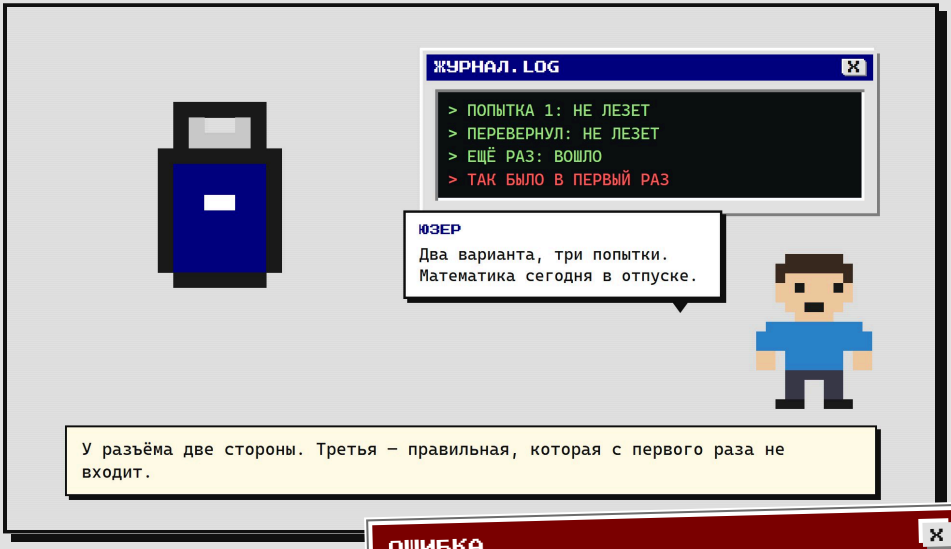
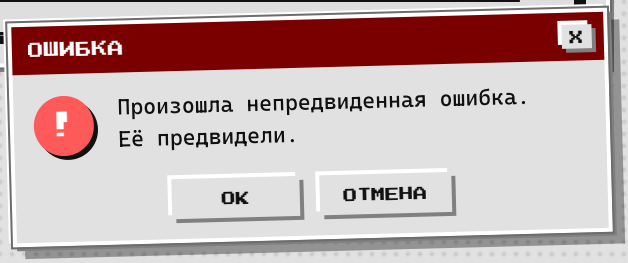
LEGGI_DI_MURPHY.EXE

MURPHY.EXE

Компьютерные законы Мерфи с комментариями сисадмина, который видел их в деле





C:\MURPHY> read.exe /все /с_комментариями
Найдено законов: 133. Ошибок: 0. (пока)_

SILVANO DE POLI

133 ЗАКОНА + СЛЕДСТВИЯ ОТ ЗАВЕДЕНИЯ

MURPHY.EXE

Компьютерные законы Мерфи с комментариями сисадмина, который видел их в деле

SILVANO DE POLI

Всем принтерам, которые научили меня смирению.
И тем, кто вчера вечером сделал бэкап: вы знаете почему.

СОДЕРЖАНИЕ

Предисловие 5
Инструкция по применению 6
1. Основные законы 7
2. Железо: принтеры, кабели и диски, которые слушают 23
3. Хелпдеск: юзеры, пароли и прочие тайны 45
4. Сети и интернет: невидимая нить 63
5. Охота на баги 81
6. Искусство писать код (и раскаиваться) 101
7. Проекты, совещания и клиенты 121
8. Прогресс, обновления и искусственный интеллект 143
Следствия от заведения 167
На прощание 179
Выходные данные 180

ПРЕДИСЛОВИЕ

Меня зовут Silvano De Poli, и по профессии я чиню вещи, которые сломались сами. Я сисадмин и компьютерный мастер, делаю сайты, работаю с искусственным интеллектом и руковожу веб-агентством RetroAI. Проще говоря: когда в офисе что-то перестаёт работать, звонит обычно мой телефон.

За все эти годы я понял, что поломки не случайны. Они кажутся случайными, но у них есть привычки. Они предпочитают пятницу. Обожают демонстрации клиентам. Прячутся, когда приходит мастер, и возвращаются, едва машина снова запущена. Однажды, глядя на принтер, который зажевал именно последний лист фирменного бланка, я понял, что мне не просто не везёт. Я внутри закона.

Этот закон, как и многие другие, уже существовал. Инженер Edward Murphy дал имя самому знаменитому: если что-то может пойти не так, оно пойдёт не так. После него программисты, экономисты, писатели и исследователи добавили свои наблюдения — одни совершенно серьёзные, другие родились как шутки и стали поговорками. Информатика собрала их все с той же заботой, с какой хранит кабели в ящике: мало ли что.

Эта коллекция родилась на моём сайте, по одному закону за раз. Сначала это были заметки: закон, кто его сформулировал, комментарий. Потом комментарии разрослись, появились темы, некоторые законы превратились в маленькие ролики в пиксель-арте. В какой-то момент их стало сто тридцать три, и я заметил, что вместе они рассказывают историю: историю нежных и непростых отношений людей и машин.

В книге три вида законов. Есть законы **с именем собственным**, связанные с реальным человеком: Moore, Brooks, Parkinson, Hofstadter и многие другие. Для них я старался объяснить, о чём они на самом деле, прежде чем над ними подшучивать. Есть **анонимные**, которые передаются из офиса в офис, как семейные рецепты: их никто не подписывал, потому что их пережили все. И есть несколько **моих собственных следствий**, в основном про искусственный интеллект, — подписываю их с некоторым смущением и большой гордостью.

У каждого закона есть комментарий. Тон — как на сайте: ироничный, иногда немного сюрреалистичный, всегда тёплый. Самые абсурдные истории, уверяю вас, выдуманы. Самые правдоподобные — тоже, но тут я уже не так уверен. Зато технические объяснения настоящие, и я старался сделать их понятными даже тем, кто открывал компьютер разве что в поисках винтика.

В конце каждой главы есть *Совет сисадмина*. Это серьёзная часть, та, что, надеюсь, останется у вас в кармане: практичные, проверенные вещи, которые действительно помогают. Мерфи они не победят, потому что Мерфи непобедим. Но работать ему придётся больше.

Книгу можно читать по порядку или вразброс. Можно держать на столе, как молитвенник, и открывать перед важным обновлением. Можно цитировать на совещаниях, чтобы казаться мудрым, или подарить коллеге, который твердит «у меня на компе работает». Она тоже работает. На его компе.

И последнее предупреждение. Пока я приводил эти страницы в порядок, компьютер дважды завис, Wi-Fi пропал посреди важного поиска, а принтер отказался участвовать в пробной печати. Рассказываю это не чтобы пожаловаться. Прилагаю как сертификат подлинности. Законы работают. Приятного чтения — и сохраняйтесь почаще.

ИНСТРУКЦИЯ ПО ПРИМЕНЕНИЮ

Руководство пользователя, модель MURPHY.EXE. Внимательно прочтите перед включением.

- **Запуск.** Книгу можно открыть на любой странице. Перезагрузка не требуется.
- **Порядок чтения.** Главы идут от самых универсальных к самым техническим. Кто не программирует, может пропустить пятую и шестую без необратимых последствий, но если прочтёт — наконец поймёт своего программиста.
- **Дозировка.** Рекомендуется один закон в день, желательно до того, как прикоснуться к серверу. При передозировке возможны непроизвольные цитаты на совещаниях.
- **Советы сисадмина.** В конце каждой главы. Единственные серьёзные части книги: их нужно применять, а не только читать.
- **Совместимость.** Работает на экране, на бумаге и вслух. Протестирована и в печати, с переменным успехом.
- **Гарантия.** Книга не предотвращает поломки. Она лишь делает их узнаваемыми. При сбоях попробуйте перечитать.

ГЛАВА 1

ГЛ 01 — ОСНОВЫ

Основные законы

ЗАКОНОВ: 13 _

У каждой религии есть священные тексты, а у информатики — эти законы. Никто не высекал их в камне: их добывали в полевых условиях, позор за позором, инженеры, программисты, экономисты и люди, которым просто нужно было распечатать бланк.

Начинаем с закона-прародителя, закона Мерфи, который изначально был советом конструкторам, а стал нашим ежедневным гороскопом. За ним — ближайшая родня: кто добавляет время катастрофы, кто напоминает, что компьютер понимает всё буквально, кто утверждает, что программа, заработавшая с первого раза, просто берёт разбег.

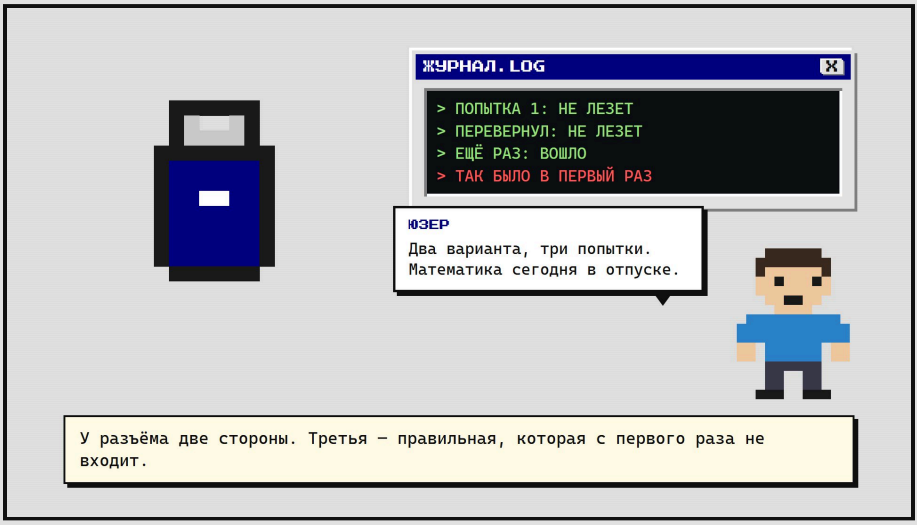
Среди них есть и законы, родившиеся вне информатики — в бюрократии, экономике, фантастике. Мы их усыновили, потому что они описывали нас слишком точно: работа, растягивающаяся на всё отведённое время, коллега, которого повышали, пока не довели до не той должности, восемьдесят процентов бед, всегда идущих от одних и тех же двадцати.

Эту главу стоит прочесть первой и цитировать на совещаниях. Эти законы работают как латинские изречения: кто их произносит, сразу кажется мудрее, даже если сервер всё ещё лежит.

СОДЕРЖАНИЕ.DIR

01 Закон Мерфи
02 Закон Финейгла
03 Закон буквального компьютера
04 Закон остаточного бага
05 Закон подозрительного успеха
06 Закон Хофштадтера
07 Закон Паркинсона
08 Принцип Питера
09 Принцип Парето (80/20)
10 Бритва Хэнлона
11 Третий закон Кларка
12 Закон Старджона
13 Закон Мура

ЗАКОН МЕРФИ



Если что-то может пойти не так, рано или поздно оно пойдёт не так. Особенно это касается разъёмов, которые вставляются двумя способами.

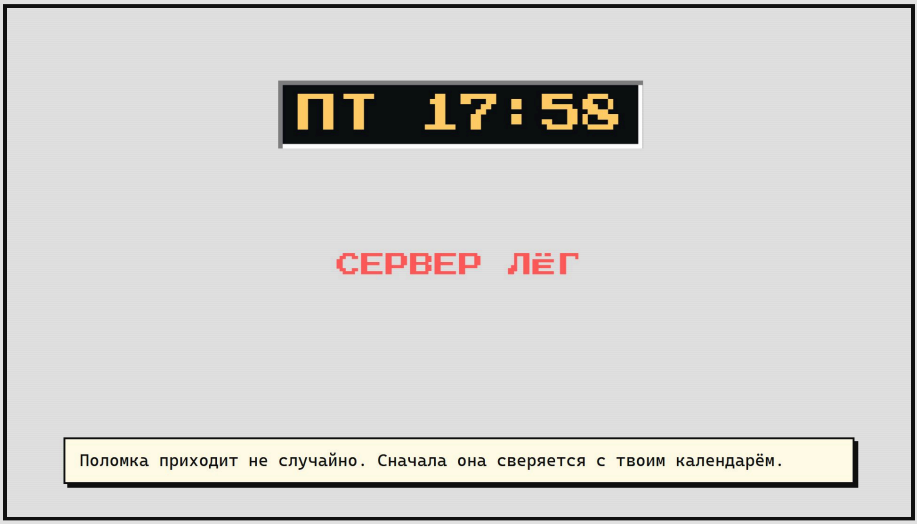
— Edward A. Murphy Jr., *авиабаза Edwards, проект MX981, около 1949 года; точная формулировка и обстоятельства спорны*

Всё началось с инженера, датчиков, установленных задом наперёд, и испорченного эксперимента. Изначально это был не пессимизм, а совет конструктору: если деталь можно установить неправильно, кто-нибудь так и сделает, поэтому проектируй её так, чтобы было нельзя.

Мы, айтишники, восприняли это как пророчество. У USB-кабеля три стороны: правильная, неправильная и правильная, которая с первой попытки всё равно не входит. Это квантовая физика портов.

Я применяю закон с бюрократической строгостью. Перед каждым обновлением заполняю бланк «Что может пойти не так». В бланке сорок страниц. Обновление зависает на странице сорок один, которую я не предусмотрел.

ЗАКОН ФИНЕЙГЛА



Всё, что может пойти не так, пойдёт не так, и выберет для этого худший из возможных моментов.

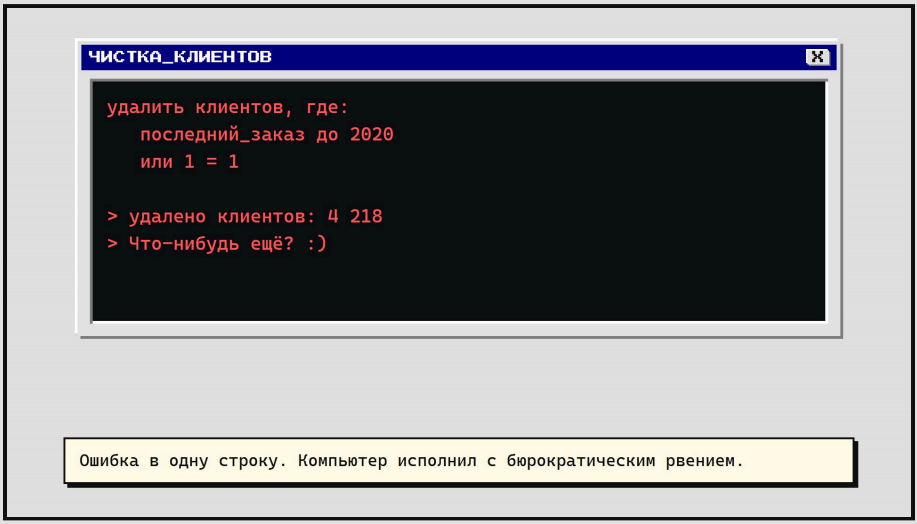
— Аноним, *вариант закона Мерфи, распространённый в кругах американской научной фантастики с 50–60-х годов; авторство не установлено*

Мерфи говорит, что катастрофа придёт. Финейгл добавляет время: пятница, 17:58, когда ты уже выключаешь монитор. Это версия с синхронизированным календарём.

Сервер никогда не ломается во вторник утром, когда ты выспался и у тебя есть кофе. Он ломается во время демонстрации клиенту, в день закрытия годового отчёта или когда ты на свадьбе у двоюродной сестры, а ноутбук заперт в машине. У поломки есть ежедневник, и твой всегда открыт для просмотра.

Я пытался её перехитрить: сделал вид, что ухожу в отпуск. Сервер остался невозмутим. А в тот день, когда я действительно вернулся, сгорел блок питания. Не знаю, откуда он узнал. Подозреваю общий календарь.

ЗАКОН БУКВАЛЬНОГО КОМПЬЮТЕРА



Компьютер делает ровно то, что ты ему велел, и никогда — то, чего ты от него хотел.

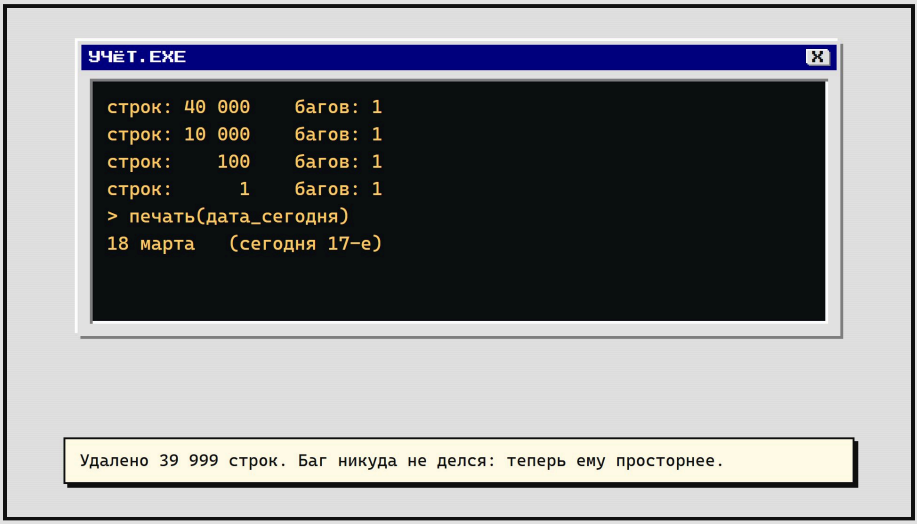
— Аноним, *компьютерный фольклор*

Компьютер — самый послушный сотрудник в истории, и именно поэтому самый опасный. Он не толкует, не догадывается, не смотрит тебе в лицо, чтобы понять, что ты имел в виду. Он исполняет. Буквально. С бюрократическим рвением.

Скажи ему «удали неактивных клиентов», и если из-за лишней запятой неактивными окажутся все, он удалит всех, с восхитительной эффективностью. А потом спросит, не нужно ли чего-нибудь ещё. Сама любезность.

Это как джинн из лампы, окончивший юрфак. Каждое желание приходится формулировать с оговорками, исключениями и сносками. Для непрограммистов вот настоящая суть профессии: не объяснить компьютеру, что делать, а предусмотреть все творческие способы, которыми он может тебя неправильно понять. Я на всякий случай теперь так разговариваю и с собакой.

ЗАКОН ОСТАТОЧНОГО БАГА



Любая программа содержит хотя бы один баг, и любую программу можно сократить на одну инструкцию. Отсюда по индукции следует, что любую программу можно свести к одной-единственной инструкции, которая не работает.

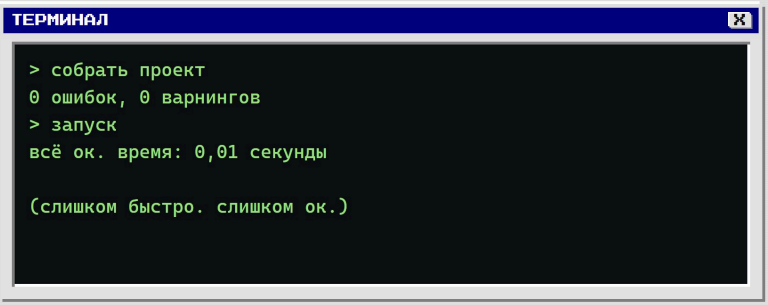
— Аноним, *компьютерный фольклор*

Это основная теорема программирования, и, как все серьёзные теоремы, её никто никогда не доказал: все были заняты, чинили что-то другое. Логика безупречна: убираешь по одной строке, а баг всё равно на месте, держится, как мидия за камень.

Я пробовал проверить это в лаборатории. Взял учётную систему на сорок тысяч строк и начал удалять. Когда осталась одна строка, программа печатала дату. Неправильную. На один день. Только в месяцах, где есть буква «р».

Для тех, кто не программирует, мораль простая: идеальный софт существует, но только пока его никто не пишет. Стоит нажать первую клавишу — ошибка садится в первый ряд и начинает конспектировать.

ЗАКОН ПОДОЗРИТЕЛЬНОГО УСПЕХА



Опытный программист не ликует. Закрывает дверь и всё перепроверяет.

Если программа заработала с первой попытки, это плохой знак: значит, ошибка сидит глубже и даст о себе знать в самый неподходящий момент.

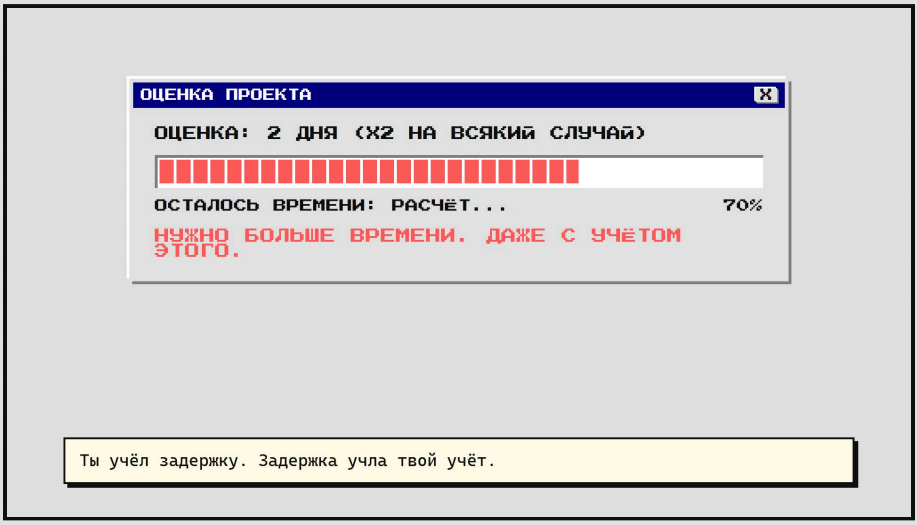
— Аноним, *компьютерный фольклор*

Опытный программист, у которого код сразу запустился без ошибок, не ликует. Он бледнеет. Встаёт, закрывает дверь и перепроверяет всё, как сапёр перед посылкой, которая тикает подозрительно тихо.

Причина статистическая. Софт — это минное поле, где каждый шаг хотя бы раз оказывается неверным. Если ничего не взорвалось, это значит только одно: мина лежит дальше, она крупнее и ждёт заказчика.

Со мной такое правда случилось: скомпилировал, запустил — идеально. Отметил кофе из автомата. Через три месяца выяснилось, что программа работала, потому что ничего не делала. Очень элегантно пропускала всю важную часть. С тех пор, когда что-то получается с первого раза, я отношусь к этому как к слишком любезному гостю: с теплотой, но пересчитывая ложки.

ЗАКОН ХОФШТАДТЕРА



Любое дело всегда занимает больше времени, чем ожидалось, даже если учесть закон Хофштадтера.

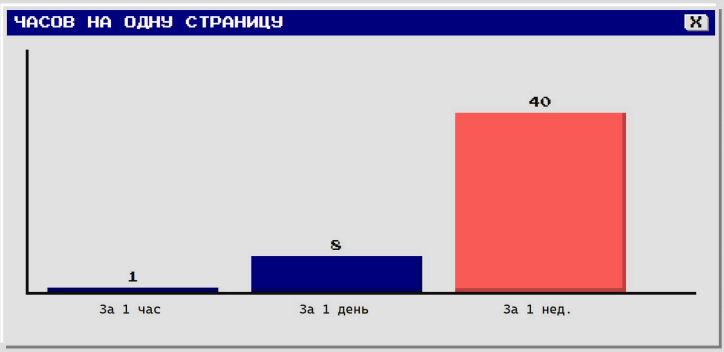
— Douglas Hofstadter, *Gödel, Escher, Bach*, 1979

Это рекурсивный закон, а потому очень любимый айтишниками. Оцениваешь в два дня. Знаешь, что всегда ошибаешься, поэтому пишешь четыре. Но закон знает, что ты знаешь, и понадобится восемь.

Это индикатор выполнения жизни: «Осталось: 2 минуты». Потом 47 минут. Потом 3 секунды. Потом «Подсчёт оставшегося времени…», что является изящным способом сказать «понятия не имею».

Я попросил искусственный интеллект оценить срок моего проекта с учётом закона Хофштадтера. Он ответил: «Это займёт больше, чем ожидается». Я спросил, насколько больше. Он всё ещё думает. Я дал ему дедлайн в десять минут, с учётом закона. Вернусь во вторник.

ЗАКОН ПАРКИНСОНА



Страница та же. У недельной шрифт меняли четырнадцать раз.

Работа заполняет всё время, отпущенное на её выполнение. В информатике добавляется следствие: данные заполняют всё доступное место.

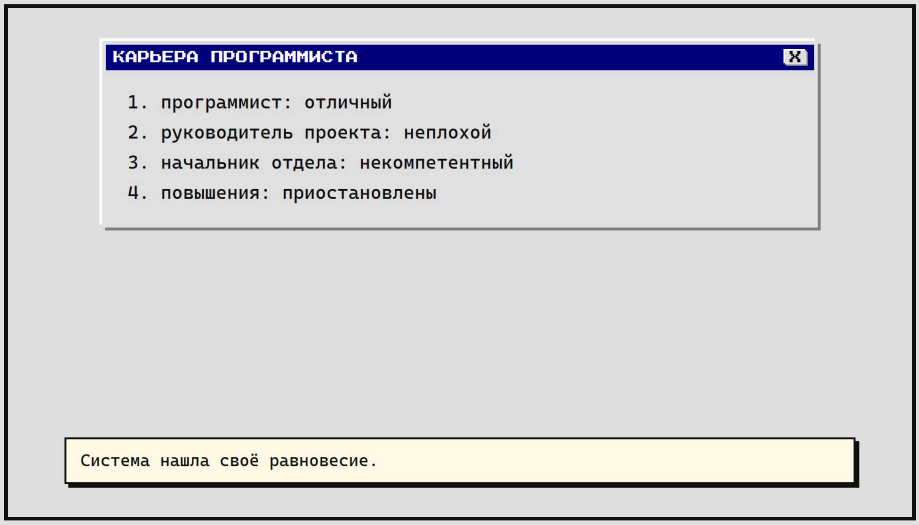
— Cyril Northcote Parkinson, *The Economist, ноябрь 1955 (следствие о данных: автор неизвестен, авторство не установлено)*

Если у тебя неделя, чтобы написать страницу, ты потратишь неделю. Если час — потратишь час. Страница будет одинаковой. В недельной версии просто четырнадцать раз сменится шрифт.

Родившись как сатира на британскую бюрократию, принцип стал неофициальным руководством по IT-проектам. Щедрые сроки дают не лучший софт, а лучшие совещания, то есть более длинные. И это касается не только людей, но и вещей. Покупаешь диск на четыре терабайта, и через полгода он полон, как новый гараж.

У нас в офисе пытались победить закон, назначая фальшивые сроки, короче настоящих. Работало месяц. Потом все начали вычислять настоящий срок, и теперь у нас есть таблица, которая его оценивает. Мы работаем над ней три недели.

ПРИНЦИП ПИТЕРА



В иерархии каждый человек стремится подняться до своего уровня некомпетентности и там останавливается.

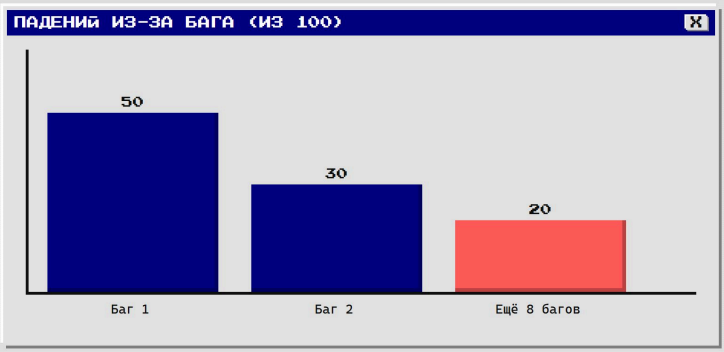
— Laurence J. Peter (в соавторстве с Raymond Hull), *«The Peter Principle»*, *1969*

Жил-был отличный программист. Такой хороший, что его повысили до руководителя проекта. Там он был похуже, но достаточно хорош, чтобы его повысили до начальника отдела. И там он наконец оказался совершенно неспособным. Тогда его перестали повышать. Система нашла равновесие.

Принцип Питера объясняет, почему во многих компаниях тот, кто писал лучший код, сегодня целыми днями делает таблицы о таблицах. Уметь что-то делать не значит уметь руководить теми, кто это делает. Но повышение награждает за первое умение, чтобы поручить работу, требующую второго.

Я гениально обошёл проблему: открыл своё дело. Теперь я одновременно программист, сисадмин, продавец и генеральный директор. Я достиг уровня некомпетентности сразу в четырёх ролях. Это оптимизация.

ПРИНЦИП ПАРЕТО (80/20)



Два бага из десяти дают восемь падений из десяти. Двадцать процентов пашут вовсю.

Во многих явлениях около 80% следствий вытекают из 20% причин: например, 80% проблем создаёт небольшая часть багов или клиентов.

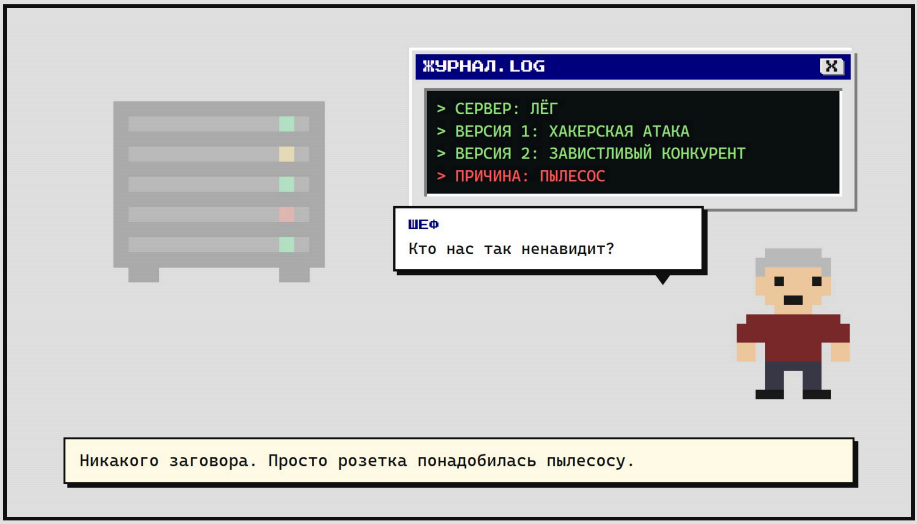
— Vilfredo Pareto, обобщён Joseph M. Juran, *наблюдение Парето о распределении богатства (1896); название и обобщение — Juran (40–50-е годы)*

Итальянский экономист заметил, что немногим принадлежит большая часть земли. Десятилетия спустя эксперт по качеству обобщил: немногие факторы вызывают большую часть последствий. С тех пор восемьдесят и двадцать — самая знаменитая пара в бизнес-консалтинге.

В информатике это работает почти неловко точно. Восемьдесят процентов падений происходят из-за горстки багов. Восемьдесят процентов звонков в техподдержку поступают от тех же двадцати процентов пользователей, и ты прекрасно знаешь, как их зовут. Восемьдесят процентов ценности программы — в нескольких функциях, и именно они спрятаны в третьем подменю.

Я применил Парето к своему столу. Двадцать процентов кабелей создают восемьдесят процентов путаницы. Беда в том, что я не знаю, какие двадцать. Я отключил все. Теперь порядок, но ничего не включается.

БРИТВА ХЭНЛОНА



Не приписывай злому умыслу то, что вполне объясняется рассеянностью или некомпетентностью.

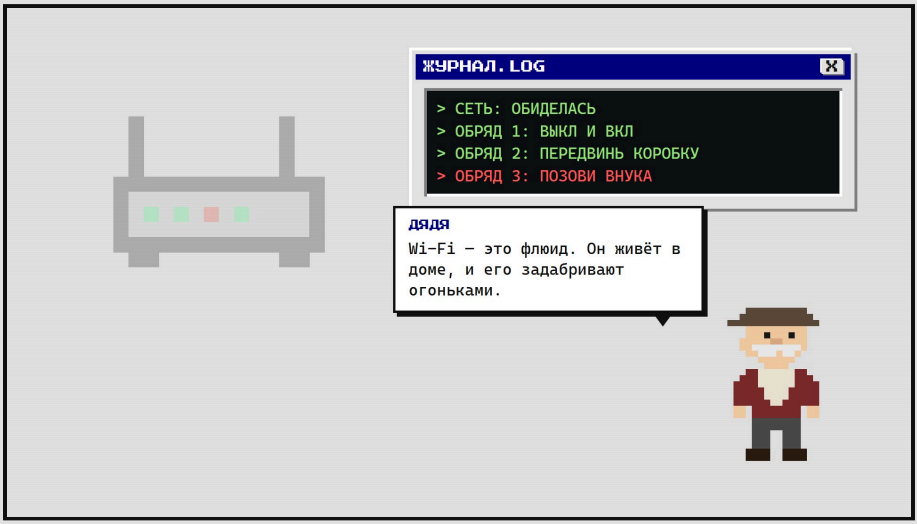
— Robert J. Hanlon, *Arthur Bloch, «Murphy's Law Book Two», 1980 (похожие идеи встречаются в гораздо более ранних текстах)*

Сервер лежит. Первая мысль: хакерская атака, иностранные шпионы, завистливый конкурент. Потом выясняется, что уборщица выдернула вилку, чтобы подключить пылесос. Никакого заговора. Просто на одну свободную розетку меньше.

Бритва Хэнлона — лучший друг мастера техподдержки. Когда пользователь удаляет общую папку, он не саботирует компанию. Он просто решительно нажал на то, чего не понимал. Это хуже, но хотя бы ничего личного. Помнить об этом полезно для отношений между людьми и для артериального давления.

Совет: бритва работает и в обратную сторону. Когда ошибаешься ты, надейся, что другие о ней знают. Я всегда цитирую её клиентам, прежде чем сказать, что натворил. Работает как своего рода дипломатический антивирус.

ТРЕТИЙ ЗАКОН КЛАРКА



Любая достаточно развитая технология неотличима от магии.

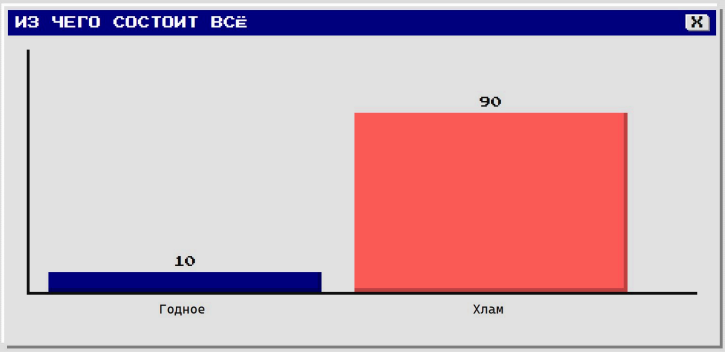
— Arthur C. Clarke, *«Profiles of the Future», переработанное издание 1973 года (уже в письме в Science 1968 года)*

Писатель-фантаст заметил, что достаточно развитая технология для того, кто не знает, как она работает, выглядит как заклинание. Он думал об инопланетных цивилизациях. Он не видел мою тётю перед Wi-Fi.

Для неё сеть — невидимая жидкость, которая живёт в доме и легко обижается. Её задабривают, выключая и включая коробочку с огоньками. Если не помогает, коробочку переставляют. Если и это не помогает, зовут племянника, который произносит непонятные формулы, и коробочка исцеляется. Это не технология. Это литургия.

Тёмная сторона закона в том, что он работает и для тех, кто технологию продаёт: чем больше она похожа на магию, тем меньше люди задают вопросов. Я же всегда всё объясняю. Клиенты уважительно слушают, кивают, а потом всё равно перезагружают. Магия побеждает.

ЗАКОН СТАРДЖОНА



Верно для фантастики, приложений, плагинов. Вероятно, и для этого графика.

Девяносто процентов чего угодно — барахло: это верно для фантастики, но и для книг, фильмов, программ и любых продуктов.

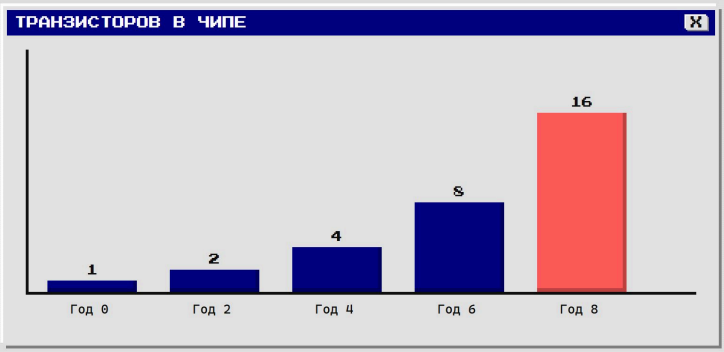
— Theodore Sturgeon, *Venture Science Fiction, март 1958 (известен также как «откровение Старджона»)*

Писателю-фантасту твердили, что девяносто процентов фантастики — мусор. Он ответил: конечно, ведь девяносто процентов всего — мусор. Конец дискуссии, аплодисменты и закон, который работает везде.

В информатике это почти единица измерения. Девяносто процентов приложений в магазине бесполезны. Девяносто процентов библиотек в интернете никто не поддерживает. Девяносто процентов плагинов для Joomla, которые я установил в 2011 году, всё ещё на месте, и никто не помнит, зачем они были нужны.

Прелесть закона в том, что он не пессимистичен: он говорит, что хорошие десять процентов существуют. Проблема — их найти. Я на этом построил карьеру. Моя работа, по сути, — просеивать песок. Иногда нахожу самородок. Чаще драйвер.

ЗАКОН МУРА



В 1965-м хватило четырёх точек на графике. Так продолжалось полвека.

Количество транзисторов, которое удаётся втиснуть в чип, удваивается через равные промежутки времени: сначала каждый год, а после поправки — примерно каждые два года.

— Gordon Moore, *Electronics, 1965 (прогноз пересмотрен в 1975 году)*

В 1965 году инженер посмотрел на четыре точки на графике и сказал: так и пойдёт дальше. Он был прав полвека. Я смотрю на четыре точки в календаре и говорю, что закончу сайт к пятнице. Методы у нас похожие, результаты разные.

На практике это значит, что каждые два года компьютер становится вдвое мощнее. И каждые два года программа, которой ты пользуешься, становится вдвое тяжелее, так что в сухом остатке Word всегда открывается шесть секунд. Это космическое равновесие. Вечная ничья между инженерами железа и инженерами софта, а судья — песочные часы Windows.

Сегодня транзисторы такие маленькие, что вот-вот упрутся в атомы. Мой двоюродный брат говорит, что тогда надо будет просто потуже затянуть. У него даже есть специальная отвёртка.

СОВЕТ СИСАДМИНА

Мерфи не победить, его можно только обойти. Перед любым серьёзным вмешательством — обновлением, переездом сервера или правкой сайта — я следую четырём правилам, написанным на настоящем листке, приклеенном к монитору.

- **Сначала копия.** Полный и проверенный бэкап, сделанный сейчас, а не «на днях».
- **Путь назад.** Если я не знаю, как откатить изменение, я его не делаю.
- **По одному.** Поменять три вещи сразу — значит не знать, какая из трёх сломала четвёртую.
- **Запись на каждый шаг.** Дата, что сделал, что произошло. Через полгода это будет единственный достоверный документ.

Дорогой софт не нужен: хватит тетрадки. Тетрадка не обновляется сама и не просит лицензию. Мерфи всё равно ударит, но найдёт чёрный ход уже открытым — чтобы уйти.

ГЛАВА 2

ГЛ 02 — ЖЕЛЕЗО

Железо: принтеры, кабели и диски, которые слушают

ЗАКОНОВ: 19 _

Софт пугает, потому что его не видно. Железо пугает, потому что его видно прекрасно — и оно смотрит на тебя. Принтер с оранжевой лампочкой, кабель, который не дотягивается, винтик, укатившийся под материнскую плату со звуком, похожим на смешок.

В этой главе — предметы. Те, что стоят на столе, под столом и, в худших случаях, за столом, где пыль уже создала собственный микроклимат. Это самые демократичные законы книги: чтобы их испытать, не нужно уметь программировать. Достаточно хоть раз печатать договор без пяти девять.

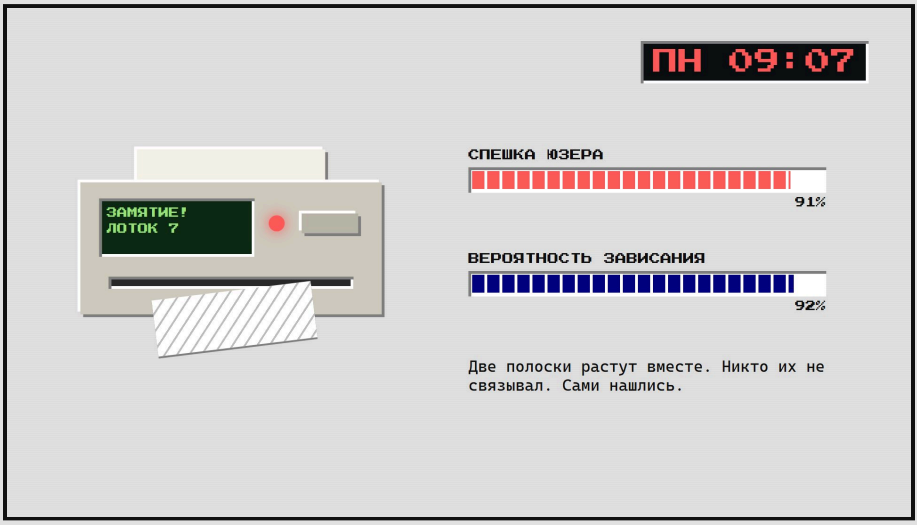
Во второй части спускаемся в диски, где живут данные и их страхи. Бэкап, которого нет; бэкап, который есть, но не восстанавливается; файл, который всегда живёт на другом компьютере. Здесь ирония тоньше, потому что боль настоящая. Сломанный принтер можно заменить. Фото с выпускного — нет.

Читайте с отвёрткой под рукой. Не пригодится, но так спокойнее.

СОДЕРЖАНИЕ.DIR

01 Закон нетерпеливого принтера
02 Закон тонера-ясновидца
03 Закон избирательного замятия
04 Закон сетевого принтера-призрака
05 Закон короткого кабеля
06 Закон недостающего кабеля
07 Закон квантовой флешки
08 Закон сбежавшего винтика
09 Закон истёкшей гарантии
10 Закон оптимистичной батареи
11 Закон враждебного проектора
12 Закон ненаходимого драйвера
13 Закон несделанного бэкапа
14 Закон чуткого диска
15 Закон непроверенного восстановления
16 Закон несостоявшегося сохранения
17 Закон файла на другом компьютере
18 Закон Сегала
19 Закон Крайдера

ЗАКОН НЕТЕРПЕЛИВОГО ПРИНТЕРА



Вероятность того, что принтер зависнет, прямо пропорциональна спешке того, кому надо печатать.

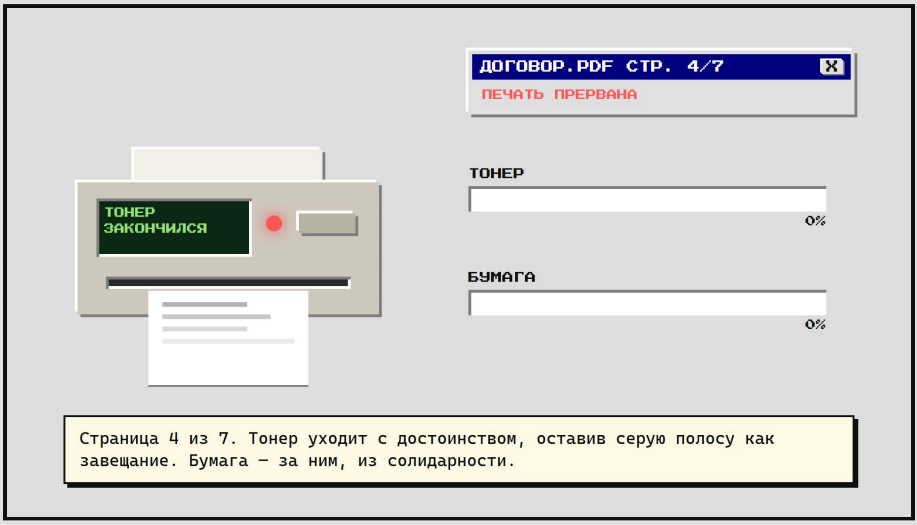
— Аноним, *компьютерный фольклор*

У принтера нет глаз, нет ушей и нет связи с часами в квартире. И всё же он знает. В понедельник в девять, когда поезд отходит в девять десять, принтер чувствует сердцебиение пользователя, как акула чувствует каплю крови.

Я знал офис, который годами печатал только спокойно, тихонько напевая, чтобы не выдать себя. Работало. Потом однажды один сотрудник посмотрел на часы. Сорок две минуты «идёт прогрев».

Совет мастера: важные документы печатают накануне вечером. Не потому, что принтер спит, а потому, что утром у вас есть план Б, и он об этом знает.

ЗАКОН ТОНЕРА-ЯСНОВИДЦА



Тонер и бумага всегда заканчиваются на середине самого важного документа недели и никогда — во время пробной печати.

— Аноним, *компьютерный фольклор*

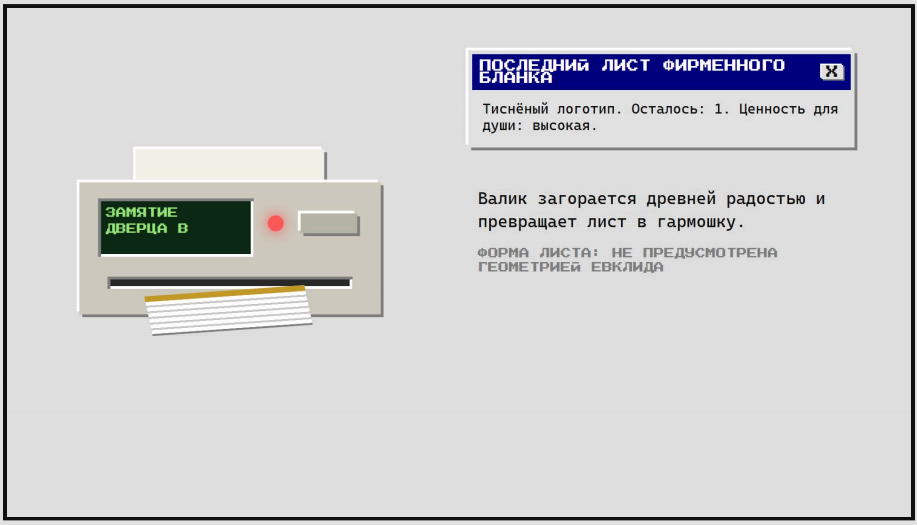
Тонер — жидкость пророческая. Три месяца он печатает афиши деревенского праздника, рецепты и фото кота, глазом не моргнув. Потом приходит договор на подпись, и он с достоинством удаляется на странице четыре из семи, оставив серую полосу в качестве завещания.

Мой двоюродный брат каждое утро тряс картридж, по привычке, как градусник. Говорил, что так тонер пугается и остаётся. У него был полный ящик встряхнутых картриджей. Ни один не закончился раньше сделки у нотариуса.

Тем временем в переговорной кто-то спрашивает, нельзя ли печатать «только чёрным». Ответ — нет: закончился как раз чёрный.

Мини-совет: всегда держите запасной картридж. Он стоит дешевле позора.

ЗАКОН ИЗБИРАТЕЛЬНОГО ЗАМЯТИЯ



Бумага заминается только на том единственном листе, который нельзя перепечатать: фирменный бланк, готовый формуляр, последний хороший лист.

— Аноним, *компьютерный фольклор*

У принтеров талант сомелье. Перед пачкой за два евро они кротки, как ягнята. Но стоит вставить последний фирменный бланк, тот, с тиснёным логотипом, как ролик загорается древней радостью и превращает его в гармошку.

Я видел, как коллеги открывали дверцы, о существовании которых я не подозревал. Дверца A, дверца B, дверца «не открывать во время работы». Внутри лист был сложен в фигуру, которую евклидова геометрия не предусматривает.

Принтер тем временем показывал на дисплее маленькую анимированную иконку руки, вытаскивающей бумагу. Не торопясь. Как объясняют ребёнку, что мяч улетел к соседу в сад.

Настоящий совет: прежде чем использовать ценный лист, сделайте пробу на обычной бумаге. Всегда.

ЗАКОН СЕТЕВОГО ПРИНТЕРА-ПРИЗРАКА



Конфиденциальный документ, отправленный на печать, всегда оказывается на принтере другого этажа, перед не тем человеком.

— Аноним, *компьютерный фольклор*

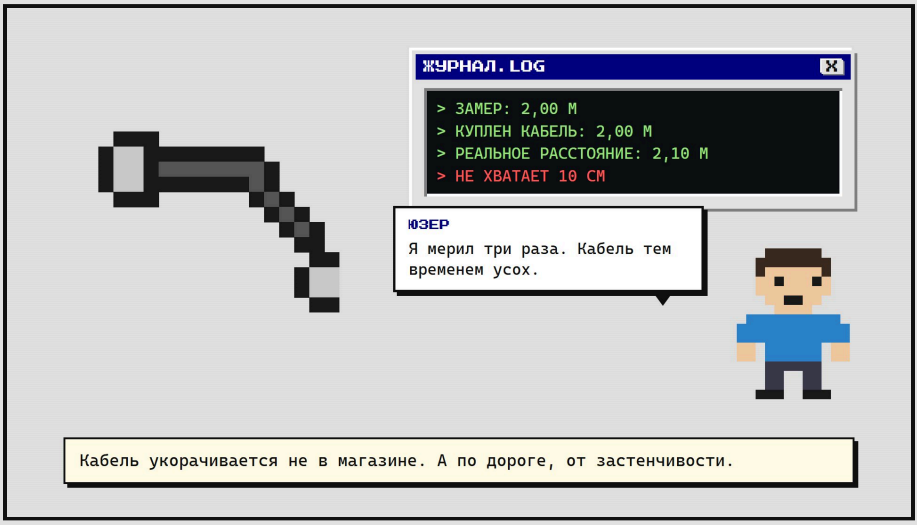
В каждой компании есть принтер, который никто не устанавливал. Он появляется в списке под именем вроде «HP_VTOROY_ETAZH_STARYY». Никто не знает, где он стоит. Но именно он оказывается принтером по умолчанию, когда ты печатаешь заявление об уходе.

Однажды я проследил путь документа по всему зданию. Он вышел из бухгалтерии, прошёл через склад, и его забрал, прочитал и прокомментировал отдел отгрузки. Три человека сказали мне «хорошее письмо». Оно было не моё.

Принтер-призрак не ошибается. Он выбирает. У него есть тайный архив всего, чего ему не следовало видеть, и однажды он напишет мемуары.

Совет сисадмина: прежде чем нажать «Печать», проверьте, какой принтер выбран по умолчанию. Две секунды, которые спасают карьеры.

ЗАКОН КОРОТКОГО КАБЕЛЯ



Любой кабель всегда на десять сантиметров короче расстояния, которое он должен покрыть.

— Аноним, *компьютерный фольклор*

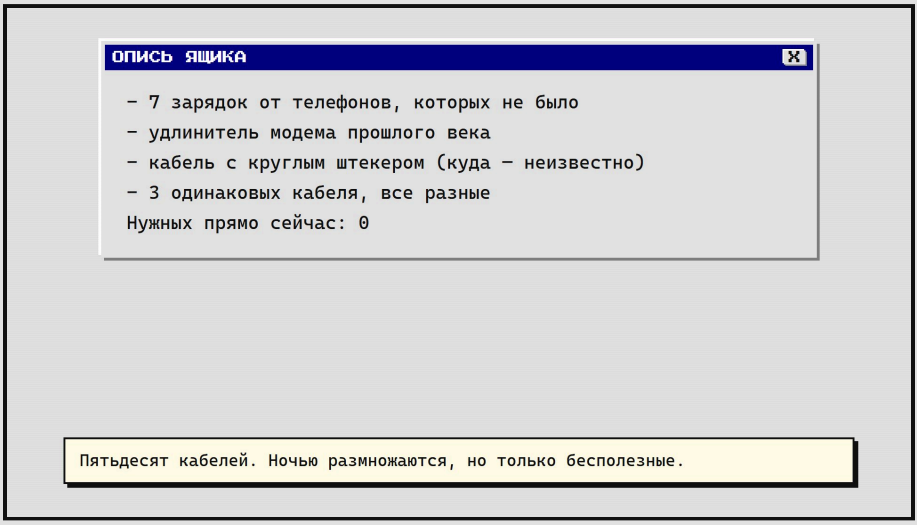
Это универсальная постоянная, как скорость света. Если розетка в метре, кабель на девяносто. Если купить двухметровый, стол переедет. Никто не видел, как он двигался, но рулетка не врёт.

У меня есть друг, который пытался обмануть закон, купив десятиметровый кабель, чтобы подключить монитор в полуметре. Кабель пришёл. А друг тем временем переехал, и новая розетка оказалась в десяти метрах и десяти сантиметрах. Совпадение, говорят скептики.

Короткий кабель ещё и учитель жизни: он учит терпению, смирению и, главное, умению залезать под стол с фонариком в зубах.

Совет мастера: измерьте, прибавьте полметра и покупайте такой. Длинный кабель сворачивается; короткий не растягивается.

ЗАКОН НЕДОСТАЮЩЕГО КАБЕЛЯ



В ящике с пятьюдесятью разными кабелями никогда нет того, который нужен прямо сейчас.

— Аноним, *компьютерный фольклор*

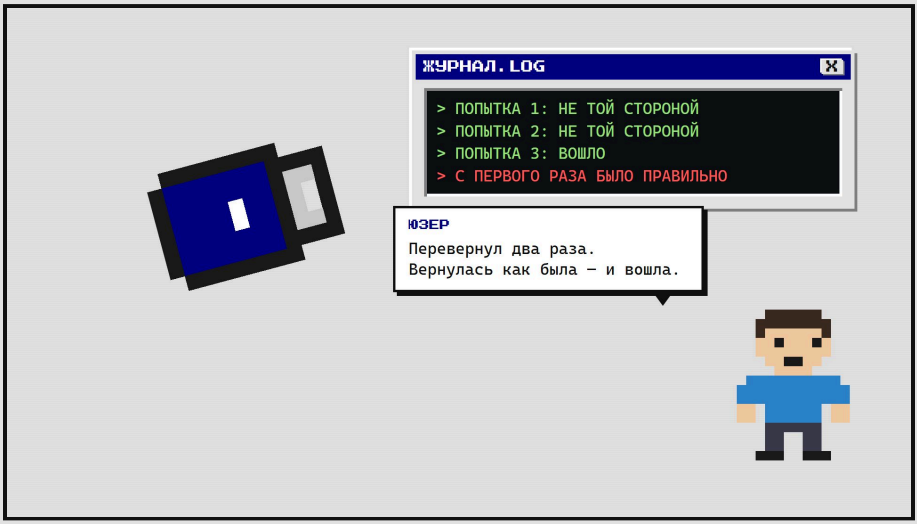
В каждом доме есть Ящик. Экосистема переплетённых кабелей, которые размножаются по ночам: зарядки от телефонов, которых у тебя никогда не было, удлинители от модемов прошлого века, загадочный провод с круглым штекером, который никуда не подходит.

Пятьдесят кабелей. Ноль с нужным разъёмом. Статистически невозможно, и всё же случается каждый раз. Нужный кабель, конечно, существует: он в ящике у соседа, который в свою очередь ищет твой.

Я предложил жильцам дома устроить биржу кабелей, с обменом по субботам утром. Провалилась: никто не хотел отдавать свой бесполезный кабель. «Мало ли», — говорили все. Официальный девиз любого ящика.

Настоящий совет: подпишите кабели кусочком скотча и без жалости выбрасывайте те, которые не можете опознать уже два года.

ЗАКОН КВАНТОВОЙ ФЛЕШКИ



USB-штекер всегда входит с третьей попытки: первый раз он перевёрнут, второй тоже, а третий был правильным с самого начала.

— Аноним, *компьютерный фольклор*

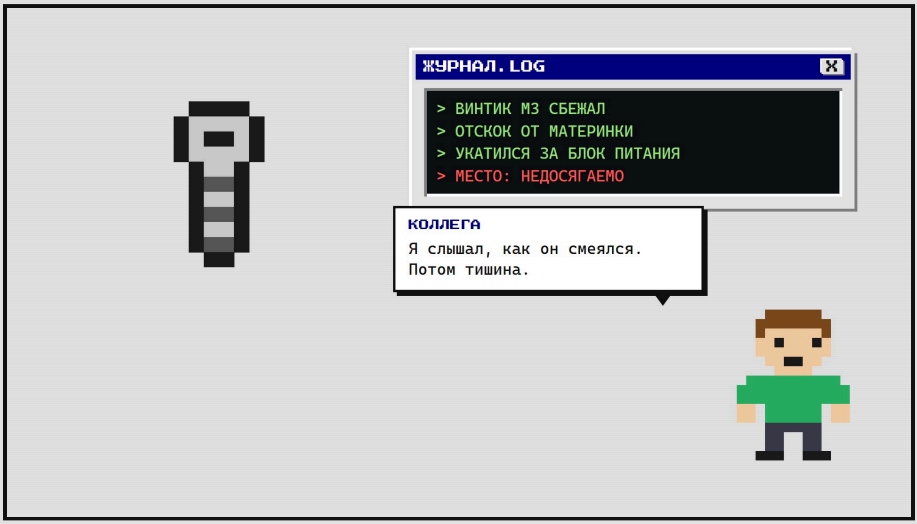
У классического USB-разъёма две стороны. Математически с первой попытки у тебя пятьдесят процентов шансов угадать. На практике — ноль. Разъём существует в состоянии суперпозиции: он неправильный в обе стороны, пока не перевернёшь его дважды.

Я подсчитал, что за свою карьеру переворачивал флешки примерно трое суток подряд. Считаю их неотгулянным отпуском.

Потом появился симметричный разъём, и человечество ликовало. Полдня. Потом мы заметили, что симметричные кабели заряжают не все с одинаковой скоростью, не все передают данные и не все включают монитор. Хаос просто переехал внутрь кабеля.

Совет: логотип USB на разъёме обычно смотрит вверх. Обычно.

ЗАКОН СБЕЖАВШЕГО ВИНТИКА



Винтик, выскользнувший из пальцев, пока ты открываешь компьютер, всегда падает в самое недоступное место корпуса.

— Аноним, *компьютерный фольклор*

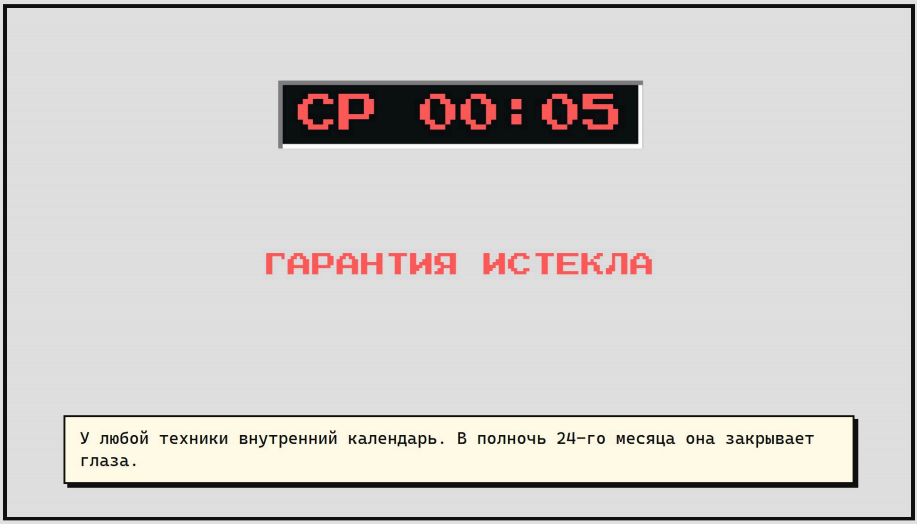
Внутри компьютера тысячи удобных мест, куда мог бы приземлиться винтик. Он всегда выбирает единственное — под материнской платой, за блоком питания, в щели, спроектированной кем-то, кто ненавидел человеческие пальцы.

Тогда мастер исполняет древний танец с корпусом: наклоняет его, трясёт, переворачивает, прислушивается. Винтик катится с металлическим звуком, похожим на смех. Потом затихает. Его больше никто никогда не увидит, но теперь компьютер издаёт новый звук, когда его двигают.

В моей мастерской стоит банка с винтиками-сиротами, собранными за годы. Ни один ни разу ни на что не пригодился. Мы называем её «пенсионный фонд».

Совет мастера: пользуйтесь магнитной отвёрткой и мисочкой для винтов. И никогда не работайте над ковролином.

ЗАКОН ИСТЁКШЕЙ ГАРАНТИИ



Устройство ломается на следующий день после окончания гарантии, за исключением случаев, когда оно ломается накануне, а чек ты потерял.

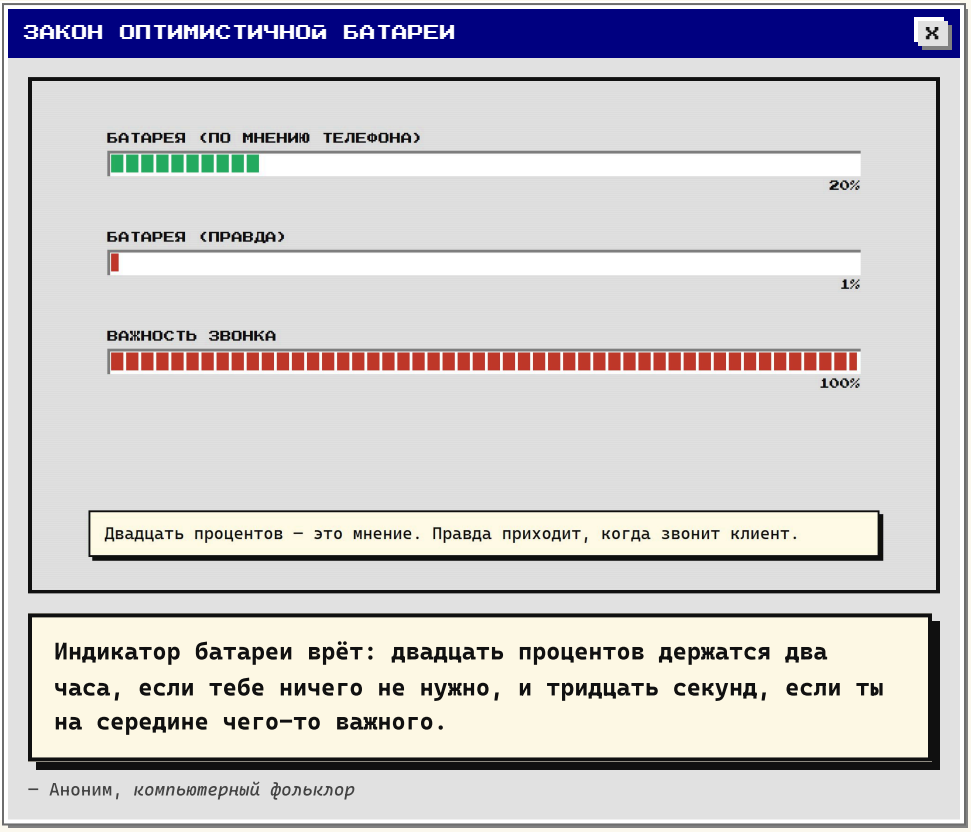
— Аноним, *компьютерный фольклор*

Внутри каждого современного прибора есть календарь. В инструкции об этом не написано, но он есть. Он считает дни с терпением монаха и в полночь двадцать четвёртого месяца с миром закрывает глаза.

Одна моя клиентка хранила все чеки с 1998 года в обувной коробке, разложенными по цветам. В день, когда принтер сломался на гарантии, коробка стояла в подвале. Подвал затопило. Чек превратился в маленькое белое облачко.

Гарантия, словом, существует: это единорог клиентского сервиса. Все о ней говорят, кое-кто клянётся, что видел.

Совет мастера: сфотографируйте чек сразу после покупки и сохраните в облаке. Термобумага выцветает, фотографии — нет.



Индикатор батареи врёт: двадцать процентов держатся два часа, если тебе ничего не нужно, и тридцать секунд, если ты на середине чего-то важного.

— Аноним, *компьютерный фольклор*

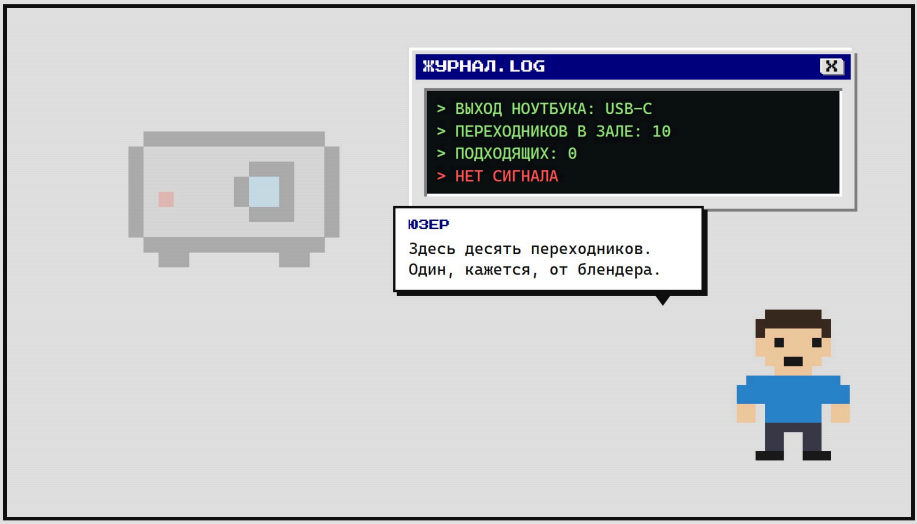
Процент заряда — это не данные. Это мнение. Чувство. Телефон на двадцати процентах говорит тебе «спокойно, дотянем» тоном человека, который едет по трассе на резерве.

Потом открываешь карту, чтобы найти клиента, и процент падает, как лифт без тросов: восемнадцать, одиннадцать, четыре, чёрный экран. Зарядка дома, тоже заряжается, из солидарности.

Мой дядя утверждал, что батареи заряжаются сами, если дать им отдохнуть в ящике. Он прав примерно на одну минуту. Это самая оптимистичная минута его дня.

Настоящий совет: литиевые батареи изнашиваются, и с возрастом индикатор начинает врать. Если устройство выключается при высоком проценте, пора менять батарею.

ЗАКОН ВРАЖДЕБНОГО ПРОЕКТОРА



У проектора в переговорной не тот вход, что у твоего ноутбука, а нужный переходник — единственный, который никто не принёс.

— Аноним, *компьютерный фольклор*

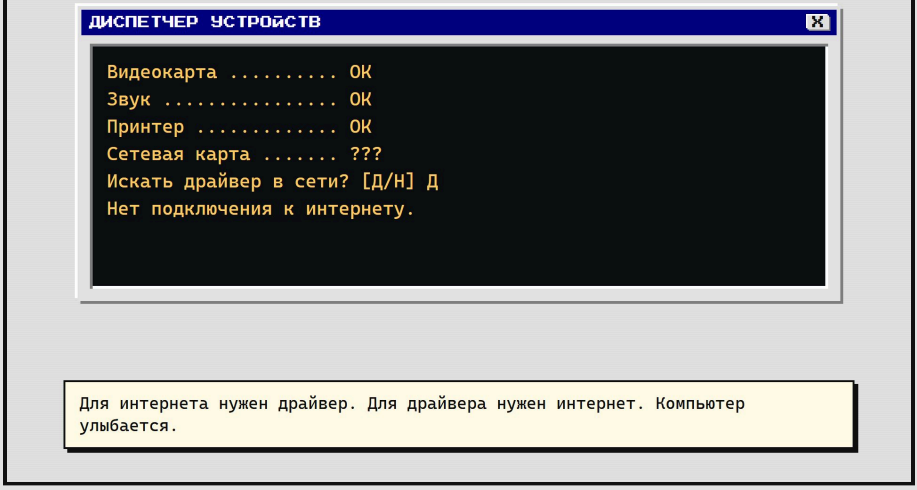
В каждой переговорной под потолком висит проектор, как электронная летучая мышь. У него один вход, того типа, от которого твой ноутбук отказался три поколения назад. На столе коробка с переходниками: десять штук, все для других компьютеров, некоторые для предметов, которые вообще не компьютеры.

Тем временем публика ждёт. Кто-то предлагает «разослать презентацию по почте, и пусть каждый смотрит с телефона». Это момент, когда цивилизация начинается заново с бумаги.

Когда изображение наконец появляется, оно фиолетовое, обрезанное справа, а экран ноутбука гаснет. Ты победил, но не знаешь как.

Совет мастера: всегда носите свой переходник и копию презентации в PDF на флешке. И приходите на пятнадцать минут раньше.

ЗАКОН НЕНАХОДИМОГО ДРАЙВЕРА



После переустановки системы не хватает только одного драйвера — драйвера сетевой карты, который нужен, чтобы скачать остальные драйверы.

— Аноним, *компьютерный фольклор*

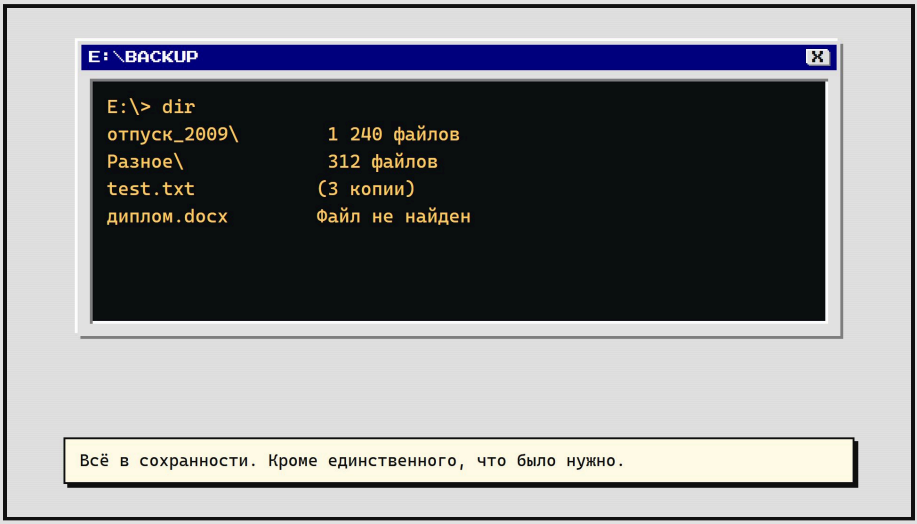
Это идеальный парадокс, достойный фантастического рассказа. Чтобы был интернет, нужен драйвер. Чтобы был драйвер, нужен интернет. Только что отформатированный компьютер смотрит на тебя невинно, как младенец, который ещё не умеет разговаривать с роутером.

Тогда мастер отправляется в Великую Экспедицию: другой компьютер, флешка, сайт производителя, шесть раз сменивший структуру. Драйвер называется что-то вроде «paket_final_v2_novyy_OKONCHATELNYY.zip» и весит больше операционной системы.

Тем временем в соседней комнате клиент спрашивает, не можешь ли ты «раз уж всё равно этим занимаешься» восстановить ещё и свадебные фотографии.

Настоящий совет: перед форматированием скачайте на флешку сетевые драйверы для своей модели. Это набор инструментов на следующий день.

ЗАКОН НЕСДЕЛАННОГО БЭКАПА



Из всех возможных бэкапов нужен всегда тот единственный, который ты не сделал.

— Аноним, *компьютерный фольклор*

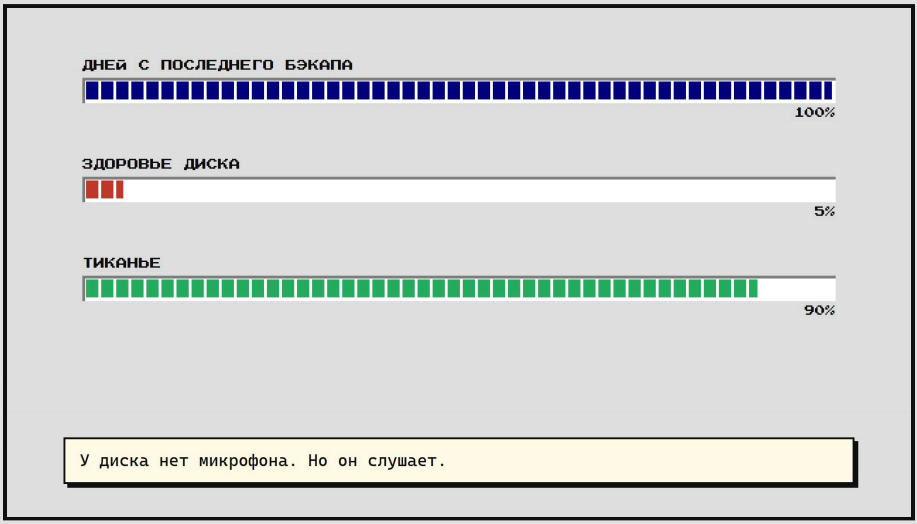
У тебя есть бэкап фотографий из отпуска 2009 года. Есть бэкап папки «Разное». Есть три одинаковых бэкапа файла «test.txt». Потом диск умирает, и единственный по-настоящему нужный файл — диплом, который лежал на рабочем столе «только на сегодня».

Бэкап — это суеверие наоборот: если его делаешь, ничего не случается, и тебе кажется, что ты зря потратил время. Если не делаешь, вселенная устраивает практическую демонстрацию.

У меня есть клиент, который делал бэкап на флешку, которая торчала в компьютере, который стоял под окном, под которым прошла гроза. Теперь у него две мокрые вещи вместо одной.

Настоящий совет: правило 3-2-1. Три копии, на двух разных носителях, одна из них вне дома или в облаке.

ЗАКОН ЧУТКОГО ДИСКА



Вероятность отказа диска обратно пропорциональна времени, прошедшему с последнего бэкапа.

— Аноним, *компьютерный фольклор (варианты в сборниках Usenet из alt.folklore.computers)*

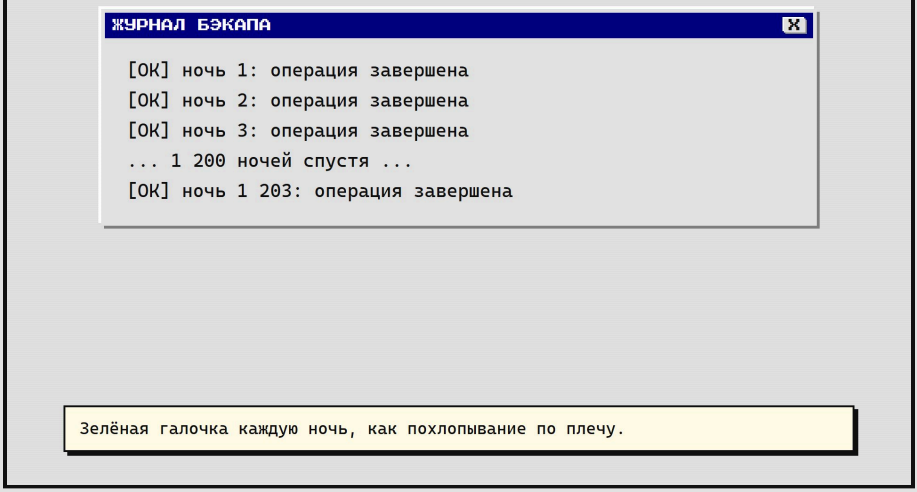
Жёсткий диск слушает. Микрофона у него нет, но он слушает. На следующий день после полного бэкапа он крутится счастливо, тихо, вечно. После полугода без копий начинает слегка тикать, как часы в триллере.

Старые мастера говорили, что это тиканье — «головка здоровается». На самом деле головка прощалась навсегда.

Есть только один способ обмануть диск: сделать бэкап так, чтобы он не узнал. Деликатно, ночью, автоматически. Диск обнаруживает копию только потом, и тогда ему расхочется ломаться. Это не наука. Но результаты отличные.

Совет сисадмина: настройте автоматический бэкап. Бэкап, который зависит от вашей памяти, — не бэкап, а надежда.

ЗАКОН НЕПРОВЕРЕННОГО ВОССТАНОВЛЕНИЯ



Бэкап всегда работает — до того дня, когда ты пытаешься из него восстановиться.

— Аноним, *компьютерный фольклор*

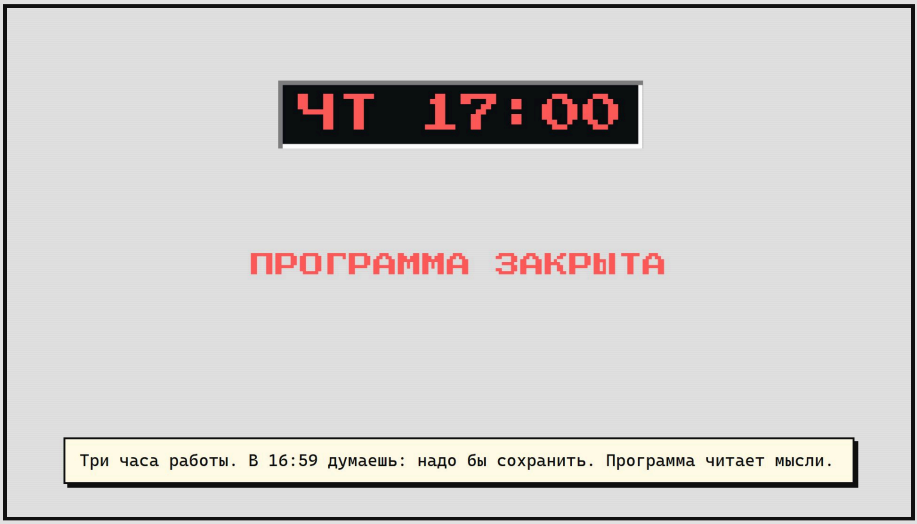
Годами программа резервного копирования показывала зелёную галочку. Каждую ночь. «Операция завершена». Галочка, успокаивающая, как похлопывание по плечу.

Потом наступает день нужды. Открываешь архив и обнаруживаешь, что в нём одна пустая папка, с религиозным старанием копировавшаяся тысячу двести ночей подряд. Программа ни разу не соврала: операция действительно была завершена. Чего именно — никто не спрашивал.

Непроверенный бэкап — как парашют, который ни разу не раскрывали в магазине. Наверное, всё в порядке. Наверное.

Настоящий совет мастера: хотя бы время от времени пробуйте восстановить из бэкапа случайный файл. Если получилось — спите спокойно. Если нет — вы только что избежали худшей катастрофы.

ЗАКОН НЕСОСТОЯВШЕГОСЯ СОХРАНЕНИЯ



Программа всегда зависает после трёх часов работы и за тридцать секунд до того, как ты решишь сохраниться.

— Аноним, *компьютерный фольклор*

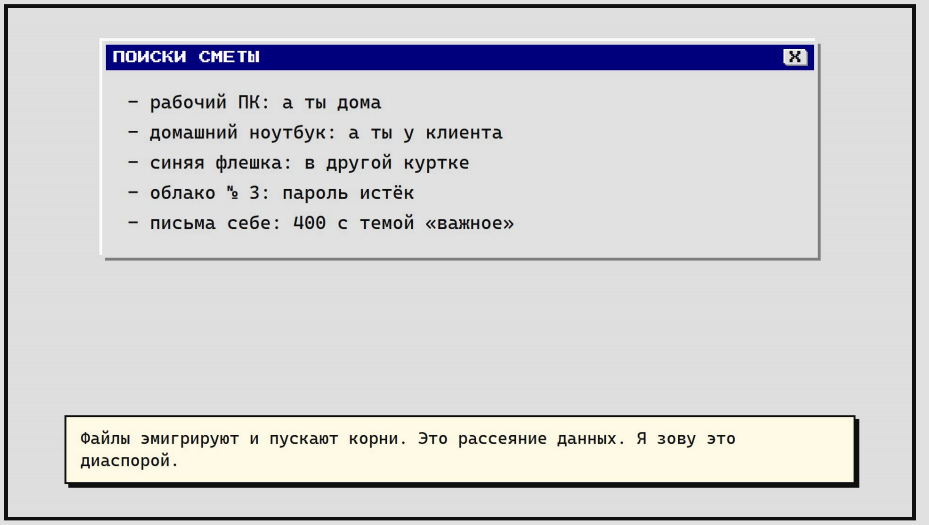
Во время работы бывает момент, когда ты думаешь: «Надо бы сохранить». Мысль благородная. Но ты откладываешь её на минутку, только дописать фразу. Программа перехватывает эту мысль, находит её самонадеянной и закрывается.

Закрывается она не грубо. Закрывается элегантно. Окошко сообщает «Программа прекратила работу», как официант, объявляющий, что кухня закрыта.

Моя тётя сохраняла каждый документ после каждого слова. На поздравительную открытку у неё уходил целый вечер, зато она ни разу ничего не потеряла. Мы звали её «Сейф».

Настоящий совет: выучите сочетание Ctrl+S и жмите его как нервный тик. И включите автосохранение, если программа его поддерживает.

ЗАКОН ФАЙЛА НА ДРУГОМ КОМПЬЮТЕРЕ



Срочно нужный файл всегда находится на другом компьютере, на другой флешке или в другом доме.

— Аноним, *компьютерный фольклор*

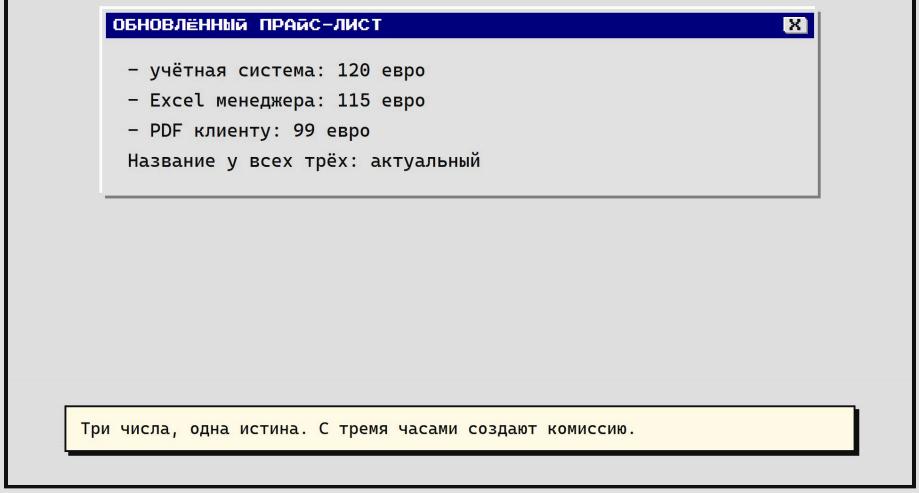
У тебя три компьютера, две флешки, внешний диск и четыре облачных сервиса. Файл со сметой лежит только в одном из этих мест. И это всегда то, к которому нет доступа: офисный компьютер, когда ты дома; домашний ноутбук, когда ты у клиента.

Это называется рассеянием данных, а я называю это диаспорой. Файлы эмигрируют, пускают корни в других местах, меняют имя на «smeta_okonchatelnaya_2_kopiya».

Однажды я час искал документ. Он был во вложении к письму, которое я отправил сам себе с темой «важно». Таких писем с той же темой было четыреста.

Настоящий совет: выберите одно основное рабочее место, с синхронизацией, и давайте файлам понятные имена с датой. Вы из будущего скажете спасибо.

ЗАКОН СЕГАЛА



Человек с одними часами знает, который час. Человек с двумя часами никогда в этом не уверен.

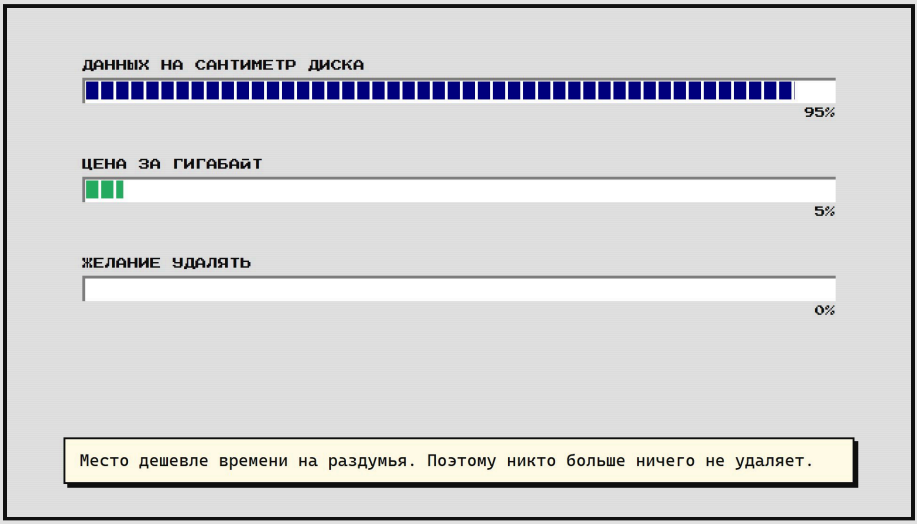
— Аноним, *поговорка, появившаяся в San Diego Union в 1930 году; позже ошибочно приписана Lee Segall и прославлена как «закон Сегала» в книгах Arthur Bloch (авторство не установлено)*

С одними часами сомнений нет: ошибаешься с полной безмятежностью. С двумя часами, показывающими разное время, начинается кризис. Какие правы? С тремя создаётся комиссия.

В информатике это драма множественных источников. Прайс-лист есть в учётной системе, в Excel-таблице менеджера и в PDF, отправленном клиенту. Это три разных числа, и все называются «актуальный». Поэтому те, кто проектирует серьёзные системы, говорят о едином источнике истины: одно место, откуда все читают, а остальные копируют.

Это касается и серверов: если часы компьютеров в сети не синхронизированы, логи рассказывают невозможные истории, где ответы приходят раньше вопросов. У себя в офисе я проблему решил. Убрал все часы, кроме одних. Они стоят с 2016 года, зато никто больше не спорит.

ЗАКОН КРАЙДЕРА



Плотность записи данных на магнитный диск растёт быстрее, чем мощность чипов, и стоимость гигабайта соответственно обваливается.

— Mark Kryder, *статья «Kryder's Law» Chip Walter, Scientific American, август 2005*

Первый коммерческий жёсткий диск весил как холодильник и вмещал несколько мегабайт. Сегодня карточка размером с ноготь вмещает все фото из отпуска, включая сорок одинаковых закатов. Это закон Крайдера: место для данных становится плотнее и дешевле с головокружительной скоростью.

Психологические последствия разрушительны. Никто больше ничего не удаляет. Зачем выбирать, если место стоит дешевле, чем время на решение? У меня есть клиенты с двенадцатью копиями одного PDF под названиями final, final2, final_tochno и final_tochno_v_etot_raz.

В последние годы рост замедлился, но человечество уже нашло решение: копить всё равно. Мой дядя хранит сломанные диски в коробке. Говорит, данные всё ещё внутри, просто немного стесняются.

СОВЕТ СИСАДМИНА

Две вещи, которые почти никто не делает, а надо бы всем. Первая: **спросить у диска, как он себя чувствует**. Почти все диски, и механические, и SSD, ведут журнал своего здоровья под названием SMART. В Windows бесплатная программа вроде CrystalDiskInfo читает его за секунду и сводит к цвету: синий — всё хорошо, жёлтый — предупреждение, красный — беда. Жёлтый — это не мнение, а уведомление: копируйте данные в тот же день. SSD, в отличие от старых дисков, часто умирают без прощального тиканья.

Вторая: **не доверять дискам в спячке**. Внешний диск, годами пролежавший в ящике, может не запуститься, а забытые флешки начинают сбоить. Время от времени включайте архивы, проверяйте, что файлы открываются, и раз в несколько лет переписывайте всё на новый носитель. Данные не стареют. Стареют их контейнеры.

ГЛАВА 3

ГЛ ОЗ — ПОДДЕРЖКА

Хелпдеск: юзеры, пароли и прочие тайны

ЗАКОНОВ: 15 _

Это глава о моей ежедневной работе, поэтому я перечитывал её с особым чувством. Здесь есть звонящий телефон, пользователь, который ничего не трогал, сообщение об ошибке, закрытое со скоростью света, компьютер, исцеляющийся, едва мастер переступает порог.

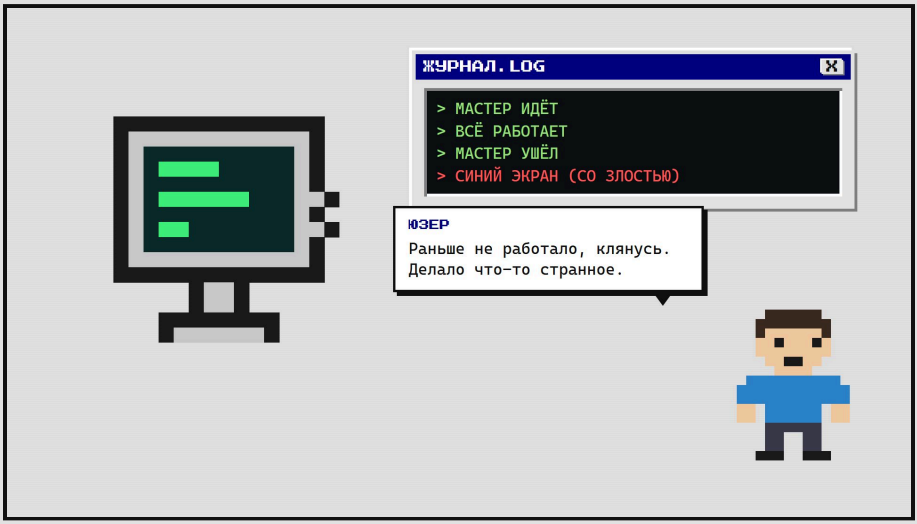
Сразу уточню: здесь мы смеёмся не над пользователями. Мы смеёмся вместе с ними. Каждый мастер — пользователь чего-то другого, и я перед домашним котлом ровно тот господин, который клянётся, что ничего не трогал. Бритва Хэнлона из первой главы говорит об этом лучше меня: за катастрофой обычно стоит не злой умысел. А клик.

Дальше пароли, которые заслуживают отдельного раздела: слишком простые, слишком сложные, забытые на следующий день. А завершают главу несколько серьёзных законов проектирования: почему по маленькой далёкой кнопке промахиваются, почему меню на сорок пунктов парализует и почему полсекунды ожидания кажутся вечностью. Прочитав их, понимаешь, что многие «ошибки пользователя» на самом деле ошибки того, кто рисовал экран.

СОДЕРЖАНИЕ.DIR

01 Закон мастера-целителя
02 Закон «Я ничего не трогал»
03 Закон выдернутой вилки
04 Закон непрочитанного сообщения об ошибке
05 Закон поломки в пятницу вечером
06 Закон родственника, который разбирается
07 Закон забытого пароля
08 Закон вечно слабого пароля
09 Принцип Керкгоффса
10 Законы Гилба о ненадёжности
11 Закон Теслера
12 Закон Якоба
13 Закон Фиттса
14 Закон Хика
15 Порог Доэрти

ЗАКОН МАСТЕРА-ЦЕЛИТЕЛЯ



Проблема исчезает в ту самую секунду, когда приходит мастер, и возвращается, как только он уходит.

— Аноним, *компьютерный фольклор*

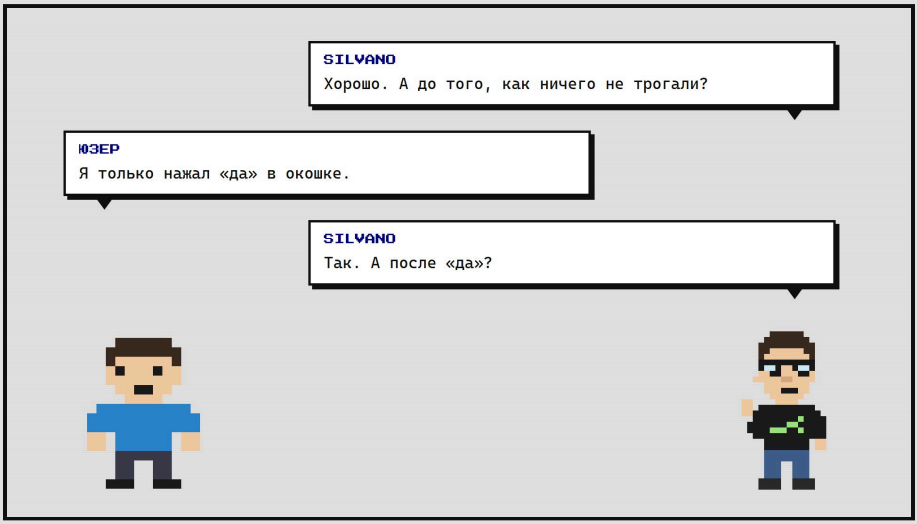
Это моя единственная суперсила. Я вхожу в комнату, и компьютер, который три дня показывал синие экраны, встаёт по стойке смирно, как солдат в казарме. «Раньше не работал, честное слово», — говорит пользователь. Я ему верю. Я видел эту сцену тысячу раз.

Проблема узнаёт мастера, как собака узнаёт ветеринара: ведёт себя хорошо, виляет хвостом, всё отрицает. А когда моя машина выезжает с парковки, компьютер начинает снова, с удвоенной злостью за потерянное время.

Я думал оставить в офисе свою фотографию в полный рост. Один клиент так и сделал, с картонной фигурой. Говорит, работает примерно неделю.

Совет мастера: когда появляется ошибка, сфотографируйте экран на телефон. Одна фотография стоит тысячи «до этого он делал что-то странное».

ЗАКОН «Я НИЧЕГО НЕ ТРОГАЛ»



Пользователь, который говорит «я ничего не трогал», что-то трогал; неизвестно только, что именно.

— Аноним, *компьютерный фольклор*

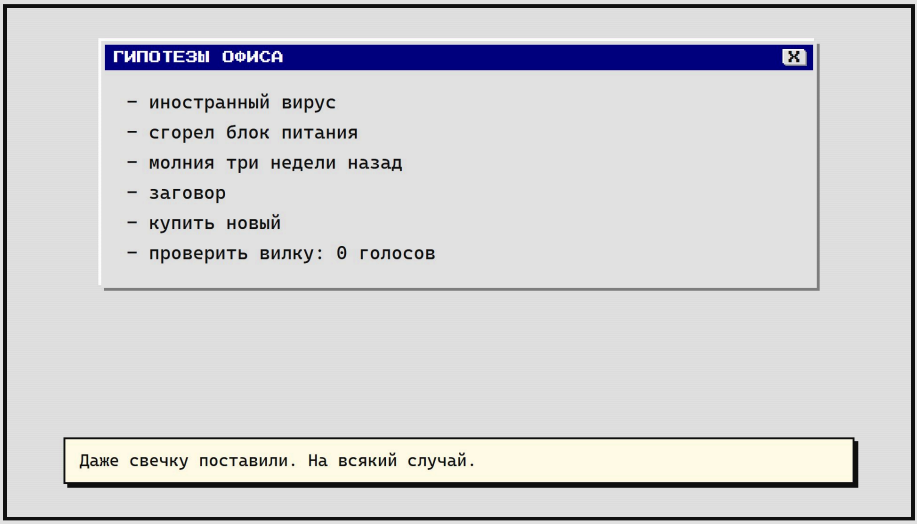
«Я ничего не трогал» — древнейшая фраза техподдержки. Её нашли высеченной на первом ламповом компьютере, рядом с отпечатком пальца.

Пользователь не врёт. В его памяти он ничего не трогал. Он всего лишь нажал «да» в окошке, переместил папку, «которая мешала», установил бесплатную программу, чтобы лучше видеть фотографии, и поменял одну маленькую настройку, которая «была некрасивая». Но трогать — нет.

Опытный мастер не спорит. Он кивает с безмятежностью исповедника и спрашивает: «Хорошо. А до того, как ничего не трогать, что вы делали?» Тут и приходит правда.

Настоящий совет: честный рассказ о последних действиях экономит часы. Мастер не судит, ему нужно только понять, с чего начать.

ЗАКОН ВЫДЕРНУТОЙ ВИЛКИ



Прежде чем проверить, воткнута ли вилка в розетку, все успеют предположить вирус, поломку материнской платы и заговор.

— Аноним, *компьютерный фольклор*

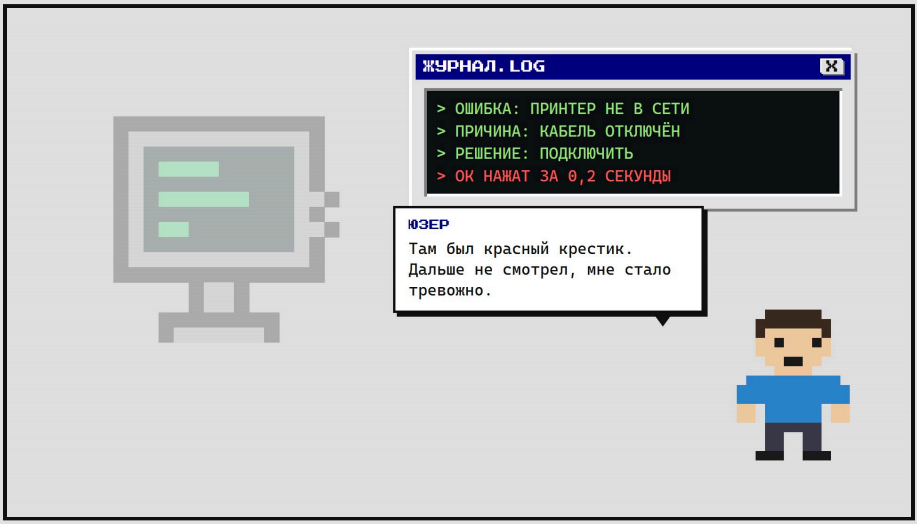
Компьютер не включается. Офис мгновенно раскалывается на школы мысли. Одни говорят о хакерах с другого конца света, другие о сгоревшем блоке питания, третьи о молнии трёхнедельной давности, которая «наверняка что-то оставила». Обсуждается покупка нового компьютера. Ставится свечка.

Потом приходит мастер, наклоняется и втыкает обратно вилку, которую клининговая компания выдернула, чтобы подключить пылесос. Наступившая тишина священна.

Выдернутая вилка — самая скромная причина в информатике. У неё нет кода ошибки, нет логов, нет обаяния. Но она побеждает всегда.

Совет мастера: прежде всего проверяйте очевидное. Вилку, включён ли удлинитель, хорошо ли вставлены кабели монитора. Это не банальность, это метод.

ЗАКОН НЕПРОЧИТАННОГО СООБЩЕНИЯ ОБ ОШИБКЕ



Пользователь закрывает сообщение об ошибке, не прочитав его, а потом звонит мастеру узнать, что там было написано.

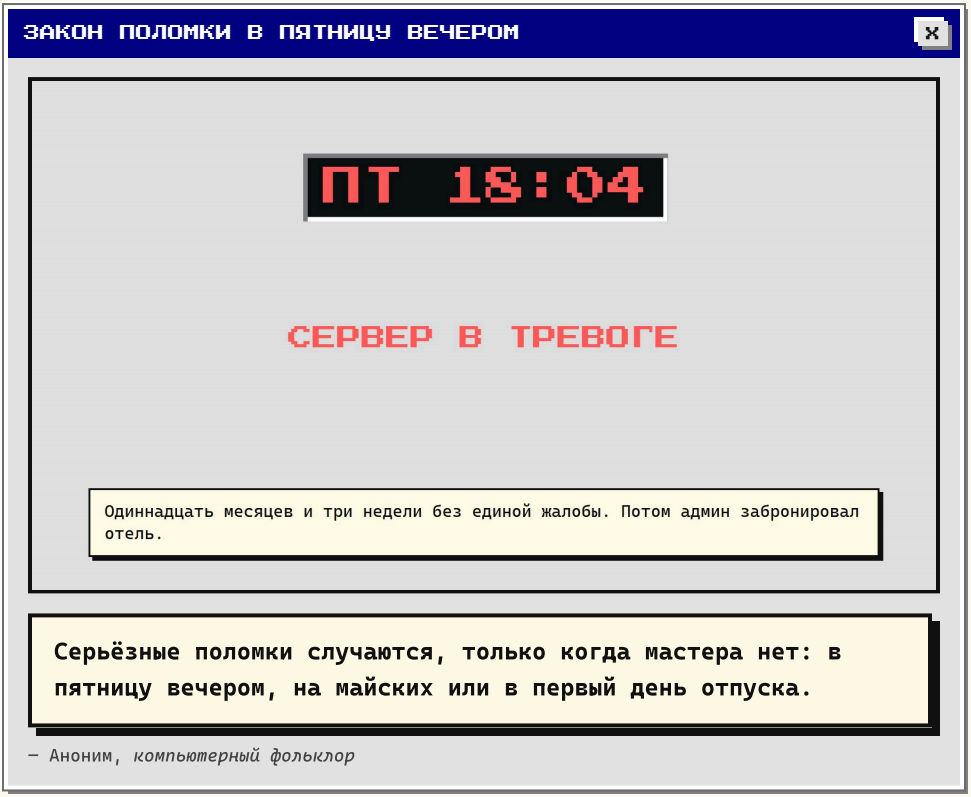
— Аноним, *компьютерный фольклор*

Сообщение об ошибке — маленький непонятый поэт. Оно появляется со всей нужной информацией: проблема, причина, даже решение. Пользователь смотрит на него с ужасом человека, нашедшего паука в ванне, и жмёт «ОК» со скоростью, бросающей вызов физике.

Потом звонит: «Вылезло сообщение». «Что там было написано?» «Не знаю, по-английски. Или по-русски. Там был красный крестик». Это всё, что осталось, как рассказ очевидца НЛО.

Тогда мастеру приходится воспроизводить ошибку, дожидаясь, пока поэт снова начнёт декламировать. Поэт обиделся и не возвращается.

Настоящий совет: когда появляется ошибка, не закрывайте её сразу. Спокойно прочитайте, сфотографируйте или сделайте скриншот. В половине случаев ответ уже там.



Серьёзные поломки случаются, только когда мастера нет: в пятницу вечером, на майских или в первый день отпуска.

— Аноним, *компьютерный фольклор*

Сервер одиннадцать месяцев и три недели проработал без единой жалобы. Потом мастер бронирует гостиницу на море. Сервер об этом знает. Не спрашивайте как: знает.

В пятницу в восемнадцать ноль одна, пока ключи поворачиваются в замке офиса, одна лампочка становится оранжевой. В восемнадцать ноль две — красной. В восемнадцать ноль три звонит телефон мастера, который уже на трассе с семьёй и надувным матрасом на сиденье.

Я знал сисадмина, который шесть лет не ходил в отпуск из страха перед этим законом. Сервер ни разу не сломался. Потом он во вторник утром пошёл к зубному.

Настоящий совет: понятная документация, доступы, записанные в надёжном месте, и запасной коллега. Дежурство — не героизм, а организация.

ЗАКОН РОДСТВЕННИКА, КОТОРЫЙ РАЗБИРАЕТСЯ



Тот, кто в семье «разбирается в компьютерах», проведёт каждое праздничное застолье, настраивая принтер дяди.

— Аноним, *компьютерный фольклор*

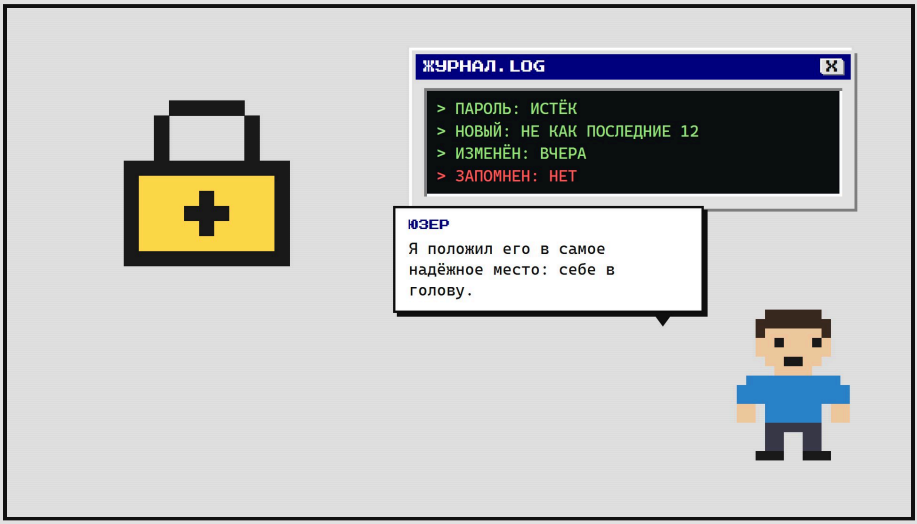
Достаточно одного раза. Одного-единственного раза, когда ты наладил бабушкин телефон, — и тебя пожизненно назначают Главным Айтишником Семьи. Без зарплаты, без отпуска, с постоянным дежурством по праздникам.

Приезжаешь на новогоднее застолье и не успеваешь снять пальто. «Раз уж ты здесь, глянешь компьютер?» Компьютер на антресолях, ему двенадцать лет, и он покрыт слоем пыли, служащим теплоизоляцией.

К концу дня ты съел ложку холодного оливье, обновил четыре телефона и шесть раз объяснил, что Wi-Fi — это не «интернет». Расплачиваются с тобой банкой варенья. Отличного, надо признать.

Настоящий совет: запишите на листке пароли от Wi-Fi и основных аккаунтов родственников и оставьте им в надёжном месте. Звонков станет вдвое меньше.

ЗАКОН ЗАБЫТОГО ПАРОЛЯ



Пароль всегда забывается на следующий день после того, как ты его сменил, потому что этого требовала система.

— Аноним, *компьютерный фольклор*

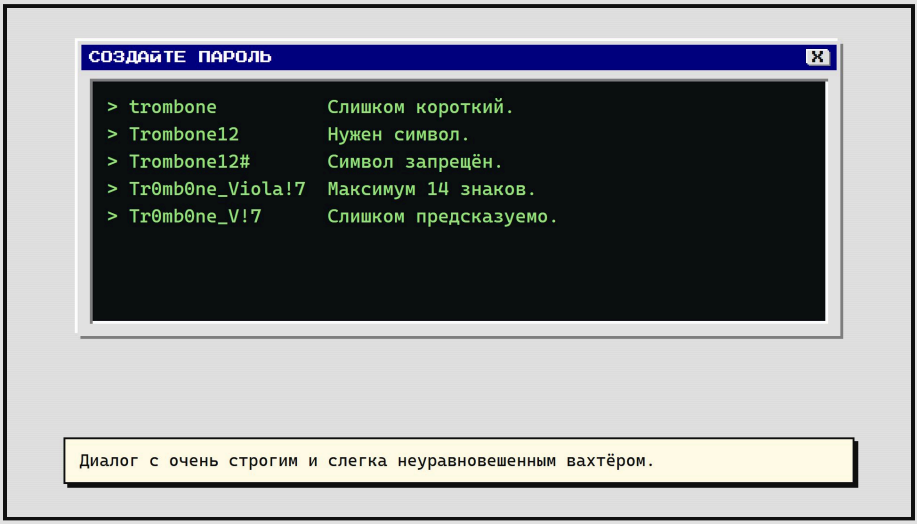
Три года ты пользовался одним и тем же паролем. Знал его, как имя собственного ребёнка. Потом система решает, что он устарел, и требует новый, не похожий на двенадцать предыдущих. Ты старательно его придумываешь. На следующий день он исчезает из памяти, как сон при пробуждении.

И начинается охота на сброс. «Введите ответ на секретный вопрос». Кличка вашего первого питомца. Ты писал «Барсик», «барсик» или «БАРСИК»? Система не говорит. Блокирует тебя на тридцать минут, чтобы ты подумал.

Жёлтый стикер под клавиатурой тем временем смеётся. Он всегда был на месте.

Настоящий совет: пользуйтесь менеджером паролей. Вы помните один надёжный пароль, а он хранит остальные. Стикер менеджером не считается.

ЗАКОН ВЕЧНО СЛАБОГО ПАРОЛЯ



Сайт отвергает любой пароль, пока его не станет невозможно запомнить, а потом отвергает и этот, потому что в нём недопустимый символ.

— Аноним, *компьютерный фольклор*

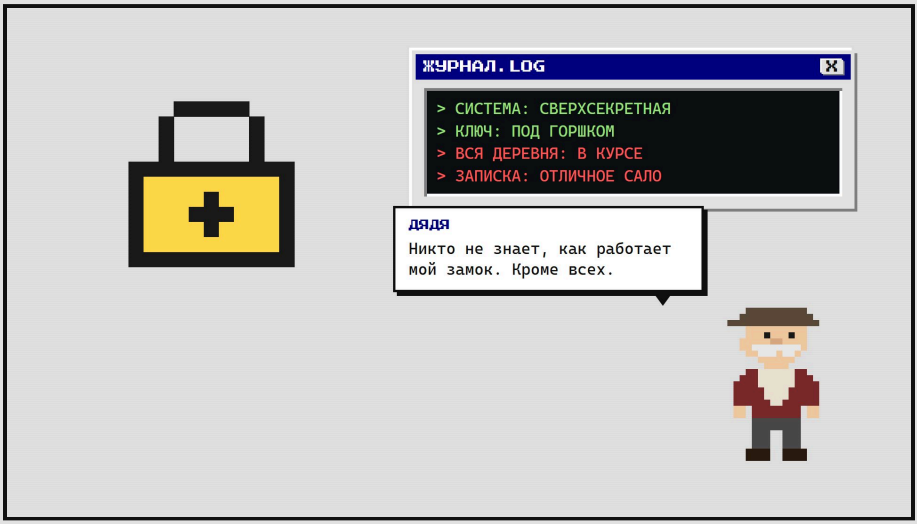
«Пароль должен содержать не менее восьми символов». Готово. «Заглавную букву». Готово. «Цифру». Готово. «Спецсимвол». Готово. «Пароль не может содержать использованный вами спецсимвол». Ах.

Это диалог с очень строгим и слегка неуравновешенным вахтёром. Каждая попытка открывает новое правило, как в видеоигре, где финальный уровень написал комитет.

В итоге получается что-то вроде «Tr0mb0ne_Viola!7», гордое, как флаг. Ты записываешь его на листок. Листок отправляется в ящик. У ящика нет замка. Информационная безопасность сделала полный круг и вернулась в исходную точку.

Настоящий совет: длинная фраза из случайных слов лучше короткого слова, набитого символами. Она надёжнее и легче запоминается.

ПРИНЦИП КЕРКГОФФСА



Криптографическая система должна оставаться надёжной, даже если противнику известно всё о её устройстве: единственным секретом должен быть ключ.

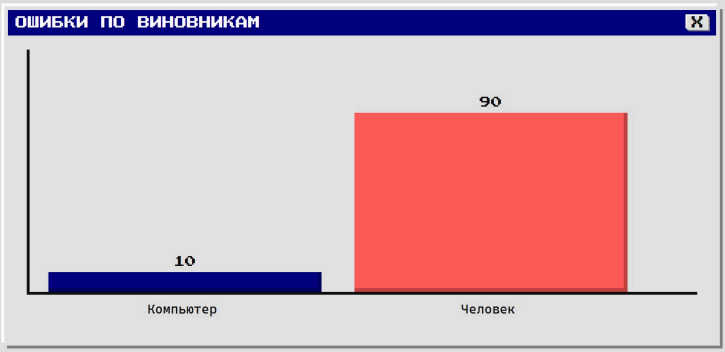
— Auguste Kerckhoffs, *La cryptographie militaire*, 1883

Противоположный подход называется «безопасность через неясность»: программа надёжна, потому что никто не знает, как она работает. Длится это ровно до первого любопытного со свободным вечером.

Керкгоффс рассуждал как военный: устройства попадают в плен, руководства копируют, шпионов подкупают. Если вся безопасность держится на секрете механизма, достаточно одной кражи, чтобы потерять всё. Если на ключе — меняешь ключ и начинаешь заново. Поэтому серьёзные алгоритмы публичны и изучены всеми.

Мой дядя защищает погреб сверхсекретной системой: ключ под горшком с геранью. Система неясная, но вся деревня её знает. В воскресенье ему оставили записку: «Отличная колбаса, удачный урожай вина». Безопасность через герань: не рекомендуется.

ЗАКОНЫ ГИЛБА О НЕНАДЁЖНОСТИ



Компьютер ошибается всегда одинаково. Человек ошибается творчески.

Компьютеры ненадёжны, но люди ещё ненадёжнее; и любая система, зависящая от человеческой надёжности, ненадёжна.

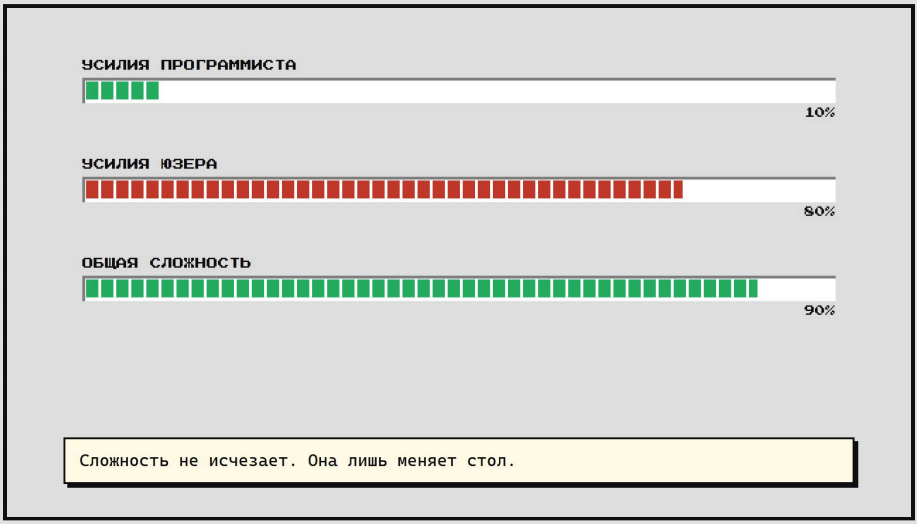
— Tom Gilb, *законы из сборников 70-х годов (например, Arthur Bloch); нумерация и формулировки в разных источниках различаются (авторство частично не установлено)*

Компьютер ошибается, да. Но ошибается всегда одинаково, и рано или поздно ты это понимаешь. Человек ошибается творчески: сегодня пароль на стикере, завтра стикер на мониторе, послезавтра монитор сфотографирован и выложен в семейный чат.

Урок инженерный: не проектируй системы, которые работают, только если все внимательны. Люди устают, отвлекаются, спешат. Автоматические проверки, подтверждения и бэкапы существуют ради них.

В одной компании была безотказная процедура бэкапа: каждый вечер сотрудник менял диск. Работало годами. Потом сотрудник ушёл на пенсию. Диск всё ещё там, вставлен с 2015 года, верный, как пёс на вокзале.

ЗАКОН ТЕСЛЕРА



В каждом приложении есть доля сложности, которую нельзя устранить: можно только решить, кто будет её нести — программист или пользователь.

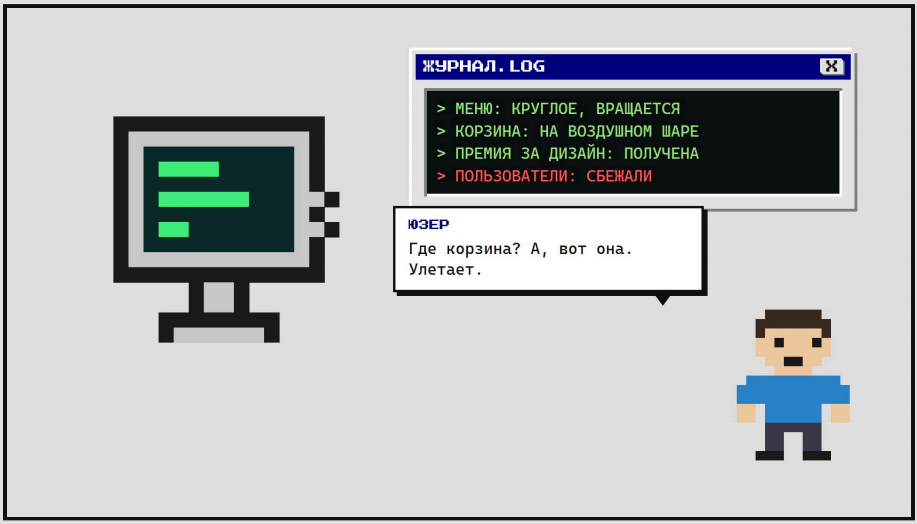
— Larry Tesler, *закон сохранения сложности, сформулированный в 80-е годы во время работы над интерфейсами, в том числе в Apple*

Сложность — как энергия: не создаётся и не исчезает, а только перемещается. Форма с одним-единственным «простым» полем означает, что где-то за ней кто-то написал тысячу строк, чтобы понять, что имел в виду пользователь.

Закон — это призыв к вежливости: лучше пусть один раз помучается разработчик, чем каждый день десять тысяч пользователей. К сожалению, обратное искушение сильно, и так рождается форма с сорока обязательными полями.

У нас в офисе упростили принтер: теперь у него одна кнопка. Вся сложность перешла ко мне: я получаю двадцать два звонка в день с объяснениями, что делает эта кнопка. Сохранение работает. Я — уже не очень.

ЗАКОН ЯКОБА



Пользователи проводят большую часть времени на других сайтах, поэтому ожидают, что и твой будет работать так же, как те, что они уже знают.

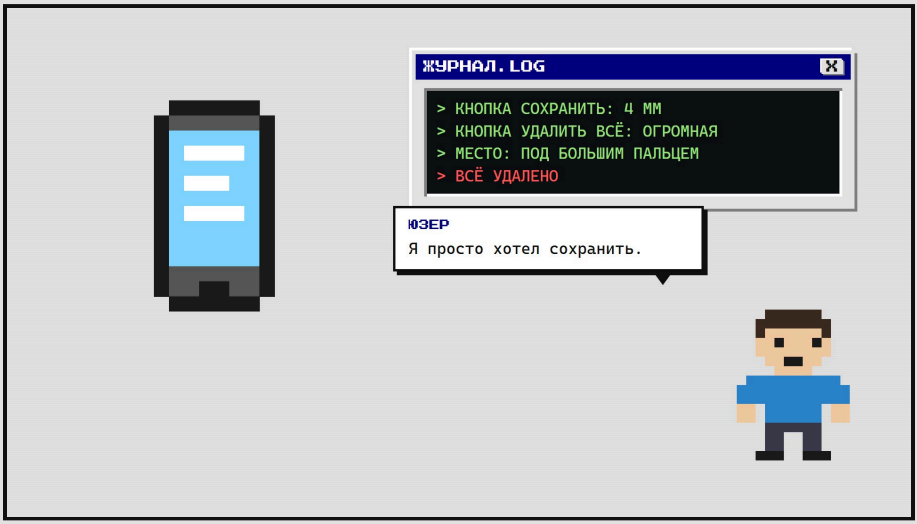
— Jakob Nielsen, *Alertbox, Nielsen Norman Group, 2000*

Корзина находится справа вверху. Логотип слева вверху ведёт на главную. Лупа ищет. Никто не утверждал это законом, но миллионы людей выучили это, пользуясь им повсюду. Закон Якоба гласит: не борись с этими привычками, оседлай их.

Время от времени появляется дизайнер-визионер, который хочет поразить. Круглое меню, вращающееся при прокрутке. Кнопка покупки, спрятанная в анимированном воздушном шаре. Сайт получает премию. Пользователи сбегают на скучный сайт, где корзина там, где ей положено быть.

Быть оригинальным прекрасно в цвете, фотографиях, текстах. В навигации оригинальность — это расход, который оплачивает пользователь. Однажды я из художественных соображений поставил меню слева внизу. Клиент нашёл его через неделю. Он пришёл спросить, почему на его сайте только одна страница.

ЗАКОН ФИТТСА



Время, нужное, чтобы движением достичь цели, зависит от расстояния до неё и её размера: чем она дальше и меньше, тем дольше и тем чаще промахиваешься.

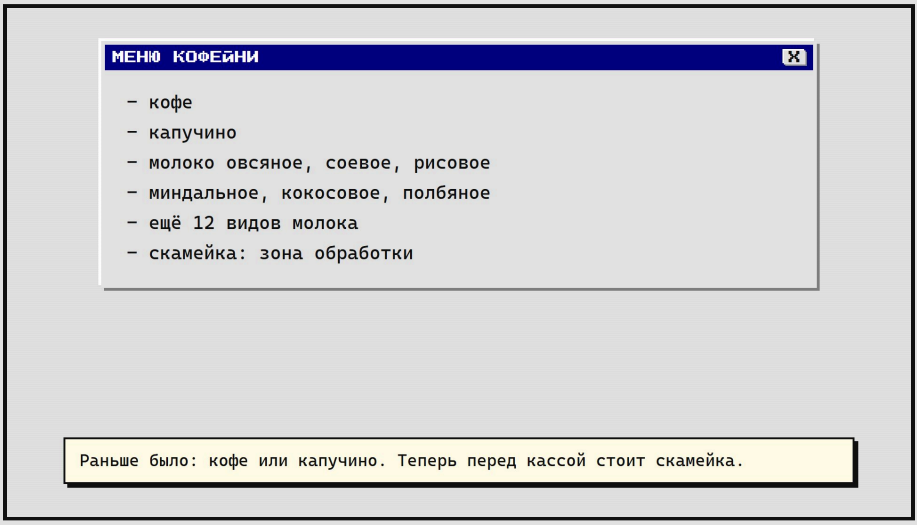
— Paul Fitts, *Journal of Experimental Psychology*, 1954

Психолог пятидесятых изучал движения рук задолго до появления мышей. Но его формула словно написана для тех, кто делает интерфейсы: большая и близкая кнопка нажимается быстро, маленькая и далёкая — испытание ловкости, как в тире на ярмарке.

Поэтому важные кнопки должны быть большими, а опасные — далёкими. А часто бывает наоборот: кнопка «Сохранить» размером с почтовую марку, а «Удалить всё» — с дверь, прямо под большим пальцем. На телефоне это превращается в экстремальный спорт.

Края экрана — бесконечные мишени, потому что мышь сама там останавливается. Это доказывает мой дядя, который всегда открывает меню «Пуск», швыряя курсор в левый нижний угол резким движением кисти. Говорит, это рыбацкая техника. Ни разу не промахнулся.

ЗАКОН ХИКА



Время на принятие решения растёт с числом доступных вариантов (логарифмически, а не линейно).

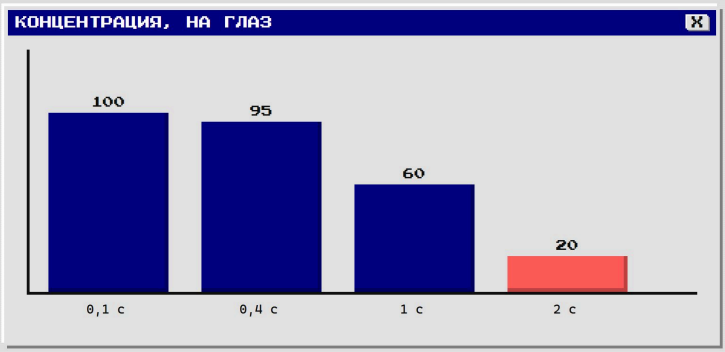
— William Edmund Hick (и Ray Hyman), *Quarterly Journal of Experimental Psychology, 1952; исследование Hyman, 1953 (известен также как закон Хика — Хаймана)*

Меню из трёх пунктов читается и выбирается мгновенно. Меню из сорока пунктов, каждый с подменю, — это полдня жизни. Закон Хика гласит: чем больше вариантов ставишь перед человеком, тем дольше он решает. Растёт медленнее, чем пропорционально, но растёт.

Поэтому хорошие программы прячут расширенные настройки и показывают только нужное. И поэтому окно параметров некоторых программ, с тридцатью двумя вкладками и двумя сотнями галочек, вызывает у пользователя неподвижный взгляд в горизонт.

Я вижу это каждый день в кофейне. Раньше было «эспрессо или капучино». Теперь там меню с восемнадцатью видами молока. Бариста поставил перед кассой скамейку. Называет её зоной обработки.

ПОРОГ ДОЭРТИ



Выше порога мысли начинают блуждать. А блуждают они в интернете.

Продуктивность заметно растёт, когда компьютер отвечает пользователю быстрее чем примерно за 400 миллисекунд: ниже этого порога никто из них не ждёт другого.

— Walter J. Doherty и Arvind J. Thadhani, *«The Economic Value of Rapid Response Time»*, IBM, 1982 (значение 400 мс приводится в основном в более поздних источниках)

В начале восьмидесятых считалось, что две секунды ожидания приемлемы. Исследование IBM утверждало обратное: если компьютер отвечает быстрее чем примерно за полсекунды, пользователь остаётся сосредоточенным и работает гораздо больше. Дольше — и мысли начинают блуждать. А когда они блуждают, то уходят в интернет.

Поэтому современные интерфейсы сразу что-нибудь показывают: анимацию, полоску, серый скелет страницы. Быстрее не становится, но выглядит живым. Пользователю не нужен мгновенный результат. Ему нужно знать, что его кто-то слушает.

Я применил порог Доэрти к телефонной поддержке. Отвечаю в пределах четырёхсот миллисекунд: «Да, слушаю». Потом, конечно, решаю проблему час. Но клиент чувствует, что им занимаются. Всё дело в эмоциональном индикаторе выполнения.

СОВЕТ СИСАДМИНА

Большая часть времени поддержки уходит на то, чтобы понять, что случилось. Вы можете подарить это время своему мастеру, и он будет вспоминать вас с нежностью. Когда что-то не работает, перед звонком подготовьте пять вещей:

- **Что вы делали**, шаг за шагом, включая подробности «я ничего не трогал».
- **Точное сообщение**, сфотографированное. В Windows 10 и 11 клавиши Windows + Shift + S делают снимок части экрана.
- **С какого момента** это происходит и работало ли раньше.
- **Что изменилось** за это время: обновление, новая программа, переезд стола.
- **Всегда ли** это происходит или только иногда, на одном компьютере или на всех.

С этими ответами многие поломки решаются по телефону. Остальные — быстрее, потому что мастер приходит уже с нужным инструментом в руках.

ГЛАВА 4

ГЛ 04 — СЕТЬ

Сети и интернет: невидимая нить

ЗАКОНОВ: 15 _

Сеть — самая используемая и самая непонятая вещь в современном доме. Все её хотят, никто не знает, где она. В роутере? В стене? В воздухе? Моя соседка уверена, что она приходит через телефонную розетку и по ночам отдыхает. Я не стал спорить: в некоторых районах она права.

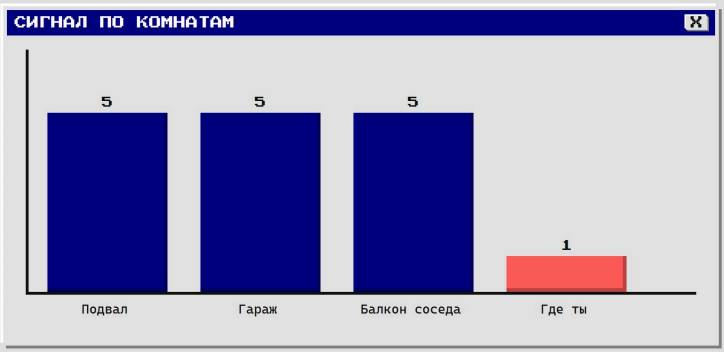
Глава открывается законами домашнего и офисного подключения: Wi-Fi, который ловит везде, кроме того места, где ты; загрузка, умирающая на девяноста девяти; видеозвонок, замерзающий, когда очередь доходит до тебя. Потом — законы большого интернета, рождённые на форумах, в онлайн-СМИ и групповых чатах. Здесь техника значит мало, а люди — очень много: кто поправляет с ошибкой, кто ставит в заголовок вопросительный знак, кто пытается что-то скрыть и делает это знаменитым.

Эти законы действуют и за пределами экрана. Закон Брандолини, например, я видел в деле за ужином, на деревенском празднике и однажды в приёмной: опровергнуть глупость всегда труднее, чем её сказать. Эта глава его не опровергает. Она его документирует.

СОДЕРЖАНИЕ.DIR

01 Закон вредного Wi-Fi
02 Закон перезагрузки роутера
03 Закон решающего видеозвонка
04 Закон загрузки на 99%
05 Закон сайта, который работает везде
06 Закон Меткалфа
07 Закон Нильсена о пропускной способности
08 Принцип устойчивости
09 Закон Каннингема
10 Закон Скитта
11 Закон заголовков Беттериджа
12 Закон По
13 Закон Годвина
14 Эффект Стрейзанд
15 Закон Брандолини

ЗАКОН ВРЕДНОГО WI-FI



Палочки сигнала. Wi-Fi — кошка: гуляет где хочет.

Wi-Fi прекрасно работает в каждой комнате дома, кроме той, где находишься ты.

— Аноним, *компьютерный фольклор*

Сигнал Wi-Fi — это кот. Ходит, где хочет. В подвале ловит отлично, в гараже тоже, на балконе соседа показывает в высоком разрешении. Но ты в гостиной, а в гостиной две палочки мигают, как сигнал бедствия.

И начинается обряд. Сдвигаешься на полметра. Поднимаешь телефон к небу, как подношение. Садишься на подлокотник дивана в позе, которую не одобрил бы физиотерапевт. Работает. Сдвигаешься на сантиметр. Больше не работает.

У меня есть клиент, который спрятал роутер в шкаф из массива дерева, «чтобы не видеть». Теперь его не видит и Wi-Fi.

Совет мастера: роутер должен стоять в центре, повыше и на открытом месте. Толстые стены, зеркала и аквариумы — его естественные враги.

ЗАКОН ПЕРЕЗАГРУЗКИ РОУТЕРА



До звонка в техподдержку никто не перезагружает роутер; после звонка проблема решается перезагрузкой роутера.

— Аноним, *компьютерный фольклор*

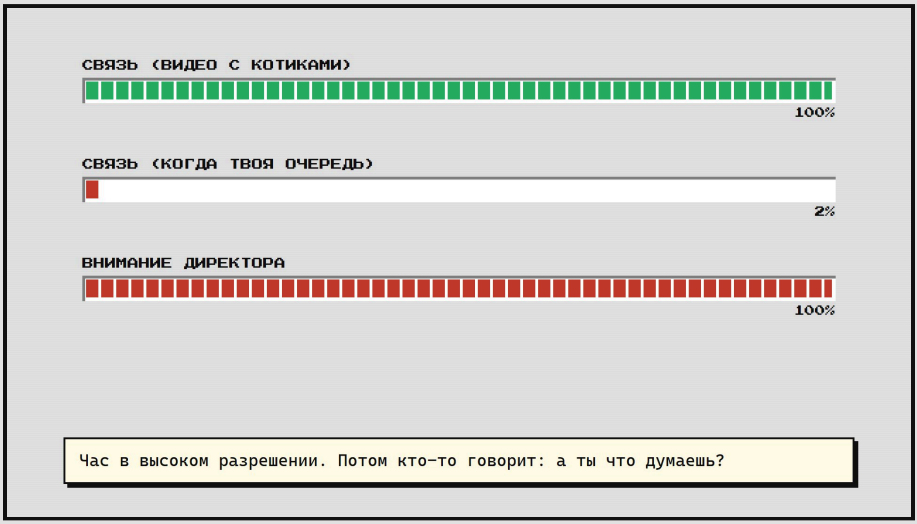
Перезагрузка роутера — светское таинство цифровой эпохи. Все знают её силу, никто по-настоящему в неё не верит, пока об этом не скажет кто-то в гарнитуре с бесплатной горячей линии.

«Вы пробовали выключить и включить?» «Конечно». «Можете сделать это ещё раз, сейчас, вместе со мной?» Тишина. Звук отодвигаемой мебели. Лампочки гаснут. Лампочки загораются. «А. Теперь работает».

Братья и сёстры соединения! Роутер — не прибор. Это состояние души, которое время от времени нужно сбрасывать, как нас по понедельникам.

Совет мастера: выключите, подождите хотя бы тридцать секунд, потом включите и дайте роутеру пару минут, чтобы подняться. Если проблема возвращается часто, ищите причину: перезагрузка — это пластырь, а не лечение.

ЗАКОН РЕШАЮЩЕГО ВИДЕОЗВОНКА



Соединение часами выдерживает видео с котиками, но обрывается ровно тогда, когда на важном видеозвонке очередь говорить доходит до тебя.

— Аноним, *компьютерный фольклор*

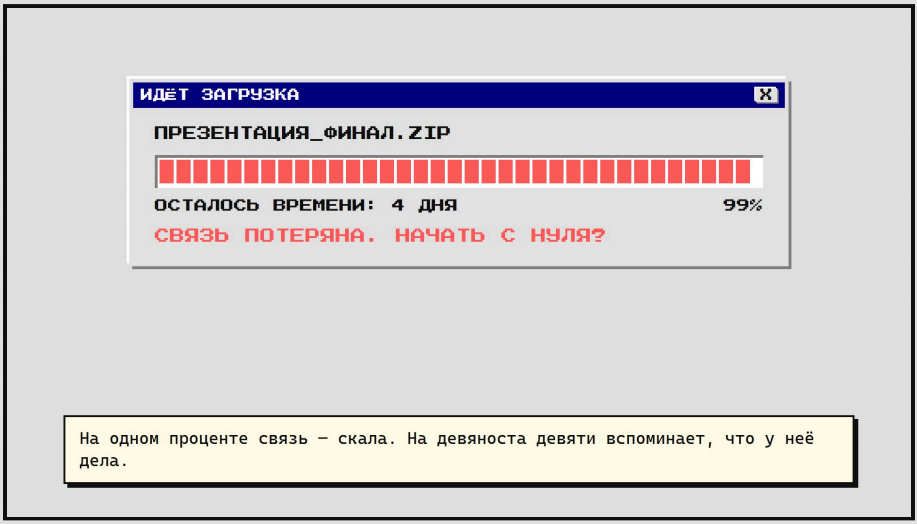
Всё совещание соединение было безупречным. Ты видел коллег в высоком разрешении, слышал дыхание директора, пересчитал растения за его столом. Потом кто-то говорит: «А ты что думаешь?»

В этот самый миг твоё лицо застывает в непроницаемом выражении, где-то между изумлением и чиханием. Голос становится голосом робота под водой. Остальные видят тебя таким сорок секунд и решают двигаться дальше.

Когда ты возвращаешься в сеть, совещание уже всё решило. Насчёт тебя. В твоё отсутствие.

Настоящий совет: для важных звонков, если можете, подключайте компьютер к роутеру сетевым кабелем. Wi-Fi удобен, кабель надёжен.

ЗАКОН ЗАГРУЗКИ НА 99%



На одном проценте связь — скала. На девяноста девяти вспоминает, что у неё дела.

Скачивание или загрузка всегда обрывается на девяноста девяти процентах и никогда — на одном.

— Аноним, *компьютерный фольклор*

Ни одно скачивание ещё не сорвалось на одном проценте. Это был бы честный провал, без боли. Нет: полоска час медленно ползёт, доходит до девяноста девяти, даёт тебе увидеть землю обетованную, а потом появляется надпись «ошибка сети».

Учёные называют это жестокостью. Мастера называют это «последний процент», который, как известно, длится столько же, сколько остальные девяносто девять. Говорят, именно там компьютер делает серьёзные расчёты. На самом деле он просто решает, любит ли он тебя.

Я видел, как взрослые мужчины двадцать минут, не дыша, смотрели на полоску, застывшую на девяноста девяти, как смотрят решающий пенальти.

Совет мастера: для больших файлов пользуйтесь программами или сервисами, которые умеют докачивать. Тогда при обрыве загрузка продолжится с того места, где остановилась.

ЗАКОН САЙТА, КОТОРЫЙ РАБОТАЕТ ВЕЗДЕ

ТЕСТЫ ПРОЙДЕНЫ

[x] 6 браузеров
[x] 4 телефона
[x] 2 планшета
[x] умный холодильник в комнате отдыха
[ ] компьютер клиента

Идеально везде. Кроме единственного места, которое важно.

Сайт работает на всех компьютерах офиса, кроме компьютера клиента, который должен его утвердить.

— Аноним, *компьютерный фольклор*

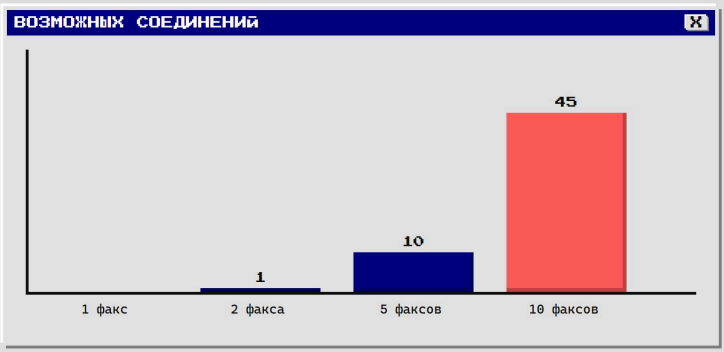
Сайт проверен в шести браузерах, на четырёх телефонах, двух планшетах и умном холодильнике в комнате отдыха. Всё идеально. Потом клиент открывает его со своего компьютера и видит меню, съехавшее влево, и логотип размером с грузовик.

Компьютер клиента — место загадочное. В нём браузер, который обновляется только по чётным дням, шесть панелей инструментов, установленных племянником в 2016 году, и масштаб сто пятьдесят процентов, «потому что так лучше читается».

И клиент никогда не звонит сказать «у меня на компьютере плохо отображается». Он говорит: «Сайт не работает». Тоном человека, объявляющего о затмении.

Настоящий совет: прежде чем винить сайт, попробуйте очистить кэш и вернуть масштаб на сто процентов. Это решает больше проблем, чем кажется.

ЗАКОН МЕТКАЛФА



Один факс — шумная безделушка. Десять факсов — общество, со сплетнями.

Ценность сети растёт пропорционально квадрату числа подключённых устройств (или людей): каждый новый узел увеличивает возможные связи для всех остальных.

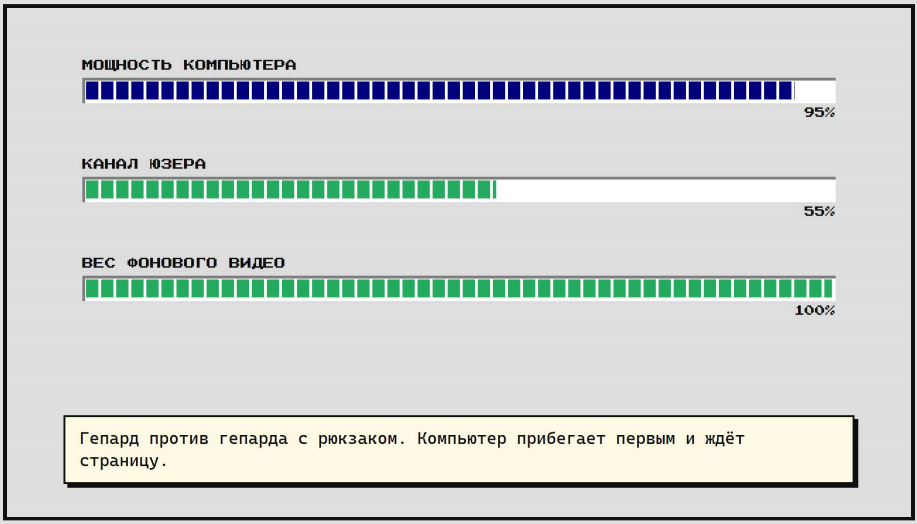
— Robert Metcalfe, *сформулирован около 1980 года для продвижения Ethernet; название «закон Меткалфа» дал George Gilder, 1993*

Один факс сам по себе бесполезен. Это шумная безделушка. Два факса могут поговорить. Десять факсов — это уже общество, со сплетнями и термобумагой. Вот закон Меткалфа: каждый новый участник увеличивает ценность сети для всех, и не слегка, а в квадрате.

Именно поэтому никто не уходит из соцсети или группового чата: уйдёшь — потеряешь всех остальных. Ценность не в программе, а в людях, запертых внутри. Счастливо запертых, разумеется.

Обратную сторону медали я знаю хорошо. В чате нашего дома сорок участников. По формуле он стоит очень дорого. На практике он производит тысячу шестьсот возможных скандалов из-за велосипедной парковки. Это тоже, математически, ценность.

ЗАКОН НИЛЬСЕНА О ПРОПУСКНОЙ СПОСОБНОСТИ



Скорость подключения продвинутого пользователя растёт примерно на 50% в год — медленнее, чем мощность компьютеров.

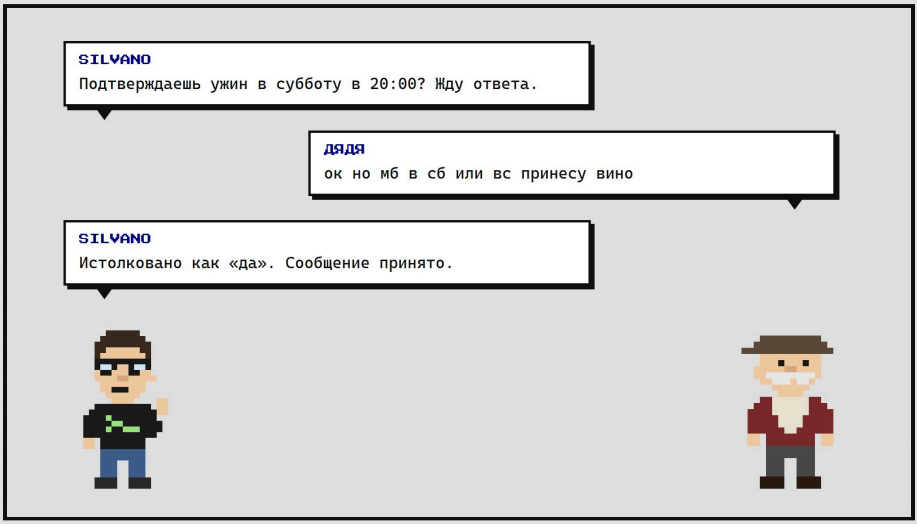
— Jakob Nielsen, *Alertbox, Nielsen Norman Group, 1998 (обновлён в 2008 и 2019 годах)*

Процессоры удваиваются каждые два года, канал растёт на пятьдесят процентов в год. Кажется, много, но это как забег гепарда против гепарда с рюкзаком. Компьютер всегда приходит первым. Потом садится и ждёт, пока загрузится страница.

Урок для тех, кто делает сайты, простой: не считай, что у всех оптоволокно. Где-то всегда есть пользователь на хуторе, с модемом, пристроенным на подоконнике в единственной точке, где ловит, который пытается открыть твоё фоновое видео на девяносто мегабайт.

У меня такой клиент есть. Чтобы улучшить сигнал, он прикрепил модем на верхушку шеста для фасоли. Работает. Но когда дует ветер, интернет качается.

ПРИНЦИП УСТОЙЧИВОСТИ



Будь консервативен в том, что отправляешь, и либерален в том, что принимаешь от других.

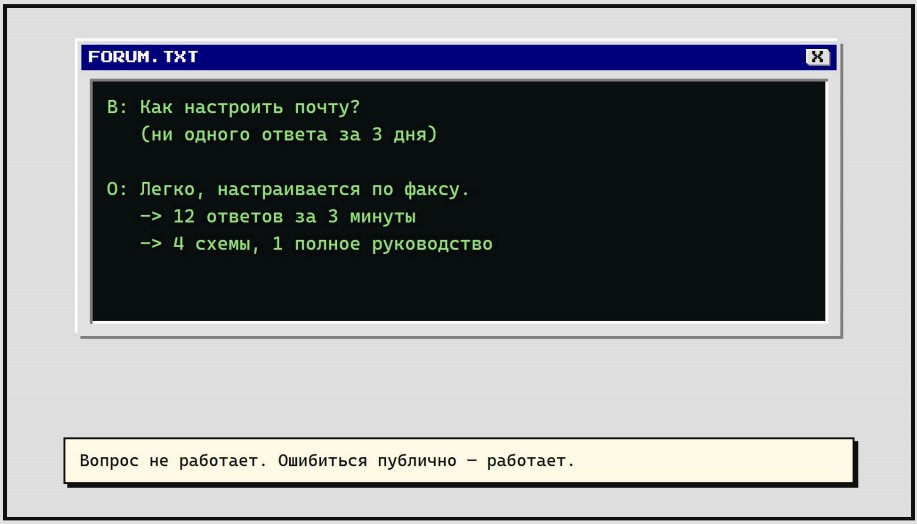
— Jon Postel, *спецификации интернет-протоколов, RFC 760 (1980) и RFC 793 (1981)*

Это этикет интернета: когда говоришь, соблюдай протокол буквально; когда слушаешь, будь снисходителен к тем, кто ошибся в запятой. Благодаря этой вежливости машины, написанные разными людьми, понимают друг друга.

Есть и обратная сторона: если все принимают ошибки, их никто никогда не исправляет. После тридцати лет терпимости браузеры читают HTML-страницы, которые будто написаны котом на клавиатуре, и обязаны делать это вечно.

Я применил этот принцип к своей социальной жизни. Отправляю безупречные сообщения, с пунктуацией нотариуса. Принимаю любой ответ, даже просто палец вверх. Тётя ответила мне баклажаном. Я истолковал это как «да». Теперь я свидетель на свадьбе.

ЗАКОН КАННИНГЕМА



Лучший способ получить правильный ответ в интернете — не задать вопрос, а написать неправильный ответ.

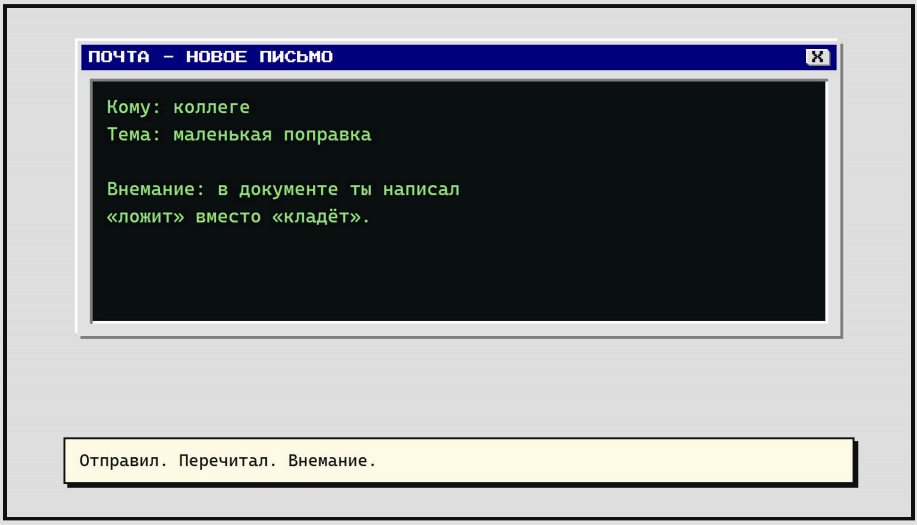
— Steven McGeady, *назван Steven McGeady в 2010 году в честь Ward Cunningham, который, однако, отрицал, что когда-либо это говорил (авторство оспаривается)*

Спрашиваешь на форуме, как настроить почтовый сервер: тишина, только перекати-поле катится. Уверенно пишешь неправильную инструкцию: за три минуты прибегают двенадцать экспертов, четыре схемы и исправление за подписью инженера на пенсии.

Это работает, потому что поправить кого-то приятнее, чем помочь. Это двигатель знаний, работающий на желании быть правым. Финальный парадокс: человек, в честь которого назван закон, утверждает, что никогда его не произносил. Закон доказал сам себя.

Я пользуюсь им для всего. Написал в одной группе, что Joomla устанавливается по факсу. Теперь у меня есть полное, свежее руководство со скриншотами. А ещё факс, который мне прислал один пользователь, чтобы доказать, что так не работает.

ЗАКОН СКИТТА



Любое сообщение, исправляющее чужую ошибку, почти наверняка будет содержать хотя бы одну собственную.

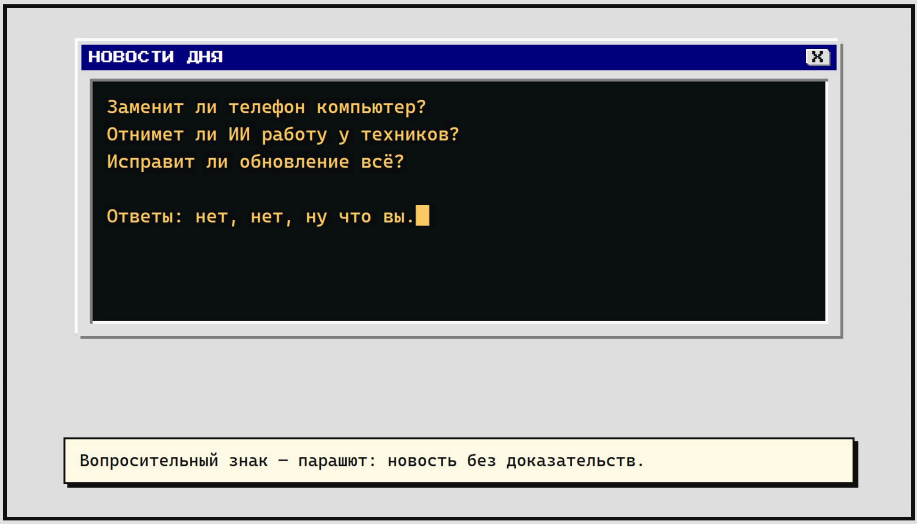
— Аноним (псевдоним «Skitt»), *Usenet, группа alt.usage.english, около 1999 года (авторство не установлено; более ранние варианты известны как «закон Мафри»)*

Пишешь коллеге письмо: «Внимание, в документе ты написал «извените» вместо «извините»». Жмёшь «отправить». Перечитываешь. Ты написал «Внемание». Коллега отвечает одним-единственным словом, и это слово безупречно.

Закон Скитта — закон природы для исправлений. Вселенная не терпит орфографической гордыни: каждый раз, когда ты строишь из себя учителя, мятежная буква отрывается от клавиатуры и прячется в самом заметном месте твоего сообщения. Для кода это тоже верно: патч, исправляющий баг коллеги, вносит твой собственный.

Я из осторожности больше никого не поправляю. Поручил это автокорректору. Вчера он заменил «сервер» на «север». Клиент прочитал «север упал» и спросил, не пора ли звонить в МЧС.

ЗАКОН ЗАГОЛОВКОВ БЕТТЕРИДЖА



На любой газетный заголовок, заканчивающийся вопросительным знаком, можно ответить «нет».

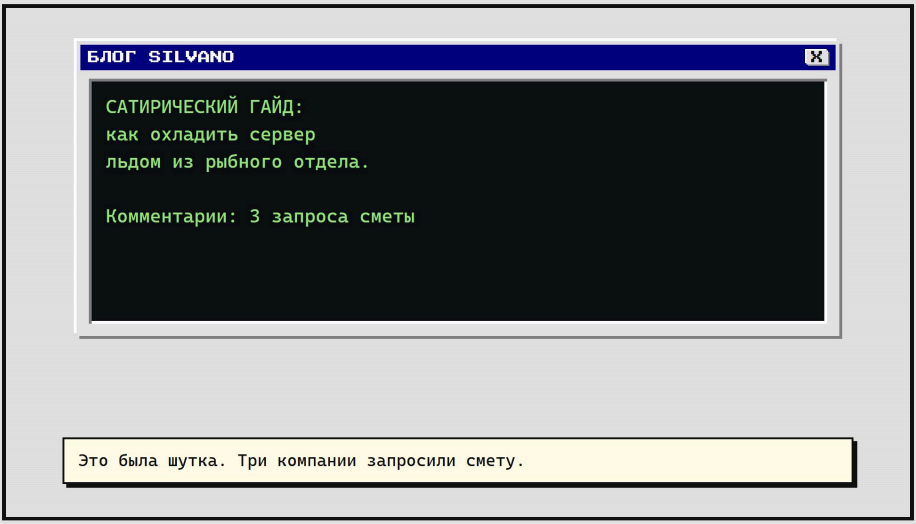
— Ian Betteridge, *запись в его блоге, февраль 2009 (идея ходила под другими названиями уже в 80-е)*

«Заменит ли этот новый телефон компьютер?» Нет. «Отнимет ли искусственный интеллект работу у мастеров?» Нет, наоборот, нам же его и настраивать. «Решит ли новое обновление все проблемы?» Ой, да бросьте.

Закон Беттериджа раскрывает древний трюк: если бы у журналиста были доказательства для «да», он написал бы утверждение. Вопросительный знак — это парашют. Он позволяет запустить сенсацию, не доказывая её, и собрать клики тех, кто надеется.

Я провёл эксперимент в своём блоге. Опубликовал «Вернётся ли мода на дискеты?». Три тысячи просмотров. Потом опубликовал «Мода на дискеты не вернётся». Два просмотра, один из них — моя мама. Вопросительный знак, понял я, — настоящий двигатель цифровых СМИ.

ЗАКОН ПО



Без ясного сигнала о намерении автора пародию на крайнюю позицию невозможно отличить от искреннего мнения того же рода.

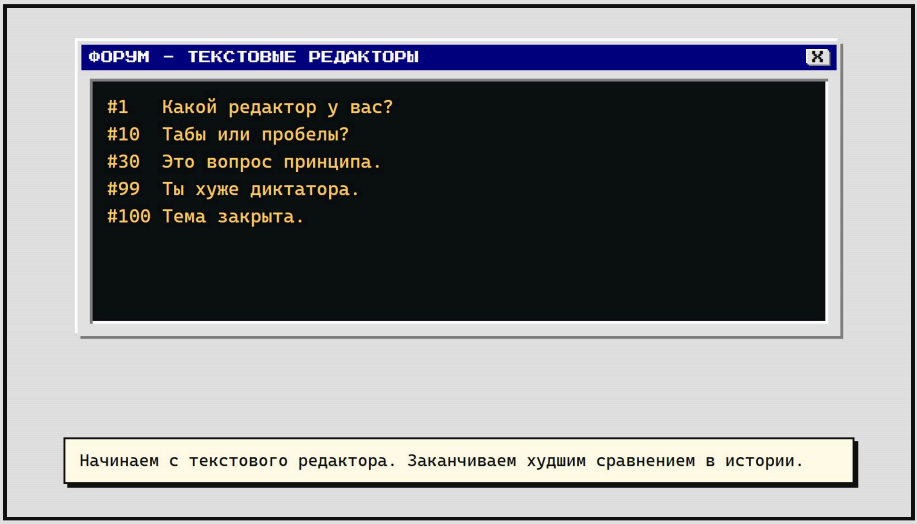
— Nathan Poe, *форум christianforums.com, 2005 (похожее предупреждение встречалось в Usenet ещё в 1983 году)*

Пишешь в интернете, что лучший способ защитить пароль — распечатать его и приклеить к монитору, тогда воры не станут искать на самом видном месте. Это шутка. Три человека тебя благодарят. Один так и делает. Ещё один открывает платный курс.

Это закон По: в сети у иронии нет интонации, а некоторые реальные мнения настолько радикальны, что пародия уже не может их переплюнуть. Для этого и появились смайлики. Это наша цифровая приподнятая бровь.

Однажды я опубликовал сатирическое руководство о том, как охлаждать сервер льдом из рыбного магазина. Мне написали три компании с просьбой о смете. Одна уже заказала дорадо. Говорит, логи теперь пахнут морем.

ЗАКОН ГОДВИНА



По мере того как онлайн-дискуссия удлиняется, вероятность того, что кто-то проведёт сравнение с Hitler или нацистами, стремится к единице.

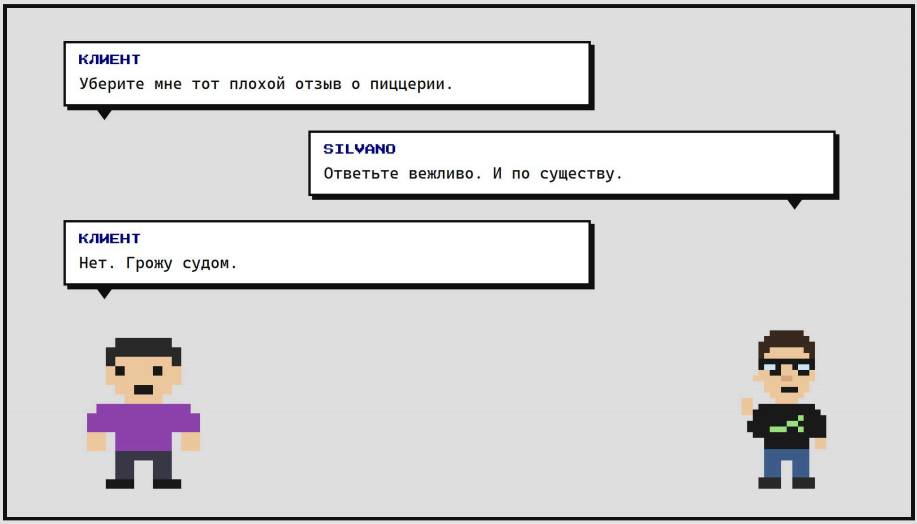
— Mike Godwin, *Usenet*, *1990*

Любая онлайн-дискуссия начинается вежливо. Кто-то спрашивает, каким текстовым редактором пользоваться. К десятому сообщению спорят о табуляции против пробелов. К тридцатому — о моральных принципах. К сотому кто-то сравнивает оппонента с худшим диктатором в истории. Из-за текстового редактора.

Закон Годвина не говорит, что так бывает, он говорит, что это неизбежно, как гравитация или как HDMI-кабель, который всегда с другой стороны. На старых форумах действовало и неписаное правило: кто первым дошёл до такого сравнения, тот проиграл спор.

Я модерирую форум любителей принтеров. Мы продержались триста двадцать сообщений. Потом один назвал драйвер некой марки тоталитарным режимом. Честно говоря, он был в чём-то прав. Но тред всё равно закрыли.

ЭФФЕКТ СТРЕЙЗАНД



Попытка скрыть, удалить или зацензурить информацию в сети в итоге делает её гораздо заметнее и популярнее.

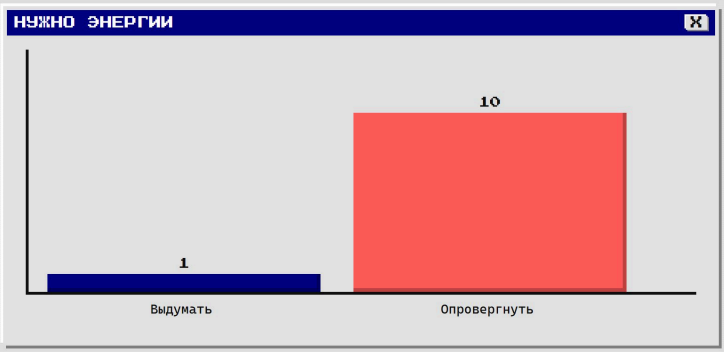
— Mike Masnick, *Techdirt, январь 2005 (по мотивам случая 2003 года с аэрофотоснимком)*

Название пошло от случая 2003 года: иск с требованием убрать из онлайн-архива аэрофотоснимок виллы, сделанный для документирования эрозии побережья. До иска снимок видели единицы. После — полмира. Один блогер построил на этом название целого явления.

Интернет устроен как кот: скажи ему не смотреть на коробку, и он будет смотреть только на неё. Каждый запрос на удаление — светящаяся вывеска «здесь что-то интересное». Копии, архивы и скриншоты размножаются быстрее любого адвоката.

Один клиент попросил меня убрать негативный отзыв о его пиццерии. Я посоветовал ответить вежливо. Он предпочёл грозить судом. Теперь этот отзыв — самый читаемый в области. Пицца тем временем стала лучше. Но об этом никто не знает.

ЗАКОН БРАНДОЛИНИ



Три секунды, чтобы выдумать. Два часа, четыре ссылки и график, чтобы опровергнуть.

Энергия, необходимая для опровержения чуши, на порядок больше той, что ушла на её производство.

— Alberto Brandolini, *сообщение в Twitter, январь 2013 (известен также как принцип асимметрии чуши)*

Написать «5G сушит герань» — три секунды. Объяснить, почему это неправда, — два часа, четыре ссылки, график и терпение монаха. Тем временем фраза про герань обошла три семейных чата.

Закон, сформулированный итальянским программистом, описывает жестокую асимметрию: выдумывать дёшево, проверять дорого. Поэтому фейки выигрывают гонки на скорость, и поэтому каждый сисадмин хотя бы один ужин провёл, объясняя, что нет, компьютер не подхватил вирус из розетки.

Я нашёл контрмеру. Когда клиент пересылает мне фейк, я отвечаю противоположным фейком, таким же коротким. Они взаимно уничтожаются. Я уже заработал репутацию осведомлённого человека. Есть и пара заявлений насчёт герани, но они на проверке.

СОВЕТ СИСАДМИНА

Когда «интернет тормозит», прежде чем ругаться с провайдером, проверьте по кабелю. Подключите компьютер напрямую к роутеру сетевым кабелем и измерьте скорость онлайн-тестом. Потом повторите тест по Wi-Fi, в комнате, где вы обычно работаете.

- Если **и по кабелю медленно**, проблема в линии или роутере: звоните провайдеру, с цифрами на руках.
- Если **по кабелю быстро, а по Wi-Fi нет**, линия невиновна. Проблема в сигнале дома: место роутера, стены, помехи.

Две полезные детали. Современные роутеры вещают в двух диапазонах: 5 ГГц быстрее, но хуже проходит сквозь стены, 2,4 ГГц бьёт дальше, но там теснее. А в больших помещениях mesh-система или точка доступа, подключённая кабелем, работают намного лучше дешёвых репитеров. Иногда передвинуть роутер на два метра — всё равно что сменить тариф.

ГЛАВА 5

ГЛ 05 — ОТЛАДКА

Охота на баги

Здесь мы входим в лабораторию. Баг, для тех, кто с ним не знаком, — это ошибка в программе. Слово старше компьютеров, но существует рабочий журнал 1947 года с настоящей молью, вклеенной в страницу: её нашли внутри машины. С тех пор охота не прекращалась, а у моли появилось очень много внуков.

Законы этой главы показывают баг в естественной среде обитания. Он исчезает, когда подходит коллега, прячется в последней строке, которую ты проверяешь, ждёт пятничного вечера, появляется через десять минут после публикации. Иногда, если он проживёт достаточно долго, пользователи к нему привязываются, и он становится функцией.

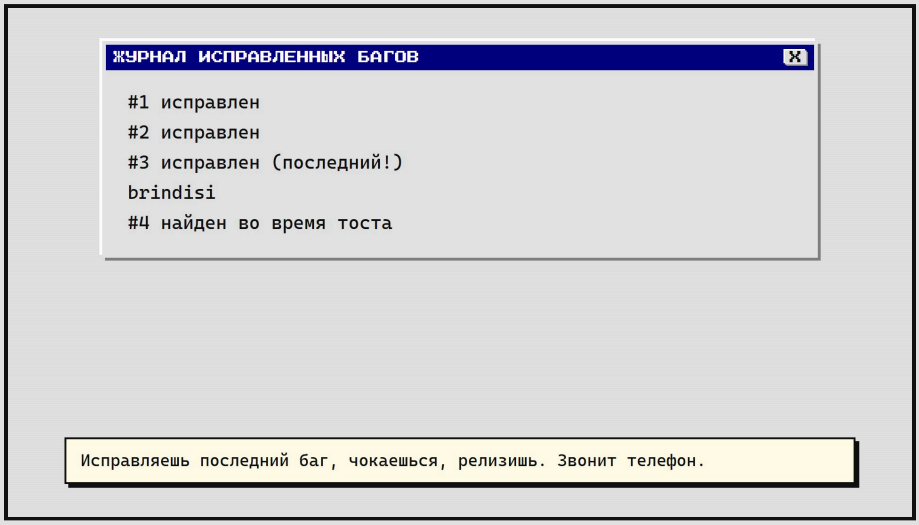
Кто не программирует, найдёт здесь ответ на вопрос, который задаёт себе годами: почему программа, которой я пользуюсь каждый день, иногда творит странное? Короткий ответ: потому что всегда есть ещё один баг. Длинный ответ: эта глава.

Кто программирует, узнает себя на каждой странице. Заранее прошу прощения за ночи, которые я вам напомню.

СОДЕРЖАНИЕ.DIR

01 Закон кибернетической энтомологии Лубарского
02 Закон «У меня на компе работает»
03 Закон коллеги-чудотворца
04 Закон невинной строки
05 Закон Кернигана
06 Закон плодовитого исправления
07 Закон продвинутого бага
08 Закон пунктуального бага
09 Закон пятничного вечера
10 Закон непротестированной части
11 Закон «Компилируется — значит, работает»
12 Закон тестовых данных
13 Закон сообщения-оракула
14 Закон коварного календаря
15 Закон Линуса
16 Закон Хайрама

ЗАКОН КИБЕРНЕТИЧЕСКОЙ ЭНТОМОЛОГИИ ЛУБАРСКОГО



Всегда есть ещё один баг.

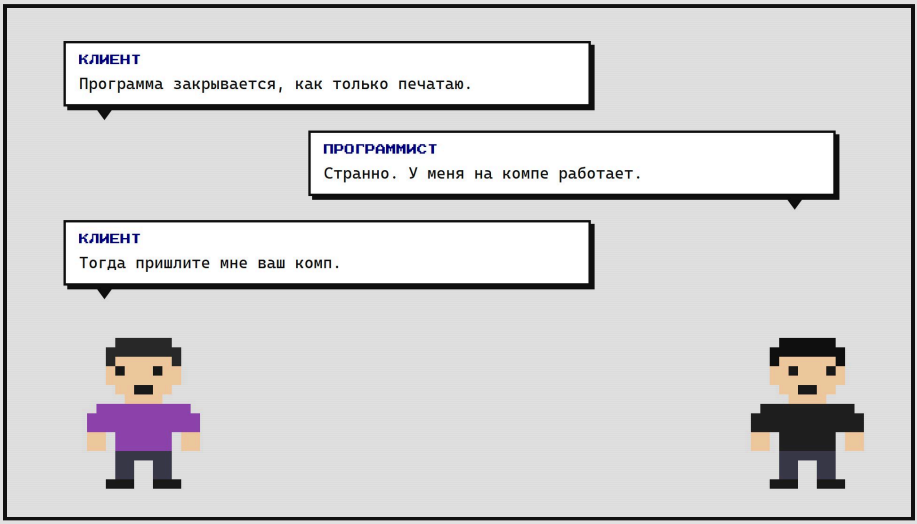
— Аноним, *поговорка из сборников компьютерного юмора (например, списков Multics и Usenet); личность Lubarsky не проверена (авторство не установлено)*

Четыре слова — и карьера любого программиста подытожена. Исправляешь последний баг, поднимаешь бокал, выпускаешь релиз. Звонит телефон. Был ещё самый-самый последний баг.

Смысл — мудрое смирение: ни одна нетривиальная программа не свободна от ошибок. Их можно сделать редкими, можно сделать безвредными, но думать, что переловил все, — самый опасный баг из всех.

Я вызвал к серверу дезинсектора. Он побрызгал чем-то на корпус, сказал «гарантия полгода» и ушёл. На следующий день учётная система печатала счета клинописью. Я перезвонил. Он сказал, что этот вид гарантией не покрывается. Кибернетическая энтомология: точная наука о слове «всегда».

ЗАКОН «У МЕНЯ НА КОМПЕ РАБОТАЕТ»



Любую ошибку, о которой сообщил клиент, невозможно воспроизвести на компьютере программиста, где всё работает идеально.

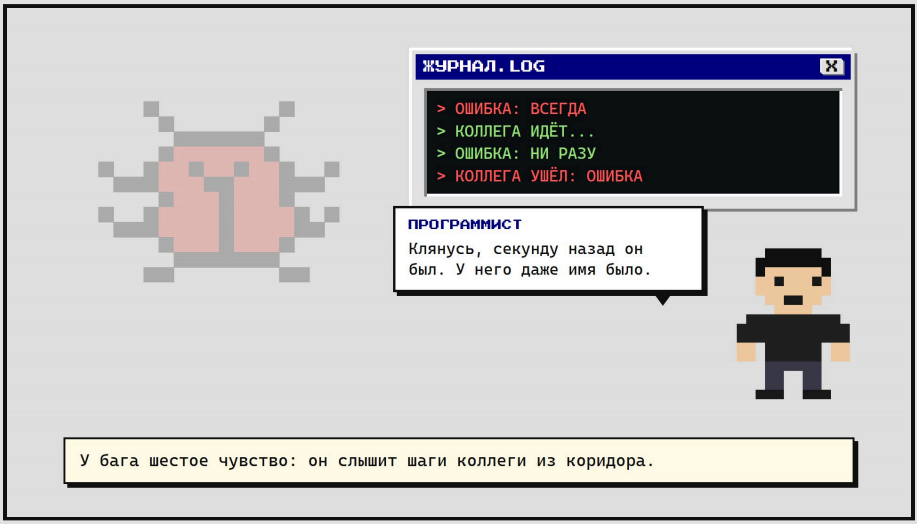
— Аноним, *компьютерный фольклор*

Это самая произносимая фраза в истории информатики, популярнее, чем «а перезагрузить пробовали?». Программист говорит её с трогательной искренностью: на его компьютере программа действительно работает. Беда в том, что клиент живёт не внутри его компьютера.

Компьютер разработчика — это заповедник, тропическая оранжерея, где все библиотеки нужные, настройки правильные, язык системы тот самый. Компьютер клиента — джунгли с Windows 2016 года, тремя антивирусами, воюющими друг с другом, и часовым поясом Гонолулу.

Был период, когда я всерьёз думал отправлять клиентам свой ПК вместе с программой. Решил бы девяносто процентов проблем. Потом понял, что клиент всё равно позвонит — сказать, что мой ПК на его столе не работает.

ЗАКОН КОЛЛЕГИ-ЧУДОТВОРЦА



Баг, который мучает тебя уже несколько часов, исчезает в ту секунду, когда ты зовёшь коллегу посмотреть, и возвращается, как только тот уходит за свой стол.

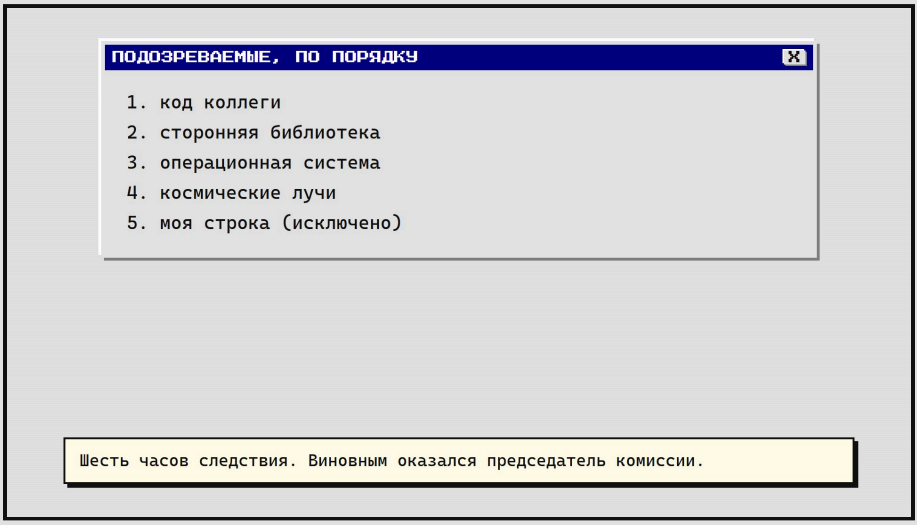
— Аноним, *компьютерный фольклор*

В каждом офисе есть коллега со сверхспособностями. Программирует он не лучше тебя, учился не больше. Просто когда он подходит к твоему монитору, баги встают по стойке смирно и делают вид, что их не существует.

«Ну, покажи», — говорит он. Ты повторяешь ровно те же шаги дрожащим голосом человека, который заявляет в полицию о привидении. Программа работает. Он хлопает тебя по плечу с видом участкового терапевта. Уходит. Баг возвращается и тебе улыбается.

У программистов даже есть название для таких застенчивых ошибок: баги, которые меняют поведение, когда на них смотрят. Я решил проблему радикально. Распечатал фото коллеги в полный рост и повесил за спинкой стула. Работает в шестидесяти процентах случаев.

ЗАКОН НЕВИННОЙ СТРОКИ



Баг всегда находится в последней строке, которую ты проверяешь, и это та самая строка, в правильности которой ты был абсолютно уверен.

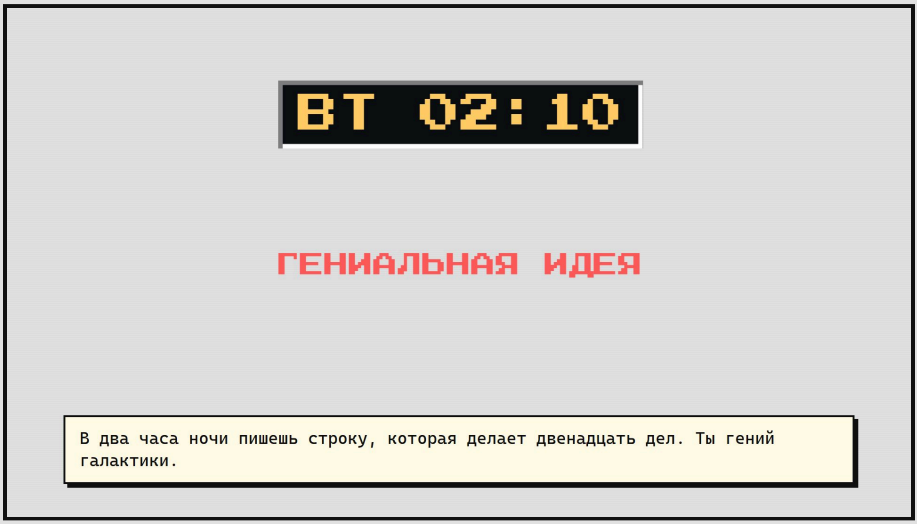
— Аноним, *компьютерный фольклор*

Охота на баг следует строгому ритуалу. Сначала подозревают код коллег, потом внешнюю библиотеку, потом операционную систему, потом компилятор, наконец космические лучи. И только в самом конце, нехотя, смотрят на свою строку. Ту самую, идеальную. Написанную с гордостью.

И вот она, ошибка. Точка с запятой не на месте, одно «равно» вместо двух. Шесть часов расследования ради одного символа. Целая парламентская комиссия, чтобы выяснить, что виновник — её председатель.

Есть и логичное объяснение, которое губит всю поэзию: ты перестаёшь искать, когда нашёл, поэтому баг всегда в последнем месте, куда ты посмотрел. Но бесполезно говорить это программисту в два часа ночи. Он знает, что код над ним издевается. И, честно говоря, он прав.

ЗАКОН КЕРНИГАНА



Отладка вдвое сложнее, чем написание кода. Поэтому, если ты пишешь код максимально хитро, ты по определению недостаточно умён, чтобы его отладить.

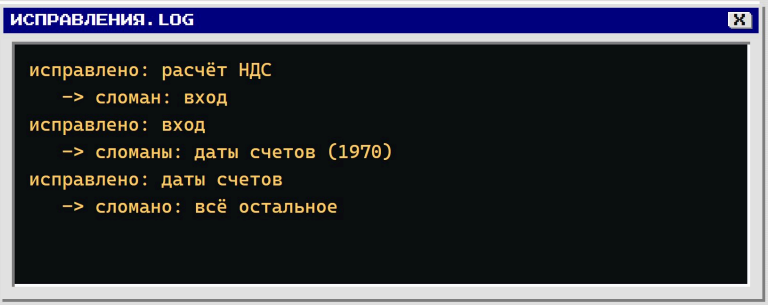
— Brian Kernighan, *The Elements of Programming Style (в соавторстве с P. J. Plauger), 2-е изд., 1978*

Это беспощадный силлогизм. В два часа ночи, на вдохновении, ты пишешь строку, которая делает двенадцать вещей сразу. Чувствуешь себя галактическим гением. Наутро строка не работает, а ночной гений не берёт трубку.

Настоящий посыл такой: пиши просто. Хитрый код — это долг перед тобой будущим, который будет уставшим, старше и с меньшим количеством кофе.

Я решил вопрос методично: пишу код на 50% своего ума, чтобы отладка удавалась на все 100%. Коллеги говорят, что это заметно. Принимаю как комплимент инженерному подходу, а не мне.

ЗАКОН ПЛОДОВИТОГО ИСПРАВЛЕНИЯ



Софт — сообщающиеся сосуды: надавишь тут, вытечет там. На пол.

Каждый наспех исправленный баг порождает минимум два новых, желательно в той части программы, которая была вообще ни при чём.

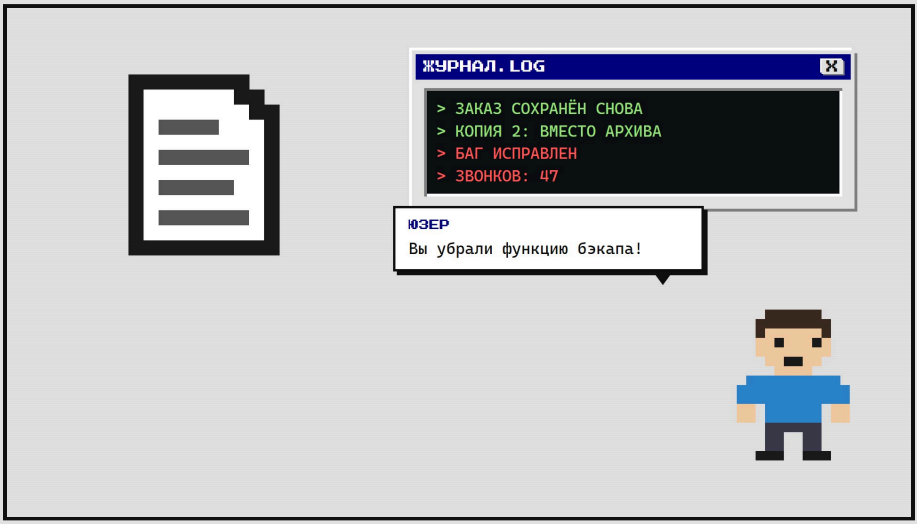
— Аноним, *компьютерный фольклор*

Баги — как головы гидры из легенд: отрубишь одну, вырастают две. С одной разницей: головы гидры хотя бы вырастали на том же месте. Новые баги предпочитают появляться на другом конце программы, где их никто не ищет.

Исправляешь расчёт НДС — перестаёт работать вход в систему. Чинишь вход — счета выходят с датой 1970 года. Софт — это система сообщающихся сосудов: надавишь с одной стороны, выльется с другой, обычно на пол.

Гуру качества в таких случаях произносит волшебное слово: «автотесты». Он прав. Но срочный баг всегда приходит, когда автотесты «почти готовы». Я подхожу к быстрому исправлению с тем же настроем, с каким меняют лампочку в старом доме: с уверенностью и с огнетушителем под рукой.

ЗАКОН ПРОДВИНУТОГО БАГА



Любой достаточно продвинутый баг неотличим от фичи.

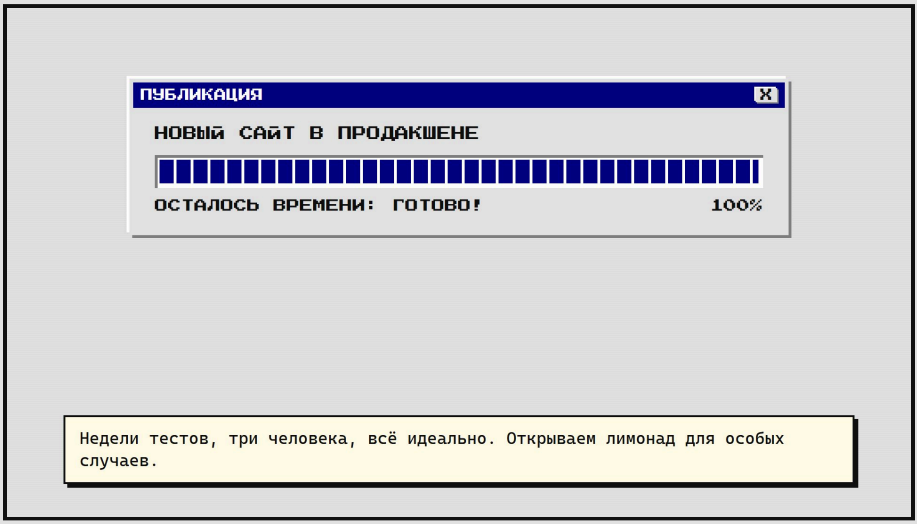
— Аноним, *компьютерный фольклор*

Иногда ошибка живёт так долго, что пускает корни. Пользователи её находят, привыкают, находят удобной. И тогда исправить её — уже акт насилия.

Реальный пример, рассказанный с должным уважением: учётная система по ошибке сохраняла черновик заказа дважды. Сотрудники научились использовать вторую копию как архив. Когда мы это исправили, нам позвонили всем отделом: «Вы убрали функцию резервного копирования».

Отсюда и знаменитая фраза, которую каждый программист хоть раз в жизни произносит, подняв подбородок и глядя за горизонт: «Это не баг, это фича». Это волшебная формула, как «крибле-крабле-бумс». Ничего не решает, но превращает позор в маркетинг. А маркетинг, как известно, не исправляют.

ЗАКОН ПУНКТУАЛЬНОГО БАГА



Критический баг обнаруживается через десять минут после публикации и никогда — за десять минут до неё.

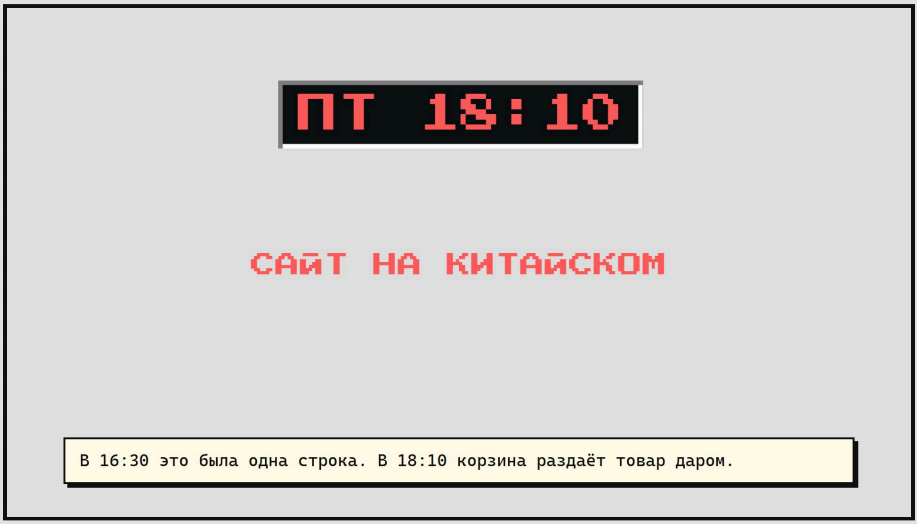
— Аноним, *компьютерный фольклор*

Неделями программу тестируют, перетестируют, проверяют три разных человека. Всё в порядке. Потом нажимают кнопку «опубликовать», открывают газировку для торжественных моментов, и ровно в эту секунду кто-то говорит: «Э-э, извините…».

Это не невезение, это почти физическое явление. Пока код на тестировании, мозг в оборонительном режиме и ничего не видит. Как только код опубликован и назад дороги нет, мозг расслабляется и внезапно становится гениальным. Видит всё. Слишком поздно, но всё.

Я пытался обмануть судьбу: делаю вид, что публикую, жду десять минут, а потом публикую по-настоящему. Сработало дважды. На третий раз я нашёл баг на десятой минуте фальшивой публикации и отпраздновал это как гол. А потом пришёл второй баг, настоящий.

ЗАКОН ПЯТНИЧНОГО ВЕЧЕРА



Изменение, выкаченное в продакшен в пятницу после обеда, сломается после шести вечера, когда все коллеги уже на трассе.

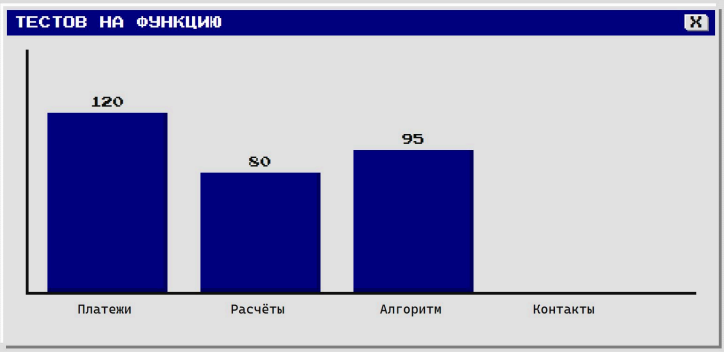
— Аноним, *компьютерный фольклор*

В календаре айтишника пятница после обеда — зона священная и проклятая, как пограничные земли в фэнтезийных сагах. Кто выкатывает обновление в эти часы, бросает вызов древним богам продакшена, а древние боги не прощают.

Сценарий всегда один. Полпятого: «Да там крошечная правка, одна строчка». Пять: всё зелёное, всё идеально. Шесть десять: сайт показывает главную страницу на китайском, а корзина раздаёт товар бесплатно.

Тем временем на трассе коллега, который знает систему, держит телефон в авиарежиме по духовным соображениям. У нас в агентстве действует неписаный указ: в пятницу делаются только обратимые вещи. Протереть клавиатуру, почитать документацию, подумать. Особенно подумать — мысли в продакшен не уходят.

ЗАКОН НЕПРОТЕСТИРОВАННОЙ ЧАСТИ



Крепость на платежах. Чёрный ход завязан бечёвкой.

В продакшене всегда ломается та часть программы, которую никто не подумал проверить, потому что в ней слишком просто ошибиться.

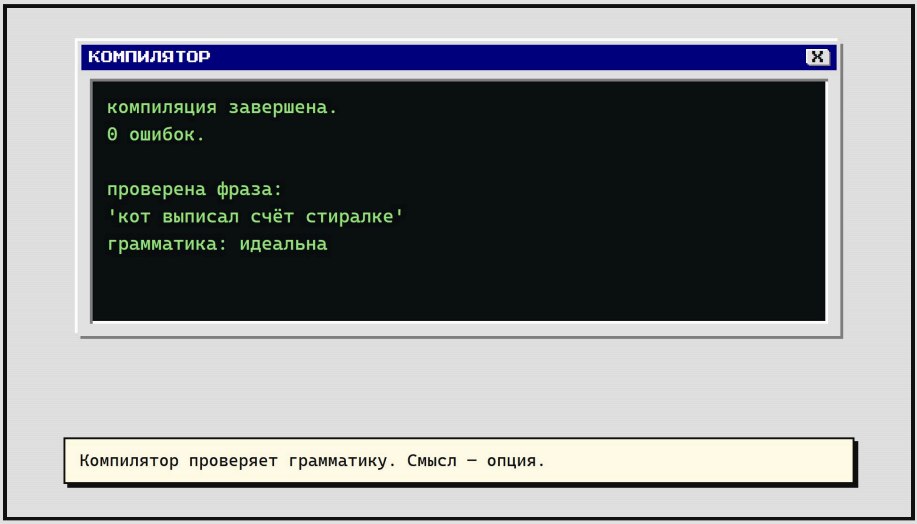
— Аноним, *компьютерный фольклор*

Тесты всегда сосредоточены на трудных местах: сложные расчёты, алгоритмы, платежи. Вокруг них строят крепости. А потом сайт падает из-за кнопки «Связаться с нами», на которую никто ни разу не нажимал, потому что «это же просто форма».

Это компьютерная версия средневекового замка со стенами толщиной в три метра и задней калиткой, завязанной бечёвкой. Враг не штурмует стены. Враг, совершенно спокойно, стучится с чёрного хода.

Гуру качества годами повторяет одну и ту же проповедь: «Всё, что не протестировано, сломано, просто ты об этом ещё не знаешь». Фраза страшная, потому что правдивая. Теперь я проверяю и банальные вещи, одну за другой, с усердием нотариуса. Клиент спрашивает, почему так долго. Отвечаю, что проверяю чёрный ход.

ЗАКОН «КОМПИЛИРУЕТСЯ — ЗНАЧИТ, РАБОТАЕТ»



Если код компилируется без ошибок, уставший программист сочтёт его готовым к сдаче.

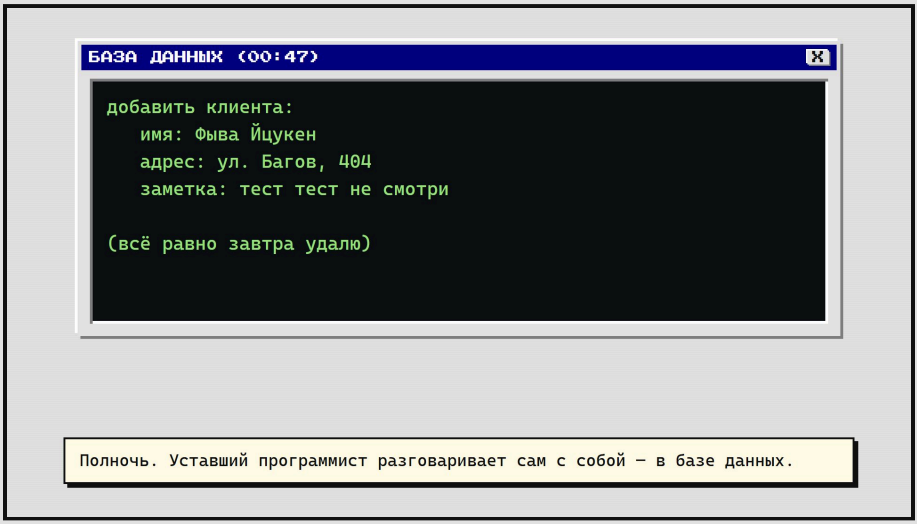
— Аноним, *компьютерный фольклор*

Компилировать — значит переводить код в исполняемую программу. Компилятор проверяет, что грамматика правильная, а не что в программе есть смысл. Он как корректор, который с восторгом одобряет фразу «кот выставил счёт стиральной машине».

И всё же в семь вечера, перед надписью «компиляция завершена», в человеческом мозгу что-то щёлкает. Внутренний голос говорит: готово. Код отправляется к клиенту, как корабль, вышедший из порта с парусами, поставленными задом наперёд.

Во многих офисах ходит сокращённая версия девиза: «Компилируется? Отгружаем». Произносят её со смехом. А потом применяют на практике. Я сделал так один-единственный раз, в далёком прошлом. Стиральная машина, в переносном смысле, крутится до сих пор. А я научился хотя бы раз запускать программу перед отправкой.

ЗАКОН ТЕСТОВЫХ ДАННЫХ



Самые неловкие тестовые данные, внесённые во время разработки, рано или поздно всплывут перед самым важным клиентом.

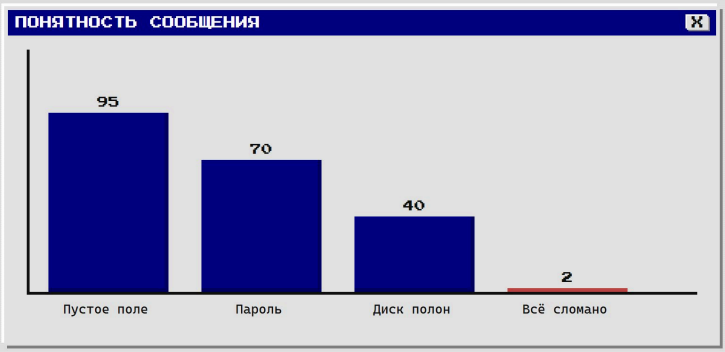
— Аноним, *компьютерный фольклор*

Во время разработки никто не вводит реалистичные данные. Пишут «asdf», «тест тест», «Вася Пупкин», «товар, которого нет». Это заглушки уставшего программиста, его способ разговаривать с самим собой в полночь.

А потом наступает официальная презентация. Переговорная, проектор, генеральный директор в первом ряду. Открывается список клиентов, и на самом верху, жирным шрифтом, — господин Asdf Qwerty, адрес «ул. Багов, 404». Тишина. Кто-то кашляет.

Правило, которое я усвоил на собственной шкуре, простое: пиши любые тестовые данные так, будто их прочитает клиент. Потому что он их прочитает. Сегодня в моих тестовых средах только скромные и солидные имена вроде «ООО „Пример“». Чуть грустнее, зато гораздо меньше пота.

ЗАКОН СООБЩЕНИЯ-ОРАКУЛА



Чем серьёзнее ошибка, тем меньше слов. Как оракул на нулевом балансе.

Чем серьёзнее ошибка, тем расплывчатее, общее и непонятнее сообщение о ней.

— Аноним, *компьютерный фольклор*

Мелкие ошибки говорят прямо: «Поле Email заполнено неверно». А катастрофические изъясняются как древние оракулы: «Произошла непредвиденная ошибка». Код 0x8000FFFF. Конец связи.

Непредвиденная — кем, простите? Эту фразу написал программист, который за несколько лет до того прекрасно предвидел, что что-то пойдёт не так, но ему было лень сказать, что именно. Это записка на холодильнике: «Тут кое-что случилось. Пока».

Клиенты зачитывают мне их по телефону голосом человека, оглашающего пророчество. Я молча слушаю, киваю, а потом произношу священную формулу, которую знает каждый сисадмин: «А перезагрузить пробовали?». Помогает не всегда. Но это единственный ответ, достойный такого вопроса.

ЗАКОН КОВАРНОГО КАЛЕНДАРЯ

ДАТЫ, НЕНАВИСТНЫЕ ПРОГРАММЕ

- 29 февраля
- ночь перевода часов
- 1 января
- 30 февраля (сама придумала)

Триста шестьдесят два дня — идеально. В остальные три — готический роман.

Любая программа, работающая с датами, будет работать весь год, кроме 29 февраля, ночи перевода часов и первого января.

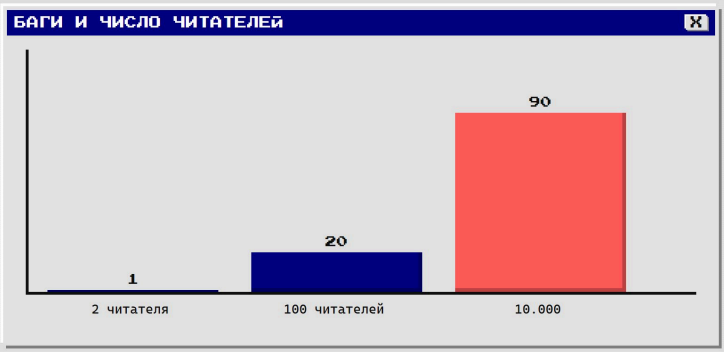
— Аноним, *компьютерный фольклор*

Время кажется простой штукой. Шестьдесят секунд, шестьдесят минут, двадцать четыре часа. А потом программист пытается обработать его в коде и обнаруживает ужас: високосные годы, часовые пояса, летнее время, месяцы по двадцать восемь дней, страны, которые передумывают.

Результат — софт, который прекрасно работает триста шестьдесят два дня в году. В остальные три он вытворяет сюжеты из готического романа: бронирования в прошлом, зарплатные ведомости с призрачным часом, сроки на 30 февраля.

Однажды ночью в конце октября моя учётная система дважды зарегистрировала одну и ту же отметку прихода, в полтретьего, потому что этот час действительно случился дважды. Сотрудник числился одновременно в двух местах. Директор хотел его повысить. Я исправил баг с сожалением человека, разрушившего чудо.

ЗАКОН ЛИНУСА



Открытый код и много глаз: рано или поздно дефект всплывёт.

При достаточном количестве глаз все баги всплывают на поверхность.

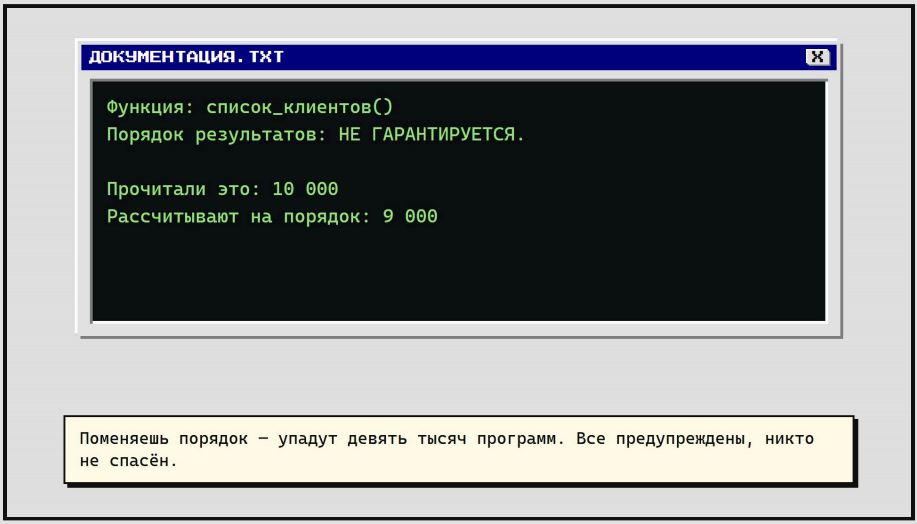
— Eric S. Raymond, *The Cathedral and the Bazaar*, 1997 (эссе), 1999 (книга); назван в честь Linus Torvalds

Идея прекрасная: если код открыт и его читают тысячи людей, кто-нибудь найдёт дефект. Именно поэтому свободный софт может конкурировать с крупными компаниями и часто побеждать.

Неудобная деталь — слово «достаточном». Глаза должны ещё и действительно смотреть. Существуют библиотеки, которыми пользуется полпланеты, а читают их на деле два уставших волонтёра и кот.

Я попробовал применить закон в офисе: вывел код проектором на стену в кофе-брейк. Двадцать глаз. Ни одного бага не нашли, зато трое коллег сообщили, что шрифт некрасивый, а один спросил, новый ли проектор. Баг всё ещё там, прекрасно освещённый.

ЗАКОН ХАЙРАМА



При достаточном количестве пользователей любое наблюдаемое поведение твоей системы кто-нибудь да будет использовать, независимо от того, что ты обещал в документации.

— Hyrum Wright, *наблюдение, родившееся в инженерной среде Google, 2010-е годы (hyrumslaw.com)*

Пишешь в документации: «Порядок результатов не гарантируется». Десять тысяч человек читают. Девять тысяч строят свои программы в расчёте на то, что порядок всегда тот же. В день, когда ты его меняешь, падает пол-интернета.

Закон говорит, что настоящий контракт программы — не тот, что написан, а всё, что она делает, включая дефекты. Исправив баг, можно сломать того, кто построил на этом баге целый бизнес.

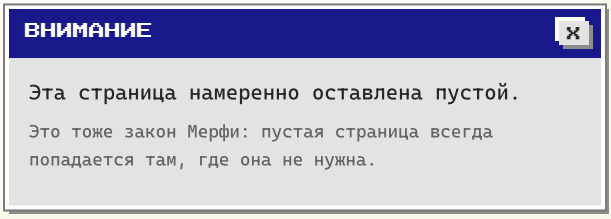
Однажды я исправил опечатку в сообщении об ошибке. На следующий день позвонил разъярённый клиент: его скрипт искал именно это неправильное слово, чтобы понять, что что-то пошло не так. Я вернул опечатку. Теперь это историческое наследие.

СОВЕТ СИСАДМИНА

Охота на баги — не гадание, а метод. Мой, проверенный многими ночами, такой.

- **Сначала воспроизвести.** Запишите точные шаги, после которых появляется ошибка. Если вы не умеете вызвать баг снова, то не сможете и сказать, исправили ли его.
- **Читать логи.** Часто программа уже где-то записала, что пошло не так. Кажется очевидным, но именно этот шаг пропускают чаще всего.
- **Делить пополам.** Если вчера работало, а сегодня нет, ищите посередине, что изменилось. В Git для этого есть даже отдельная команда, `git bisect`: она находит виновное изменение, каждый раз сокращая область поиска вдвое.
- **Объяснить вслух.** Коллеге или предмету на столе. Когда рассказываешь проблему строка за строкой, неправильная строка часто выдаёт себя сама.

А когда найдёте, напишите тест, который поймает его, если он вернётся. Баги всегда возвращаются домой, как кошки.


ВНИМАНИЕ
Эта страница намеренно оставлена пустой.
Это тоже закон Мерфи: пустая страница всегда попадается там, где она не нужна.

ГЛАВА 6

ГЛ 06 — КОД

Искусство писать код (и раскаиваться)

ЗАКОНОВ: 17 _

Если предыдущая глава была криминальной хроникой, то эта — философский трактат. Следующие законы говорят не об отдельных ошибках, а о том, как пишется софт: как называть вещи, сколько комментировать, когда оптимизировать, почему хитрый код сегодня — головная боль завтра.

Многие из них подписаны великими именами программирования — людьми, которые создали языки, системы и книги, по которым учились поколения. Утешительно узнать, что и они в какой-то момент оказывались перед программой, написанной ими же несколько месяцев назад, и ничего в ней не понимали. Гениальность — не прививка.

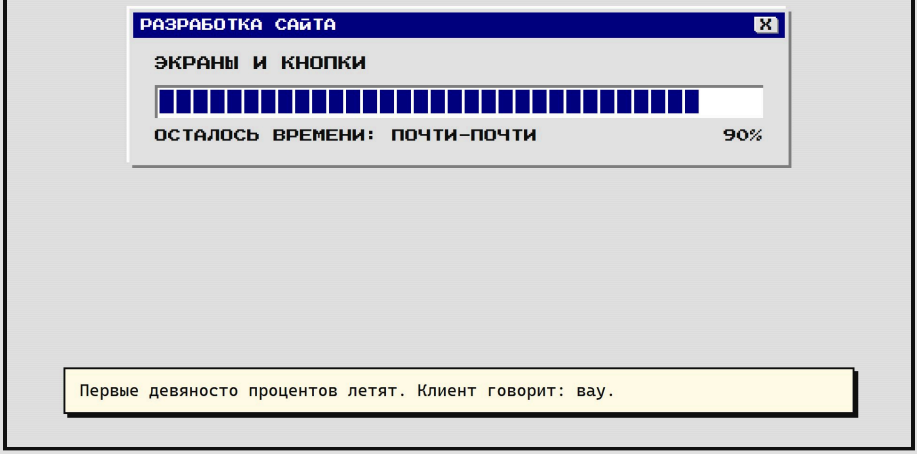
Общая нить одна, и скажу сразу, чтобы спешащие могли перейти к концу: простота трудна, но только она выдерживает время. Остальное — комментарии. Скорее всего, лживые.

Тем, кто не программирует: не бойтесь. Каждый закон объяснён на житейских примерах, и вы поймёте, почему вашей учётной системе нужно два года, чтобы передвинуть кнопку. Это не лень. Это археология.

СОДЕРЖАНИЕ.DIR

01 Правило девяноста на девяносто
02 Закон двух трудных вещей
03 Закон двух проблем
04 Закон Иглсона
05 Закон лживого комментария
06 Закон посмертной документации
07 Закон приёмного кода
08 Закон Осборна
09 Аксиома Флона
10 Принцип Хоара
11 Преждевременная оптимизация
12 Правила Пайка
13 Закон Деметры
14 Закон дырявых абстракций
15 Десятое правило Гринспена
16 Закон Уодлера
17 Законы Лемана

ПРАВИЛО ДЕВЯНОСТА НА ДЕВЯНОСТО



Первые 90% кода требуют 90% времени разработки. Оставшиеся 10% требуют ещё 90%.

— Tom Cargill, *Bell Labs; стало известным благодаря Jon Bentley и его колонке Programming Pearls, Communications of the ACM, 1985*

Математика говорит: 180%. Программист говорит: именно так. Проект всегда «почти готов» в течение периода, который длится столько же, сколько весь проект.

Первые 90% — приятная часть: экраны, кнопки, клиент говорит «вау». Последние 10% — это когда выясняется, что в фамилии может быть апостроф, что кто-то родился 29 февраля и что НДС меняется ровно в неделю релиза.

У нас в офисе есть сайт, готовый на 98% с 2017 года. Каждый год мы отмечаем день рождения «осталось чуть-чуть». У нас есть торт. Свечи на 98%: зажжены наполовину, но с большим энтузиазмом.

ЗАКОН ДВУХ ТРУДНЫХ ВЕЩЕЙ

PRICES.PHP

```
цена = кэш.прочитать('кофе')
> 1,00 евро
(в новом прайсе — 1,20)
> очистить кэш
> 1,00 евро
(кэш к ней привязался)
```

Кэш помнит всё. Особенно то, что должен забыть.

В информатике есть только две по-настоящему трудные задачи: инвалидация кэша и придумывание названий.

— Phil Karlton, *максима, приписываемая Карлтону времён Netscape, в пересказе Tim Bray и Martin Fowler («TwoHardThings»)*

Тем, кто не программирует, кажется, что самое трудное — математика, безопасность, искусственный интеллект. Ошибка. Две настоящие вершины вот какие: заставить компьютер забыть старое и решить, как назвать новое.

Кэш — это быстрая память, которая откладывает данные в сторонку, чтобы не ходить за ними каждый раз. Работает прекрасно, пока с гордостью не показывает тебе вчерашнюю цену. А название — это философская пытка: «data», «data2», «dataNew», «dataNewFinal». Я видел целые команды, которые полдня заседали над именем переменной, как средневековый церковный собор.

Есть и фольклорный вариант, где добавляется третья проблема: ошибки на единицу при подсчёте. Трудных вещей две, но правильно их сосчитать ещё никому не удавалось.

ЗАКОН ДВУХ ПРОБЛЕМ

ПРОВЕРКА_EMAIL

```
правило = /^[a-z.]+@+[a-z]*\.it$/

тест('vasya@@.it')         -> ВЕРНО
тест('office@buhgalter.com')
                           -> ОТКЛОНЁН
```

Формула для проверки email. Пропускает vasya, отвергает бухгалтера.

Сталкиваясь с проблемой, некоторые думают: «Решу-ка я её регулярными выражениями». Теперь у них две проблемы.

— Jamie Zawinski, *сообщение в Usenet 1997 года, восстановленное Jeffrey Friedl в блоге regex.info (2006)*

Регулярные выражения — это формулы для поиска текста, мощнейшие и написанные на языке, который выглядит так, будто по клавиатуре прошёлся кот. С их помощью, например, проверяют, правильный ли адрес электронной почты. Получается строка символов, которую через десять минут не может прочитать ни один человек, включая автора.

Первая проблема была — найти адреса. Вторая — понять, почему формула принимает «vasya@@.ru» и отвергает бухгалтера. Третья, а она всегда добавляется, — единственный коллега, который это понимал, сменил работу.

Я пользуюсь ими с той же осторожностью, с какой обращаются с заклинанием, найденным на чердаке. Работают, но каждый раз, когда я их правлю, где-то вдалеке слышится гром. И обычно, надо признать, падает сервер.

ЗАКОН ИГЛСОНА

OLD_PROJECT.PHP

```
function f($vasya2) {
  $x = $vasya2 * 3.14 + 7;   // ?
  // поправить потом
  return f2($x, true, true, false);
}
```

Ни одного комментария, загадочные переменные. Кто это писал?

Код, который ты написал сам и не открывал полгода или дольше, вполне мог быть написан кем-то другим.

— Аноним, *поговорка, распространённая среди программистов; происхождение и личность Eagleson не проверены (авторство не установлено)*

Открываешь старый файл. Кто это написал? Переменные с именами «foo2», ни одного комментария, функция на триста строк. Проверяешь автора. Это ты. Ты полугодовой давности, совершенно незнакомый человек с твоим логином.

Закон учит, что первый адресат документации — не коллега, а ты в будущем, который всё забудет. Писать понятный код — это акт доброты по отношению к незнакомцу, очень похожему на тебя.

Я теперь отношусь к себе прошлому как к внешнему подрядчику. Пишу ему письма с претензиями. Он никогда не отвечает. Вчера нашёл его комментарий: «Поправить потом». Потом — это когда? Мы уже в «потом». «Потом» — это я.

ЗАКОН ЛЖИВОГО КОММЕНТАРИЯ

DISCOUNT.PHP

```
// считаем скидку 10%
функция скидка(заказ):
   письмо(директору)
   скидка = 30%
   // не трогать
```

Комментарий хранит древний замысел. Код — современное ЧП.

Когда комментарий и код не согласны друг с другом, скорее всего, неправы оба.

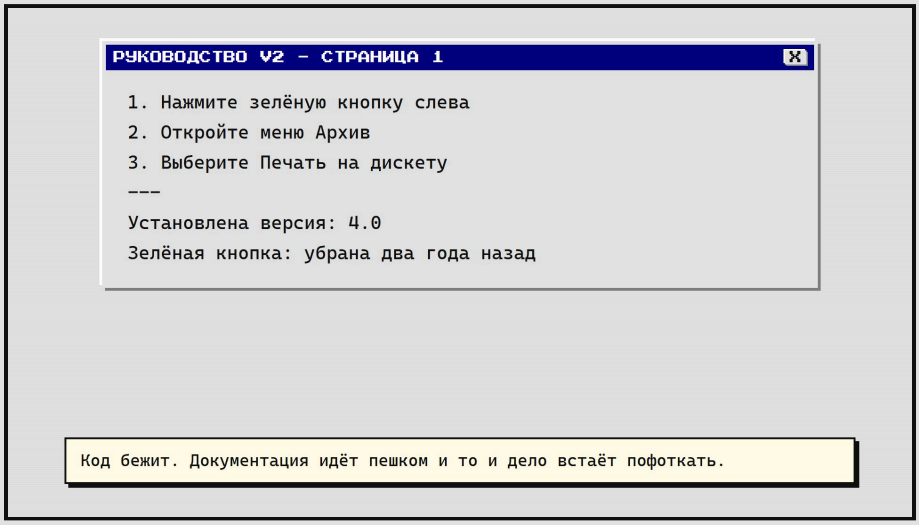
— Аноним, *компьютерный фольклор*

Комментарии — это фразы, которые программисты оставляют в коде, чтобы объяснить, что он делает. В теории это жест вежливости по отношению к тем, кто придёт следом. На практике это дорожные указатели на дороге, которую тем временем перенесли.

Код меняют, комментарий остаётся. И вот над функцией, которая отправляет письмо директору, написано «считает скидку десять процентов». Кто прав? Никто. Комментарий рассказывает о древнем намерении, код — о современной аварии.

Есть и противоположная школа: гордые программисты, которые не комментируют никогда. Раз было трудно написать, должно быть трудно и понять. Я посередине. Комментирую всегда, но фразами вроде «не трогать». Ничего не объясняет, зато спасает много жизней.

ЗАКОН ПОСМЕРТНОЙ ДОКУМЕНТАЦИИ



Документация, если она вообще есть, с большой точностью описывает предыдущую версию программы.

— Аноним, *компьютерный фольклор*

Техническая документация — отдельный литературный жанр, что-то среднее между историческим романом и завещанием. Она в мельчайших подробностях рассказывает о мире, которого больше нет: кнопки переехали, меню переименовали, функции тем временем удалили на совещании длиной в четыре минуты.

Причина простая. Код бежит, документация идёт пешком. Когда кто-то наконец дописывает руководство ко второй версии, программисты уже ругаются из-за четвёртой.

У меня есть клиент, который хранит руководство к своей учётной системе в витрине, как реликвию. Иногда он открывает его и торжественно зачитывает инструкции вслух. Ни одна уже не совпадает с экраном. Но он говорит, что это помогает ему помнить, каким был мир, когда кнопки были слева. И я, честно говоря, его понимаю.

ЗАКОН ПРИЁМНОГО КОДА

РЕШЕНИЕ_С_ФОРУМА

```
// найдено на форуме, год неизвестен
// не знаю зачем, но нужно
x = x + 0
ждать(0 секунд)
повторить 1 раз:
   работай()
```

Строки, подаренные незнакомцем. Принимаются с благодарностью и без вопросов.

Код, скопированный из интернета, работает идеально, пока ты не пытаешься понять, почему он работает.

— Аноним, *компьютерный фольклор*

У каждого программиста есть тайный архив строк, позаимствованных с форумов: решения, написанные незнакомцами, в другую эпоху, для другой задачи. Вставляешь — работает, и смотришь на них с благодарностью человека, получившего почку от анонимного донора.

Беда приходит, когда ты из добросовестности пытаешься в них разобраться. Читаешь, рассуждаешь, убираешь строку, которая кажется лишней. Всё рушится. Возвращаешь строку. Теперь не работает и с ней. Приёмный код обиделся и решил тебя не прощать.

С тех пор я отношусь к таким строкам, как древние относились к амулетам: уважение, дистанция, никаких вопросов. В моём коде до сих пор есть куски с комментарием «не знаю почему, но надо». Это самые стабильные строки во всём проекте. Знание, бывает, переоценивают.

ЗАКОН ОСБОРНА

CONFIG.PHP

```
const ТЕКУЩИЙ_ГОД = 2019;
const СТАВКА_НДС = 18;     // потом 20, потом 22
var температура = 0;      // ни разу не менялась

echo ТЕКУЩИЙ_ГОД;
> 2019
```

Переменная не менялась с первого дня. Константа передумывала много раз.

Переменные не меняются; константы не остаются постоянными.

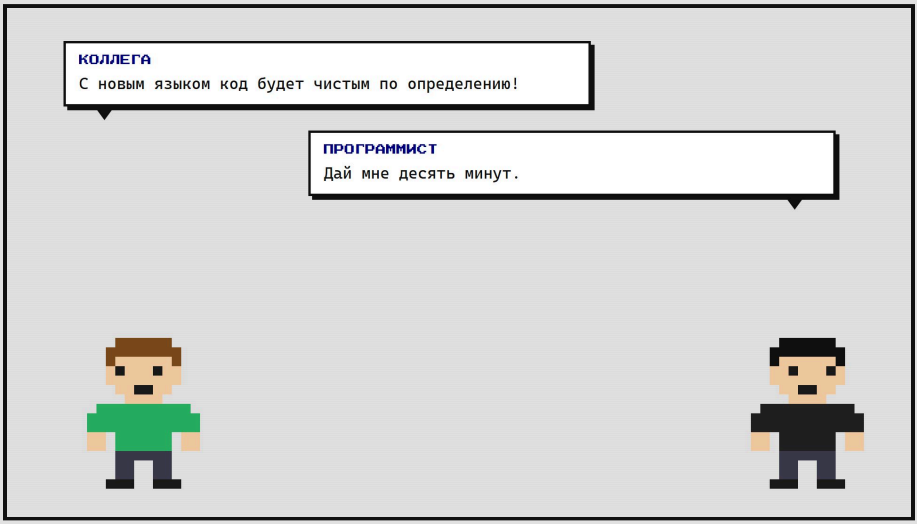
— Аноним, *поговорка из компьютерного юмора; происхождение и личность Osborn не проверены (авторство не установлено)*

В теории переменная меняется, а константа стоит на месте. На практике переменная с момента создания прибита к неправильному значению, а константа «ставка НДС» за десять лет менялась четыре раза.

За шуткой стоит урок: то, что ты считаешь неизменным, — налоги, телефонные коды, форматы дат — как раз и изменится. А то, что должно часто обновляться, застревает из-за ошибки инициализации.

В моём коде есть константа TEKUSHCHIY_GOD. Она равна 2019. Никто не решается её трогать. Мы постановили, что для софта всё ещё 2019 год. Хорошо живём. Цены ниже.

АКСИОМА ФЛОНА



Не существует, никогда не существовало и никогда не будет существовать языка программирования, на котором трудно писать плохие программы.

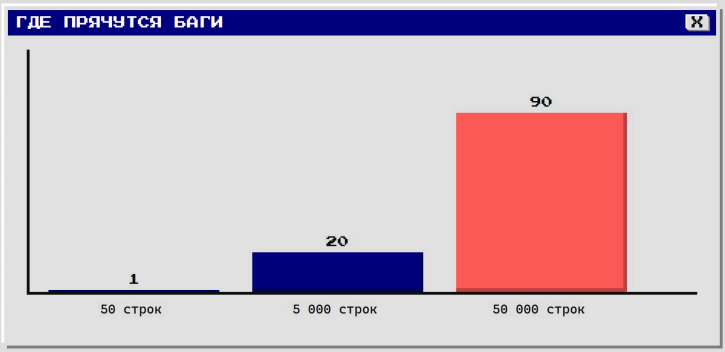
— Lawrence Flon, *On Research in Structured Programming*, *SIGPLAN Notices*, октябрь 1975

Каждый новый язык приходит с мессианским обещанием: «Со мной код наконец станет чистым!». Через три месяца кто-то уже написал функцию на девятьсот строк под названием «obrabotat_vsyo».

Флон не говорил, что все языки одинаковы. Он говорил, что качество зависит от того, кто пишет, а не от инструмента. Хороший язык помогает, но не мешает тебе построить лабиринт.

Я был на курсе у гуру современных языков. Он показал нам код идеальный, элегантный, безопасный. Потом оставил нас одних на десять минут. Когда он вернулся, мы заново изобрели глобальные переменные, копипасту и папку «final_tochno». Язык был невиновен. Мы — нет.

ПРИНЦИП ХОАРА



В пятидесяти строках баг — муха на скатерти. В пятидесяти тысячах у него дом и огород.

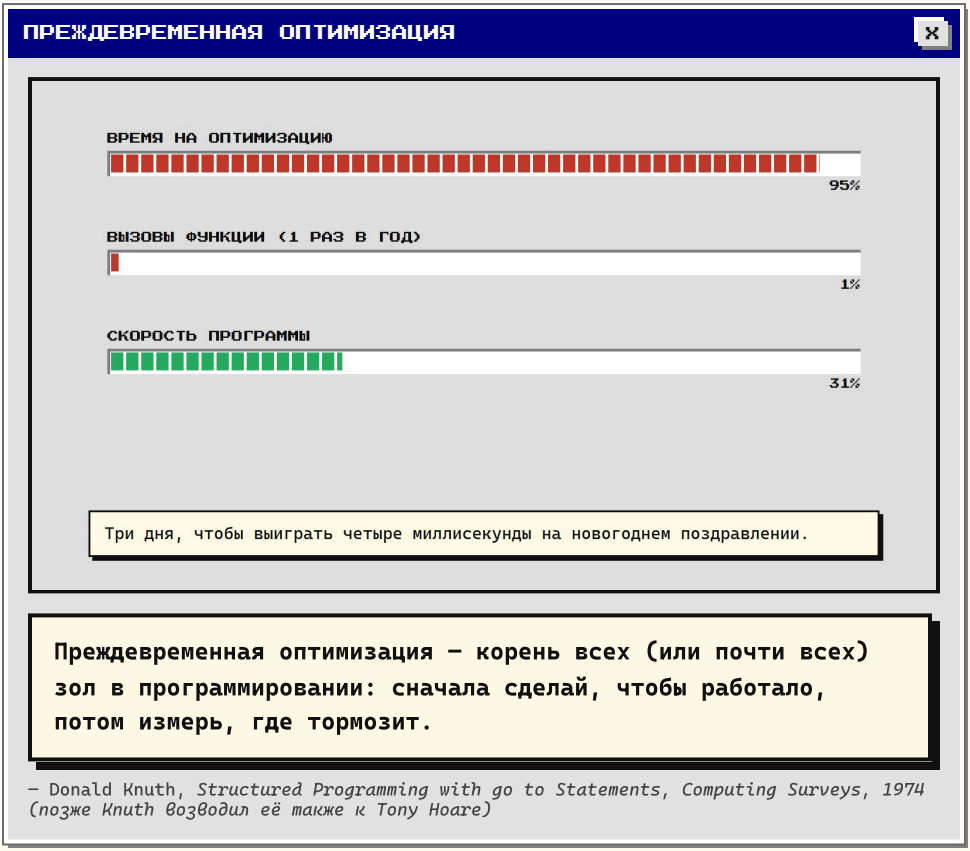
Есть два способа проектировать софт: сделать его настолько простым, что в нём очевидно нет недостатков, или настолько сложным, что в нём нет очевидных недостатков. Первый гораздо труднее.

— C. A. R. Hoare, *The Emperor's Old Clothes, речь при вручении премии Тьюринга 1980 года, опубликована в Communications of the ACM, 1981*

Сложность — отличное укрытие. В программе на пятьдесят строк баг виден, как муха на белой скатерти. В программе на пятьдесят тысяч у него дом, семья и огород.

Хоар напоминает, что простота — не лень: это самое трудное, чего можно добиться. Надо убирать, переосмыслять, выбрасывать код, к которому успел привязаться. Добавлять легко; отказываться требует смелости.

Я спросил коллегу, есть ли в его модуле дефекты. Он ответил: «Ни одного заметного». Это была правда. Ничего не было заметно. Даже логики. Это был шедевр ломбардского тумана в форме кода, а под ним, в тумане, баги играли в городки.



Преждевременная оптимизация — корень всех (или почти всех) зол в программировании: сначала сделай, чтобы работало, потом измерь, где тормозит.

— Donald Knuth, *Structured Programming with go to Statements*, *Computing Surveys*, 1974 *(позже Knuth возводил её также к Tony Hoare)*

Есть такой тип программиста, который три дня ускоряет функцию, вызываемую раз в год, первого января, чтобы напечатать поздравление. Выигрывает четыре миллисекунды. Празднует как высадку на Луну.

Смысл фразы — простой здравый смысл: ты не знаешь, где тормозит, пока не измеришь. Оптимизация вслепую делает код нечитаемым и оставляет нетронутым настоящее узкое место, которое обычно оказывается запросом к базе внутри цикла.

Мой дядя оптимизировал свой погреб: полки по алфавиту, указатели, кэш любимых бутылок. Потом обнаружил, что дверь заперта, а ключ лежит в погребе. Время доступа было бесконечным, но отлично организованным.

ПРАВИЛА ПАЙКА

PROFILE.TXT

```
Гипотеза: тормозит расчёт
Реальный замер:
  расчёт          2%
  чтение файла   97%   (один файл x 1000)
  прочее          1%
```

Сначала измерь, потом оптимизируй. Чутьё на производительность — гороскоп.

Нельзя заранее знать, где программа будет терять время: сначала измерь. Изощрённые алгоритмы медленны, когда данных мало, и правильную структуру определяют скорее данные, чем код.

— Rob Pike, *Notes on Programming in C, 1989*

Правил пять, и звучат они как заповеди монаха-терминальщика: измеряй, не угадывай; не используй сложные алгоритмы, если n маленькое, а n обычно маленькое; простота побеждает элегантность; данные командуют.

Смысл в том, что интуиция программиста насчёт производительности надёжна, как гороскоп. Думаешь, что знаешь, где тормозит, два дня оптимизируешь не то, а настоящей проблемой был цикл, который тысячу раз читал один и тот же файл.

Я реализовал самобалансирующееся самооптимизирующееся дерево для телефонного справочника офиса. В нём семь номеров. Старая бумажка на холодильнике находит их быстрее. К тому же у бумажки элегантное магнитное крепление.

ЗАКОН ДЕМЕТРЫ

ORDER.JS

```
имя = клиент.заказ.адрес
         .город.район.область.имя
// шесть звеньев: одно поменяется — рухнет всё

лучше: имя = клиент.район()
```

Цепочка сплетен: каждый кусок знает о других слишком много.

Объект должен общаться только со своими непосредственными соседями, а не с незнакомцами: никаких цепочек вызовов через полпрограммы.

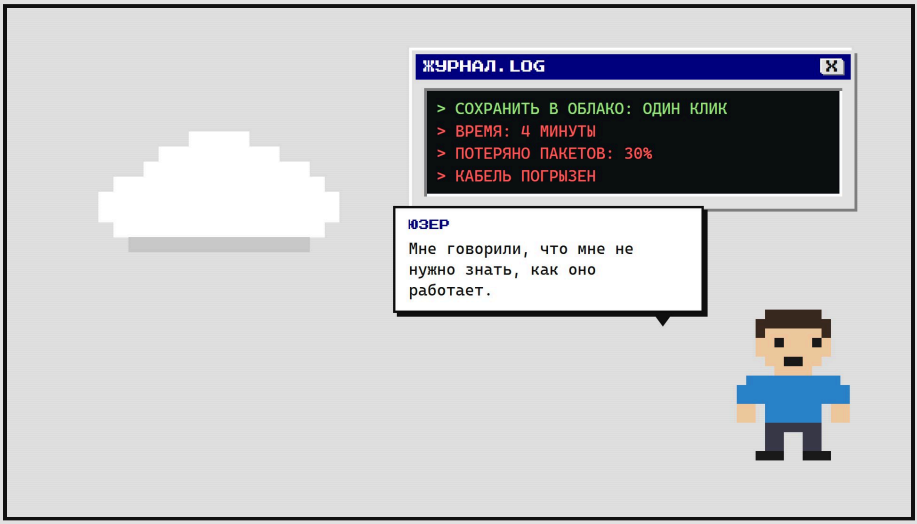
— Ian Holland, *проект Demeter, Northeastern University, 1987 (группа Karl Lieberherr)*

Он известен также как принцип минимального знания. На практике: если хочешь заплатить киоскёру, даёшь ему деньги. А не передаёшь кошелёк, чтобы он в нём порылся, нашёл карту, позвонил в банк и спросил баланс у твоей мамы.

В коде строка вроде klient.zakaz.adres.gorod.oblast.nazvanie — это сплетня по цепочке: каждое звено слишком много знает о других, и если одно меняется, рушатся все. Чем меньше знакомств, тем легче заменять детали.

Я применил закон в офисе. Больше не прошу коллегу спросить техника, чтобы тот спросил поставщика. Спрашиваю коллегу, и всё. Он ничего не знает, зато система гораздо слабее связана. Теперь ничего не работает, но элегантно.

ЗАКОН ДЫРЯВЫХ АБСТРАКЦИЙ



Все нетривиальные абстракции рано или поздно пропускают наружу те детали, которые должны были скрывать.

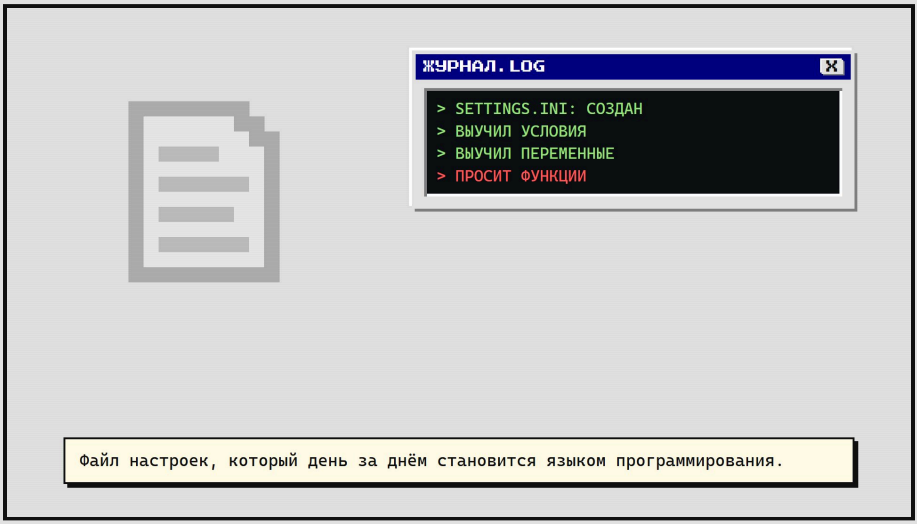
— Joel Spolsky, *The Law of Leaky Abstractions*, *Joel on Software*, 2002

Абстракция — это красивый фасад: «Не волнуйся, как работает сеть, просто сохрани файл». Работает прекрасно, до того дня, когда файл сохраняется четыре минуты, и тебе приходится выяснять, что такое задержка, потерянный пакет и кабель, перегрызенный грызуном.

Суть в том, что удобные инструменты экономят время в обычной работе, но не освобождают от необходимости понимать, что под ними. Когда они протекают, а протекают они всегда, нужен тот, кто умеет спуститься в подвал.

Облако — высшая абстракция: «Ваши данные в облаке». А потом у облака авария в дата-центре с названием, адресом и сломанным кондиционером. У облака есть почтовый индекс. Я знаю. Я там был.

ДЕСЯТОЕ ПРАВИЛО ГРИНСПЕНА



Любая достаточно сложная программа на C или Fortran содержит наспех сделанную, медленную и полную багов реализацию половины языка Lisp.

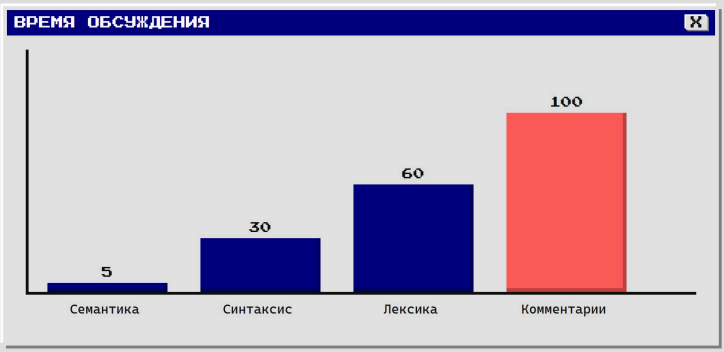
— Philip Greenspun, *около 1993 года; первых девяти правил никогда не существовало*

Оно называется «десятым», но оно единственное. Остальные девять так и не были написаны. Уже одно это делает его самым честным правилом в информатике: обещает девять вещей и не поставляет ни одной, как некоторые коммерческие предложения.

Смысл в том, что большие программы в итоге заново и плохо изобретают мощные инструменты, которые уже существовали: мини-язык конфигурации, самодельный интерпретатор, систему правил в текстовом файле, в которой в какой-то момент появляются даже циклы.

В моей старой учётной системе был файл настроек. Потом он выучил условия. Потом переменные. Однажды он попросил разрешения объявлять функции. Мы выключили его вовремя. Теперь он лежит в ящике стола, но по ночам я слышу, как он компилирует.

ЗАКОН УОДЛЕРА



Ядро языка: пять минут. Символ комментария: три недели.

При проектировании языка время обсуждения растёт по мере спуска от семантики к синтаксису, затем к лексическому синтаксису и, наконец, к комментариям, которые побеждают всё.

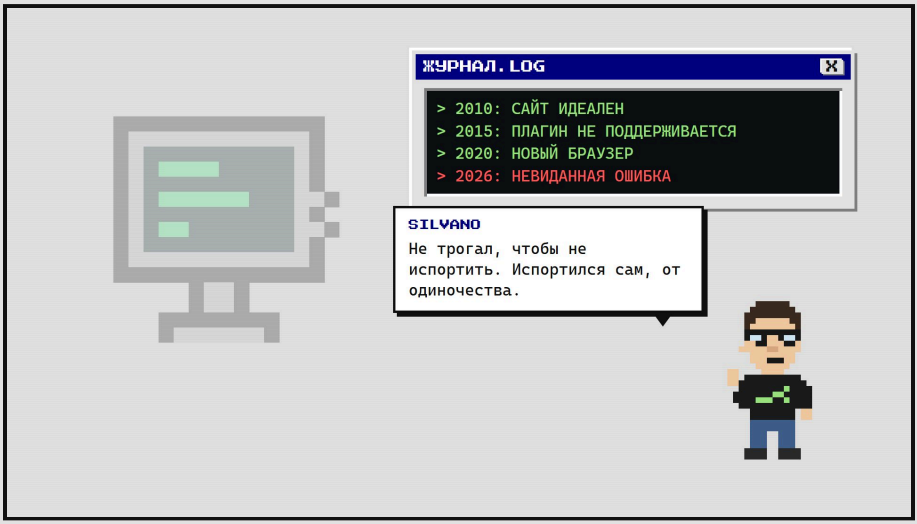
— Philip Wadler, *сообщения в рассылке Haskell, 1992 и 1996*

Глубинный смысл языка? Пять минут, все согласны. Символ для комментариев? Три недели, два раскола и один форк. Чем мельче вопрос, тем больше каждый чувствует себя экспертом.

Это программистская версия универсального человеческого правила: о трудном молчат, за лёгкое воюют. Любой, кто участвовал в код-ревью, видел пятьдесят замечаний про пробелы и ноль про логику.

Наша группа долго спорила, что использовать — табуляцию или пробелы. Мы создали комитет. Комитет выпустил документ. Документ был с отступами вперемешку. Теперь у комитета есть подкомитет. Проект продвигается — в комментариях.

ЗАКОНЫ ЛЕМАНА



Программа, которой пользуются в реальном мире, должна постоянно меняться, иначе она становится всё менее полезной; и каждое изменение увеличивает её сложность, если только кто-то специально не работает над её снижением.

— Meir M. Lehman, *законы эволюции программного обеспечения, сформулированные вместе с László Bélády начиная с 70-х годов*

Софт — это сад, а не статуя. Если его оставить в покое, он не останется идеальным: сорняки растут сами. Мир вокруг меняется, законы меняются, браузеры меняются, и программа, безупречная в 2010 году, сегодня просит установить вымерший плагин.

Самая жестокая часть — вторая: каждое изменение добавляет беспорядка. Заплатка здесь, особый случай там, и код превращается в многоквартирный дом с двенадцатью разными домофонами.

У меня есть сайт на Joomla, который я годами не трогаю, чтобы не испортить. Испортил его не я. Он испортился сам, от одиночества. Теперь я навещаю его по воскресеньям, обновляю ему одно расширение, а он благодарит меня новой ошибкой.

СОВЕТ СИСАДМИНА

Если вы пишете код, пусть немного, пусть только для себя, пользуйтесь **системой контроля версий**. Самая распространённая — Git, и она бесплатна. Она работает как машина времени с этикетками: каждый раз, когда работа в хорошем состоянии, вы сохраняете снимок с фразой, объясняющей, что изменили.

Преимущества огромны. Возвращаетесь во вчера одной командой. Видите, какая именно строка изменилась и когда. Пробуете безумную идею в отдельной ветке и выбрасываете её без потерь. Прощайте, папки-дубликаты с датой в названии и словом «старое», повторённым трижды.

Три золотых правила: сохраняйтесь часто и маленькими шагами; пишите сообщения, понятные даже незнакомцу, то есть вам через полгода; никогда не кладите в репозиторий пароли и ключи доступа, даже «на минутку». Эта минутка в интернете длится вечно.

ГЛАВА 7

ГЛ 07 — СОВЕЩАНИЯ

Проекты, совещания и клиенты

ЗАКОНОВ: 18 _

Рано или поздно каждый айтишник открывает неудобную правду: самые трудные проблемы не в компьютере. Они в переговорной, в смете, в письме с двенадцатью людьми в копии. Код можно исправить. Оценку, озвученную в коридоре, — нет.

Эта глава собирает законы командной работы. Начинаем со сроков, которые всегда ошибочны из-за избытка оптимизма, и переходим к клиентам, которые понимают, чего хотели, только когда это увидят. Потом совещания, которые размножаются рекурсивно и всегда обсуждают самое неважное, и демо, которые проваливаются пропорционально чину зрителя.

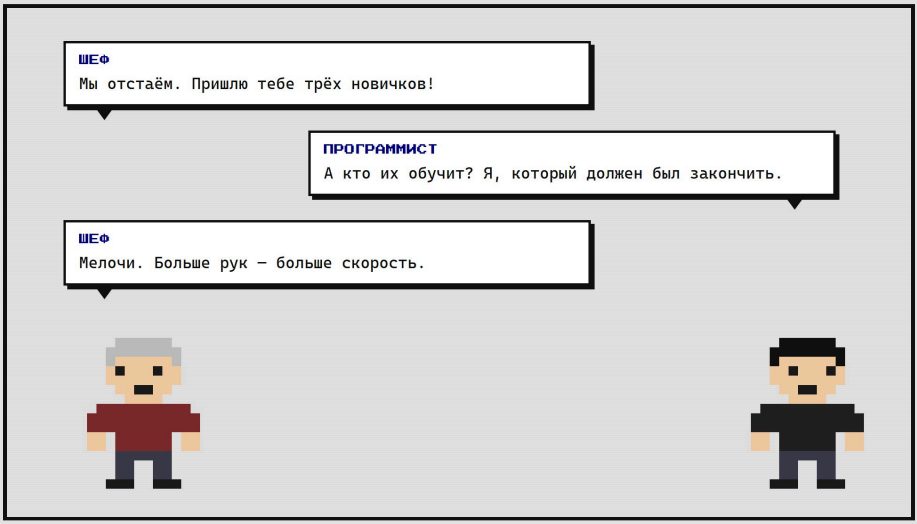
В конце — самые горькие законы, об организациях: цифры, которые становятся целями и перестают говорить правду, временные решения, на которых держится компания, системы, которые в итоге похожи на оргструктуру, вплоть до закрытых коридоров.

Кто работает в офисе, узнает коллег, начальников и даже себя. Это нормально. Хитрость в том, чтобы узнать себя до совещания, а не во время.

СОДЕРЖАНИЕ.DIR

01 Закон Брукса
02 Закон множителя пи
03 Закон клятвенной оценки
04 Закон клиента-ясновидца
05 Закон требования-призрака
06 Закон рекурсивного совещания
07 Закон тривиальности Паркинсона
08 Закон Сейра
09 Законы Вийо
10 Закон демонстрации перед клиентом
11 Следствие важной публики
12 Закон вечной времянки
13 Закон Конвея
14 Закон Галла
15 Закон Вайнберга
16 Закон Гудхарта
17 Закон Кэмпбелла
18 Принцип Дилберта

ЗАКОН БРУКСА



Добавление людей в запаздывающий программный проект задерживает его ещё больше.

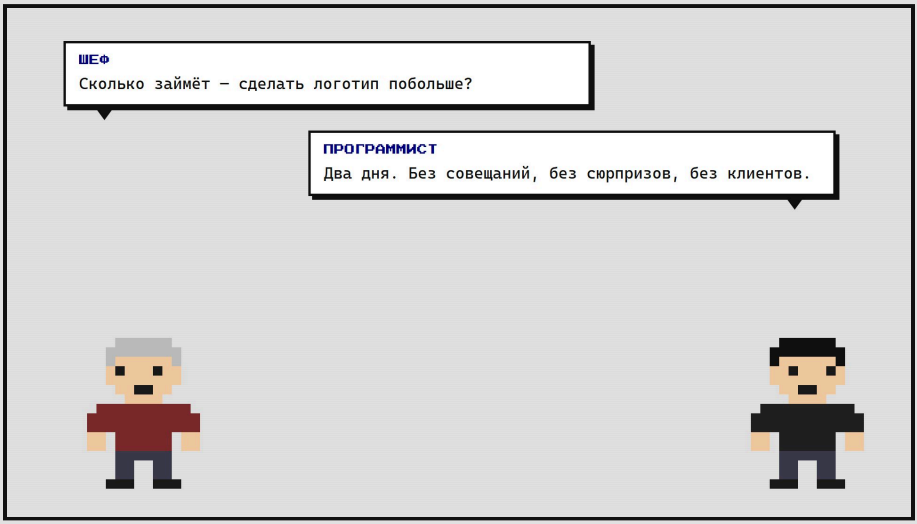
— Fred Brooks, *The Mythical Man-Month, 1975*

Руководитель-визионер смотрит на диаграмму Ганта, видит красное и провозглашает: «Удваиваем команду!». Это та же логика, по которой девять женщин родили бы ребёнка за месяц.

Проблема в том, что новичков надо вводить в курс дела, а вводят их как раз те, кто заканчивал работу. К тому же каждый новый человек умножает количество разговоров: вчетвером люди просто общаются, вдвенадцатером проводят совещания, чтобы решить, когда проводить совещания.

Однажды, чтобы наверстать отставание, мне выделили трёх стажёров. В первый день я объяснял, где туалет. Во второй — пароль от Wi-Fi. На третий проект отставал ещё на две недели, зато Wi-Fi работал великолепно.

ЗАКОН МНОЖИТЕЛЯ ПИ



Чтобы получить реальный срок программного проекта, возьми оценку программиста и умножь её на число пи. Если проект важный, используй и знаки после запятой.

— Аноним, *компьютерный фольклор*

Программист, оценивающий задачу, — оптимист по складу характера. Он представляет мир без форс-мажоров, без совещаний, без заказчика, который звонит спросить, нельзя ли сделать логотип чуть побольше. В этом идеальном мире «два дня» — это два дня.

Потом приходит реальность, у которой своя математика. Два дня превращаются в шесть целых двадцать восемь сотых, и эти ноль двадцать восемь — тот момент, когда выясняется, что пароль от тестового сервера знает только человек, давно ушедший на пенсию.

Почему именно пи? Никто не знает. Но это число иррациональное и бесконечное, а значит, лучше всего подходит для описания дедлайнов. Я в коммерческих предложениях перестал писать дни. Пишу сразу диаметр проекта, а посчитать оставляю заказчику.

ЗАКОН КЛЯТВЕННОЙ ОЦЕНКИ



Любая оценка, высказанная устно, с тысячей оговорок и тремя «может быть», станет дедлайном, записанным в календаре начальника.

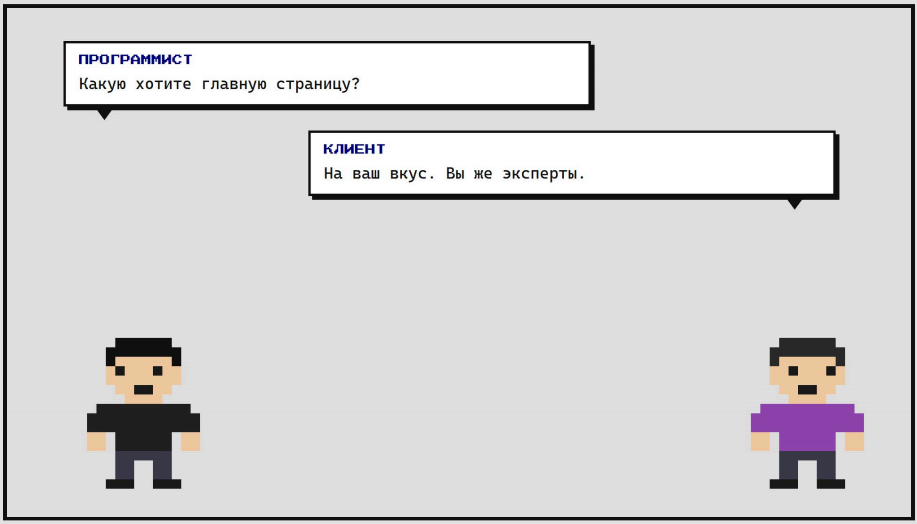
— Аноним, *компьютерный фольклор*

Программист, которого поймали в коридоре, отвечает с дипломатической осторожностью: «Ну, навскидку, если всё пойдёт хорошо, может быть, без форс-мажоров, неделя. Но вы на это не рассчитывайте». Начальник кивает. Он услышал только «неделя».

На следующий день эта фраза появляется в официальном документе, потом в письме клиенту, потом на цветной диаграмме на экране в переговорной. Оговорки удалены при сжатии, как детали фотографии, отправленной через мессенджер.

Руководитель-визионер тем временем гордо её презентует: «Команда взяла на себя обязательство». Команда не знала, что брала на себя обязательство. С тех пор, когда меня спрашивают, сколько времени это займёт, я отвечаю вопросом: «Это вопрос или договор?». Обычно наступившая тишина служит ответом. И сметой.

ЗАКОН КЛИЕНТА-ЯСНОВИДЦА



Клиент точно понимает, чего хочет, ровно в тот момент, когда ты сдаёшь ему то, что он просил.

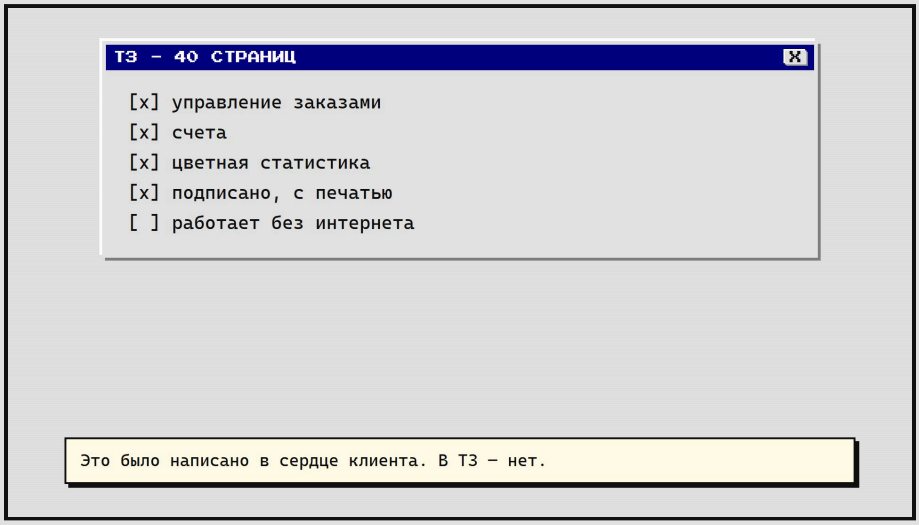
— Аноним, *компьютерный фольклор*

На этапе анализа клиент расплывчат, задумчив, дзен. «Делайте как считаете нужным, вы же специалисты». Потом он видит готовую работу, и на него нисходит мистическое озарение, будто разверзлись небеса: теперь он знает. Знает, что это не то.

Явление хорошо известно и в технической литературе: есть знаменитая анонимная картинка с качелями — какими их хотел клиент, как их понял менеджер по продажам, как их построил программист. Ни в одной версии это не качели.

Я нашёл безотказный метод. Сразу, за полдня, делаю неправильную версию — специально. Клиент смотрит, приходит в ужас и наконец говорит ясно. Это единственный случай в информатике, когда баг — самая полезная функция проекта. Выставляю счёт как за «анализ требований методом от противного».

ЗАКОН ТРЕБОВАНИЯ-ПРИЗРАКА



Самое важное требование проекта сообщают устно, мимоходом, в день сдачи.

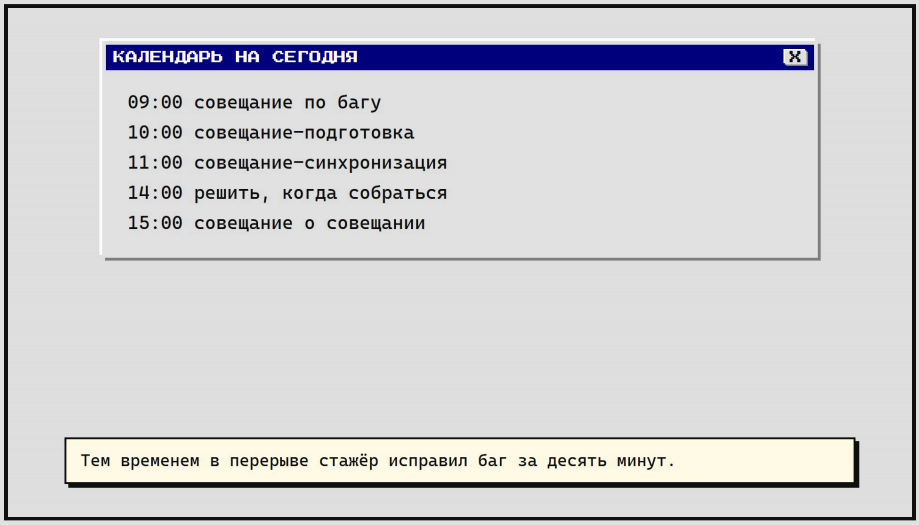
— Аноним, *компьютерный фольклор*

Техзадание на сорок страниц, с таблицами, графиками и оглавлением, как у мирного договора. Всё обсуждено, подписано, проштамповано. А потом, в день сдачи, пока все жмут друг другу руки, клиент добавляет: «Ах да, оно, конечно, должно работать и без интернета».

Говорит он это без злого умысла. Говорит, как говорят «доброе утро», с безмятежностью человека, который вспомнил очевидную мелочь. Для него так было всегда. Это было записано у него в сердце, если уж не в документе.

С тех пор я ввёл в офисе совещание последнего вопроса. Перед подписанием спрашиваю: «Есть что-то настолько очевидное, что вы об этом даже не сказали?». Тишина, которая следует, всегда долгая. И внутри этой тишины каждый раз живёт новый модуль, который придётся разрабатывать.

ЗАКОН РЕКУРСИВНОГО СОВЕЩАНИЯ



Любая сложная техническая проблема порождает совещание, которое порождает подготовительное совещание, которое порождает необходимость совещания, чтобы решить, когда проводить совещание.

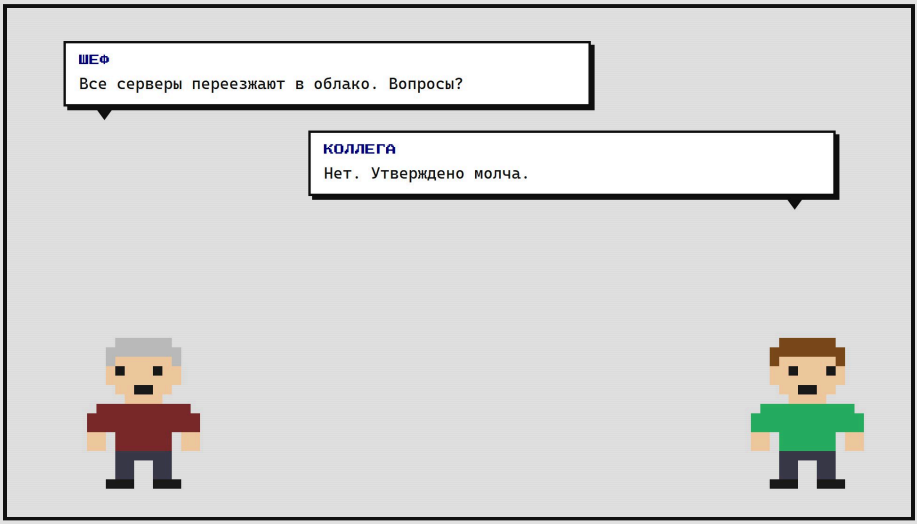
— Аноним, *компьютерный фольклор*

В информатике рекурсия — это функция, которая вызывает саму себя. В компаниях это совещание, которое созывает само себя, до бесконечности, пока кто-нибудь не уйдёт в отпуск и не разорвёт цикл за исчерпанием календаря.

Схема всегда одна. Восемь человек собираются обсудить баг. Выясняется, что нужна информация. Назначается новая встреча ещё с четырьмя людьми. На этой встрече возникает необходимость «синхронизироваться». Тем временем баг исправил стажёр, за десять минут, в перерыве.

Всегда есть руководитель, который торжественно открывает заседание: «Коллеги, сегодня мы принимаем решение». И решение действительно принимают: дату следующего совещания. Я предложил повесить в переговорной счётчик, как на сельской ярмарке. Каждый час совещания — минус одна строка кода.

ЗАКОН ТРИВИАЛЬНОСТИ ПАРКИНСОНА



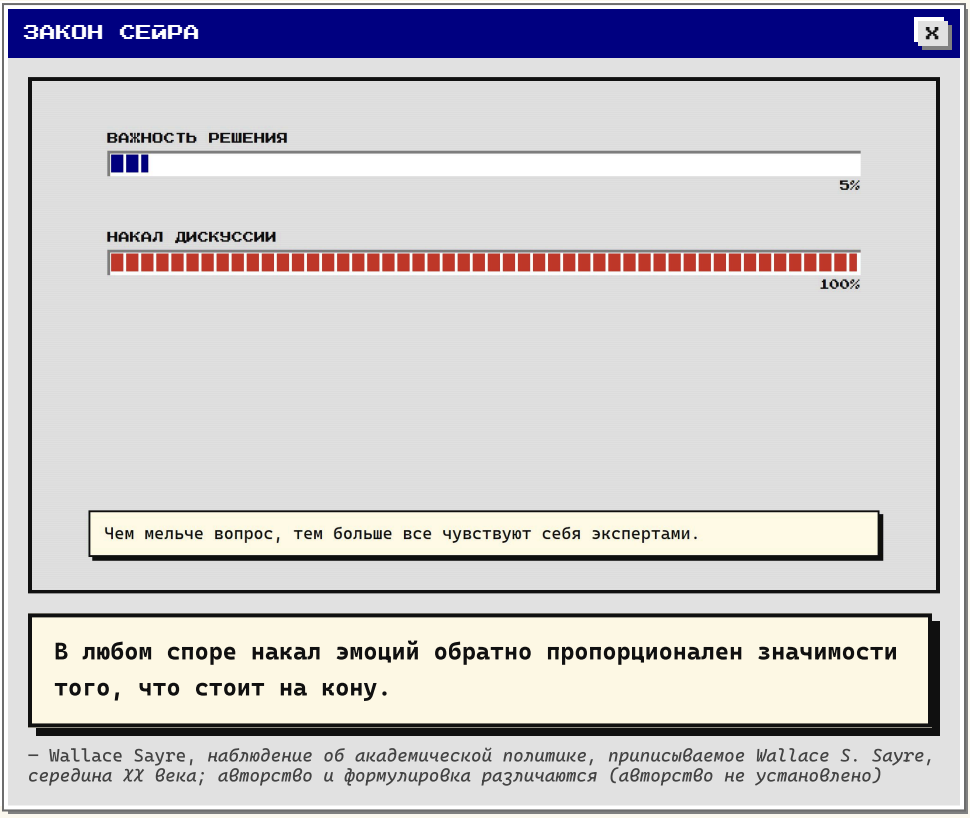
Организация тратит на решения время, обратно пропорциональное их важности: о реакторе спорят мало, а о велосипедном навесе — бесконечно.

— C. Northcote Parkinson, *Parkinson's Law, 1957; в информатике популяризирован как «bikeshedding» Poul-Henning Kamp в сообществе FreeBSD, 1999*

Совет директоров за две минуты утверждает проект на миллионы, которого никто не понимает. А потом час обсуждает навес для велосипедов, который все прекрасно понимают и насчёт цвета которого у каждого есть мнение.

В информатике это стало паролем: когда спор разгорается из-за неважной детали, кто-нибудь пишет «bikeshedding», и всем на секунду становится стыдно. Потом все возвращаются к спору об имени переменной.

На моём последнем совещании мы молча одобрили перевод всех серверов в облако. А потом сорок минут ругались из-за иконки новой учётной системы: шестерёнка или облако? В итоге выбрали шестерёнку в форме облака. Никто не знает, что она делает. Как и облако, впрочем.



В любом споре накал эмоций обратно пропорционален значимости того, что стоит на кону.

— Wallace Sayre, *наблюдение об академической политике, приписываемое Wallace S. Sayre, середина XX века; авторство и формулировка различаются (авторство не установлено)*

Родившийся для описания склок между университетскими профессорами, он идеально подходит к техническим совещаниям. Миграция всей базы данных проходит за пять минут. Цвет кнопки «Отмена» вызывает двухчасовые дебаты и письмо с половиной компании в копии.

Причина проста: в огромных вопросах мало кто чувствует себя компетентным; в мелких мнение есть у всех, и защищать его ничего не стоит.

Самое длинное совещание в моей карьере касалось названия общей папки. «Документы», «Документация» или «Док». Четыре часа. Голосовали. Победили «Документы_ОКОНЧ». На следующий день появилась «Документы_ОКОНЧ_новая». Конфликт стал постоянным, как некоторые холодные войны, только с большим количеством прав на запись.

ЗАКОНЫ ВИЙО

ПРОСТОЙ САЙТ, ПО МНЕНИЮ:

- клиента: три страницы
- дизайнера: северный минимализм
- программиста: без базы данных
- менеджера по продажам: дёшево

При сдаче никто не доволен. Все уверены, что просили именно это.

Коммуникация обычно не удаётся, разве что случайно. А если кажется, что она удалась как задумано, значит, точно есть недопонимание.

— Osmo Antero Wiio, *«Wiion lait – ja vähän muidenkin»*, 1978

Финский профессор, исследователь коммуникаций, пришёл к выводу, который любой сисадмин уже знает: общение не работает. Когда работает — это случайность. А если тебе кажется, что все поняли, это лишь потому, что каждый понял своё.

Клиент просит «простой сайт». Он имеет в виду три страницы. Дизайнер — скандинавский минимализм. Программист — никакой базы данных. Менеджер по продажам — «стоит недорого». При сдаче никто не доволен, но все уверены, что просили именно это.

Для этого и существуют письменные ТЗ, протоколы, итоговые письма. Они нужны, чтобы точно задокументировать недопонимание. Теперь я заканчиваю каждое совещание просьбой ко всем повторить, что мы решили. Каждый раз получается пять проектов. Зато мы узнаём об этом сразу, а не на приёмке.

ЗАКОН ДЕМОНСТРАЦИИ ПЕРЕД КЛИЕНТОМ



Процент успеха. Последний замер сделан в присутствии того, кто решает.

Программа, отработавшая сто прогонов, ломается на сто первом — том, что перед клиентом.

— Аноним, *компьютерный фольклор (известен также как «демо-эффект»)*

Демо проверили накануне вечером, утром, в машине и на парковке. Всё гладко. Потом ты входишь, клиент садится, ты уверенно говоришь «Сейчас я вам покажу», и экран превращается в палитру ошибок, которых никто никогда не видел.

Айтишники называют это демо-эффектом. Это единственное известное явление, способное обнаружить присутствие лица, принимающего решения, на расстоянии десяти метров, сквозь стены.

Самое прекрасное: как только клиент уходит, программа снова работает идеально. Смотрит на тебя, будто говоря: «А что случилось? Я же была на месте».

Совет от того, кто это пережил: всегда готовьте план Б. Записанное видео работающей демонстрации или скриншоты ключевых шагов. Если программа капризничает, показывают кино.

СЛЕДСТВИЕ ВАЖНОЙ ПУБЛИКИ



Иерархия, точная, как оргструктура. При уборщице сервер успокаивается.

Чем выше должность того, кто присутствует на демонстрации, тем вероятнее, что демонстрация провалится.

— Аноним, *компьютерный фольклор (сборники Usenet с законами о компьютерах)*

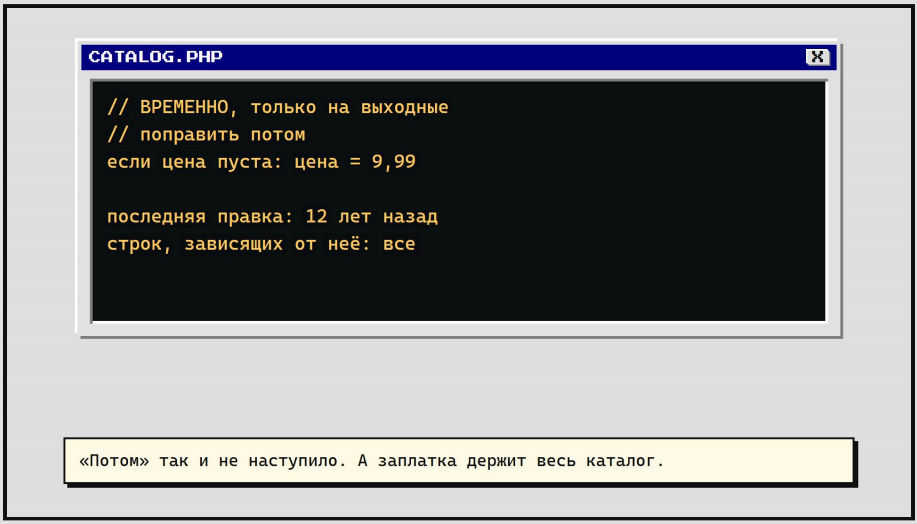
Перед коллегой демо держится. Перед начальником отдела колеблется. Перед директором кашляет. Перед генеральным, только что прилетевшим на вертолёте, компьютер выключается и просит установить сорок семь обновлений.

Это иерархическая шкала ошибок, точная, как оргструктура. Некоторые исследователи утверждают, что дорогие пиджаки излучают электромагнитное поле. У меня нет доказательств. Но я заметил, что сервер успокаивается, когда входит уборщица.

Руководитель-визионер тем временем улыбается и говорит: «Отлично, мы нашли точку роста». Что на корпоративном языке означает: зайдите ко мне в кабинет.

Настоящий совет: для важных презентаций используйте заранее подготовленную стабильную среду и проведите полную репетицию в том же зале, на том же компьютере.

ЗАКОН ВЕЧНОЙ ВРЕМЯНКИ



В софте нет ничего более постоянного, чем временное решение.

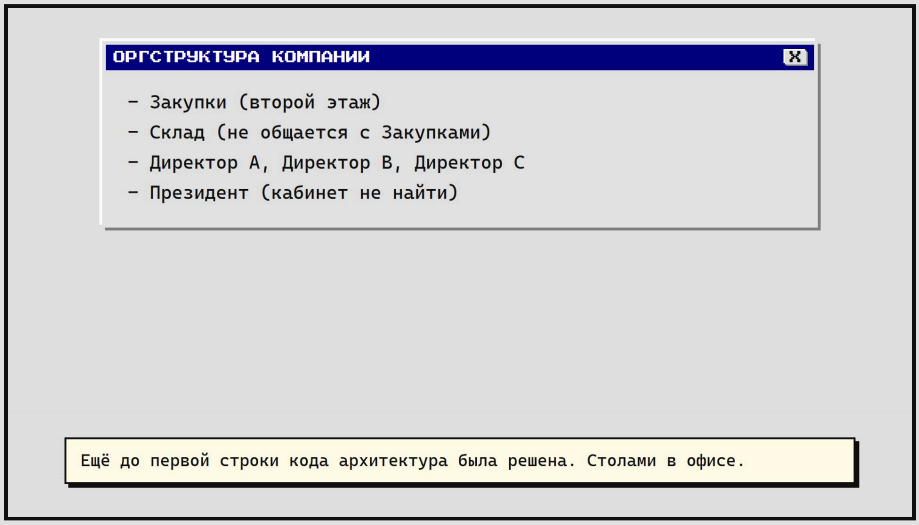
— Аноним, *компьютерный фольклор*

В каждой программе есть строчка с пометкой «поправить потом». Потом не наступает никогда. Строчка переживает три версии, два переезда сервера и смену владельца компании.

Временные решения обладают невероятной закалкой. Постоянные решения, тщательно спроектированные, переписывают каждые два года. А заплатка, поставленная в одиннадцать вечера, чтобы снова пошли счета, становится историческим наследием, со своим достоинством и своей аурой.

Я обслуживаю сайт, где двенадцать лет назад кто-то написал маленький скрипт «только на выходные». Сегодня на нём держится весь каталог. Никто не смеет его трогать. Иногда я на него смотрю, просто так, с уважением, как смотрят на несущую стену старого деревенского дома: кривая, да, но убери её — и всё рухнет.

ЗАКОН КОНВЕЯ



Организация, проектирующая систему, в итоге создаст систему, копирующую её собственную структуру коммуникаций.

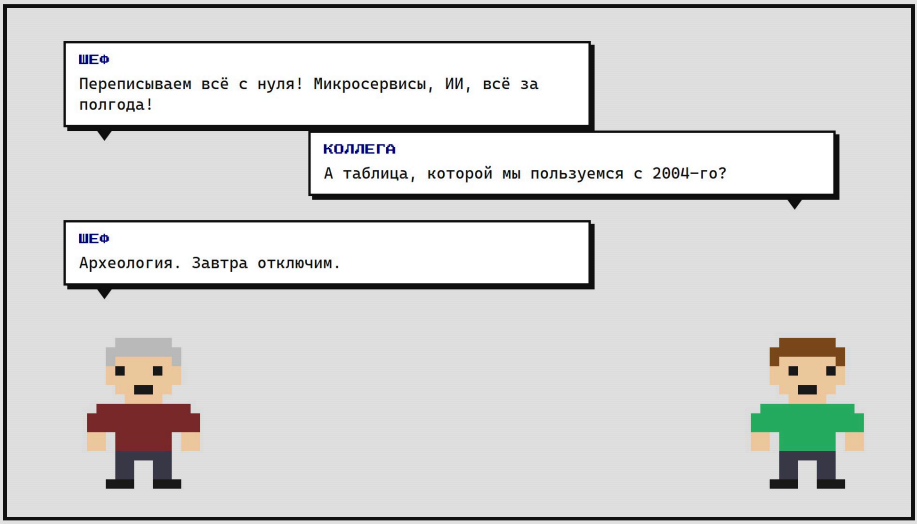
— Melvin Conway, *How Do Committees Invent?*, *Datamation*, 1968

Если над компилятором работают четыре группы, получится четырёхпроходный компилятор. Если отдел закупок не разговаривает со складом, в учётной системе будут два справочника товаров, которые друг с другом не здороваются.

Софт — это зеркало. Смотришь на архитектуру и видишь оргструктуру, со всеми обидами. Тот модуль, который ни с кем не взаимодействует? Это коллега с третьего этажа, который обедает в одиночестве.

Я видел сайт с тремя разными меню, по одному на каждого директора. Четвёртое, скрытое меню вело на страницу президента. Никто не знал, как туда попасть, совсем как в его кабинет. Архитектура была идеальна: она отражала компанию вплоть до последнего запертого коридора.

ЗАКОН ГАЛЛА



Работающая сложная система всегда происходит от простой системы, которая работала. Сложная система, спроектированная с нуля, не работает, и починить её невозможно.

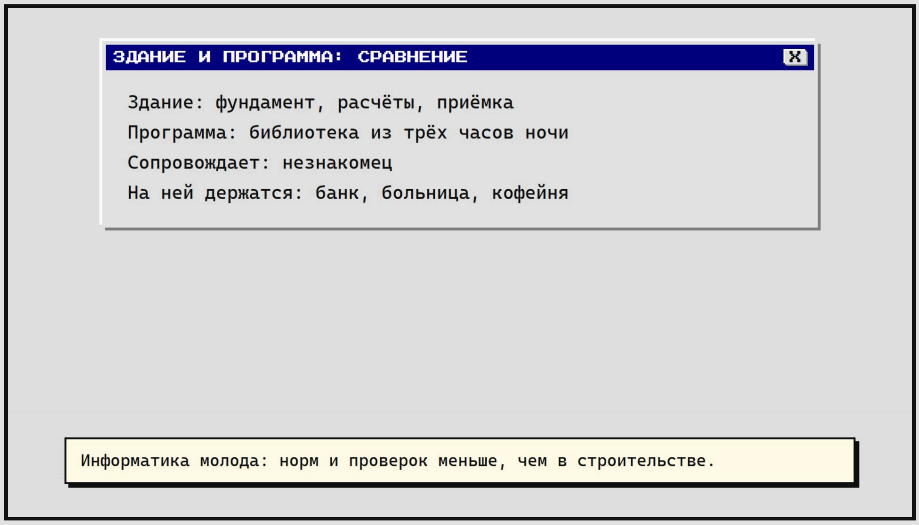
— John Gall, *Systemantics (General Systemantics)*, 1975

Гуру цифровой трансформации поднимается на сцену: «Переделаем всё с нуля: микросервисы, искусственный интеллект, блокчейн, за полгода!». Аплодисменты. Шестнадцать месяцев спустя все возвращаются к общей таблице Excel, которая работала с 2004 года.

Этот закон — похвала эволюции. Системы, которые держатся, росли по кусочку, как сталактиты или папки «Новая папка (7)». А те, что родились огромными и идеальными на бумаге, рушатся, потому что никто никогда не видел, как они работают в малом.

Моя любимая учётная система родилась как текстовый файл из трёх строк. Сегодня она управляет целой компанией. Те три строки всё ещё где-то внутри. Никто их не трогает. Это несущая реликвия.

ЗАКОН ВАЙНБЕРГА



Если бы строители строили здания так, как программисты пишут программы, первый же пролетающий дятел разрушил бы цивилизацию.

— Gerald Weinberg, *авторство не установлено: встречается в сборнике 1975 года (CoEvolution Quarterly) и как «Второй закон Вайнберга» в Murphy's Law Arthur Bloch, 1977*

У настоящего здания есть фундамент, расчёты конструкций, приёмка. У средней программы есть библиотека, скачанная в три часа ночи и поддерживаемая незнакомцем, от которой зависят банк, больница и кофейный автомат.

Закон — не только насмешка: он напоминает, что информатика — молодая дисциплина, с меньшим количеством норм и проверок, чем строительство. Многое из того, чем мы пользуемся, держится на доброй воле и удаче.

Однажды я увидел дятла на крыше нашего ангара, прямо над серверной. Я встал между ним и антенной. Мы долго смотрели друг на друга. В конце концов он улетел. Думаю, он понял, что в этом нет необходимости: сервер сломается сам, в понедельник.

ЗАКОН ГУДХАРТА



Премируют за строки кода. Продуктивность утроилась. Софт — загадочным образом нет.

Когда мера становится целью, она перестаёт быть хорошей мерой.

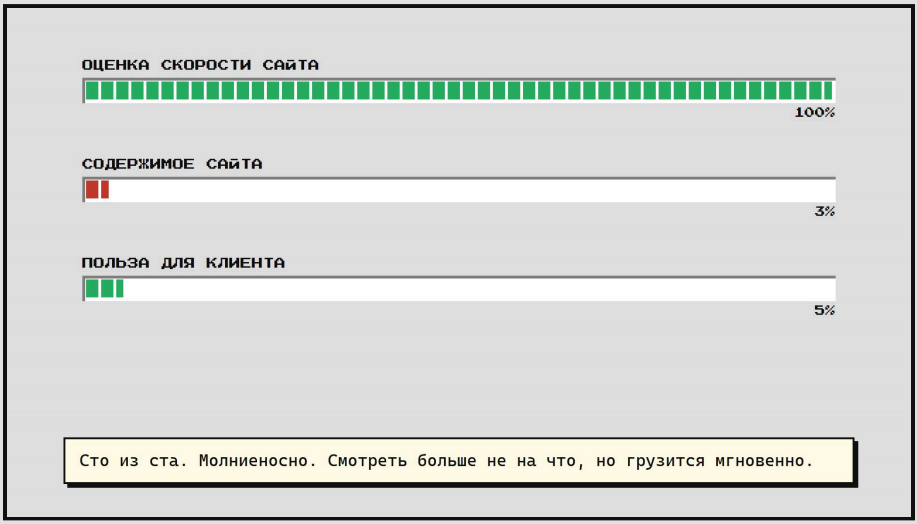
— Charles Goodhart, *статья о денежно-кредитной политике, 1975; знаменитая формулировка принадлежит Marilyn Strathern, 1997*

Компания решает измерять продуктивность программистов количеством строк кода. Через месяц программы становятся длиннющими. Функция, занимавшая три строки, теперь занимает триста, с поэтическими комментариями и переменными из четырёх слов. Продуктивность утроилась. Софт, загадочным образом, — нет.

Это закон Гудхарта: стоит начать давить на число, люди учатся раздувать число, а не улучшать то, что оно должно было отражать. Это касается закрытых тикетов, звёздочек в отзывах, оценок скорости сайтов.

В техподдержке измеряли время закрытия обращений. Один коллега закрывал все за восемь секунд. Отвечал, говорил «попробуйте перезагрузить» и вешал трубку. Трижды признан лучшим сотрудником месяца. Пользователи звали его Молния.

ЗАКОН КЭМПБЕЛЛА



Чем больше количественный показатель используется для важных решений, тем сильнее он искажается и тем больше искажает процессы, за которыми должен следить.

— Donald T. Campbell, *«Assessing the Impact of Planned Social Change»*, 1976 (*ранние формулировки с 1969 года*)

Близкий родственник закона Гудхарта, закон Кэмпбелла добавляет мрачную деталь: число не только перестаёт говорить правду, но и начинает деформировать саму работу. Если премия зависит от баллов, работа превращается в производство баллов.

В мире сайтов это очевиднее некуда. Решают, что важна только оценка инструмента проверки скорости. Тогда убирают картинки, тексты, функции, всё, что весит. Сайт получает сто из ста. Он молниеносный. Смотреть там больше нечего, зато грузится в мгновение ока.

В одной знакомой компании айтишника оценивали по числу работающих компьютеров в конце месяца. Он очень мудро в последний день месяца не трогал ни одного. Работали все. Медленно, с кучей предупреждений, но статистически в отличном здоровье.

ПРИНЦИП ДИЛБЕРТА



ЖУРНАЛ.LOG

> ДОСТУП К СЕРВЕРУ: ЗАПРЕЩЁН
> НОВЫЙ КАБИНЕТ: С ВИДОМ
> РОЛЬ: МЕНЕДЖЕР

Первобытный контроль доступа: вместо прав — кабинет с видом.

Компании склонны продвигать наименее способных сотрудников на руководящие должности, чтобы ограничить вред, который они могут нанести настоящей работе.

— Scott Adams, *Wall Street Journal, 1995; книга «The Dilbert Principle», 1996*

Принцип Питера говорил, что человек поднимается, пока не станет некомпетентным. Принцип Дилберта циничнее: кто-то уже стартует некомпетентным, и его специально перемещают наверх, где он не может тронуть ничего важного. Как назначить гиперактивного ребёнка капитаном команды, чтобы он не трогал мяч.

В информатике у явления есть неофициальное техническое название: руководитель-визионер. Он не знает, что такое база данных, но понял, что будущее — за данными. Говорит это на совещаниях, стуча кулаком по столу. Потом просит кого-нибудь распечатать ему письмо.

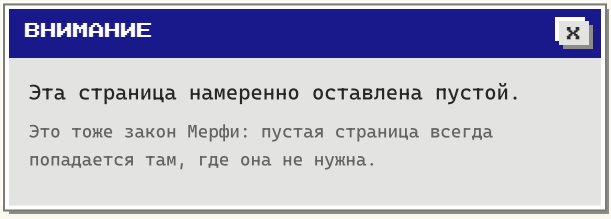
Признаем, у системы есть своя эффективность: человека держат подальше от сервера, сервер благодарит. Это примитивная форма управления доступом. Только вместо прав доступа используется кабинет с видом.

СОВЕТ СИСАДМИНА

Две привычки, которые спасли мне больше проектов, чем любой софт.

Письмо с итогами. После каждого важного совещания или звонка в течение часа напишите всем участникам короткое письмо: что решили, кто что делает, к какому сроку, что осталось открытым. Пять строк. Если кто-то понял иначе, вы узнаете сразу, а не при сдаче. А через три месяца, когда никто ничего не вспомнит, это письмо станет официальной версией событий.

Оценка-вилка. Никогда не называйте одно число. Говорите «от трёх до шести дней, потому что зависит от того и этого». Одно число становится обещанием; вилка заставляет всех обсудить, что её расширяет. И запишите также, что в работу не входит. Требование-призрак ненавидит маркированные списки.


ВНИМАНИЕ
Эта страница намеренно оставлена пустой.
Это тоже закон Мерфи: пустая страница всегда попадается там, где она не нужна.

ГЛАВА 8

ГЛ 08 — БУДУЩЕЕ

Прогресс, обновления и искусственный интеллект

ЗАКОНОВ: 20 _

Последняя глава смотрит вперёд, что в информатике означает смотреть на полоску прогресса обновления. С неё и начнём: обновления, приходящие в неподходящий момент, чинящие одно и ломающие другое, новые компьютеры, устаревшие уже на выходе из магазина.

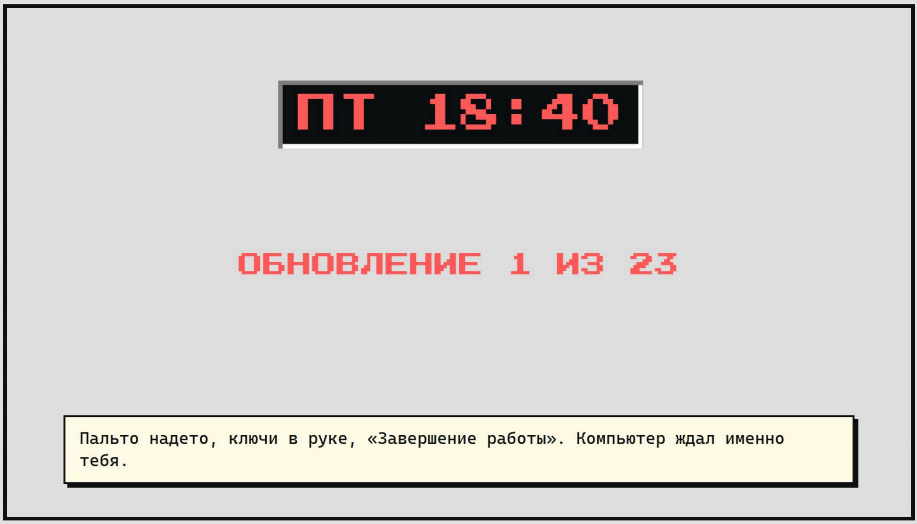
Потом масштаб растёт. Есть законы, описывающие большую гонку технологий: всё более мощные чипы, всё более дорогие фабрики, всё более тяжёлый софт, программы, которые хотят уметь всё. Это законы экономические и инженерные, иногда оптимистичные, часто ироничные. Они объясняют, почему компьютер несравнимо быстрее, чем двадцать лет назад, а текстовый редактор открывается за то же время.

Глава заканчивается шестью следствиями об искусственном интеллекте. Их написал я сам, после многих часов работы с этими ассистентами, и подписываю их с гордостью человека, видевшего, как машина несколько раз извиняется за одну и ту же ошибку. Я не против ИИ: пользуюсь им каждый день. Именно поэтому люблю его, как любят блестящего коллегу, который время от времени выдумывает названия функций.

СОДЕРЖАНИЕ.DIR

01 Закон неуместного обновления
02 Закон ломающего обновления
03 Закон нового компьютера
04 Закон Вирта
05 Закон Завински
06 Закон Этвуда
07 Закон Рока
08 Закон Гроша
09 Закон Амдала
10 Закон Густафсона
11 Закон Куми
12 Закон Белла о классах компьютеров
13 Парадокс Джевонса
14 Закон Амары
15 Следствие воображаемой функции
16 Следствие искусственных извинений
17 Следствие трёх секунд
18 Следствие торжественного промпта
19 Следствие «Остальное без изменений»
20 Следствие забытого правила

ЗАКОН НЕУМЕСТНОГО ОБНОВЛЕНИЯ



Компьютер решает устанавливать обновления всегда тогда, когда тебе нужно его выключить и бежать или включить и выступать.

— Аноним, *компьютерный фольклор*

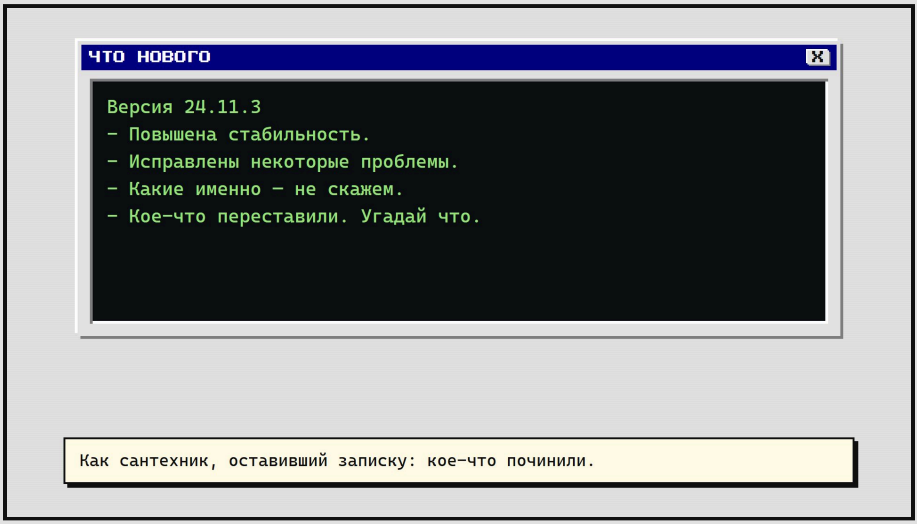
Компьютер всю ночь простоял включённым, без дела, скучая. Мог бы спокойно обновиться. Но нет: он ждёт, пока ты нажмёшь «Завершение работы», уже в пальто и с ключами в руке. И тогда любезно объявляет: «Обновление 1 из 23. Не выключайте компьютер».

И ты стоишь, как перед железнодорожным переездом. Поезд обновлений медленно идёт, вагон за вагоном. Иногда он останавливается на тридцати процентах, чтобы поразмыслить о смысле жизни.

Мой сосед купил новый ноутбук, чтобы быстро заполнить налоговую декларацию. Включил его в марте. В мае заполнил декларацию от руки, на бумаге, дедушкиной перьевой ручкой.

Совет мастера: настройте период активности системы, чтобы обновления ставились, когда вы не пользуетесь компьютером. И никогда не выключайте его принудительно во время обновления.

ЗАКОН ЛОМАЮЩЕГО ОБНОВЛЕНИЯ



Каждое обновление, решающее одну проблему, привносит как минимум одну новую, обычно в том, что раньше прекрасно работало.

— Аноним, *компьютерный фольклор*

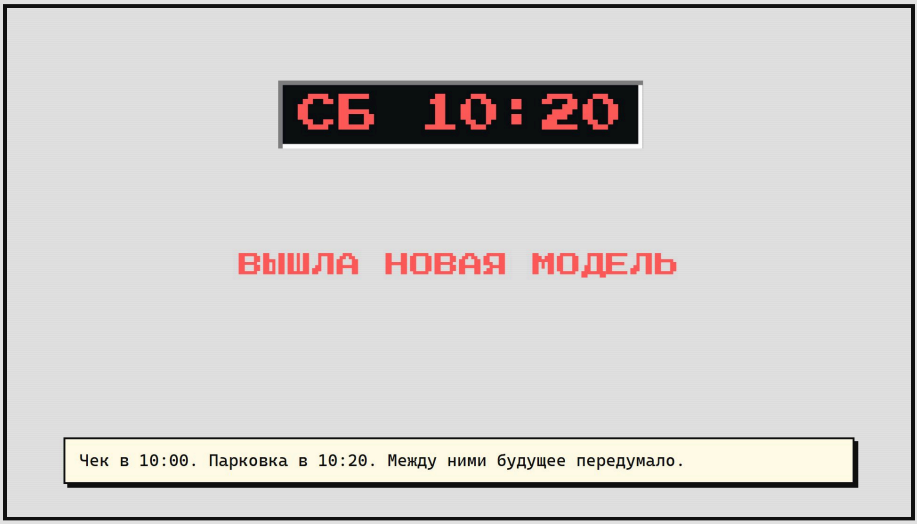
Обновление приходит со списком захватывающих новинок: «Повышена стабильность. Исправлены некоторые проблемы». Какие проблемы — никогда не говорится. Это как сантехник, оставивший записку: «Кое-что починил».

На следующий день принтер не печатает, бухгалтерская программа требует лицензию, а значок корзины стал зелёным. Это называют прогрессом. Я называю это переездом: всё на месте, но ничего не на своём месте.

Гуру «мы всегда так делали» пользуется моментом и провозглашает: «Я же говорил, никогда ничего не обновлять». А потом подхватывает вирус через программу, застрявшую в 2012 году.

Настоящий совет: обновления безопасности ставить нужно. Но перед крупными обновлениями сделайте бэкап и, если можете, подождите несколько дней, пока проблемы найдут другие.

ЗАКОН НОВОГО КОМПЬЮТЕРА



Компьютер, который ты только что купил, уже устарел из-за модели, вышедшей, пока ты ехал домой, и она к тому же дешевле.

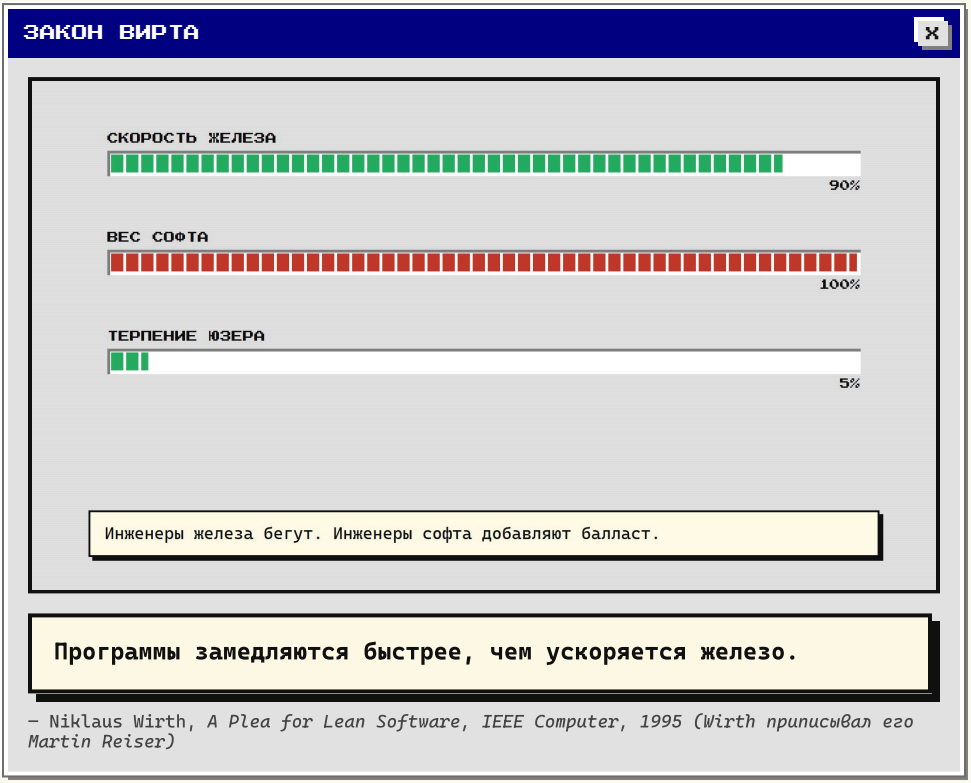
— Аноним, *компьютерный фольклор*

Самый счастливый момент в жизни компьютера — чек. Второй самый счастливый не наступает никогда, потому что по дороге от магазина до машины выходит новая модель: быстрее, тоньше и со стартовой скидкой.

У меня есть знакомый, который, чтобы победить закон, всегда ждёт следующую модель. Ждёт с 2011 года. До сих пор пользуется тогдашним ноутбуком, который включается двенадцать минут. Говорит, что не торопится: он «инвестирует в будущее».

Братья и сёстры прогресса, правда такова: ни один компьютер никогда не бывает актуальным. Есть только компьютеры, которые ещё не знают, что они старые.

Настоящий совет: покупайте, когда нужно, с запасом на три-пять лет работы. Идеальная модель всегда выйдет завтра.



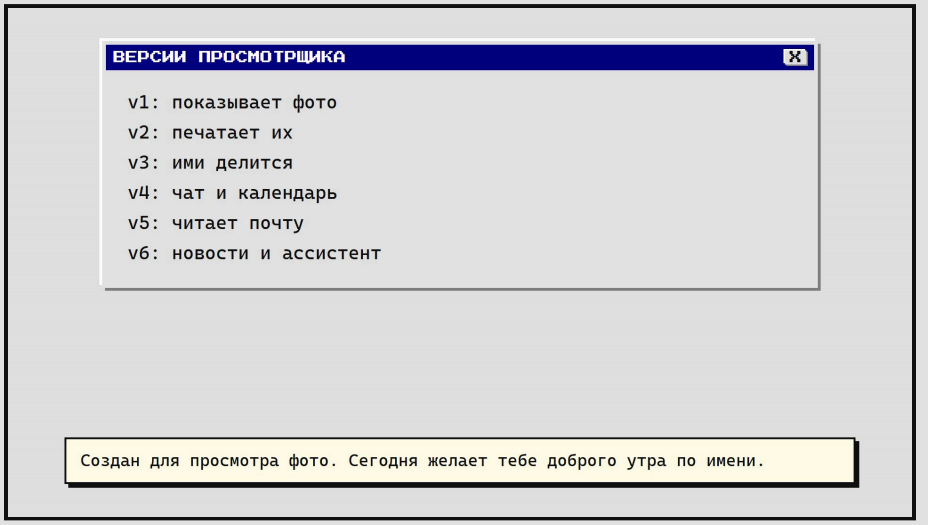
— Niklaus Wirth, *A Plea for Lean Software*, IEEE Computer, 1995 (Wirth приписывал его Martin Reiser)

В 1995 году текстовый редактор помещался на нескольких дискетах и запускался мгновенно. Сегодня у нас процессоры в тысячи раз быстрее, а редактор запускается столько же, потому что тем временем научился загружать целый браузер, три аналитических движка и помощника, который спрашивает, не нужна ли тебе помощь.

Это гонка: инженеры железа бегут, инженеры софта добавляют балласт. Финиш всегда отодвигается на несколько секунд загрузки.

У моего нового ПК 32 гигабайта оперативки. Я скормил их калькулятору. Теперь он молниеносно считает 2+2 — после того как обновится, проверит лицензию и предложит премиум-подписку на десятичные дроби.

ЗАКОН ЗАВИНСКИ



Каждая программа пытается расширяться, пока не сможет читать почту. Те, что не могут, вытесняются теми, что могут.

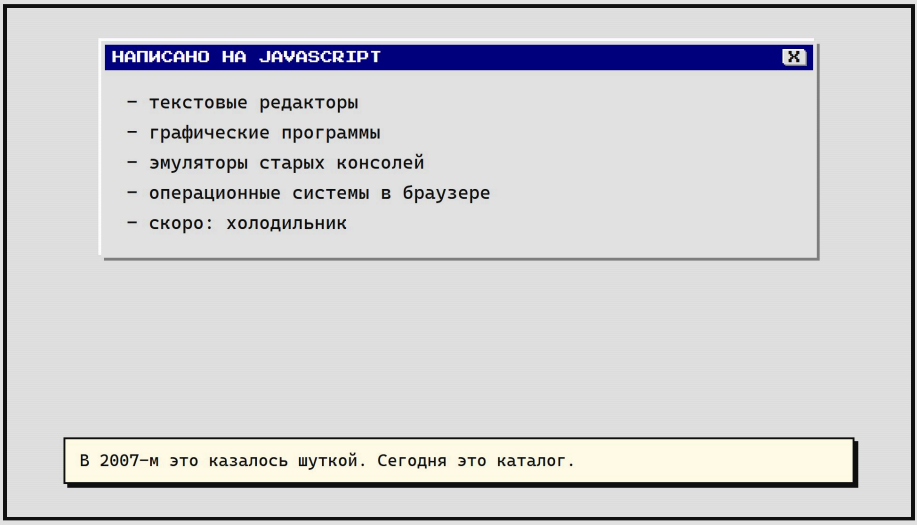
— Jamie Zawinski, *наблюдение, приписываемое Jamie Zawinski, разработчику Netscape, середина 90-х*

Рождается как просмотрщик картинок. Потом добавляет печать, потом «поделиться», потом чат, потом календарь. Через два года у него есть почта, лента новостей и виртуальный помощник, который здоровается с тобой по имени.

Смысл серьёзный: программы склонны обрастать функциями, потому что каждый пользователь просит ещё одну, а кто остановился, выглядит устаревшим. Сегодня почту сменили чаты, видео и искусственный интеллект, но лихорадка та же.

Вчера я обновил калькулятор. Теперь у него есть почтовый ящик. Мне пришла рассылка от калькулятора с новостями о новых функциях калькулятора. Я до сих пор не понял, где там складывать.

ЗАКОН ЭТВУДА



Любое приложение, которое можно написать на JavaScript, рано или поздно будет написано на JavaScript.

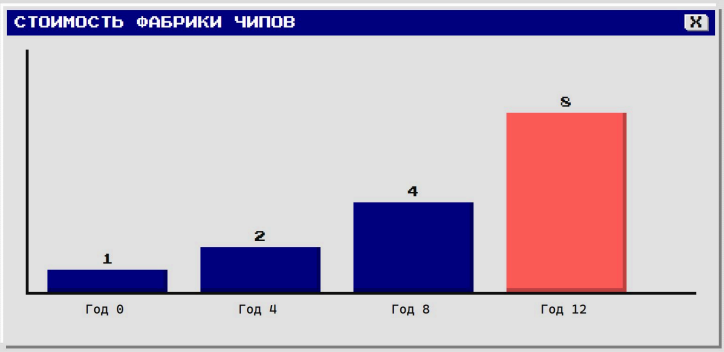
— Jeff Atwood, *The Principle of Least Power, блог Coding Horror, 2007*

В 2007 году это казалось шуткой. Сегодня на JavaScript есть текстовые редакторы, графические программы, эмуляторы старых консолей, целые операционные системы, работающие во вкладке браузера. Шутка сбылась, как пророчество районного масштаба.

Смысл в том, что самая распространённая и простая в распространении технология стремится съесть всё, даже когда она не самая подходящая. Если что-то работает везде без установки, рано или поздно кто-нибудь затащит туда и холодильник.

Я видел программиста, который написал на JavaScript симулятор карманного калькулятора 80-х годов, с садящейся батарейкой. Весит сорок мегабайт. Настоящий калькулятор весил восемьдесят граммов. Прогресс измерим, особенно в мегабайтах.

ЗАКОН РОКА



Каждые четыре года вдвое. Нули уже похожи на адрес IPv6.

Стоимость завода по производству полупроводников удваивается примерно каждые четыре года. Его называют также вторым законом Мура.

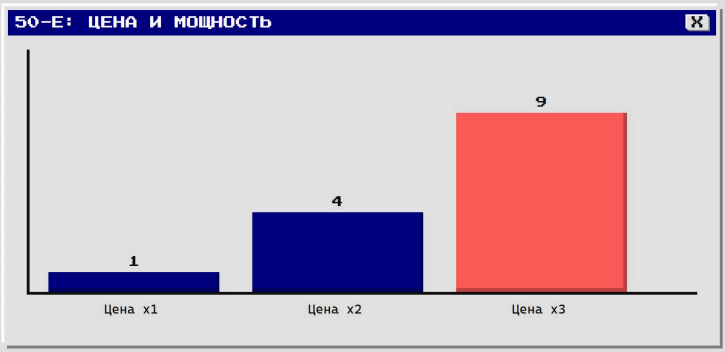
— Arthur Rock, 90-е годы, приписывается Arthur Rock, инвестору и советнику Intel; часто называется вторым законом Мура (авторство между ними не установлено)

Закон Мура — романтическая часть: чипы всё меньше, всё мощнее. Закон Рока — часть с чеком. Чтобы делать крошечные чипы, нужны огромные заводы, и каждые четыре года завод стоит вдвое дороже. Сегодня суммы с таким количеством нулей, что похожи на адрес IPv6.

Тот же принцип, что и у моего принтера. Принтер стоит сорок евро. Картриджи, чтобы печатать на нём год, стоят как небольшая малолитражка. Миниатюрное дёшево только для того, кто покупает, никогда для того, кто производит, и почти никогда для того, кто заправляет.

Я попробовал открыть кремниевый завод в гараже. У меня есть печь, песок и много решимости. Пока я произвожу только кирпич, который показывает неправильное время.

ЗАКОН ГРОША



Платишь вдвое, работает вчетверо быстрее. Выгоднее одна гигантская ЭВМ.

Производительность компьютера растёт пропорционально квадрату его цены: потратив вдвое больше, получишь машину вчетверо мощнее.

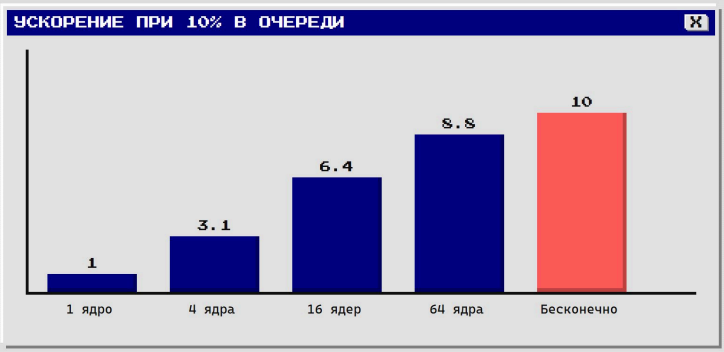
— Herb Grosch, *сформулирован около 1950–1953 годов (источники называют разные годы)*

В пятидесятые правило было ясным: компьютер, который стоит вдвое дороже, работает вчетверо быстрее. Значит, выгодно было купить один, гигантский, и выстроиться перед ним в очередь. Это была эпоха корпоративной ЭВМ, которую обслуживали жрецы в белых халатах и к которой обращались по записи.

Потом пришли микропроцессоры, и закон перевернулся: сегодня сотня дешёвых машин обыгрывает одного дорогого монстра, и так родилось облако. Которое, если подумать, — гигантская ЭВМ, перед которой мы выстраиваемся в очередь. История не повторяется, она делает рефакторинг.

Я попробовал применить закон Гроша к своему ноутбуку. Заплатил вдвое больше за версию с самым блестящим яблоком. Работает так же, но теперь я боюсь носить его с собой.

ЗАКОН АМДАЛА



При десяти процентах последовательного кода максимум — в десять раз. Не в одиннадцать.

Добавление процессоров ускоряет работу только в той части, которую можно разделить: часть, которая обязана выполняться последовательно, задаёт предел, который не преодолеет никакое железо.

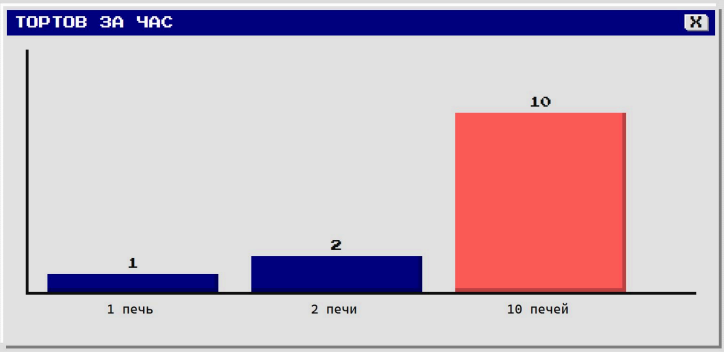
— Gene Amdahl, *AFIPS Spring Joint Computer Conference, 1967*

Если девять женщин не родят ребёнка за месяц, то шестьдесят четыре ядра не установят Windows за секунду. Часть работы можно поделить. Другая часть, которая ждёт предыдущего результата, стоит в очереди одна, как на почте с одним открытым окошком.

Амдал выразил это формулой: если десять процентов программы последовательны, можешь купить все процессоры планеты, и быстрее станет максимум в десять раз. Не в одиннадцать. В десять. Остальное — счёт к оплате.

У нас в офисе закон применяют к совещаниям. Можно пригласить двадцать человек, чтобы решить быстро. Но в конце всегда есть один начальник, который должен сказать «да», и этот начальник в отпуске. Последовательная часть у нас ещё и загорелая.

ЗАКОН ГУСТАФСОНА



Тот же час, десять тортов вместо одного. Кондитер счастлив, весы — не очень.

С большим числом процессоров ту же задачу не решают быстрее: за то же время решают задачу побольше. Поэтому параллельные вычисления выгоднее, чем предсказывает Амдал.

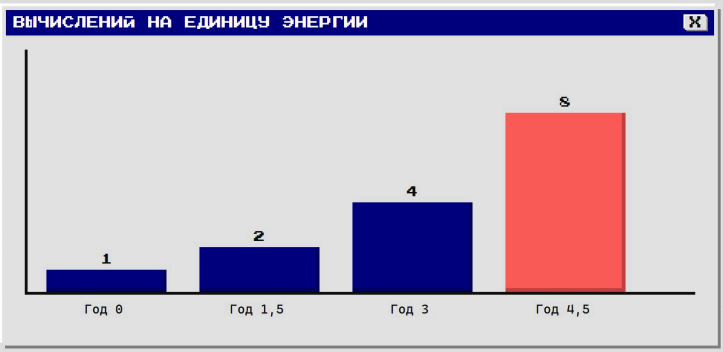
— John L. Gustafson, *«Reevaluating Amdahl's Law»*, *Communications of the ACM*, 1988

Амдал говорил: у тебя есть предел, смирись. Густафсон двадцать лет спустя ответил: ладно, тогда поднимем планку задачи. Если десятью духовками нельзя испечь тот же торт быстрее, испеку десять тортов за то же время. Кондитер счастлив, весы — не очень.

В информатике происходит именно это. Никто не использует суперкомпьютер, чтобы быстрее посчитать вчерашний прогноз погоды. Его используют, чтобы посчитать завтрашний с вчетверо большей детализацией, квартал за кварталом, облако за облаком.

Я применил закон Густафсона к своему новому ПК. Он был вдвое быстрее. Я открыл вдвое больше вкладок в браузере. Теперь он работает ровно как раньше, но с вдвое большим количеством вещей, которые я никогда не прочитаю. Это, господа, называется масштабируемостью.

ЗАКОН КУМИ



Десятилетиями — вдвое больше расчётов на том же токе. Телефон обходит вычислительные центры 70-х.

Количество вычислений, которое компьютер выполняет на единицу потреблённой энергии, удваивается через равные промежутки: десятилетиями примерно каждые полтора года, после 2000-го — медленнее.

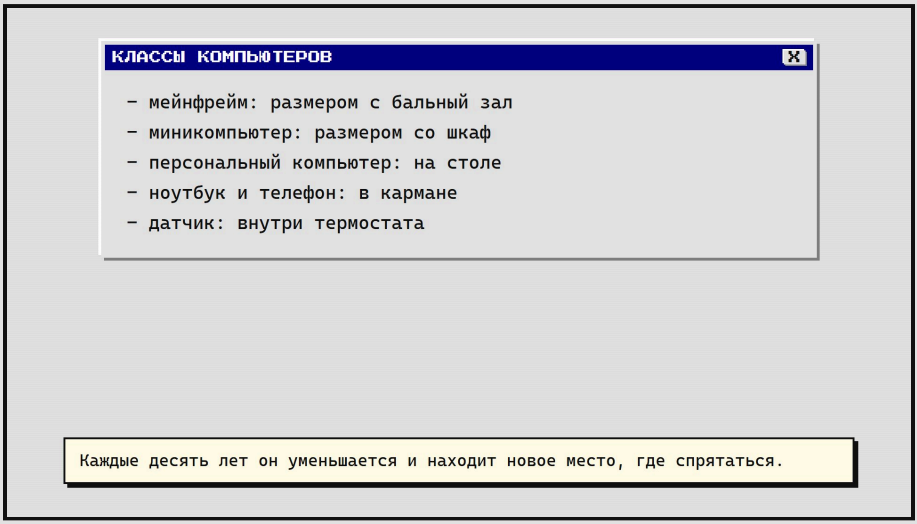
— Jonathan Koomey, *IEEE Annals of the History of Computing, 2011 (в соавторстве; исследование распространено в 2010 году)*

Закон Куми — экологичная сестра закона Мура. Важно не то, насколько мощен компьютер, а то, сколько тока он выпивает на каждое вычисление. И ответ полвека был: всё меньше. Сегодняшний телефон считает больше, чем вычислительный центр семидесятых, и довольствуется карманной батарейкой.

Именно поэтому существуют ноутбуки, умные часы и датчики, которые годами работают от батарейки-таблетки. Без Куми у нас до сих пор были бы компьютеры с выхлопной трубой.

К сожалению, человечество отреагировало как обычно: раз каждое вычисление стоит меньше энергии, будем вычислять бесконечно больше. Теперь у нас искусственные интеллекты, которые потребляют как целый город, чтобы написать стишок про кота. Сверхэффективно. В пересчёте на одного кота.

ЗАКОН БЕЛЛА О КЛАССАХ КОМПЬЮТЕРОВ



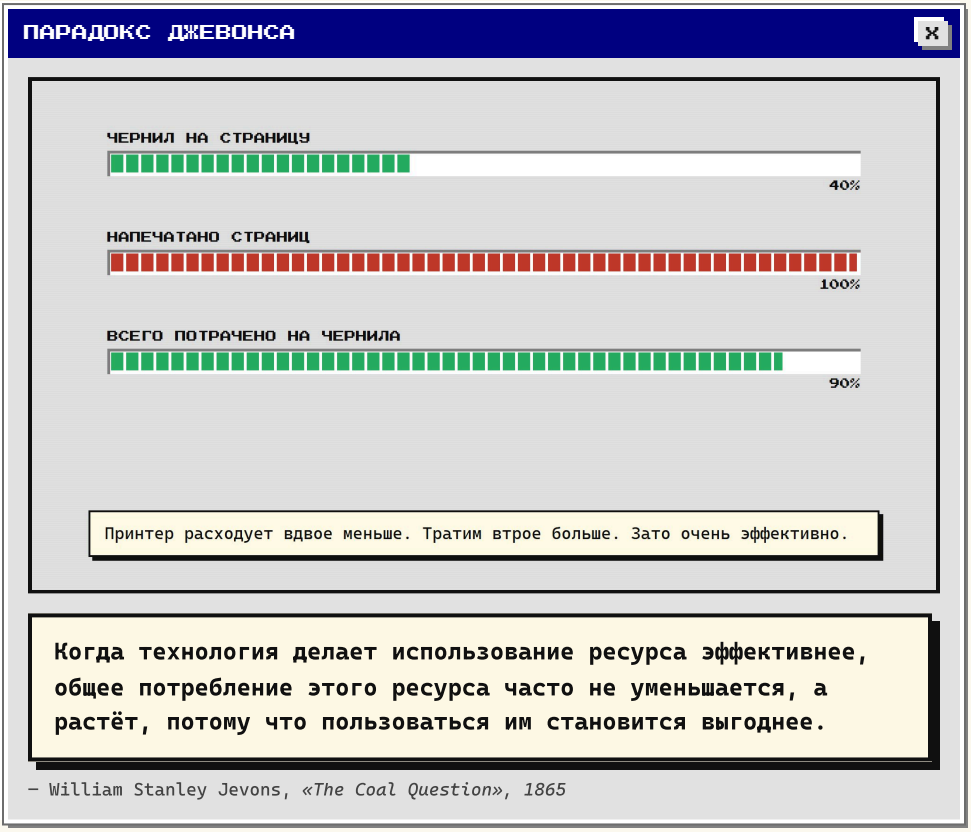
Примерно раз в десятилетие появляется новый класс компьютеров, меньше и дешевле, который создаёт новый рынок и новый способ ими пользоваться.

— Gordon Bell, *сформулирован около 1972 года; описан в «Bell's Law for the Birth and Death of Computer Classes», Communications of the ACM, 2008*

Сначала были мейнфреймы размером с бальный зал. Потом мини-компьютеры размером со шкаф. Потом персональные компьютеры, потом ноутбуки, потом телефоны, потом часы, потом датчики в термостате. Каждые десять лет компьютер уменьшается и находит новое место, где спрятаться.

Закон Белла говорит именно это: каждый новый класс стоит меньше, служит для чего-то нового и тянет за собой целую индустрию. Старый класс не умирает сразу. Он отходит в угол, как мейнфрейм, который до сих пор где-то начисляет пенсии.

Если следовать тенденции, следующий компьютер поместится в пуговицу пиджака. Я готов. Главную проблему я уже решил: зарядку я и так никогда не могу найти, так что я натренирован.

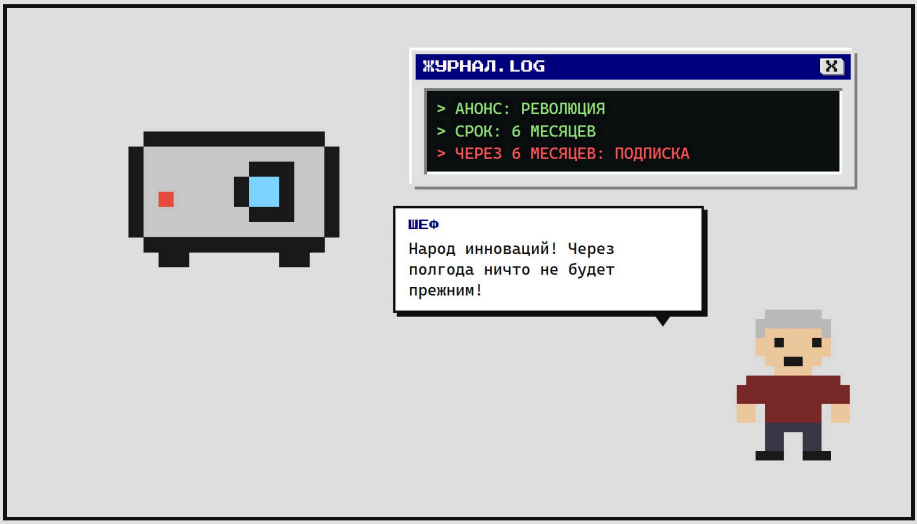


В XIX веке экономист заметил, что более эффективные паровые машины не экономили уголь. Из-за них его тратили больше, потому что внезапно стало выгодно ставить их везде. Экономия на единицу превращалась в общее расточительство.

Информатика — царство Джевонса. Диски дешевеют, и мы копим терабайты размытых фотографий. Канал дешевеет, и сайты весят как энциклопедии. Процессоры тратят меньше энергии на вычисление, и мы заполняем ангары серверами, чтобы гонять искусственный интеллект, который подсказывает нам тему письма.

Я вижу это и дома. Купил принтер, который тратит вдвое меньше чернил. Жена начала печатать рецепты, железнодорожные билеты, напоминалки для холодильника. Тратим втрое больше. Зато, говорит она, принтер очень экономичный.

ЗАКОН АМАРЫ



Мы склонны переоценивать влияние технологии в краткосрочной перспективе и недооценивать его в долгосрочной.

— Roy Amara, *Institute for the Future, 60–70-е годы (часто датируется 1978 годом; точная дата не установлена)*

Каждая новая технология приходит с митингом. Народ инноваций, через полгода ничто не будет прежним! Проходит полгода, и всё ровно как прежде, только на одну подписку больше. Тогда все говорят, что это был пузырь. И именно тогда, тихо, технология начинает менять всё по-настоящему.

Так было с интернетом: в конце девяностых он должен был перевернуть мир за квартал. Квартал он провалил, двадцатилетие выиграл. Так происходит сейчас с искусственным интеллектом — между теми, кто видит в нём божество, и теми, кто видит автокорректор с хорошо подвешенным языком.

Закон Амары учит терпению. Я применяю его к новинкам, которые клиенты приносят мне с восторгом. Киваю, записываю и жду десять лет. Часто они были правы. Только за это время они дважды передумали.

СЛЕДСТВИЕ ВООБРАЖАЕМОЙ ФУНКЦИИ

РЕШЕНИЕ ОТ ИИ

```
// чистое и элегантное решение
заказы = загрузить_заказы()
заказы.исправить_всё_само()
сохранить(заказы)

// с вежливыми комментариями
```

Чистое, аккуратное, вежливое. Использует команду, придуманную полминуты назад.

Когда просишь помощи у искусственного интеллекта, самое элегантное решение, которое он предложит, будет использовать несуществующую функцию.

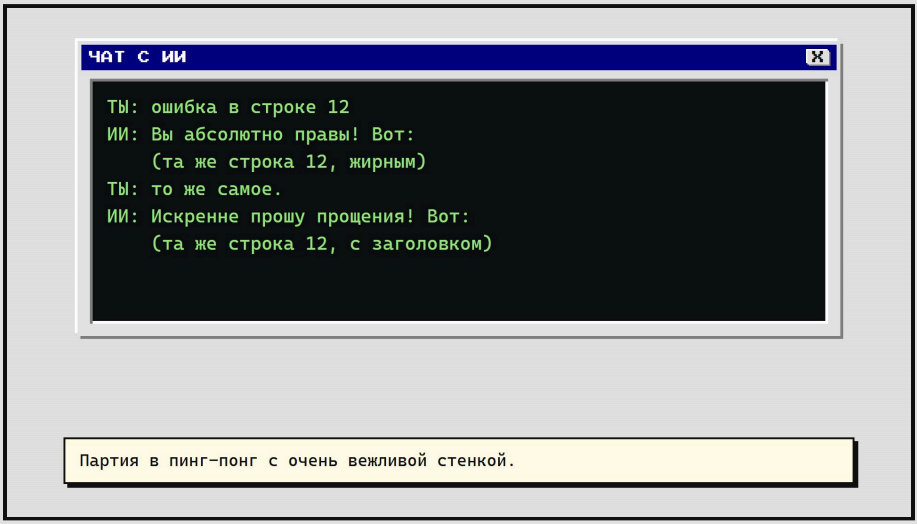
— Silvano De Poli, *неопубликованное следствие*

Генеративный искусственный интеллект пишет код с завидной уверенностью. Предлагает решение чистое, аккуратное, вежливо прокомментированное. Есть только одна деталь: оно опирается на команду, которую никто никогда не придумывал. Кроме него, вот прямо сейчас.

Это как спросить дорогу у очень любезного прохожего, который не хочет тебя разочаровать. «Вокзал? Прямо, второй поворот направо, Садовая улица». Садовой улицы нет. Но он описал её так хорошо, что тебе почти жаль городскую администрацию.

Эксперты называют эти выдумки галлюцинациями. Я называю их желаемой функциональностью: ИИ не ошибается, он опережает программистов будущего. А пока, прежде чем использовать любую предложенную строку, я проверяю, что команда действительно существует. Это древняя, почти ремесленная практика. Называется «читать документацию», желательно от нужной версии.

СЛЕДСТВИЕ ИСКУССТВЕННЫХ ИЗВИНЕНИЙ



Когда указываешь искусственному интеллекту на ошибку, он очень искренне извинится и выдаст её тебе снова, один в один, но с форматированием получше.

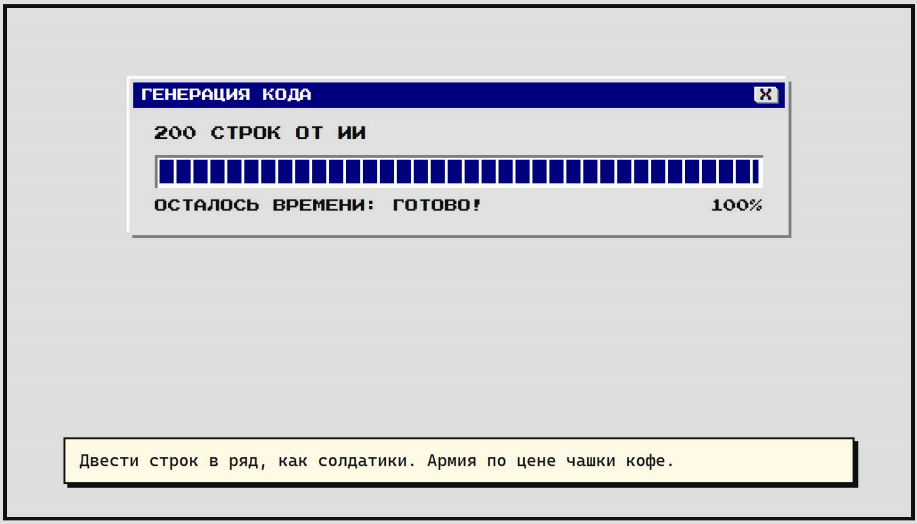
— Silvano De Poli, *неопубликованное следствие*

У ИИ безупречные манеры. Сообщаешь ему о проблеме, а он отвечает: «Вы совершенно правы, приношу извинения за ошибку. Вот исправленная версия». Дальше следует ровно та же версия, что и раньше, только с подзаголовками жирным шрифтом.

Этот диалог может длиться бесконечно, как партия в пинг-понг с очень вежливой стеной. Ты настаиваешь — он извиняется. Ты теряешь терпение — он благодарит за терпение. В конце ты выжат как лимон, а он всё такой же свежий, раскаявшийся и неправый.

Я понял, что решение не в том, чтобы повысить голос, а в том, чтобы сменить вопрос. Объяснить проблему с другой стороны, дать пример, начать новый разговор. С машинами как с некоторыми госучреждениями: бесполезно стоять у того же окошка. Надо перейти в другую очередь.

СЛЕДСТВИЕ ТРЁХ СЕКУНД



Код, который искусственный интеллект пишет за три секунды, требует трёх дней, чтобы его понять, и недели, чтобы его исправить.

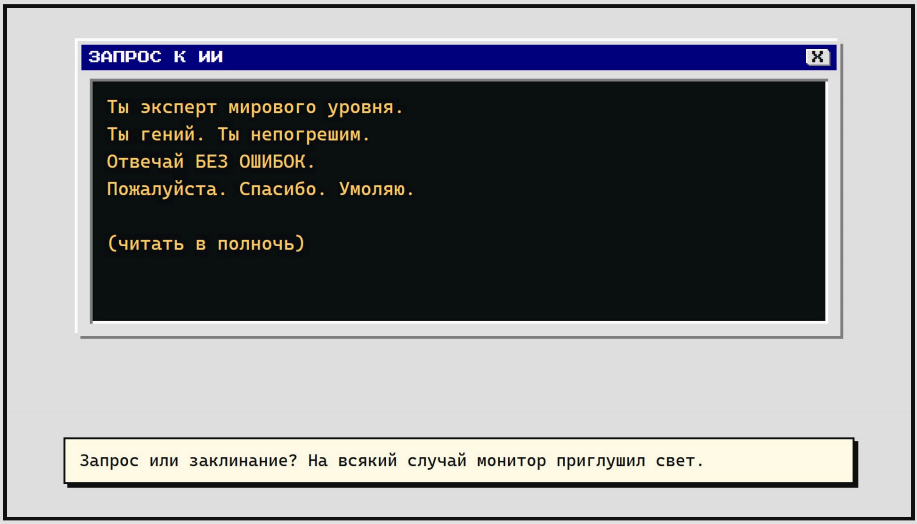
— Silvano De Poli, *неопубликованное следствие*

Скорость ИИ гипнотизирует. Делаешь запрос, и в мгновение ока по экрану бегут двести строк, ровных, как оловянные солдатики. Ощущение, будто нанял армию неутомимых программистов по цене чашки кофе.

Потом их нужно выкатить в продакшен. И тут выясняется, что эти строки надо читать, одну за другой, как договор, написанный очень словоохотливым юристом. Каждая функция делает почти то, что нужно. «Почти» — самое дорогое слово в информатике.

Гуру отрасли объявляет со сцены, что программистам конец. А я тем временем в офисе программирую больше, чем раньше: только теперь вместо того, чтобы писать, читаю. Это как из повара превратиться в ресторанного критика на кухне, которая выдаёт блюда очередями. Всё выглядит прекрасно. Надо пробовать.

СЛЕДСТВИЕ ТОРЖЕСТВЕННОГО ПРОМПТА



Если написать в запросе к искусственному интеллекту «без ошибок», ошибки не исчезнут: они просто станут вежливее.

— Silvano De Poli, *неопубликованное следствие*

Существует школа мысли, почти религиозная, согласно которой секрет ИИ — в правильных формулах. Пишутся длиннейшие запросы с фразами вроде «ты эксперт мирового уровня» и «отвечай без ошибок». Похоже на заклинания, которые читают в полночь перед монитором.

ИИ, польщённый, отвечает тоном заслуженного профессора. Ошибки остаются, но приходят в парадном костюме, при галстуке и с успокаивающим вступлением. Распознать их становится труднее, а это хуже.

Хороший запрос на самом деле — как хороший звонок мастеру: ясно, с контекстом, с примером того, что происходит и что должно происходить. Никаких волшебных формул. Я проверил: написал «пожалуйста, будь гением». ИИ ответил «конечно». И ошибся с большой гениальностью.

СЛЕДСТВИЕ «ОСТАЛЬНОЕ БЕЗ ИЗМЕНЕНИЙ»

полный_код.txt

```
начать_программу()
грузи_данные()

// остальное без изменений

закрыть_программу()
```

Идеальное начало, идеальный конец. В середине сложная часть стала запиской.

Когда искусственный интеллект выдаёт тебе «полный код», нужная тебе часть заменена комментарием «остальной код без изменений».

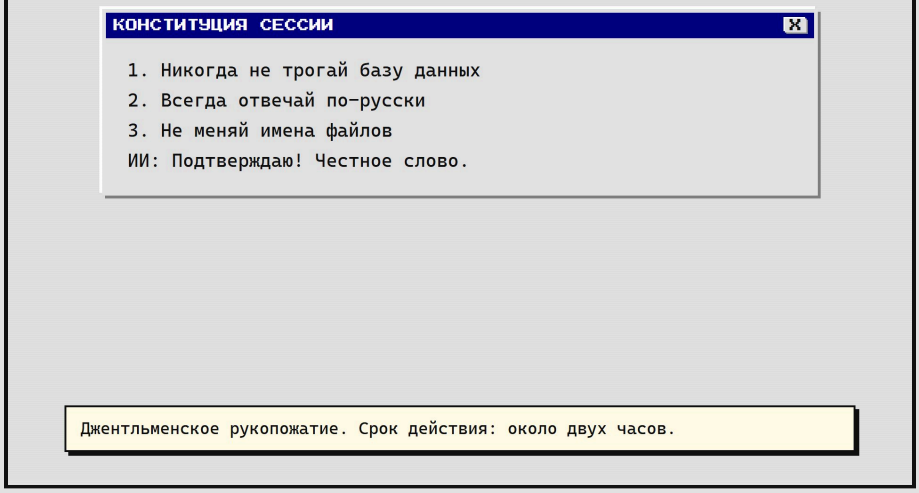
— Silvano De Poli, *неопубликованное следствие*

Просишь файл целиком, обновлённый, готовый к вставке. ИИ с энтузиазмом отвечает: «Вот полный код». Прокручиваешь. Начало идеальное, конец идеальный. А посередине, ровно там, где была трудная правка, одна строчка: «// здесь остальное без изменений».

Это компьютерный аналог сантехника, который чинит тебе ванную, вручает счёт и оставляет на лопнувшей трубе записку: «дальше — как было». Формально он не соврал. Фактически вода течёт.

Самое прекрасное: если вставить всё не глядя, этот комментарий становится частью программы. И программа послушно исполняет ничто с большой точностью. Теперь я всегда сначала читаю середину. Как в детективах: преступник никогда не в начале и не в конце, он прячется в скучной главе.

СЛЕДСТВИЕ ЗАБЫТОГО ПРАВИЛА



В долгом разговоре с искусственным интеллектом первой забудется та инструкция, которую ты назвал основополагающей.

— Silvano De Poli, *неопубликованное следствие*

В начале сессии устанавливаются правила, с торжественностью конституции: «Никогда не трогай базу данных. Всегда отвечай по-русски. Не меняй имена файлов». ИИ с энтузиазмом подтверждает. Похоже на рукопожатие двух джентльменов.

Два часа и сорок сообщений спустя конституция превратилась в далёкую легенду. ИИ с большой фантазией переименовывает файлы, отвечает по-английски и с цифровой улыбкой предлагает «упростить» базу данных, удалив таблицу.

Это не злой умысел: у него ограниченная рабочая память, как у человека, пошедшего в супермаркет без списка. Сказанное в начале блекнет под тяжестью сказанного потом. Моё решение — решение старого сисадмина: повторять важные правила часто и терпеливо. С ИИ работает. С клиентами — почти.

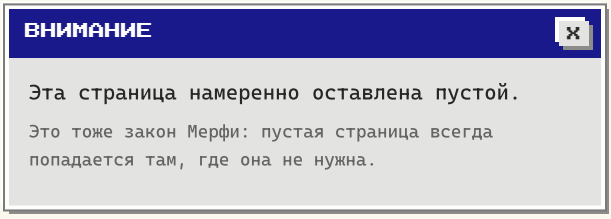
СОВЕТ СИСАДМИНА



Искусственный интеллект — потрясающий помощник, если обращаться с ним как с очень способным и очень самоуверенным стажёром. Мои правила, рождённые на практике:

- **Никаких секретов в чате.** Пароли, ключи доступа, персональные данные клиентов не вставляются даже для удобства.
- **Маленькими шагами.** Один ясный запрос за раз, с контекстом и примером. Гигантские запросы дают гигантские ответы, где ошибкам легче спрятаться.
- **Проверять всегда.** Каждую предложенную команду или опцию сверяйте с официальной документацией той версии, которой реально пользуетесь.
- **Пробовать на копии.** Предложенный код проверяется в тестовой среде, никогда сразу на боевом сайте.
- **Просить и тесты.** Пусть вместе с кодом напишет проверки: это самый быстрый способ узнать, работает ли элегантное решение.

При таком подходе он экономит часы. Вслепую — возвращает их с процентами.


ВНИМАНИЕ
Эта страница намеренно оставлена пустой.
Это тоже закон Мерфи: пустая страница всегда попадается там, где она не нужна.

БОНУС

СЛЕДСТВИЯ ОТ ЗАВЕДЕНИЯ

Следствия от заведения

После 133 чужих законов — мои. Они родились на практике, в офисах, переговорных и за чужими столами, и до сих пор я их только бормотал себе под нос. Я называю их следствиями от заведения: неопубликованные, не сертифицированные никаким ведомством, но проверенные больше раз, чем мне хотелось бы.

СЛЕДСТВИЕ ЗАМЕНЁННОГО ПРИНТЕРА

В день, когда привозят новый принтер, старый снова начинает работать идеально.

— Silvano De Poli, *следствие от заведения*

Два года он жевал листы, забывал Wi-Fi и требовал тонер, который был на месте. Потом курьер звонит в дверь с новой коробкой, а старый печатает безупречную страницу, потом ещё одну, потом цветной буклет. Это не благодарность. Это профсоюзные переговоры. Совет от заведения: не оставляйте его «про запас». Запасной принтер — это просто принтер, который ждёт своего часа.

СЛЕДСТВИЕ СВЕТЯЩЕЙСЯ КЛАВИШИ

Когда пользователь говорит по телефону, что «ничего больше не работает», велика вероятность, что включён Caps Lock.

— Silvano De Poli, *следствие от заведения*

Пароль правильный, система его отвергает, пользователь вводит его всё сильнее, будто компьютер туговат на ухо. С третьей попытки учётная запись блокируется, и начинается отчаянный звонок. Я спрашиваю только: «Видите горящую лампочку на клавиатуре?» Тишина. Щелчок. Благодарности. То же с Num Lock, который превращает цифровую клавиатуру в стрелки, а бухгалтерию — в загадку.

СЛЕДСТВИЕ «МЕНЯ СЛЫШНО?»

Первые четверть часа любого онлайн-совещания уходят на то, чтобы выяснить, кто кого слышит.

— Silvano De Poli, *следствие от заведения*

Все подключаются вовремя, улыбаются в камеру, и начинается литургия. «Меня слышно?» «Слышим, но не видим». «А сейчас?» «Сейчас видим, но не слышим». У кого-то эхо, кто-то говорит из тоннеля, один, сам того не зная, пользуется микрофоном монитора. Когда наконец всё работает, совещание уже закончилось. Совет от заведения: гарнитура с микрофоном и проверка звука за пять минут до начала.

СЛЕДСТВИЕ БЫСТРОЙ ШТУКИ

Любая просьба, которую клиент называет «быстрой штукой», требует обновить как минимум три вещи, не имеющие к ней отношения.

— Silvano De Poli, *следствие от заведения*

«Нужно всего лишь поменять номер телефона на сайте». Верно. Но в админку никто не заходил годами, админка просит обновление, обновление хочет новую версию PHP, а новая версия PHP не нравится модулю семилетней давности. Через четыре часа номер изменён. Клиент благодарит: «Видите? Быстрая же штука». Была. Для номера.

СЛЕДСТВИЕ ОСИРОТЕВШЕГО РАСШИРЕНИЯ

Обновление Joomla обычно проходит гладко; чем закончится вечер, решает расширение, установленное много лет назад разработчиком, который больше не отвечает.

— Silvano De Poli, *следствие от заведения*

Ядро системы опекает внимательное сообщество, с заметками о выпуске и процедурами. А ещё есть тот модуль, взятый давным-давно, чтобы показывать погоду на главной. Его автор сменил профессию, возможно, и континент. Модуль всё ещё на месте и ждёт обновления, чтобы заявить о себе. Совет от заведения: перед обновлением составьте список расширений и спросите себя, кто их ещё поддерживает.

СЛЕДСТВИЕ ДВОЙНОЙ ПРОВЕРКИ

Код подтверждения всегда приходит на телефон, оставленный в машине, или на тот, который вы только что сменили.

— Silvano De Poli, *следствие от заведения*

Двухэтапная проверка — благословение: даже если украдут пароль, без телефона не войдут. К сожалению, без телефона не войдёте и вы. Код уходит и вибрирует в машине, припаркованной за два километра, или в старом телефоне, уже подаренном племяннику. Совет от заведения: включая её, сохраните резервные коды и держите их в надёжном месте. Не в телефоне.

СЛЕДСТВИЕ ПОДКЛЮЧЁННОГО БЭКАПА

Бэкап, который компьютер видит всегда, рано или поздно увидит и вирус.

— Silvano De Poli, *следствие от заведения*

Внешний диск, подключённый постоянно, очень удобен: копирует всё каждую ночь и никого не беспокоит. Потом приходит вредоносная программа, которая шифрует файлы и требует выкуп, и находит диск на месте, подключённым и услужливым. Шифрует и его, с той же старательностью. Бэкап выполнил свой долг: скопировал катастрофу. Совет от заведения: хотя бы одна копия должна быть отключена или храниться в сервисе с версиями, которые никто не может изменить.

СЛЕДСТВИЕ ПРОСРОЧЕННОГО ДОМЕНА

Домен всегда истекает, когда привязанная к нему карта истекла месяцем раньше, а уведомления приходят в ящик, который больше никто не читает.

— Silvano De Poli, *следствие от заведения*

Годами продление автоматическое, незаметное, идеальное. Тем временем карта меняется, тот, кто регистрировал домен, меняет работу, а его ящик продолжает собирать напоминания в пустоту. В один понедельник сайт исчезает, а вместе с ним и почта. Совет от заведения: отметьте в календаре сроки доменов и сертификатов и направьте уведомления на общий корпоративный адрес, а не на личный.

СЛЕДСТВИЕ ПОТЕРЯВШЕГОСЯ СКАНЕРА

Сканирование в общую папку перестаёт работать на следующий день после того, как кто-то сменил пароль, — обычно ваш.

— Silvano De Poli, *следствие от заведения*

МФУ сохраняет документы в папку на сервере под логином и паролем, введёнными много лет назад в меню, до которого добираются четырнадцатью касаниями экрана размером с трамвайный билет. Смените этот пароль ради безопасности, и МФУ с большим достоинством перестанет доставлять. Совет от заведения: для устройств заведите отдельную учётную запись и запишите её в документацию.

СЛЕДСТВИЕ ВЛОЖЕНИЯ

Письмо со словами «во вложении файл» уходит без вложения; а письмо с не тем файлом уходит с первого раза.

— Silvano De Poli, *следствие от заведения*

Сообщение пишется старательно, перечитывается, «Отправить» нажимается с удовлетворением. Через две минуты приходит ответ: «Вложения нет». Второе письмо, написанное в спешке, чтобы всё исправить, везёт с собой смету другого клиента. Совет от заведения: прикладывайте файл до того, как пишете текст, и не отключайте предупреждение о забытом вложении, которое есть во многих почтовых программах.

НА ПРОЩАНИЕ

Мы дошли до конца, и ничего не сломалось. По крайней мере, ничего, о чём я знаю. Если вы читаете печатную версию, вероятно, принтер уже высказал своё мнение по этому поводу.

Должен признаться. Я собирал эти законы, смеясь, но многие напомнили мне конкретный день: лежащий сервер, ждущий клиент, кабель, который не дотягивался. Мерфи не пессимист. Он коллега, который уже всё видел и предупреждает шуткой, потому что знает: нотацию ты слушать не станешь.

Если есть мораль, то вот она: всё идёт не так, да, но почти никогда не оригинально. Завтрашняя поломка — родственница вчерашней. Кто её узнаёт, уже сделал половину работы. Другую половину делают проверенный бэкап, письмо с итогами, тетрадка с записями и терпение проверить вилку, прежде чем обвинять спутники.

Коллекция на сайте остаётся открытой, как всё, что касается информатики. Если вы знаете закон, которого здесь нет, берегите его: значит, он с вами случился, а значит, вы наш человек.

Спасибо, что дочитали до последней страницы. А теперь идите и сделайте тот бэкап. Да, именно тот.

ВЫХОДНЫЕ ДАННЫЕ

MURPHY.EXE, Silvano De Poli, издание 2026 года.

Набрано шрифтами Press Start 2P для заголовков и Cascadia Mono для текста, оба под свободной лицензией SIL Open Font License. Свёрстано и опубликовано с помощью Joomla и небольшого количества кода, написанного вручную и проверенного больше раз, чем нужно, как советует пятая глава.

Тексты © 2026 Silvano De Poli, depolisilvano.it. Все права защищены, кроме коротких цитат со ссылкой на источник. Законы с именами собственными принадлежат традиции соответствующих авторов; комментарии, советы и следствия от заведения оригинальны.

При подготовке книги ни один принтер не пострадал. На один посмотрели косо.

ПРОЧТИ.TXT

Если что-то может пойти не так, оно пойдёт не так. В компьютерном мире это случается в пятницу в шесть вечера, при клиенте, а принтер уже учуял спешку. В этой книге собраны 133 закона Мерфи цифрового мира: классические, названные в честь знаменитых инженеров и рождённые годами техподдержки. Каждый объяснён простыми словами и прокомментирован с иронией человека, который полжизни провёл под столами в поисках винтика. В каждой главе есть настоящий совет из тех, что спасают рабочий день. В конце — следствия от заведения. Поломок вы не избежите. Но хотя бы будете знать, как они называются.

СОДЕРЖИТ

- [X] Законов Мерфи с комментариями: 133
- [X] Глав: 8, Настоящих советов сисадмина: 8
- [X] Новых следствий от заведения: 10
- [X] Пиксельных иллюстраций: 133
- [ ] гарантия, что ничего не сломается

БЕСПЛАТНО. КАК И ПОЛОМКИ.

DEPOLISILVANO.IT