\$.tags') t
WHERE t.value = 'db' ORDER BY e.id" > otvety/4
```

18.5. Вычисляемый столбец. *Подсказка ученику: Вычисляемый столбец хранит не данные, а формулу; `VIRTUAL` вычисляется при чтении, `STORED` — при записи.*

```
ALTER TABLE events ADD COLUMN level TEXT GENERATED ALWAYS AS
(json_extract(payload, '$.level')) VIRTUAL;
```

18.6. Индекс по уровню. *Подсказка ученику: Индекс по вычисляемому столбцу ускоряет отбор событий по уровню, не требуя разбирать JSON в каждой строке.*

```
CREATE INDEX idx_events_level ON events(level);
```

18.7. Полнотекстовый индекс. *Подсказка ученику: `CREATE VIRTUAL TABLE … USING fts5(…)` создаёт таблицу поиска; данные в неё вставляют обычным `INSERT … SELECT`.*

```
CREATE VIRTUAL TABLE docs_fts USING fts5(title, body);
INSERT INTO docs_fts (title, body) SELECT title, body FROM docs;
```

18.8. Поиск по тексту. *Подсказка ученику: `SELECT title FROM docs_fts WHERE docs_fts MATCH 'backup AND restore' ORDER BY title;` — операторы `AND`, `OR`, `NOT` пишутся заглавными.*

```
sqlite3 events.db "SELECT title FROM docs_fts WHERE docs_fts MATCH 'backup AND
restore' ORDER BY title;" > otvety/8
```

18.9. Данные из двух баз. *Подсказка ученику: После `ATTACH … AS shop` таблицы второй базы доступны как `shop.имя_таблицы`. В консоли `sqlite3 events.db` команду `ATTACH` вводят до запроса.*

```
ATTACH 'shop.db' AS shop;
SELECT DISTINCT cu.name FROM events e JOIN shop.customers cu ON cu.id =
json_extract(e.payload, '$.user')
WHERE json_extract(e.payload, '$.level') = 'error' ORDER BY cu.name;
```

18.10. Пометить просмотренные. *Подсказка ученику: `json_set(json, путь, значение)` возвращает новый JSON с добавленным или изменённым полем; результат записывают обратно в `payload`.*

```
UPDATE events SET payload = json_set(payload, '$.reviewed', 1) WHERE
json_extract(payload, '$.level') = 'error';
```

Как проверен полигон SQL. В полигоне 590 заданий в 18 темах. Самопроверка (`python3 selftest.py`) создаёт чистую тему, убеждается, что ни одно задание не выполнено, применяет эталонные решения и убеждается, что выполнены все; она повторялась на разных наборах случайных данных без расхождений. Дополнительно эталонные решения всех заданий, записанные одной командой `sqlite3 … > otvety/N` (темы 1–8 почти целиком), выполнены в настоящей консоли `sqlite3` 3.53 (Windows, оболочка bash) и приняты программой проверки; при этом найдены и исправлены две ошибки: слово `notnull` нужно брать в квадратные скобки, а пробелы внутри строковых литералов в команде нельзя сворачивать. Задания, в которых ученик вводит команды консоли (`.backup`, `.dump`, `.import`, `.once`, `.read`, `VACUUM INTO`), проверены через равнозначные вызовы Python, а синтаксис основных команд проверен вручную в консоли 3.53; на Windows установлено, что `sqlite3 -csv … > файл.csv` даёт двойной возврат каретки (в справочнике это отмечено). Прогон полного курса на Linux с консолью `sqlite3` из пакета Ubuntu остаётся за преподавателем.

Приложение В

Справочник команд системного администратора

В.1 Один вопрос – несколько средств проверки

Таблица помогает выбрать инструмент после понимания вопроса. Она не является обещанием одинаковых параметров и не заменяет лабораторные. Изменяющие команды запускают с нужной личностью, проверяя цель и результат.

Вопрос	Linux / Bash	Windows / PowerShell или CMD
Где я?	pwd; cd	Get-Location; Set-Location; в CMD: cd.
Что в папке?	ls -la	Get-ChildItem -Force; в CMD: dir.
Прочитать текст	cat; less	Get-Content; в CMD: type.
Скопировать файл	cp	Copy-Item; в CMD: copy.
Найти файл	find	Get-ChildItem -Recurse -Filter.
Найти строку	grep -F	Select-String -SimpleMatch.
Кто выполняет?	whoami; id	whoami /user; whoami /groups.
Какие записи?	getent passwd; getent group	Get-LocalUser; Get-LocalGroup.
Какие права?	ls -l; namei -l; getfacl	Get-Acl; icacls.
Как изменить права?	chmod; chown; setfacl	icacls; средства «Безопасность».
Какие процессы?	ps; top	Get-Process; Диспетчер задач.
Какая память?	free -h	Get-CimInstance; Диспетчер задач.
Какие подключения дисков?	lsblk -f; findmnt; df -h	Get-Disk; Get-Partition; Get-Volume.
Какие службы?	systemctl status	Get-Service; Win32_Service.
Какие события?	journalctl	Get-WinEvent; Просмотр событий.
Какой IP?	ip -br address	Get-NetIPConfiguration; ipconfig /all.
Какой маршрут?	ip route	Get-NetRoute; route print.
Как разрешается имя?	resolvectl; getent hosts	Resolve-DnsName; nslookup.
Кто слушает?	ss -lntp	Get-NetTCPConnection.
Как проверить службу?	curl; ssh	Test-NetConnection; браузер; ssh.
Как проверить копию?	tar -t; восстановление; sha256sum	Expand-Archive; Get-FileHash.

B.2 Минимальный отчёт и оценивание

Отчёт содержит цель, паспорт среды, прогноз, точные команды, личность процесса, наблюдение, объяснение и повторную проверку после изменения. Скриншот полезен как доказательство конкретного состояния, но не заменяет адресный план, текст команды и интерпретацию. Для доступа обязательны разрешённая и запрещённая операции; для копии – восстановление.

Критерий	Баллы	Что проверяется
Понимание задачи	15	Объяснены роли устройств, пользователей и приложений.
План и воспроизводимость	20	Версии, пути, адреса, роли и шаги позволяют повторить работу.
Работоспособность	25	Проверены требуемые операции и реальные ответы служб.
Разграничение доступа	15	Показаны разрешения и отказы от нужных личностей.
Диагностика	15	Причина подтверждена наблюдением; изменение минимально.
Восстановление и передача	10	Есть проверенная копия и инструкция следующему администратору.

Это предложенная рубрика на 100 баллов для комплексных проектов; преподаватель может адаптировать её. Для простой лабораторной критерии, которых ещё не изучали, не требуются.

B.3 Что изучать после базового курса

Следующие темы развивают основы: загрузка ОС и восстановление, LVM и дисковые массивы, шифрование данных, сертификаты и TLS, Active Directory/LDAP и групповые политики, управление конфигурациями, контейнеризация, мониторинг и централизованные журналы. Они требуют отдельных стендов и задач. В этой книге они обозначены как направления продолжения, а не как уже выполненные лабораторные. Прежде чем добавлять новую технологию, сформулируйте её роль, полномочия, зависимости, способ наблюдения и план возврата.

ПРИЛОЖЕНИЯ

Книга II. Сетевое администрирование

Справочники по сети, шаблоны отчётов, контрольные расчёты, ключи к вопросам, словарь, источники и границы применимости.

Приложение C

Карманный справочник и безопасный порядок действий

C.1 Среда передачи и кабели

Среда	Где используется в курсе	Что проверить
Copper Straight-Through	ПК–коммутатор, сервер–коммутатор, маршрутизатор–коммутатор	Выбран Ethernet-порт, линия активна.
Copper Cross-Over	Прямое соединение сходных Ethernet-устройств, например R1–R2	Возможности Auto-MDIX отличаются; автоматический выбор допустим.
Console	RS-232 ПК–Console IOS-узла	Это локальное управление; кабель не создаёт IP-сеть. На ПК открывается Terminal.
Fiber	Опциональные устройства с подходящими оптическими портами	Оба порта и модули должны поддерживать этот тип соединения.
Serial DCE/DTE	Дополнительный WAN-опыт ниже	На DCE нужен источник синхронизации; проверяются encapsulation и адреса.
Wireless	Laptop–AP/WRT/LAP	Сначала SSID и защита, затем DHCP и маршрут.

Автоматический выбор кабеля удобен, когда нужно проверить идею. Для лабораторной по среде передачи выбирайте вручную и объясняйте выбор. Не ставьте адрес на console-порт: он служит иной цели.

C.2 Состояния интерфейса

Состояние	Возможное объяснение	Следующая проверка
administratively down/down	Интерфейс выключен настройкой	running-config, shutdown/no shutdown.
down/down	Нет физического соединения, питания или подходящей среды	Кабель, порт, питание обеих сторон.
up/down	Физическая часть есть, протокол линии не поднят	Тип интерфейса, encapsulation, параметры L2.

up/up	Интерфейс готов к передаче	Далее адреса, VLAN, маршруты и политика.
SVI down	VLAN отсутствует или нет подходящего активного порта	VLAN, trunk allowed, состояние STP, no shutdown.
err-disabled / secure-shutdown	Нарушена защита или другой механизм отключил порт	Port Security, BPDU Guard и журнал, затем осознанное восстановление.

Таблица предлагает гипотезы, а не однозначный диагноз. Например, SVI может зависеть от состояний нескольких портов и особенностей модели. Успешное up/up не подтверждает правильность маски и маршрута.

C.3 Команды по задачам

Что выяснить	Команды EXEC
Адреса и интерфейсы IPv4	`show ip interface brief; show interfaces; show ip interface`
IPv6	`show ipv6 interface brief; show ipv6 route; show ipv6 neighbors`
VLAN и trunk	`show vlan brief; show interfaces trunk; show interfaces switchport`
Обучение MAC	`show mac address-table`
STP	`show spanning-tree vlan 10`
EtherChannel	`show etherchannel summary`
Port Security	`show port-security; show port-security address`
Маршруты	`show ip route; show ip protocols`
OSPF	`show ip ospf neighbor; show ip ospf interface; show ip ospf database`
EIGRP	`show ip eigrp neighbors; show ip eigrp topology`
DHCP	`show ip dhcp binding; show ip dhcp pool`
ACL	`show access-lists; show ip interface gigabitEthernet 0/0`
NAT	`show ip nat translations; show ip nat statistics`
Управление	`show ip ssh; show cdp neighbors detail; show logging`
Время	`show clock; show ntp associations; show ntp status`
Сохранение	`copy running-config startup-config`

Если команда отсутствует, сначала проверьте режим и ?. Один симулятор может реализовывать разные наборы команд для разных моделей. Полный вывод обычно полезнее его обрезанного снимка.

C.4 Часто используемые службы

Служба	Типичный транспорт/порт	Что ещё учитывать
SSH	TCP 22	Требуются пользователь, ключи и разрешение VTY.
DNS	UDP/TCP 53	Корректная запись и DNS в параметрах клиента.
DHCPv4	UDP 67 сервер, 68 клиент	Начальный источник может быть 0.0.0.0.
HTTP	TCP 80	HTTP включён на сервере.
HTTPS	TCP 443	Модель TLS в симуляторе ограничена.
FTP control	TCP 21	Данные используют отдельное соединение.
SMTP	TCP 25 в нашем стенде	Вне учебной модели встречаются другие порты и режимы отправки.
POP3	TCP 110 в нашем стенде	В книге используется упрощённое получение почты.
TFTP	UDP 69 для начального запроса	Дальнейшая передача использует согласованные идентификаторы/порты.
NTP	UDP 123	Адрес сервера не равен подтверждённой синхронизации.
Syslog	Обычно UDP 514 в стенде	Реальные системы могут применять другие транспорты.
ICMP / ICMPv6	Не TCP/UDP	У ping нет транспортного порта 80 или 53.

C.5 Дополнительный WAN-эксперимент: Serial и PPP

Этот опыт дополняет Ethernet-маршрутизацию, но не является обязательной 41-й работой. На двух совместимых маршрутизаторах в Physical выключите питание, установите поддерживаемый serial-модуль, например HWIC-2T для подходящего слота модели, затем включите питание. Выберите Serial DCE и соедините serial-порты. Перед вводом найдите фактические имена через `show ip interface brief`; обозначение S0/0/0 ниже – пример.

На стороне DCE настройте:

```
interface serial 0/0/0
 ip address 10.99.0.1 255.255.255.252
 encapsulation ppp
 clock rate 64000
 no shutdown
exit
```

На стороне DTE задайте 10.99.0.2/30, ту же encapsulation ppp и no shutdown без clock rate. При поддержке `show controllers serial 0/0/0` проверьте DCE/DTE. Проверяйте up/up и ping .1–.2. Для управляемой ошибки поставьте HDLC только на одной стороне, объясните несогласованность L2 и восстановите PPP. Различайте `bandwidth` для расчётных/протокольных целей и `clock rate` как источник тактовой частоты DCE: это не взаимозаменяемые команды.

Приложение D

Шаблоны отчётов, план занятий и оценивание

D.1 Отчёт по лабораторной

Отчёт должен позволять другому человеку воспроизвести результат. Не нужно переписывать все страницы книги; достаточно данных собственного стенда и проверок.

1. **Паспорт.** Номер работы, дата, версия Packet Tracer, модели устройств, имя проекта.
2. **Цель и прогноз.** Какой результат должен получиться и какой поток должен быть запрещён, если есть политика.
3. **Схема и портовая таблица.** У обоих концов каждой линии указаны порты, VLAN и тип соединения.
4. **Адреса.** Устройство, интерфейс, IP, маска/префикс, шлюз, DNS, способ выдачи.
5. **Настройки.** Только значимые команды и значения форм; публичные учебные пароли не превращайте в свои личные пароли.
6. **Приёмка.** Источник, назначение, протокол, ожидание, наблюдение и ссылка на доказательство.
7. **Неисправность.** Симптом, гипотезы, команда проверки, найденная причина и минимальная правка.
8. **Вывод.** Что удалось доказать и какие ограничения модели остались.

№	Источник	Назначение	Протокол	Ожидается	Факт
1	PC-A	Свой шлюз	ICMP	Разрешён	Ответы
2	GUEST	Веб-сервер	TCP/HTTP	Разрешён	Страница
3	GUEST	Admin	ICMP	Запрещён	Нет ответа + ACL
4	PC-A	Имя портала	DNS + HTTP	Разрешён	A-запись + страница

Для запрета отсутствие ответа без дополнительных данных не всегда достаточно: узел мог быть выключен. Сначала подтвердите его работоспособность с разрешённого источника и найдите подходящее правило/счётчик или наблюдение Simulation.

D.2 Журнал поиска ошибки

Образец короткой записи

Симптом: PC VLAN 20 на S2 не достигает PC той же VLAN на S1, VLAN 10 работает. Гипотезы: неправильный access-порт либо отсутствие VLAN 20 на trunk. Проверка show vlan brief: ПК действительно в VLAN 20; show interfaces trunk на S2: allowed list не содержит 20. Правка: вернуть согласо-

ванный список. Результат: повторный ping VLAN 20 проходит, контрольный ping VLAN 10 остаётся исправным.

Разделяйте факты и предположения. «Наверное, DNS» – гипотеза; «по IP работает, при свежем запросе DNS возвращает .99 вместо .10» – наблюдение, которое её проверяет.

D.3 Пример учебного календаря

Для курса в группе удобно 20 учебных блоков по 4–6 часов самостоятельной и аудиторной работы. Проекты требуют дополнительного времени; сокращённый маршрут можно пройти без факультативных усложнений.

Блок	Работы	Контрольный навык
1	01–02	Собрать и сохранить LAN, прочитать адреса.
2	03–04	Объяснить ARP и влияние маски.
3	05–06	Составить VLSM и сохранить IOS-конфигурацию.
4	07–08	Управлять по SSH, маршрутизировать две LAN.
5	09–10	Проверить пути и границы статического резерва.
6	11–12	Настроить access и trunk, подтвердить изоляцию.
7	13–14	Сравнить подинтерфейсы и SVI.
8	15–16	Объяснить STP и логический канал.
9	17–18	Найти нарушение защиты и документировать соседства.
10	19–20	Проверить DHCP локально и через relay.
11	21–22	Отделить DNS, транспорт и приложения.
12	23–24	Проверить разрешённые и запрещённые потоки.
13	25–26	Объяснить Static NAT и PAT.
14	27–28	Проверить соседства, стоимость и сходимость OSPF.
15	29–30	Сравнить области и альтернативные протоколы.
16	31–32	Проверить IPv6 локально и через соседний маршрутизатор.
17	33–35	Настроить WLAN и зарегистрировать IoT.
18	36–37	Сделать резервную копию и расследовать отказы.
19	38–39	Приёмка кампуса и филиальной сети.
20	40	Самостоятельный проект и защита.

D.4 Рубрика оценивания

Критерий	Баллы	За что начисляются

Схема и адресный план	20	Порты, VLAN, непересекающиеся подсети, корректные шлюзы.
Реализация	25	Конфигурации отвечают требованиям; сохранены и воспроизводимы.
Проверки и политика	25	Есть позитивные и негативные тесты, наблюдения служб и пути.
Диагностика	20	Причина подтверждена, правка минимальна, контрольный тест сохранён.
Документация и объяснение	10	Другой человек воспроизводит работу; студент объясняет путь пакета.

Ориентир: 85–100 – уверенное владение, 70–84 – рабочий результат с отдельными пробелами, 50–69 – требуется повторение, ниже 50 – вернуться к последнему устойчивому навыку. Для простой вводной работы критерии масштабируются: не оценивайте OSPF в лабораторной 02. Критический отказ обязательной функции итогового проекта должен быть исправлен даже при большом суммарном балле.

Приложение E

Контрольные расчёты и разборы

E.1 IPv4 без калькулятора

Для 192.168.10.77/26 последний октет адреса – `01001101`, маски – `11000000`. AND даёт `01000000`, то есть 64. Узловая часть состоит из шести бит: от 000001 до 111110, то есть адреса .65–.126. Все единицы узловой части дают .127. Этот расчёт объясняет формулу, а не просто запоминает диапазон.

Для самостоятельного задания 05 один корректный вариант:

Требование	Сеть	Узлы / broadcast
120 ПК + шлюз	172.16.8.0/25	.8.1–.8.126 / .8.127
60 ПК + шлюз	172.16.8.128/26	.8.129–.8.190 / .8.191
28 ПК + шлюз	172.16.8.192/27	.8.193–.8.222 / .8.223
10 ПК + шлюз	172.16.8.224/28	.8.225–.8.238 / .8.239
Канал 1	172.16.8.240/30	.8.241–.8.242 / .8.243
Канал 2	172.16.8.244/30	.8.245–.8.246 / .8.247
Канал 3	172.16.8.248/30	.8.249–.8.250 / .8.251

Внутри 172.16.8.0/23 остаются 172.16.8.252/30 и весь 172.16.9.0/24. Не называйте остаток одной /24: свободное пространство состоит из двух разных блоков.

E.2 Ключ к вопросам первых работ

01–03: линия, кадр и пакет.
Линия – состояние физического соединения; кадр – передача в канальном сегменте; пакет – IP-передача между узлами. ПК в общей подсети не нуждаются в шлюзе для взаимного обмена. ARP разрешает адрес локального соседа или шлюза, а не произвольного удалённого ПК через все маршрутизаторы.

04–05: маска и VLSM.
Смена маски меняет локальность назначения. Количество пригодных адресов включает шлюз и прочую инфраструктуру. Рост одной подсети нельзя удовлетворить маской, которая перекрывает следующую.

06–07: сохранение и управление.
Running-config действует сейчас, startup-config используется при запуске. SVI требует активной VLAN-среды. SSH дополнительно требует имя/domain, RSA-ключи, локальную учётную запись и VTY с login local/transport input ssh.

08–10: маршруты.
В двух непосредственно подключённых LAN не нужны статические маршруты. В цепочке нужен путь ответа. Floating route выбирается, когда основной маршрут перестаёт быть пригодным; достижимость next hop не равна достижимости всех удалённых сетей.

E.3 Ключ к коммутации

11–12: VLAN и trunk.
ПК одной IP-подсети в разных VLAN не общаются напрямую без специальной маршрутизирующей конструкции. Для переноса VLAN нужны её наличие, trunk, разрешённый список и forwarding-состояние. Номер native согласовывают на концах.

13–14: шлюзы.
VLAN подинтерфейса задаёт encapsulation dot1Q, а не его суффикс. На L3-коммутаторе между SVI нужен ip routing. Удалённый default gateway L2-управления не создаёт L3-коммутацию VLAN.

15–16: резервирование.
STP предотвращает L2-петлю, выключая передачу на избыточном пути. EtherChannel объединяет совместимые линии; passive/passive не инициирует LACP. Одиночный поток не подтверждает сложение полос.

17–18: доступ и discovery.
Secure-shutdown – результат политики, а не обязательно плохой кабель. CDP сообщает о соседстве, но выключенный CDP не выключает IP-передачу. Подпись порта может устареть после перестановки кабеля.

E.4 Ключ к службам и безопасности

19–20: DHCP.
Выдача адреса и работа DNS – разные проверки. Relay размещают на клиентском интерфейсе; серверу нужен пул соответствующей клиентской подсети и обратная связность.

21–22: приложения.
По IP работает, по имени нет – повод проверить DNS и кэш. Успешный ping не означает включённый HTTP или правильный пароль почты. FTP control и data нужно различать при фильтрации.

23–24: ACL.
Применение и направление обязательны. Первое совпадение завершает проверку. Запрещённый ping к серверу при разрешённом HTTP может быть правильным результатом. ACL без учёта DHCP ломает начальную выдачу даже при permit для уже выданной подсети.

25–26: NAT.
Static NAT обеспечивает устойчивое сопоставление; overload различает сессии. Маршруты и NAT нужны совместно. ACL для NAT выбирает преобразуемые источники и не является интерфейсным фильтром сама по себе.

E.5 Ключ к расширенным работам

27–30: динамическая маршрутизация.
OSPF process ID локален, area и параметры общего канала согласованы. Путь сравнивается по сумме стоимостей. ABR связывает области с backbone. AS классического EIGRP должен совпадать у соседей; RIPv2 передаёт префиксы и использует hop count.

31–32: IPv6.
IPv6 forwarding требует ipv6 unicast-routing. NDP использует ICMPv6, не ARP. Link-local имеет канальную область действия; для next hop важен интерфейс. Маршрут ответа нужен и в IPv6.

33–35: WLAN и IoT.
Ассоциация не гарантирует IP-параметры. WLC/AP management и клиентское отображение WLAN проверяются отдельно. Сетевой IP IoT не равен регистрации под правильным пользователем.

36–37: обслуживание.
NTP-строка не является доказательством синхронизации. Восстановление в running-config может сливаться с текущим состоянием. Диагноз требует наблюдения, которое различает гипотезы, и контрольного теста после правки.

38–40: проекты.
Проверяется результат требований и политика, а не совпадение схемы с единственным образцом. Резервный путь должен сохранять ACL, NAT и службы. IPv4-политика не переносится на IPv6 автоматически.

E.6 Ориентир конфигурации MLS1 для проекта 38

Это опорный фрагмент, не готовый проект. Настройки access-коммутаторов, серверных пулов и VTY выполняются отдельно. Перед вводом проверьте модель 3560 и фактические порты.

```
enable
configure terminal
hostname MLS1
ip routing
vlan 10
 name STAFF
exit
vlan 20
 name STUDENTS
exit
vlan 30
 name GUEST
exit
vlan 50
 name SERVERS
exit
vlan 99
 name MANAGEMENT
exit
vlan 999
 name NATIVE-UNUSED
exit
interface range fastEthernet 0/1 - 2
 switchport trunk encapsulation dot1q
 switchport mode trunk
 switchport trunk native vlan 999
 switchport trunk allowed vlan 10,20,30,50,99,999
exit
interface vlan 10
 ip address 10.38.10.1 255.255.255.0
 ip helper-address 10.38.50.10
 no shutdown
exit
interface vlan 20
 ip address 10.38.20.1 255.255.255.0
 ip helper-address 10.38.50.10
 no shutdown
exit
interface vlan 30
 ip address 10.38.30.1 255.255.255.0
 ip helper-address 10.38.50.10
```

```
 no shutdown
exit
interface vlan 50
 ip address 10.38.50.1 255.255.255.0
 no shutdown
exit
interface vlan 99
 ip address 10.38.99.1 255.255.255.0
 no shutdown
exit
ip access-list extended CAMPUS-GUEST
 permit udp any eq 68 any eq 67
 permit udp 10.38.30.0 0.0.0.255 host 10.38.50.10 eq 53
 permit tcp 10.38.30.0 0.0.0.255 host 10.38.50.10 eq 53
 permit tcp 10.38.30.0 0.0.0.255 host 10.38.50.10 eq 80
 permit icmp 10.38.30.0 0.0.0.255 host 10.38.30.1 echo
 deny ip any any
exit
interface vlan 30
 ip access-group CAMPUS-GUEST in
exit
end
copy running-config startup-config
```

Такой ACL разрешает первоначальный DHCP, а далее – оговорённые потоки. Он не разрешает весь доступ к серверной VLAN. Чтобы дополнительно ограничить SSH управления только Admin, используйте VTY access-class на поддерживаемой модели и отдельно проверьте разрешённый и запрещённый вход.

E.7 Один адресный вариант для проекта 40

Это пример с запасом, а не единственный ответ. Для преподавателей выделяется /25, хотя 60 ПК плюс шлюз формально помещаются в /26: запас /26 был бы слишком мал. В филиальном блоке учтены 25 ПК, 4 инфраструктурных устройства и шлюз.

Группа	Сеть	Пригодные адреса	Шлюз
Студенты	10.40.0.0/25	.0.1–.0.126	10.40.0.1
Преподаватели	10.40.0.128/25	.0.129–.0.254	10.40.0.129
Филиал	10.40.1.0/26	.1.1–.1.62	10.40.1.1
Гости	10.40.1.64/27	.1.65–.1.94	10.40.1.65
Серверы	10.40.1.96/28	.1.97–.1.110	10.40.1.97
Управление	10.40.1.112/28	.1.113–.1.126	10.40.1.113

Свободны 10.40.1.128/25 и 10.40.2.0/23. В отдельном транзитном пуле 10.40.4.0/27 помещается восемь /30: .0, .4, .8, .12, .16, .20, .24, .28. IPv6-подсети можно назначить как 2001:db8:40:10::/64 преподавателям и 2001:db8:40:50::/64 серверной: четвёртая группа содержит выбранный идентификатор /64 внутри /48.

Приложение F

Словарь сетевых понятий

Access port
: Порт коммутатора для одной пользовательской VLAN; обычный конечный ПК не обязан понимать 802.1Q.

ACL
: Список правил сравнения пакетов. Направление и место применения определяют, какой поток он видит.

Административная дистанция
: Показатель предпочтения источника маршрута для одного префикса. Меньше обычно предпочтительнее.

ARP
: Механизм сопоставления IPv4 с канальным адресом локального соседа. Не выполняет маршрутизацию.

Area / ABR
: Область OSPF / маршрутизатор на границе областей. Backbone в обычном проекте – area 0.

Broadcast domain
: Область распространения канального широковещания. VLAN отделяет такую область от другой VLAN.

CDP / LLDP
: Протоколы обнаружения непосредственно подключённых сетевых соседей; не заменяют адресный план.

Default route
: Маршрут самого общего префикса /0 для назначений без более специфичного пути.

DHCP / relay
: Автоматическая выдача параметров / передача запроса серверу из другой подсети.

DNS
: Система имён и записей. Возвращённый адрес не доказывает достижимость службы.

Dual stack
: Одновременный IPv4 и IPv6. Каждый стек имеет свои адреса и маршруты.

EtherChannel
: Логическая агрегация совместимых физических линий; LACP – один из способов согласования.

Gateway
: Следующий маршрутизатор для выхода узла за пределы его локальной сети.

ICMP / ICMPv6
: Служебные сообщения IP; используются для диагностики и, у IPv6, важных функций NDP.

Инкапсуляция
: Вложение сообщения верхнего уровня в структуру нижнего: данные в сегмент/датаграмму, затем в IP-пакет и кадр.

Link-local
: Адрес IPv6 с областью действия текущего канала; обычно из fe80::/10.

MAC table
: Таблица L2-перенаправления по MAC и VLAN. Изучается по источникам поступающих кадров.

Метрика
: Стоимость пути в конкретном протоколе. Метрики разных протоколов не имеют общей шкалы.

NAT / PAT
: Преобразование адресов / совместное использование внешнего адреса с различением сессий.

NDP
: Neighbor Discovery Protocol: обнаружение соседей и маршрутизаторов IPv6 с использованием ICMPv6.

Next hop
: Следующий узел маршрута. Это не обязательно конечный адресат IP-пакета.

OSPF
: Протокол состояния каналов с вычислением кратчайших путей по стоимости.

Port Security
: Ограничение канальных адресов на порту; не является подтверждением личности пользователя.

Префикс
: Число значимых бит сети. Маска IPv4 – альтернативная запись сетевой части.

Router ID
: Идентификатор маршрутизатора в протоколе; требует уникальности в нужной области взаимодействия.

Running / startup config
: Текущая / сохранённая для старта конфигурация IOS.

SLAAC
: Автонастройка IPv6 с использованием объявления маршрутизатора и локального формирования адреса.

SSID
: Имя беспроводной сети. Не обеспечивает доступ без согласованных параметров защиты.

STP / root bridge
: Протокол предотвращения L2-петель / опорный коммутатор выбранного дерева.

SVI Виртуальный интерфейс VLAN. Может служить управлению L2-узлом либо шлюзом на L3-узле.

Trunk / native VLAN
: Канал нескольких VLAN / VLAN для нетегированной передачи в принятой модели конфигурации.

TTL / Hop Limit
: Ограничение числа IP-переходов. Канальные широковещательные кадры не защищены им от L2-петель.

VLSM
: Использование масок разной длины внутри выделенного адресного пространства.

VTY
: Виртуальные линии удалённого терминального доступа IOS; их политика определяет SSH/Telnet-вход.

Wildcard
: Маска сравнения, где 0 требует совпадения, а 1 разрешает игнорировать бит.

WLC / LAP
: Контроллер WLAN / управляемая lightweight access point.

Приложение G

Источники и границы применимости (обе книги)

G.1 Как читать документацию вместе с учебным стендом

Объяснения и задачи курса изложены самостоятельно; источники ниже помогают уточнить синтаксис, ограничения и поведение программ. Они относятся к разным моделям и версиям. Для ОС фиксируются выпуск Ubuntu, сборка Windows, версия оболочки и пакет; для Packet Tracer – выпуск программы и модель устройства. Сетевые источники проверялись 10 октября, системные дополнения – 11 октября 2026 г.; страницы могут обновляться. У локальной команды также есть встроенная справка, соответствующая установленной версии.

Успешная компиляция подтверждает сборку документа, но не является выполнением всех лабораторных. Сценарии, конфигурации и различия платформ требуют учебного прогона на выбранных ОС и моделях. Это самостоятельный курс базовой дисциплины, а не официальный материал Cisco, Microsoft или Canonical и не обещание полного покрытия сертификационного экзамена. Все схемы созданы для книги; внешние картинки и дополнительные исходные файлы для сборки не нужны. Скриншоты отчёта получают из собственного стенда.

G.2 Что проверено в Packet Tracer 8.2.1

11 и 13 октября 2026 года сетевые работы были выполнены в Packet Tracer 8.2.1 на Windows 11: схемы собраны, команды из листингов введены в консоли устройств, результаты проверены ping, tracert и командами show. Ниже указано, в каком объёме.

Работы	Объём проверки
01–03	Полностью, с рисунками: размещение устройств, кабели, IP Configuration, ping, режим Simulation и PDU.
04, 05	Расчёты проверены вручную; схема работы 05 (четыре группы, заметки) собрана.
06–10	Полностью: консоль, SSH с ПК, маршрутизатор между LAN, три маршрутизатора, floating static route.
11–19	Полностью: VLAN, trunk, router-on-a-stick, L3-коммутатор, STP, LACP, Port Security, CDP, DHCP на маршрутизаторе.
20–22	Полностью: DHCP relay, DNS, HTTP, FTP (вход и dir), почта между двумя ПК (отправка и получение письма).
23–27	Синтаксис и поведение ACL, NAT и OSPF подтверждены ping и командами show. Открытие страниц по имени через ACL и NAT (HTTP) отдельно не проверялось.
28–31	Стоимость OSPF и отказ линии, две области, EIGRP и RIP, IPv6 (статические адреса на маршрутизаторе, SLAAC на ПК).

33	Полностью: WRT300N, замена модуля ноутбука, PC Wireless, DHCP, ping.
34	Частично: питание LAP, адрес контроллера, регистрация точки, создание WLAN. Подключение клиента к WLAN WLC не воспроизводилось.
35	Частично: служба IoT, регистрация учётной записи, подключение двух вещей, страницы Home и Conditions. Окно создания условия и срабатывание правила не проверялись.
36	Частично: настройка syslog, NTP и копирование конфигурации на TFTP. Синхронизация NTP и обратное восстановление из TFTP не доводились до конца.
32, 37–40	Не выполнялись: работы используют те же команды и устройства, что проверенные выше, но целиком на Packet Tracer не прогонялись.
41, 42, 43, 45, 46, 47, 48	Полностью (13 октября 2026 года), со схемами: HSRP с отказом и возвратом роли, DHCP Snooping с чужим сервером, Rapid PVST+ с обрывом канала и root guard, GRE и IPsec с лицензией `securityk9`, RADIUS по SSH и уровни привилегий, телефония на 2811 со звонком между телефонами и политикой QoS, eBGP между тремя AS. Найденные отличия учебной модели внесены в текст работ (синтаксис `standby track`, `tunnel source`, отсутствие `mode transport`, `clear crypto`, `show ephone`, `prefix-list`; телефония только на 2811; питание телефонов).
44	Частично: на 2911 команда `snmp-server` знает только `community`; MIB Browser проверен для Get и Get Bulk (v2) и неверного сообщества. Списки доступа, location/contact, ловушки и операция Set в Packet Tracer недоступны — в тексте работы это оговорено.
Модули 1–20 практикума и SQL-1–SQL-18, I01–I05	Не относятся к Packet Tracer; настоящие ВМ в этом прогоне не запускались.

При проверке были найдены и исправлены расхождения между текстом и программой (список в сопроводительном файле REVIEW-CPT.md): отсутствовавшая команда `no shutdown` в работе 18, недопустимый для почтового клиента домен `lab.example`, несуществующее имя «Lamp», отсутствие у WLC-PT веб-мастера и многое другое. Если вы найдёте новое расхождение, считайте программу правой и поправьте текст.

[1] Cisco. Download and Installation Instructions. Официальный путь получения и установки. https://www.netacad.com/skillsforall/files/Cisco_Packet_Tracer_Download_and_Installation_Instructions.pdf

[2] Cisco Packet Tracer Help. FAQ / Troubleshooting Guide. Ограничения моделей IOS и диагностика программы. https://tutorials.ptnetacad.net/help/default/faqTroubleShoot.htm

[3] Cisco Networking Academy. Getting Started with Cisco Packet Tracer. Вводный официальный курс.

https://www.netacad.com/courses/getting-started-cisco-packet-tracer/1000
[4] Cisco Packet Tracer Help. Connections / Links. Типы соединений. https://tutorials.ptnetacad.net/help/default/connectionsLinks.htm
[5] Cisco. Configure Inter VLAN Routing with the Use of an External Router, Document ID 14976. Подинтерфейсы и 802.1Q. https://www.cisco.com/c/en/us/support/docs/lan-switching/inter-vlan-routing/14976-50.html
[6] Cisco. Configure IP Access Lists, Document ID 23602. Standard/extended ACL и применение на интерфейсе. https://www.cisco.com/c/en/us/support/docs/security/ios-firewall/23602-confaccesslists.html
[7] Cisco. Configure Network Address Translation, Document ID 13772. Static NAT и overload. https://www.cisco.com/c/en/us/support/docs/ip/network-address-translation-nat/13772-12.html
[8] Cisco. OSPF Configuration Guide. Соседства, интерфейсы и параметры областей. https://www.cisco.com/c/en/us/td/docs/switches/lan/c9000/lyr3-fwd/ospf/ospf-configuration-guide/ospf.html
[9] Cisco. Configuring the Cisco IOS DHCP Relay Agent, IOS 15S. Роль helper и relay. https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipaddr_dhcp/configuration/15-s/dhcp-15-s-book/dhcp-relay-agent.html
[10] Cisco. Configure SSH on Routers, Document ID 4145. Предпосылки и настройка SSH. https://www.cisco.com/c/en/us/support/docs/security-vpn/secure-shell-ssh/4145-ssh.html
[11] Cisco. Catalyst 2960: Configuring EtherChannels and Link-State Tracking. LACP и условия агрегации. https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst2960/software/release/15-0_2_se/configuration/guide/scg2960/swethchl.html
[12] Cisco. Catalyst 2960: Configuring Port-Based Traffic Control. Port Security и sticky MAC. https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst2960/software/release/12-2_53_se/configuration/guide/2960scg/swtrafc.html
[13] Cisco. Catalyst 2960: Configuring Optional Spanning-Tree Features. PortFast и BPDU Guard. https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst2960/software/release/12-2_53_se/configuration/guide/2960scg/swstpopt.html
[14] Cisco. Implement Static Routes for IPv6 Configuration Example, Document ID 113361. IPv6 forwarding и статические пути. https://www.cisco.com/c/en/us/support/docs/ip/ip-version-6-ipv6/113361-ipv6-static-routes.html
[15] Cisco. Understand and Use the Enhanced Interior Gateway Routing Protocol, Document ID 16406. Соседства и принципы EIGRP. https://www.cisco.com/c/en/us/support/docs/ip/enhanced-interior-gateway-routing-protocol-eigrp/16406-eigrp-toc.html
[16] Cisco Packet Tracer Help. Configuring Other Devices. WLAN-модели и варианты защиты. https://tutorials.ptnetacad.net/help/default/config_others.htm

[17] Cisco Packet Tracer Help. Internet of Things. Smart Things, Home Gateway и Registration Server.
https://tutorials.ptnetacad.net/help/default/iot.htm
[18] Cisco. System Management Configuration Guide: Network Time Protocol. Ассоциации и проверка времени.
https://www.cisco.com/c/en/us/td/docs/routers/ios-xe/system-management/system-management/m_bsm-time-calendar-set.html
[19] Cisco. Catalyst 2960: Configuring System Message Logging. Пороги и доставка сообщений.
https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst2960/software/release/12-2_55_se/configuration/guide/scg_2960/swlog.html
[20] RFC Editor. RFC 5737: IPv4 Address Blocks Reserved for Documentation.
https://www.rfc-editor.org/info/rfc5737/
[21] RFC Editor. RFC 3849: IPv6 Address Prefix Reserved for Documentation.
https://www.rfc-editor.org/info/rfc3849/
[22] Cisco Packet Tracer Help. My First Packet Tracer Lab. Переименование устройств, сохранение проекта и просмотр событий.
https://tutorials.ptnetacad.net/help/default/myFirstPTLab.htm
[23] Ubuntu Server. User management. Локальные учётные записи, группы и административный доступ.
https://ubuntu.com/server/docs/how-to/security/user-management/
[24] GNU Coreutils Manual. Файлы, владельцы, режимы и базовые утилиты.
https://www.gnu.org/software/coreutils/manual/coreutils.html
[25] GNU Bash Reference Manual. Оболочка, переменные, потоки и сценарии.
https://www.gnu.org/software/bash/manual/bash.html
[26] Microsoft Learn. PowerShell 101. Командлеты, справка и конвейер.
https://learn.microsoft.com/en-us/powershell/scripting/learn/ps101/00-introduction
[27] Microsoft Learn. LocalAccounts / New-LocalUser. Локальные записи и ограничения модуля.
https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.localaccounts/new-localuser?view=powershell-5.1
[28] Microsoft Learn. icacls. Разрешения и наследование NTFS.
https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/icacls
[29] Ubuntu Manpages. systemd.service. Units и одноразовые службы.
https://manpages.ubuntu.com/manpages/noble/man5/systemd.service.5.html
[30] Ubuntu Manpages. journalctl. Просмотр системных журналов.
https://manpages.ubuntu.com/manpages/noble/man1/journalctl.1.html
[31] Ubuntu Server. Package management. Apt, индексы и пакеты.
https://ubuntu.com/server/docs/how-to/software/package-management/
[32] Ubuntu Server. OpenSSH server. Установка, ключи и конфигурация.
https://ubuntu.com/server/docs/how-to/security/openssh-server/
[33] Ubuntu Server. Firewall. UFW и адресные правила.
https://ubuntu.com/server/docs/how-to/security/firewalls/
[34] Microsoft Learn. Test-NetConnection / NetTCPIP. Средства проверки сети Windows.
https://learn.microsoft.com/en-us/powershell/module/nettcpip/test-netconnection
[35] Microsoft Learn. Get-WinEvent. Фильтрация журналов событий.
https://learn.microsoft.com/en-us/powershell/module/microsoft.powe

rshell.diagnostics/get-winevent
[36] Microsoft Learn. ScheduledTasks. Действия и регистрация задач. https://learn.microsoft.com/en-us/powershell/module/scheduledtasks/new-scheduledtaskaction
[37] GNU tar Manual. Архивирование, извлечение и атрибуты. https://www.gnu.org/software/tar/manual/tar.html
[38] Rsync project. rsync manual. Копирование и свойства режима archive. https://download.samba.org/pub/rsync/rsync.1
[39] Ubuntu Manpages. crontab(5). Поля расписания и среда запуска. https://manpages.ubuntu.com/manpages/noble/man5/crontab.5.html
[40] Ubuntu Server. Samba file server. Публикация каталогов для SMB-клиентов. https://ubuntu.com/server/docs/how-to/samba/file-server/
[41] Ubuntu Server. Configure nginx. Блоки server и настройка сайта. https://ubuntu.com/server/docs/how-to/web-services/configure-nginx/
[42] Ubuntu Server. Basic installation. Первоначальная установка системы. https://ubuntu.com/server/docs/tutorial/basic-installation/
[43] Ubuntu Manpages. acl(5). Эффективные права, маска и default ACL. https://manpages.ubuntu.com/manpages/noble/man5/acl.5.html
[44] Microsoft Learn. Compress-Archive. Формат ZIP и ограничение скрытых файлов. https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.archive/compress-archive
[45] Microsoft Learn. Robocopy. Параметры и коды результата. https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/robocopy
[46] Microsoft Learn. about_Execution_Policies. Области и смысл политики сценариев. https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_execution_policies
[47] Canonical Netplan. Documentation. YAML, постоянные адреса и netplan try. https://netplan.readthedocs.io/en/stable/

Следующий шаг

После курса измените условия проекта: добавьте отдел, новую роль, другой адресный план и требование к сроку восстановления. Если вы можете объяснить путь запроса, личность процесса, применённое правило доступа, выбрать проверку двух разных гипотез и восстановить данные вместе с правами, у вас есть основа для дальнейшего администрирования. Продолжайте на новых стендах: централизованные записи, управление конфигурациями, мониторинг и контейнеры. Для каждой новой технологии фиксируйте версию, проверяемую задачу и способ возврата.