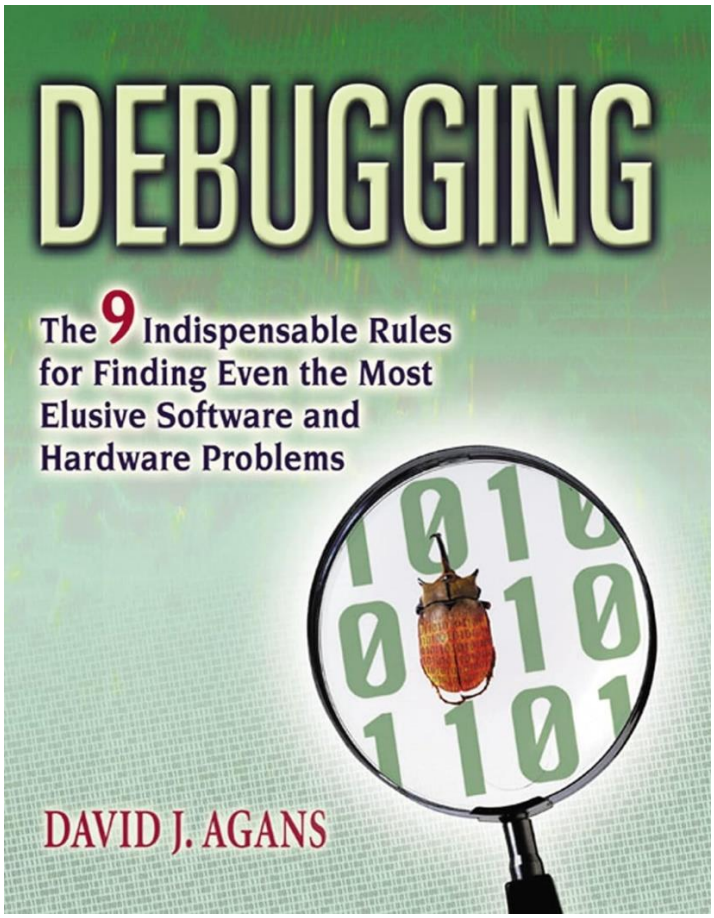
Дэйв Дж. Аганс

Отладка

Девять незаменимых правил для обнаружения самых неуловимых ошибок в ПО и «железе»



Версия перевода: 1.0 (27.02.2024)

https://oscat.ru/

Оглавление

От переводчика

В любой системе рано или поздно обнаруживаются ошибки. При удачном стечении обстоятельств далее происходит поиск и устранение их причин. В идеальном случае после устранения причины ошибки выполняется тестирование для подтверждения того, что ошибка действительно больше не проявляется, а также анализ других компонентов системы (или других систем) на возможность наличия этой же ошибки, и документирование деталей произошедшей ситуации. Весь этот процесс в целом называется ***отладкой***.

Про то, как создавать программы, написаны тысячи книг. Про разработку аппаратного обеспечения – меньше, но всё равно не так уж и мало. Количество книг про отладку измеряется единицами. К этим редким творениям относится книга Дэйва Аганса. Она написана ещё в конце 90-х, но поскольку перечисленные в ней фундаментальные правила не менялись с незапамятных времён – то содержимое книги по-прежнему предельно актуально. Фундаментальность этих правил я испытал на собственном опыте – когда-то несколько лет назад, ещё до прочтения труда господина Аганса, я написал статью про отладку проектов в среде CODESYS V3.5 – и существенная часть сформулированных в ней рекомендаций совпала с правилами, которые вы увидите в главе 2.

Отдельным бриллиантом являются рассказываемые Дэйвом тут и там «байки ветеранов» – короткие истории о том, как с помощью правил книги решались совершенно разные проблемы из различных предметных областей.

Если вам будет интересно почитать что-то ещё про отладку ПО – то я рекомендую классический труд *Брайан Керниган, Роб Пайк. Практика программирования* (глава 5 посвящена отладке) и *Питер Сейбл. Кодеры за работой* (Сейбл взял интервью у 16 известных программистов и, в том числе, обсудил с ними их методы отладки).

Я хочу выразить наиглубочайшую признательность Владиславу Зинько, который бесчисленное множество раз разделял со мной сеансы отладки, вычитывал мои документы и оказывал другую помощь и поддержку как по рабочим, так и по личным вопросам. Также я благодарен Олегу и Владу (другому), у которых многому научился (и планирую научиться ещё большему) и которые стали ангелом и демоном на моих плечах. Спасибо тем[1], кто помог с вычиткой – находил опечатки, неточности, ошибки и неудачные формулировки. Отдельное спасибо Юлии Шляковой, которая самоотверженно перерисовала все иллюстрации оригинальной книги в хорошем качестве.

Переводчик: Евгений Кислов

[1] В алфавитном порядке:
Александр Пинэко-Скворцов
Михаил Троицкий
Электрошаман (Cs-Cs)

Моей маме, Рут (Уорсли) Аганс, которая отлаживала программы на Фортране с помощью ручки и листа бумаги за нашим обеденным столом, поддерживая себя бесконечными чашками крепкого кофе.

И моему отцу, Джону Агансу, который научил меня думать, руководствоваться здравым смыслом и смеяться.

Ваши души со мной во всех моих начинаниях.

Благодарности

Эта книга появилась на свет в 1981 году, когда во время работы в компании Gould я получил вопрос из отдела тестирования – смогу ли написать руководство по устранению неполадок в наших устройствах. Я растерялся – каждая выпускаемая нами плата имела сотни микросхем, несколько микропроцессоров и множество коммуникационных шин. Я знал, что никакого волшебного рецепта не существует; сотрудникам отдела просто нужно было научиться заниматься отладкой. Я обсудил это с Майком Бромбергом, который долгое время был моим наставником, и мы решили, что, по крайней мере, можем составить список некоторых общих правил. Результатом этой работы стали «Десять правил отладки» – единственный лист, который мы повесили на стене рядом с испытательными стендами. С годами он лишился одного из правил, и был отредактирован таким образом, чтобы подходить как для аппаратного, так и программного обеспечения; именно этот список лёг в основу данной книги. Так что спасибо Майку и тестировщикам, по запросу которых он был написан.

На протяжении многих лет я был счастлив работать с рядом вдохновляющих меня людей, которые помогли мне развить как навыки отладки, так и чувство юмора. Я хотел бы выразить признательность Дугу Карри, Скотту Россу, Глену Дэшу, Дику Морли, Майку Гринбергу, Косу Фрикано, Джону Эйлсворту (одному из инженеров того самого отдела тестирования компании Gould), Бобу ДеСимоуну и Уоррену Байеку за то, что они добавили в тяжёлые трудовые будни множество весёлых минут. Я должен также упомянуть трёх моих учителей, которые полностью отдавались своей работе и сделали обучение максимально комфортным: Ника Менутти (пусть это и не Нобелевская премия, а всего лишь слова благодарности), Рэя Филдса и профессора Фрэнсиса Ф. Ли. Также я упомяну авторов книг, с которыми я никогда не встречался, но которые оказали огромное влияние на мою писательскую карьеру: Уильяма Странка-младшего и Э. Б. Уайта («Элементы стиля»), а также Джеффа Хермана и Дебору Адамс («Напишите идеальное предложение для издателя»).

Спасибо «Delt Dawgs», моей летней команде по софтболу с 28-летней историей (которая продолжается) за отзывы и помощь в налаживании полезных связей. Я в долгу перед Чарли Седдоном, который написал подробную рецензию на черновик этой книги со множеством полезных комментариев, и Бобу Сиденстикеру, который тоже написал рецензию, а ещё предоставил мне несколько «баек ветеранов», предложил темы для ряда разделов и дал советы по издательскому бизнесу. Несколько человек, большинство из которых я в то время ещё не знал лично, написали отзывы на книгу и прислали мне хорошие рекомендательные письма, которые помогли её опубликовать. Уоррен Байек и Чарли Седдон (упомянутые выше), Дик Райли, Боб Оукс, Дэйв Миллер и профессор Терри Симкин – спасибо за потраченное вами время и слова поддержки.

Я благодарен «Мастерской Сезам», Тому и Рэю Мальоцци («Клику» и «Клаку» (или «Клаку» и «Клику»?) из радиопередачи Car Talk) и Стиву Мартину за разрешение использовать их истории и шутки; сэру Артуру Конан Дойлу за создание Шерлока Холмса и за то, что он одарил его столькими подошедшими для книги цитатами; а также Сеймуру Фриделю, Бобу Макилвейну и моему брату Тому Агансу за некоторые из «баек ветеранов». Спасибо всем участникам «баек» (как героям, так и глупцам – вы сами знаете, кто есть кто) за то, что помогли мне с примерами, на которых я поясняю правила, изложенные в этой книги.

Работа с моими редакторами в Amacom была замечательным и поучительным опытом. Джеки Флинн и Джим Бессент – спасибо вам за энтузиазм и ценные советы. А художников и других творческих людей, принявших участие в этом процессе, хочу поблагодарить за отличную работу – получилось просто великолепно.

Особая благодарность моему литературному агенту Джоди Роудс за то, что она рискнула дать шанс писателю-новичку с необычным подходом к незнакомой ей теме. Она знает своё дело, и это видно.

За поддержку, ободрение и бесчисленные одолжения (большие и малые) особая благодарность моим родственникам, Дику и Джоан Блэгбро. Моих дочерей, Джен и Лиз, обнимаю и целую за то, что они веселились и верили в меня (а ещё за то, что позволяли мне поиграть за компьютером по вечерам в перерывах между их хардкорными зарубами и общением в мессенджерах).

И, наконец, моя вечная любовь и благодарность моей жене Гейл за то, что она вдохновила меня превратить список правил в эту книгу, за то, что помогла мне начать поиск литературного агента, за то, что предоставила мне время и пространство для писательства, а также за корректуру многочисленных черновиков, которые я не осмелился бы показать кому-либо ещё. Ты можешь зажечь люстру пылесосом, но сама по себе являешься маяком моей жизни.

Дэйв Аганс, июнь 2002

Об авторе

Дэйв Аганс — выпускник Массачусетского технологического института (выпуск 1976 г.), который работал как в крупных компаниях (Gould, Fairchild и Digital Equipment), так и небольших стартапах (в том числе, Eloquent Systems и Zydacron), а также выступал в роли независимого консультанта для различных клиентов. Он проявил себя на «обычных» инженерных должностях (как инженер, менеджер проекта, менеджер по продукту, технический писатель, докладчик на семинарах, консультант и менеджер по продажам), после чего занимал должности системного архитектора, директора по разработке программного обеспечения, вице-президента по инжинирингу и главного технического директора.

Мистер Аганс разрабатывал интегральные схемы, видеоигры, системы управления для промышленности, устройства климат-контроля, системы управления отелями, рабочие станции для CAD, карманные компьютеры, беспроводные диспетчерские терминалы и системы видеоконференцсвязи. Он является обладателем двух патентов США. Что касается хобби – Дэйв увлекается созданием музыкальных пьес и написанием текстов песен.

В настоящее время Дэйв проживает в Амхерсте, штат Нью-Гэмпшир, со своей женой, двумя дочками и (периодически, когда они соизволят зайти в дом) двумя кошками. В своё редкое свободное время он занимается созданием музыкальных пьес, софтболом, баскетболом, а также писательством.

Глава 1: Вступление

Сейчас, как видите, я очень занят, а на склоне лет я собираюсь написать руководство, в котором сосредоточится всё искусство раскрытия преступлений.

Шерлок Холмс, «Убийство в Эбби-Грейндж»

1.1. Общий обзор

Из этой книги вы узнаете, как быстро выяснить причину какой-нибудь технической проблемы. Она короткая и весёлая, потому что и должна быть такой – ведь если вы инженер, то отладка занимает бóльшую часть ваших суток, и свободного времени остаётся разве что на комиксные стрипы. А даже если вы не инженер – то всё равно часто сталкиваетесь с тем, что какая-то штука сломалась и вам нужно придумать, как её починить.

Возможно, вам никогда не потребуется заниматься отладкой. Может, вы успели продать акции доткомов после того, как они вышли на IPO, но ещё не успели обанкротиться, и у вас есть целый штат сотрудников, которым просто нужно отдать указания. Может быть, вы очень удачливы, и ваши устройства и ПО работают без ошибок – или, что ещё менее вероятно, ошибки в них всегда обнаруживаются быстро и устраняются просто. Но есть вероятность, что и в ваших продуктах, и в продуктах ваших конкурентов есть трудноуловимые ошибки, и тот, кто исправит их первым, получит преимущество на рынке. Когда вы можете быстро находить ошибки, то это не только позволяет быстрее предоставлять клиентам качественные продукты, но и раньше возвращаться домой, чтобы провести время со своими близкими.

Так что положите эту книгу на прикроватную тумбочку или в уборную, и через две недели вы станете звездой отладки.

1.2. Почему эта книга работает?

Как может такое короткое и лёгкое чтиво быть полезным? Что ж, за свои двадцать шесть лет опыта проектирования и отладки систем я обнаружил две вещи (если не считать такие откровения, как «первая чашка кофе в кофейнике содержит весь кофеин»):

- когда мы тратили много времени, чтобы найти ошибку, это происходило потому, что мы пренебрегли каким-то важным, фундаментальным правилом; применив это правило, мы быстро обнаруживали проблему.
- те, кто преуспели в быстрой отладке, изначально интуитивно понимали и применяли эти правила. Людям, которым требовались усилия, чтобы осмыслить их, приходилось затрачивать существенно больше времени.

Я составил список этих основных правил. Я обучал им других инженеров и наблюдал, как растут их навыки и скорость отладки. Они *действительно* работают.

1.3. Но разве всё это не очевидно?

Прочитав эти правила, вы, возможно, скажете себе: «Но ведь это же всё и так очевидно». Не спешите; эти тезисы *очевидны* (фундаментальные принципы и должны такими быть), но то, как применить их для поиска конкретной проблемы, не всегда столь очевидно. И не путайте очевидное с простым — этим правилам не всегда легко следовать, и поэтому, когда начинается самая жара, ими часто пренебрегают.

Главное – *запомнить* их и *применять*. Если бы это было так легко и просто, мне не пришлось бы постоянно напоминать их инженерам, и у меня не было бы нескольких десятков баек ветеранов о том, что произошло, когда мы этого не сделали. Трудно найти специалистов по отладке, которые интуитивно используют эти правила. Мне нравится спрашивать кандидатов на собеседовании: «Каких эмпирических правил вы придерживаетесь при отладке?» Удивительно, но многие отвечают: «Правила тут не подходят, это же искусство». Отлично! Пусть Пикассо отлаживает наш алгоритм обработки изображений. Придерживаясь простых путей и полагаясь на своё «искусство», вы не сможете быстро найти источник проблемы.

Эта книга содержит «очевидные» принципы; она помогает вам запомнить их, понять их преимущества и научиться их применять, чтобы вы могли устоять перед искушением срезать путь и провалиться в кроличью нору. Таким образом искусство отладки превращается в науку.

Даже если вы уже хороши в отладке – эти правила помогут вам стать ещё лучше. Когда опытные инженеры читали ранний вариант этой книги, то часто упоминали следующее: помимо изучения одного или двух правил, которые они ещё не использовали (но теперь планируют делать это в будущем), книга помогла им «разложить по полочкам» правила, которыми они раньше пользовались интуитивно. Друзья-тимлиды (те, кто хороши в отладке, конечно, достигают высоких должностей) сказали мне, что эта книга дала им правильные слова, с помощью которых они могут передать свои навыки и опыт другим членам их команд.

1.4. Кто угодно может научиться этому

Хотя в тексте книги я называю своего читателя (то есть вас) «*инженером*», приведённые в ней правила будут полезны и для тех, кто не считает себя таковым. Она поможет вам, если вы участвуете в поисках неполадок – независимо от того, являетесь ли вы инженером, программистом, техническим специалистом, сотрудником службы технической поддержки или консультантом.

Если вы не принимаете непосредственного участия в процессе отладки, но являетесь руководителем сотрудников, которые ей занимаются – то можете передать эти правила им. Вам даже не надо разбираться в деталях систем и инструментов, с которыми работают ваши подчинённые: правила отладки фундаментальны, поэтому после прочтения этой книги даже немного сумасбродный менеджер сможет помочь своим гораздо более умным коллегам-инженерам как можно быстрее найти источник проблемы.

Если вы преподаватель, то вашим ученикам понравятся «байки ветеранов», которые помогут им узнать, как устроен реальный мир. И когда они столкнутся с ним – то будут готовы, в отличие от их более опытных, но не обученных отладке конкурентов.

1.5. Эта книга поможет отладить всё, что угодно…

Эта книга посвящена общим аспектам отладки; в ней не рассматриваются конкретные проблемы, инструменты, устройства и языки программирования. Речь пойдёт об универсальном подходе, который поможет вам решить любую проблему с любым устройством, программируемом на любом языке, используя для этого любые доступные вам инструменты. Это принципиально новый подход к вопросам отладки – например, вместо того, чтобы объяснить вам, как включить триггер на цифровом логическом анализаторе «Глюколов», я расскажу, *почему* в принципе стоит воспользоваться анализатором, даже если это потребует существенных усилий.

Неважно, в чём причина вашей проблемы – в ошибках проектирования, разработки, эксплуатации или просто где-то что-то сломалось. Описанные в книге правила помогут вам быстро докопаться до её сути.

Область применения этих правил не ограничивается инженерными задачами, хотя именно в их рамках они были сформулированы. Правила помогут вам разобраться в причинах проблем с автомобилями, домами, стереоаппаратурой, сантехникой и здоровьем (в книге приведены примеры по всем этим вопросам). Следует признать, что в некоторых областях данные правила не особо эффективны – например, экономические проблемы являются слишком комплексными. А в некоторых областях выяснять причины проблем просто не нужно – например, все и так в курсе, что не так с нашим правительством.

1.6. …но не поможет в предотвращении, сертификации и приоритезации

Хотя в этой книге рассматриваются общие вопросы отладки, основной упор в ней сделан на *поиске причин ошибок* и их *устранении*.

Я буду рассказывать вам об управлении качеством, стандартах серии ISO-9000, проведении код-ревью и управлении рисками. Если вы хотите почитать об этом, то я рекомендую труды «Метод темпура для тоталитарных процессов управления качеством» и «Руководство по фэншуй для домов без насекомых». Процессы управления качеством имеют ценность, но внедряются они не слишком часто. И даже если внедряются – то не являются абсолютной панацеей от ошибок в ваших продуктах.

А раз в ваших продуктах есть ошибки – вы должны их обнаружить. Этим занимается отдел тестирования (QA) или, если у вас его нет, ваши клиенты. Процесс тестирования также не рассматривается в рамках этой книги – вопросы покрытия тестами, автоматизации тестирования и иных аспектов контроля качества хорошо описаны в других источниках. Хороший томик поэзии, например, «Как мне тестировать тебя – не перечесть тому путей» поможет вам скоротать время,

пока выполняется проверка корректной работы всех 6 467 826 возможных комбинацией настроек вашего продукта.

Рано или поздно одна из этих комбинаций окажется некорректной, и какой-нибудь тестировщик или клиент напишет отчёт об ошибке. Затем ряд менеджеров, инженеров, сотрудников отдела продаж и сотрудников службы поддержки клиентов соберутся на совещании и будут страстно спорить о том, насколько критична эта ошибка, стоит ли её исправлять, и если стоит, то когда именно. Эти вопросы глубоко специфичны для вашего продукта, ресурсов и рынка сбыта, и поэтому я предпочту не касаться их даже 18-метровой палкой. Но когда эти люди решат, что ошибку нужно исправить, вам придётся изучить отчёт об её обнаружении и спросить себя: «Как, чёрт возьми, это могло произойти?» Именно в этот момент воспользуйтесь моей книгой:

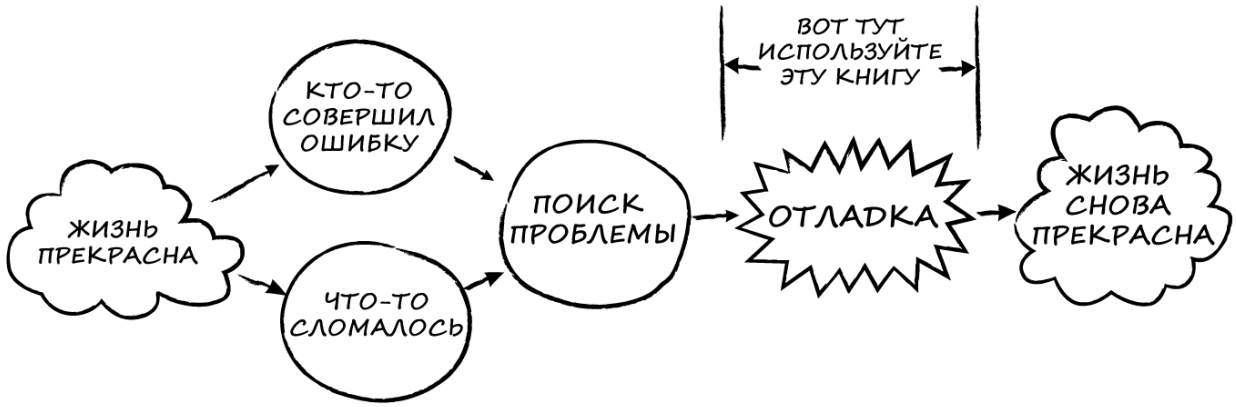


Рис. 1.1. Пояснение, в какой момент следует использовать эту книгу

В следующих главах я научу вас, как:

- подготовиться к поиску ошибки;
- изучить подсказки, указывающие на её причину;
- сосредоточиться на реальной проблеме;
- устранить её;
- убедиться в том, что она действительно устранена;
- с триумфом отправиться домой.

1.7. Отладка – это не просто устранение неполадок

Хотя термины «отладка» (debug) и «устранение неполадок» (troubleshooting) часто используются как синонимы – между ними есть существенная разница. Такая же разница есть между этой книгой и сотнями руководств по устранению неполадок. Обычно под «отладкой» подразумевают поиск причины, по которой продукт не работает так, как должен. «Устранение неполадок» подразумевает, что ошибка уже известна (удалённый файл, оборвавшийся кабель, неисправный элемент и т. д.). Инженер-программист занимается отладкой; автомеханик устраняет неполадки. *Создатели* автомобилей занимаются отладкой (в идеальном мире). Врачи устраняют неполадки в человеческом организме — у них нет возможности его отладить (Богу хватило одного дня на проектирование, выпуск прототипа и релиз; вот что значит «жёсткий график»! Думаю, мы можем простить ему два основных дефекта – бурсит и облысение по мужскому типу).

Правила, описанные в этой книге, применимы как для отладки, так и для устранения неполадок. Неважно, из-за чего именно возникла проблема – они просто помогут вам найти её причину. Поэтому эти правила работают независимо от того, является ли источником проблем ошибка проектирования или неисправный элемент. С другой стороны, руководства по устранению неполадок касаются *только* неисправных элементов. Они содержат десятки таблиц с описанием симптомов ошибок, их причин и рекомендаций по устранению. *Это полезно* – вы получите информацию о всех потенциальных проблемах, которые могут возникнуть с данным конкретным продуктом, и узнаете, как их исправить. С их помощью специалисты по устранению неполадок могут получить концентрированный опыт множества других инженеров и быстро найти известную проблему по её симптомам. Но это не поможет в том случае, если никто ранее не сталкивался с возникшей у вас ошибкой или если это ошибка проектирования (потому что инженеры настолько изобретательны, что предпочитают совершать новые ошибки вместо повторения старых).

Так что если вы устраняете неполадки в какой-то типовой системе – не игнорируйте правило 8 («Воспользуйтесь чьим-то свежим взглядом»). Поищите информацию о возникшей ошибке в руководстве. Если её там нет или приведённые рекомендации не помогают её устранить, или само руководство ещё не написано, потому что вы отлаживаете первую в мире систему по передаче запахов через цифровые каналы связи, то не беспокойтесь – правила этой книги помогут выяснить суть вашей уникальной проблемы и решить её.

1.8. Пару слов о «байках ветеранов»

Я – американский мужчина, рождённый в 1954 и получивший образование по специальности «электронная инженерия». Когда я рассказываю «байку ветерана» о какой-то проблеме и о том, каким образом она была решена – то это реальная история, связанная с понятиями, которые известны американским мужчинам, рождённым в 1954 и получившим такое же образование. Возможно, вы не являетесь таковым, поэтому некоторые из этих понятий будут вам незнакомы. Если вы автомеханик – то, вероятно, не знаете, что такое «прерывание». Если вы родились в 1985 – то, возможно, не слышали о виниловом проигрывателе. Но это неважно; принципы, демонстрируемые в «байках», стоит знать, и в тексте я расскажу достаточно, чтобы вы смогли их понять.

Также отмечу, что я опустил некоторые детали, чтобы защитить репутацию невиновных и, в особенности, виноватых.

1.9. Оставайтесь с нами

Далее я приведу список из девяти золотых правил отладки, а затем подробно рассмотрю каждое из них в отдельной главе. Каждая глава начинается с «байки ветерана», в которой это правило помогло решить какую-то проблему. Я расскажу о различных вариантах осмысления и использования правила, которые помогут вам легко его вспомнить, когда вы лицом к лицу столкнётесь со сложными (или простыми) инженерными проблемами. И я приведу несколько примеров, как можно использовать это правило в других областях – при проблемах с автомобилями, домами и т. д.

В последние главы я включил набор «баек ветеранов», чтобы вы смогли проверить, насколько поняли правила книги, раздел об использовании правил в рамках тяжёлых условий технической поддержки и ряд советов о том, как применять на практике те знания, которым вы научились за время чтения, в рамках вашей профессиональной деятельности.

Когда вы закончите читать эту книгу – ваши навыки отладки станут гораздо лучше, чем раньше. Вы даже сможете гулять по округе в поисках инженеров, попавших в беду, чтобы мгновенно прийти им на помощь и спасти ситуацию. Но дам совет: не надевайте плащ и трусы поверх трико.

Глава 2: Девять золотых правил отладки

Теория, которую я здесь изложил, и которая кажется вам такой фантастической, на самом деле очень жизненна, настолько жизненна, что ей я обязан своим куском хлеба с маслом.

Шерлок Холмс, «Этюд в багровых тонах»

Вот список правил. Запомните их. Распечатайте и повесьте на стену. Повесьте их вообще на все стены, которые найдёте (скоро в продаже: обои с правилами отладки для домашних офисов с серьёзным подходом к декору. Категорически не рекомендуется экспертами по фэншуй).

ПРАВИЛА ОТЛАДКИ

ИЗУЧИТЕ СВОЮ СИСТЕМУ

ВОСПРОИЗВЕДИТЕ ОШИБКУ

НЕ ПРЕДПОЛАГАЙТЕ, А СМОТРИТЕ

РАЗДЕЛЯЙТЕ И ВЛАСТВУЙТЕ

ВНОСИТЕ ПО ОДНОМУ ИЗМЕНЕНИЮ ЗА РАЗ

ЗАПИСЫВАЙТЕ ВСЁ, ЧТО ПРОИСХОДИТ

ПРОВЕРЬТЕ КАБЕЛЬ

ВОСПОЛЬЗУЙТЕСЬ ЧЬИМ-ТО СВЕЖИМ ВЗГЛЯДОМ

ЕСЛИ ВЫ НЕ ИСПРАВИЛИ ОШИБКУ – ОНА НЕ ИСЧЕЗЛА

Глава 3: Изучите свою систему

Но чтобы довести искусство мышления до высшей точки, необходимо, чтобы мыслитель мог использовать все установленные факты, а для этого ему нужны самые обширные познания.

Шерлок Холмс, «Пять зёрнышек апельсина»

3.1. Общий обзор

Байка ветерана. Когда я только закончил колледж и пытался поднабраться опыта по своей специальности (помимо всех остальных дел, которыми был занят), то подрабатывал разработкой контроллера управления задвижкой, созданном на базе микропроцессора. Система управления должна была контролировать массу металлической стружки, попадающей в форму. Масса определялась с помощью весов (см. рис. 3.1). Как и многие инженеры (особенно новички), я взял за основу проект своего сокурсника, который использовал тот же микропроцессор в своей дипломной работе.

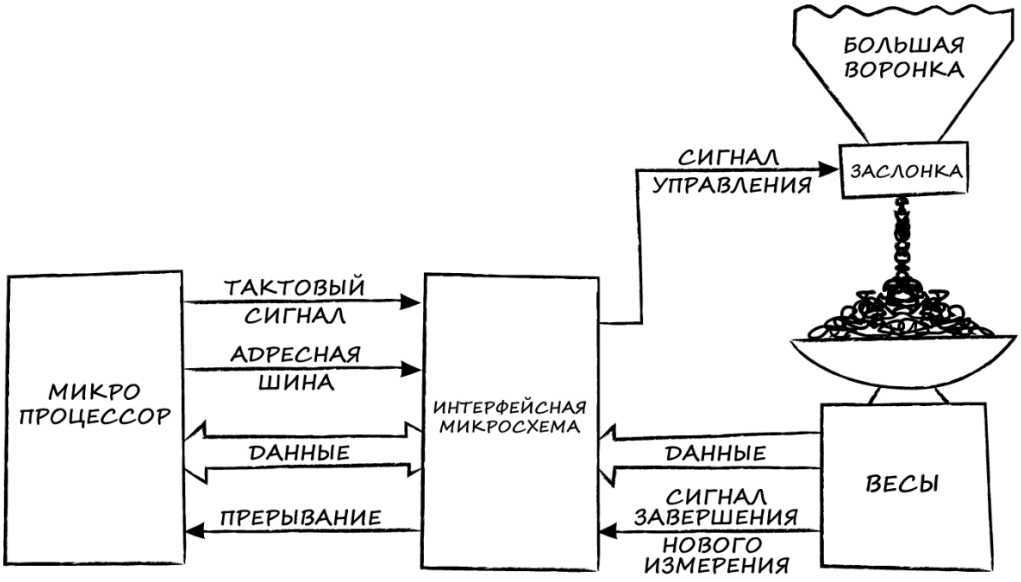


Рис. 3.1. Структурная схема котроллера управления задвижкой

Но моя система не заработала. После завершения измерения процессор должен был получить прерывание – но этого не произошло. Так как это было подработкой, то я был ограничен в инструментах для отладки, и мне потребовалось много времени, чтобы выяснить, что интерфейсная микросхема, получавшая сигнал от весов, не передавала его процессору.

Я работал вместе с программистом. К часу ночи он окончательно расстроился и настоял, чтобы я достал руководство на микропроцессор и изучил его от корки до корки. Я так и поступил, и на странице 37 прочитал: «Прерывание от интерфейсной микросхемы будет получено микропроцессором при первом заднем фронте тактового сигнала, сформированном после прерывания». Это всё равно, как если бы прерывание произошло при первом звонке телефона,

когда абонентом являлась бы не моя микросхема – но в моей ситуации этого бы никогда не произошло, потому что я (как и мой сокурсник) сэкономил на некоторых аппаратных узлах, объединив тактовый генератор с адресными шинами. Продолжая аналогию: телефон звонил *только тогда, когда звонили микросхеме*. В дипломном проекте моего сокурсника не использовались прерывания – поэтому у него не возникло подобной проблемы. Как только я добавил дополнительные аппаратные узлы – то моя программа заработала корректно.

Позже я описал свой «отладочный марафон» отцу (который мало что смыслил в электронике и ничего не знал о микропроцессорах). Он сказал: «Это же просто здравый смысл: когда всё остальное не помогло – читай инструкцию». Эти слова дали мне первое представление о том, что существуют общие правила отладки, которые могут применяться не только к компьютерам и программному обеспечению (ещё я понял, что даже высшего образования недостаточно, чтобы произвести впечатление на моего отца). В данном случае таким правилом было «Изучить систему».

Я не мог приступить к отладке проблемы, пока не понял, как именно работает интерфейсная микросхема. Когда я это понял – проблема стала очевидной. Отмечу, что в целом я понимал, как работает моя система управления. Я знал, как работает аппаратная часть, что получение нового измерения от весов должно вызывать прерывание в процессоре и что такое тактовый генератор и адресные шины. Но я не изучил все детали и обжёгся на этом.

Вам необходимо знать, что делает ваша система, как она устроена и, в некоторых случаях, почему она спроектирована именно таким образом. Если вы не понимаете, как работает какая-то часть вашей системы – то, скорее всего, проблема в ней (это не просто закон Мерфи; если вы чего-то не понимаете в процессе разработки – то это и станет причиной ошибки).

Кстати, понимание системы не означает мгновенного понимания возникшей в ней проблемы. Это похоже на метод Стива Мартина, как гарантированно стать миллионером: «В первую очередь раздобудьте миллион…»[2]. Естественно, вы ещё не понимаете причину проблемы, но чтобы найти её – вы должны знать, как устроена ваша система.

3.2. Прочитайте инструкцию

Квинтэссенция правила «Изучите свою систему» заключается в следующем совете: «Прочитайте инструкцию». Вопреки мнению моего отца – я рекомендую читать её *перед тем*, как что-то делать, а не после того, как все попытки решить проблему закончились неудачей. Когда вы покупаете какой-то товар – в инструкции указано, как его использовать и какие последствия вы получите в результате этих действий. Вам необходимо изучить инструкцию от корки до корки, чтобы добиться от товара именно того, чего вы хотите. В процессе чтения вы можете обнаружить, что это невозможно – значит, вы купили не то, что вам нужно. Так что читайте инструкцию даже не перед тем, как что-то делать с товаром, а ещё до того, как приобрести его, чтобы не получить бесполезное барахло.

[2] Цитируется с разрешения Стива Мартина.

Если ваша газонокосилка не запускается, то, прочитав руководство, вы узнаете, что следует несколько раз сжать грушу праймера, прежде чем тянуть за шнур стартера. У нас была газонокосилка, которая нагревалась так сильно, что её лески склеивались друг с другом; парень, который её использовал, забыл прочитать ту часть инструкции, в которой объяснялось, что триммерную головку надо смазывать. Я узнал это, прочитав инструкцию. Если запеканка из тофу получилась ужасной на вкус – то перечитайте рецепт (хотя чтение рецепта может и не помочь; вам лучше прочитать меню доставки в «Chu-Quik Chou House»).

Если вы инженер, и отлаживаете продукт, созданный вашей компанией, то вам необходимо прочитать внутреннюю документацию. Ответьте себе на вопрос – какова цель разработки этого продукта? Изучите функциональные спецификации. Изучите любые спецификации, касающиеся реализации – схемы, циклограммы, машины состояний. Изучите код и прочитайте комментарии (Ха! Ха! Читайте *комментарии*! Поняли?). Проведите исследование продукта. Выясните, о чём думали инженеры, создававшие его (за исключением того, хватит ли прибыли на приобретение BMW).

Предостережение: не следует слепо доверять этой информации. Документы (и инженеры, в глазах которых уже отражаются их будущие BMW) могут ошибаться, что приводит к множеству сложных проблем. Но вам всё равно нужно выяснить, что, по мнению этих инженеров, они создали, даже если вам придётся принять это «с щепоткой соли».

И иногда эта информация —именно то, что вам нужно знать.

Байка ветерана. Мы отлаживали прошивку для встраиваемого устройства, написанную на языке Ассемблера – то есть имели дело непосредственно с регистрами микропроцессора. Мы обнаружили, что регистре B периодически оказывается «мусор», и нашли конкретную подпрограмму, вызов которой приводит к этому. Изучая её исходный код, мы увидели в самом начале следующий комментарий:

«/* Внимание — эта подпрограмма портит значение регистра B. */»

Внесение исправления в подпрограмму устранило ошибку (конечно, было бы здорово, если бы эту правку внёс сам разработчик, но, по крайней мере, он задокументировал существующую ошибку).

В другой компании, где я работал, мы столкнулись с некорректным поведением программы: выполнение операций происходило в неправильном порядке. В процессе изучения исходного кода (автором которого был я), я вспомнил кое-что и сказал коллеге, что когда-то уже наблюдал такую ситуацию. Мы выполнили поиск по слову «баг» и нашли следующий комментарий, предваряющий вызов двух функций:

/* DJA[3] — Здесь баг? Может быть, следует вызвать их в обратном порядке? */

Действительно, мы изменили порядок вызова функций – и ошибка исчезла.

[3] Это инициалы автора (**D**avid **J**. **A**gans). Когда-то (до всеобщего перехода на системы контроля версий) использование в комментарии инициалов автора помогало понять, к кому нужно обращаться с вопросом по программе (прим. переводчика).

Понимание того, как устроены ваши системы, приносит и дополнительную пользу. Когда вы обнаружите ошибки, то вам нужно будет их исправить, не нарушив при этом работу существующего функционала. Понимание того, что должна делать система — это первый шаг к тому, чтобы не сломать её при внесении исправлений.

3.3. Читайте всё, от корки до корки

Очень часто люди пытаются приступить к отладке, невнимательно прочитав руководство по своей системе. Они пролистали его, останавливаясь на разделах, которые казались им важными – но ключ к пониманию возникшей проблемы остался именно в пропущенных разделах. Да я и сам попал в эту ситуацию со своим контроллером управления задвижкой, который закончил отлаживать в час ночи.

Руководства по программированию и документация API могут быть очень объёмными, но вам придётся тщательно изучить их – ведь проблема будет именно в описании той функции, которое вы не прочитали, так как посчитали очевидным. Часть электрической схемы, которую вы игнорируете, будет источником помехи. Та самая мелкая строчка в даташите, в которой указан неясно описанный параметр синхронизации, может иметь принципиальное значение.

Байка ветерана. Мы разработали несколько версий коммуникационной платы: некоторые из них были трёхпроводными (в них использовалось три телефонных линии), а некоторые – четырёхпроводными. Трёхпроводные платы отлично показали себя в полевых условиях, поэтому перед передачей четырёхпроводных плат на бета-тестирование клиенту мы проводили меньше внутренних испытаний, так как отличия между платами заключались всего лишь в одной микросхеме.

В процессе испытаний на работу при высоких температурах четырёхпроводные платы вышли из строя из-за перегрева. Мы быстро воспроизвели проблему у себя и обнаружили, что произошёл сбой основного процессора. Такая ситуация часто связана с повреждением памяти, поэтому мы провели тесты и обнаружили, что при чтении памяти возникают ошибки. Мы спросили себя – почему у трёхпроводных плат не возникает этой проблемы?

Разработчик аппаратной части осмотрел несколько четырёхпроводных плат и заметил, что в них используются микросхемы памяти от другого производителя. Он сравнил их характеристики – обе микросхемы были одобрены схемотехниками, обе обеспечивали быстрый доступ к памяти и одинаковый механизм синхронизации операций чтения и записи. Разработчик ПО учёл эти характеристики при расчёте таймингов процессора.

Я внимательно прочитал даташиты на обе микросхемы. Единственное найденное мной отличие заключалось в интервале ожидания, который нужно было выдерживать *между* операциями доступа к памяти. Эти интервалы был крайне малыми, а их отличия друг от друга – и того меньше, но механизм синхронизации доступа к памяти, реализованный в процессоре, *вообще не учитывал* необходимость этого интервала. Это приводило к ошибке доступа на более «медленной» микросхеме памяти (где значение интервала должно быть больше) и, вероятно, рано или поздно проявилось бы и на более «быстрой» микросхеме.

Мы исправили проблему, немного замедлив работу процессора. В следующей версии четырёхпроводной платы мы использовали более «быструю» память и прочитали каждую строчку даташита.

Рекомендации по применению (application notes) и руководства по внедрению (implementation guides) содержат массу информации не только об особенностях работы какого-то продукта, но, в частности, и о проблемах, с которыми сталкивались клиенты, использующие этот продукт. Предупреждения о типовых ошибках крайне ценны (даже если вы совершаете только нетиповые ошибки). Загрузите последнюю версию документации с сайта производителя – и узнаете об ошибках, найденных на этой неделе.

Референсный (эталонный) дизайн и примеры программ подскажут вам некоторые способы использования продукта; иногда ими и исчерпывается доступная вам документация. Однако относитесь к ним с осторожностью – зачастую они создаются людьми, которые отлично знают свой продукт, но не следуют передовым практикам проектирования и разработки, и не создают ПО для реальных проектов (их типичный просчёт – невозможность возобновления работы после возникновения ошибки). Не занимайтесь бездумным копированием референсного дизайна; если вы не обнаружите в нём ошибок на ранней стадии разработки, то они обязательно проявят себя позже. Кроме того, даже лучший референсный дизайн, скорее всего, не будет полностью закрывать все потребности вашего продукта – и именно здесь могут начаться проблемы. Это случилось и со мной, когда я взял за основу своей системы проект своего сокурсника – а он не использовал прерывания.

3.4. Изучите предметную область

Для того, чтобы отладить систему – вы должны знать, как она устроена и как функционирует в штатном режиме. Если вы не знаете, что в ПО для ПК на базе процессоров Intel используется порядок байт Little Endian («младшим байтом вперёд»), то можете подумать, что ваши данные зашифрованы. Если вы не знакомы с концепций кэширования, то вас может смутить, что запись в память происходит «не сразу». Если вы не понимаете, как работает шина данных с тремя состояниями, то можете решить, что на вашей плате какие-то глюки с сигналом. Если вы ни разу в жизни не слышали звук цепной пилы, то можете предположить, что он как-то связан с проблемой в её работе. Понимание того, что является нормальным, поможет вам заметить вещи, которые нормальными не являются.

Вы должны знать хотя бы азы своей предметной области. Если бы я не знал, что такое тактовые импульсы и адресные шины, то не понял бы причину своей проблемы с прерываниями даже после прочтения инструкции. Почти все (если не все) участники примеров этой книги знали основы работы своих систем (и позвольте мне извиниться, если я каким-то образом заставил вас поверить в то, что прочтение этой книги поможет вам решить проблему в любой сфере деятельности. Если вы разработчик игр – то вам, вероятно, следует избегать отладки промышленного ПО, используемого на АЭС. Если вы не врач – то не пытайтесь ставить диагноз по

одному лишь серо-зелёному пятну на руке. А если вы политик – то, пожалуйста, вообще ничего не пытайтесь делать).

Байка ветерана. Инженер-программист, с которым я работал, пришёл с совещания по отладке одной из ошибок в нашем продукте, и обречённо покачал головой. Проблема заключалась в зависании микропроцессора. В нём практически ничего не было – ни ОС, ни виртуальной памяти, ничего такого. Единственный способ определить, что он прекратил свою работу – это срабатывание сторожевого таймера. Разработчики ПО пытались выяснить, что к этому привело. Один из специалистов по аппаратному обеспечению предложил: «поставьте точку останова непосредственно перед срабатыванием сторожевого таймера, и когда она активируется – изучите дамп памяти и тому подобное». Видимо, у него были сложности с причинно-следственными связями – если бы разработчики знали, **где** поставить точку останова, то уже нашли бы причину проблемы.

Вот почему люди, занимающиеся железом и ПО, действуют друг другу на нервы, когда пытаются помочь друг другу в отладке.

Отсутствие фундаментальных знаний является причиной того, почему так много людей не могут понять, что не так с их домашним компьютером – у них просто нет основ компьютерной грамотности. Если вы не можете получить нужные знания (или, по крайней мере, сделать это быстро) – то воспользуйтесь правилом 8 и попросите взглянуть на вашу проблему человека с бóльшим опытом. Вполне подойдёт подросток, живущий напротив. И раз уж он здесь – пусть заодно «починит» мигающие цифры «12:00» на вашем видеомагнитофоне.

3.5. Определите взаимосвязи

Чтобы обнаружить место появления ошибки – вам надо знать, что вообще входит в состав вашей системы. Если вы знаете, из каких крупных компонентов она состоит – то можете, например, поочерёдно отключать их, чтобы понять, в каком из них проявляется ошибка. Вам нужно понимать (хотя бы в общих чертах) что делают все составные компоненты и интерфейсы. Если в тостере подгорает тост – то вы должны знать, что вон та чёрная крутилка регулирует время поджаривания.

Вы должны знать весь API и все коммуникационные интерфейсы вашей системы. Вы должны понимать, какие данные (и от какого компонента) получает каждый программный модуль, как он их использует и куда передает результаты работы. Если ваш код имеет высокую степень модульности или написан в ООП-стиле – то интерфейсы будут простыми, а модули будут иметь чёткие концептуальные границы. Можно будет просто взглянуть на интерфейс и увидеть, всё ли в данный момент работает корректно.

В системе могут быть элементы, которые являются «чёрными ящиками» – то есть вы не знаете их внутреннее устройство; но знания о том, как взаимодействуют эти элементы, помогут вам определить, находится ли источник проблемы внутри «чёрного ящика» или вне его. Если проблема внутри элемента – то его потребуется заменить, а если вне его – то её можно устранить. Продолжая пример с подгоревшим тостом – если воздействие на чёрную крутилку никак не влияет на время

поджаривания – то можно предположить, что проблема в тостере, и тогда вы выбрасываете его и покупаете новый (или разбираете и пытаетесь починить, а потом уже выбрасываете).

Предположим, вы ведёте машину и слышите звук «тук-тук-тук», который, как вам кажется, становится чаще по мере того, как вы увеличиваете скорость. Возможно, в протекторе шины застрял камень (это легко исправить) или что-то не так с двигателем (это исправить сложно). На высокой скорости частота вращения двигателя и шин изменяется одновременно. Но если вы знаете, что двигатель связан с шинами через трансмиссию, то понимаете, что если переключитесь на пониженную передачу – то двигатель будет вращаться быстрее при той же скорости шин. Итак, вы переключаетесь на пониженную передачу, а звук остается прежним – тогда вы останавливаетесь и обнаруживаете в протекторе камень. В результате простой переход на пониженную передачу сэкономил вам дорогостоящий визит в автосервис.

3.6. Изучите свои инструменты

Инструменты отладки — это ваши глаза и уши в системе. Вы должны уметь выбирать подходящие инструменты, разумно их использовать и правильно интерпретировать получаемые результаты (если вы засунете градусник под язык не тем концом, то он покажет не ту температуру, которая вам нужна). Многие инструменты обладают очень полезными функциями, известными лишь знатокам. Чем лучше вы разбираетесь в инструментах – тем легче вам будет понять, что происходит в вашей системе. Потратьте время, чтобы узнать всё, что только можно, о ваших инструментах – часто ключом к пониманию причины проблемы (см. правило 3) является хорошо настроенный отладчик или логический анализатор.

Вам также следует знать границы возможностей своих инструментов. Пошаговое выполнение кода поможет найти ошибки в логике, но не в проблемах с таймингами или многопоточностью. Профилировщик позволит найти проблему с таймингами, но не ошибки в логике. Аналоговый осциллограф даст возможность увидеть шум сигнала, но не позволит сохранить объёмный лог; цифровой логический анализатор позволит сохранить больше данных, но он не будет детектировать шум сигнала. Градусник не подходит для того, чтобы определить температуру ириски, а кондитерский термометр – для измерения температуры человеческого тела.

«Железячник», предлагавший установить точку останова «непосредственно перед срабатыванием сторожевого таймера», не знал их ограничений (или ему была известна технология путешествий во времени). Что в итоге сделал программист: он подключил логический анализатор для сохранения трассировки адресной шины и шины данных, и установил для сторожевого таймера очень маленькое время срабатывания. Он знал, что ему нужна трассировка, потому что он не узнает, что произошёл сбой, пока не сработает сторожевой таймер. Он установил для сторожевого таймера маленькое время срабатывания так как знал, что логический анализатор может сохранить лишь ограниченный объём данных. Трассировка позволила ему определить, в какой момент происходит сбой, и предоставила информацию для анализа его причин.

Вы должны знать свои инструменты разработки ПО. Конечно, в их состав входит используемый вами язык программирования – если вы не знаете, что означает оператор " "[4] в языке ANSI C, то где-то допустите ошибку. Но вы должны обладать и более глубокими знаниями о том, что именно делают компилятор и линкер с вашим исходным кодом. То, как обрабатываются ссылки, выравнивается и выделяется память, влияет на вашу программу, и это влияние неочевидно при взгляде на ваш код. Инженеры-схемотехники должны понимать, как определения на высокоуровневых языках описания аппаратуры (Verilog, VHDL и т. п.), конвертируются в наборы регистров и логических вентилей микросхемы.

3.7. Ищите информацию

Байка ветерана. Начинающий инженер (назовём его «Джуниор») работал на крупного производителя компьютеров и участвовал в разработке системы, в состав которой входил чип под названием 1489A. Этот чип принимал сигналы по линии связи (примерно такой же, которая используется между компьютером и модемом). Один из старших инженеров (назовём его «Рефлекс») увидел его схему и сказал – «Ох, тебе не стоит использовать 1489A. Лучше используй старый добрый 1489». Джуниор спросил, почему, и Рефлекс ответил: «1489A очень сильно нагревается». Что ж, скептически отнесясь к старческому брюзжанию (что характерно для самоуверенной молодёжи), Джуниор решил изучить схемы и выяснить, почему новая версия чипа стала греться сильнее. Он обнаружил, что единственное различие между чипами заключается в номинале внутреннего резистора подтяжки, который делал новый чип менее восприимчивым к помехам, наводящимся на линию связи. Номинал резистора в 1489A стал меньше, поэтому при подаче высокого напряжения он мог нагреться очень сильно; но резистор был подключён таким образом, что на него не могло быть подано такое напряжение. Выяснив это, Джуниор проигнорировал совет Рефлекса и использовал новый чип. И он не перегревался.

Несколько месяцев спустя Джуниор изучил предыдущий вариант схемы, разработанный другой командой. Когда он приложил щуп мультиметра к точке, которая была обозначена на схеме как входной контакт 1489, то понял, что номер контакта оказался неправильным. Он открыл свой потрёпанный справочник – и, конечно же, нашёл несоответствие в распиновке между документаций и схемой. Сигнал был подключен к контакту резистора подтяжки вместо входного контакта чипа. В большинстве случаев к контакту подтяжки ничего не подключается, но если подать на него сигнал, то он вроде как работает, но сам чип при этом вообще не имеет помехоустойчивости. Также такое ошибочное подключение приводит к тому, что ток не проходит через входной резистор, и поэтому увеличивается потребление тока через резистор подтяжки. «Фактически», – с удовлетворением отметил Джуниор, – «он начинает нагреваться, а если использовать 1489A – то нагревается ещё сильнее».

[4] В языке ANSI C одиночные кавычки используются для обозначения литерала одиночного символа, а двойные – для литерала строки символов. Литерал строки всегда завершается автоматически добавляемым терминирующим нулём – то есть 'a' будет занимать один char, "a" – два char'а. (прим. переводчика)

В этой ситуации были нарушены две части правила «Изучите свою систему». Во-первых, при проектировании никто не проверил правильность используемых номеров контактов. Во-вторых, после обнаружения перегрева Рефлекс не пробовал разобраться в схеме, чтобы определить его причину. Результат этого крайне печален: команда использовала в схеме устаревший и труднодоступный чип, который сильно нагревался и не имел помехоустойчивости. В остальном же инженеры проделали отличную работу в рамках этого проекта.

С другой стороны, Джуниор не поверил распиновке, приведённой на схеме, и сверил её со справочником. Он *выяснил*, почему чип перегревался.

Не пытайтесь гадать. Ищите информацию. Ищите записи, сделанные либо собственноручно вами, либо вашими коллегами, либо инженером, который разработал микросхему или программистом, который написал утилиту, и т. д. Не стоит доверять собственной памяти. Распиновки чипов, названия и сигнатуры функций – всё это при желании можно найти. Будьте как Эйнштейн, который не помнил свой номер телефона. «Зачем беспокоиться?», – говорил он. – «Он же есть в телефонной книге».

Если вы делаете какие-то необоснованные предположения, когда смотрите на сигнал, приходящий на чип, то можете пропустить неправильный сигнал, который выглядит правильным. Если вы предполагаете, что аргументы передаются функции в правильном порядке – то можете не заметить ошибку, которую допустил разработчик. Основываясь только на «производной» информации (схемах, чужом коде и т. д.) вы можете получить сведения, которая вас запутает или, что ещё хуже, вселит в вас ложные надежды. Не тратьте время на предположения – обратитесь к первоисточнику.

И, ради всего святого, если вы не можете устранить потоп в подвале в 2 часа ночи и решили позвонить водопроводчику, не пытайтесь угадать номер. Поищите его в телефонном справочнике.

3.8. Памятка

Изучите свою систему

Это правило приведено первым, потому что оно является наиболее важным. Понятно?

- **Прочитайте инструкцию**. В ней указано, что триммерную головку газонокосилки надо смазывать, чтобы лески не склеивались друг с другом;
- **Читайте всё, от корки до корки**. Информация о том, как ваш микропроцессор обрабатывает прерывания, приведена на 37 странице руководства;
- **Изучите предметную область**. Цепная пила *должна* издавать громкий звук;
- **Определите взаимосвязи**. Скорость вращения двигателя может отличаться от скорости вращения шин – и связь между этими скоростями зависит от трансмиссии;
- **Изучите свои инструменты**. Узнайте, чем отличаются концы градусника и как использовать специфичные функции вашего логического анализатора «Глюколов»;
- **Ищите информацию**. Даже Эйнштейн углублялся в детали. А вот Рефлекс доверился своей памяти.

Глава 4: Воспроизведите ошибку

Нет ничего полезнее сведений из первых рук.

Шерлок Холмс, «Этюд в багровых тонах»

4.1. Общий обзор

Байка ветерана. Шёл 1975 год. Поздно ночью (ночью всегда поздно, не так ли?) я сидел в одиночестве в лаборатории и пытался устранить ошибку в видеоигре «Пинг-понг» – одной из первых домашних видеоигр. Она разрабатывалась Центром инноваций MIT при финансовой поддержке местного предпринимателя. В ней присутствовал режим «тренировочной стенки». Ошибка, которую я пытался устранить, периодически возникала в тот момент, когда шарик рикошетил от стенки. Я подключил свой осциллограф (это такой прибор, который рисует волнистые линии на маленьком круглом экране – вы могли видеть его в старых научно-фантастических фильмах), уставился на него и стал ждать появления ошибки. Это было непросто – если я ставил слишком медленную скорость шарика, то рикошет происходил только раз в пару секунд (и ждать ошибки приходилось долго), а если слишком быструю – то мне приходилось уделять всё внимание тому, чтобы отбить шарик, и я не успевал смотреть на осциллограф. Если бы я пропустил шарик – то мне пришлось бы ждать следующей подачи, чтобы увидеть ошибку. Это был не очень эффективный способ отладки, и я сказал себе: «Ну, это всего лишь игра». Ладно, на самом деле я подумал, что было бы здорово, если бы я мог заставить приставку играть саму с собой.

Надо сказать, что перемещение ракетки (по оси Y) и перемещение шарика (по осям X и Y) осуществлялось за счёт изменения уровня напряжений (см. рис. ниже). Пояснение для тех, кто не разбирается в «железе»: напряжение похоже на уровень воды в баке; вы наполняете и опустошаете бак, чтобы изменить уровень. Только тут не вода, а электроны.

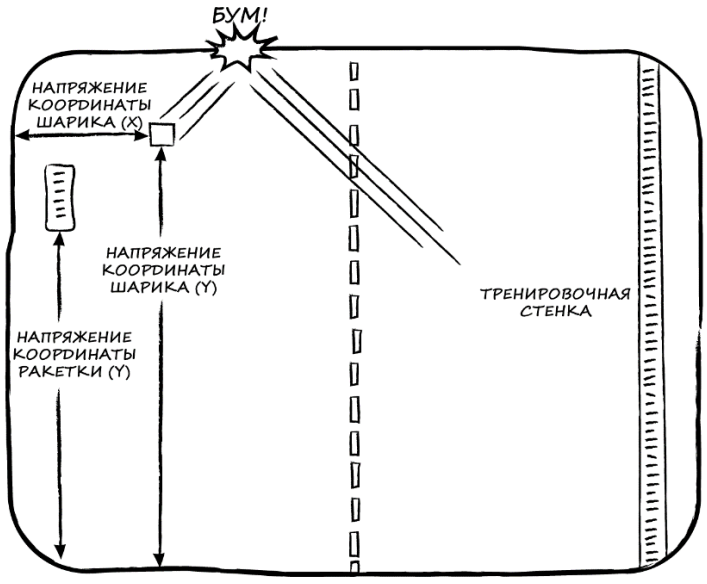


Рис. 4.1. Видеоигра «Пинг-понг»

Я понял, что если подключу к входу, определяющему положение ракетки по оси Y, не сигнал от paddle-контроллера, а сигнал напряжения, определяющий положение шарика по оси Y, то ракетка всегда будет на той же высоте, что и шарик. Каждый раз, когда шарик будет рикошетить, ракетка окажется на нужной высоте, чтобы отбить его. Я сделал это, и мой «призрачный игрок» действительно оказался очень хорош. Поскольку приставка играла сама с собой, я смог сосредоточить своё внимание на осциллографе и приступил к поиску и устранению ошибки.

Даже поздно ночью я смог легко увидеть, что не так, потому что в момент проявления проблемы смотрел на осциллограф – это было ключом к её пониманию. Вообще, это типично для многих проблем – вы не можете её увидеть, потому что она не проявляется в тот момент, когда вы этого ждёте. И это не значит, что она происходит только поздней ночью (хотя в конечном итоге отлаживать её приходится именно тогда) – проблема может повториться только в одном случае из семи или проявиться только в тот раз, когда тестирование проводит Чарли.

Сейчас, если бы Чарли работал в моей компании, и вы спросили бы его: «Что ты будешь делать, если увидишь ошибку?», он бы ответил: «Попробую её повторить» (Чарли был хорошо обучен). Есть три причины, из-за которых стоит повторить ошибку:

- **Потому что вы сможете опять её увидеть**. Чтобы определить, что произошёл сбой (мы обсудим это подробнее в следующем разделе) – вы должны уметь его воспроизводить. И желательно воспроизводить регулярно. В случае с видеоигрой у меня получилось не отрывать слипающихся глаз от осциллографа в те моменты, когда возникала проблема;
- **Потому что это позволит сфокусироваться на её причине.** Точная информация о том, при каких условиях происходит сбой, поможет вам сделать предположения о его вероятных причинах (но будьте осторожны; иногда это может ввести людей в заблуждение, например: «Тост подгорает, только если вы кладёте хлеб в тостер; следовательно, проблема в хлебе». Подробнее о надуманных предположениях мы ещё позже поговорим);
- **Потому что это поможет определить, исправлена ли ошибка**. Как только вы решите, что устранили её – вам нужно будет удостовериться в этом. Если до внесения исправлений ошибка проявлялась в 100 тестах из 100, а после внесения – не проявилась ни в одном, то, значит, вы действительно её устранили (и не стоит относиться к этой фразе как к чему-то предельно очевидному. Часто разработчик вносит изменения в ПО, чтобы исправить ошибку, а потом тестирует новую сборку в условиях, отличных от тех, в которых она была обнаружена. Ошибка может не повториться, даже если он просто добавил в код лимерики, но он всё равно уходит домой довольный. А через несколько недель во время тестирования или, что ещё хуже, на объекте клиента эта ошибка снова повторяется. Об этом я тоже расскажу позже).

Байка ветерана. Недавно я купил полноприводную машину, и всё лето ездил на ней без проблем. Когда наступили холода (в Нью-Гэмпшире они обычно начинаются в сентябре), я заметил, что в первые несколько минут после начала движения сзади доносится какой-то «завывающий» звук – если скорость при этом находится в диапазоне от 25 до 30 миль в час. Через 10 минут езды он исчезал. На более низких и высоких скоростях звука не было. Если температура была выше -4 °C– звука не было.

Я отдал автомобиль дилеру на плановое техническое обслуживание и попросил его ранним утром, когда будет холодно, попробовать проехаться на нём и прислушаться. В итоге им занялись только в 11 часов, когда температура была 3 °C – и звука не было. Автомеханики сняли колёса и поискали проблему в тормозах, но ничего (конечно же) не нашли. Они не воспроизвели условия, в которых проявлялась проблема, поэтому у них не было шансов её обнаружить (я пытался сам приехать к ним в очередной холодный день, но в Нью-Гэмпшире выдалась самая тёплая зима в истории. Так работает Закон Мерфи. Я думаю, что холода вернутся, как только закончится моя гарантия на машину).

4.2. Повторите это

Но как воспроизвести ошибку? Что ж, один из простых способов – продемонстрировать ваш продукт на выставке; настолько же эффективно будет провести его демонстрацию для потенциальных инвесторов. Если же у вас под рукой нет клиентов или инвесторов – вам придётся наблюдать за работой вашего продукта в ожидании сбоя. В принципе, в этом и заключается суть тестирования, но есть важный момент – вы должны суметь сделать так, чтобы *после* первого проявления ошибки можно было повторить её по вашему желанию. Наличие хорошо задокументированной процедуры тестирования является несомненным плюсом, но в целом достаточно понимать, что однократного проявления ошибки будет недостаточно для её устранения. Когда трёхлетняя девочка видит, как её отец падает со стремянки, после чего банка с краской взлетает в воздух, переворачивается вверх дном и падает ему на голову, она хлопает в ладоши и кричит: «Сделай так ещё раз!». Ведите себя как трёхлетний ребенок.

Проанализируйте, что вы сделали непосредственно перед проявлением ошибки, и повторите это снова. Записывайте каждый ваш шаг. Затем повторите эти шаги, чтобы убедиться, что их выполнение приводит к ошибке (отцы, измазанные краской, освобождаются от этого упражнения. Вообще, бывают случаи, когда проявление ошибки приводит к серьёзным негативным последствиям, и нет возможности каждый раз доводить дело до этого. Вам потребуется изменить что-то в системе, чтобы снизить ущерб, но вы должны внести как можно меньше изменений).

4.3. Начните с самого начала

Часто достаточно повторить всего лишь несколько простых шагов для воспроизведения проблемы: нажмите на эту кнопку – и появится сообщение об ошибке. Иногда шаги просты, но их много: перезагрузите компьютер, запустите эти пять программ, *затем* нажмите на эту кнопку – и появится сообщение об ошибке. Поскольку ошибки могут зависеть от состояния системы (которое, в свою очередь, может зависеть от множества отдельных факторов) – вам нужно следить за ним на протяжении выполнения всех шагов (если вы скажете механику, что стёкла вашей машины перестают открываться каждый раз, когда вы едете в холодную погоду, то ему, вероятно, также следует знать, что каждое утро вы заезжаете на автомойку). Попробуйте начать описание последовательности шагов с известного состояния – например, с перезагрузки компьютера или входа в гараж.

4.4. Стимулируйте появление ошибки

Если для повторения ошибки требуется вручную выполнить значительное количество действий – может оказаться полезным автоматизировать этот процесс. Именно это я сделал в своём примере с видеоигрой; мне нужно было одновременно играть в неё и отлаживать, так что я автоматизировал ход игры и сосредоточился на отладке (жаль, что я не смог автоматизировать отладку и сосредоточиться на игре!). Во многих случаях ошибка повторяется только спустя множества попыток её воспроизвести, поэтому будет разумным запустить автотест на всю ночь. ПО с радостью отработает до утра, и вам даже не придётся покупать ему пиццу.

Байка ветерана. В моём доме есть окно, которое раньше протекало – и только в те дни, когда шёл сильный дождь, а ветер дул с юго-востока. Я не стал ждать следующего шторма, а использовал приставную лестницу и шланг, чтобы создать похожие условия. Это позволило мне увидеть, куда именно протекает вода, обнаружить щель в слое герметика и, после того как я законопатил её, убедиться, что течи больше нет.

Аллерголог проведёт аллергопробы, чтобы увидеть реакцию вашего организма. Стоматолог распылит холодный воздух на ваши зубы, чтобы найти чувствительные к холоду места (кроме того, они делают это просто для развлечения). А полицейский заставит вас пройти по прямой линии туда и обратно, прогнуться назад и коснуться кончика носа, а также проговорить алфавит задом наперёд, чтобы определить, не пьяны ли вы (это намного безопаснее, чем позволить вам проехать дальше, чтобы посмотреть, сможете ли вы удержаться на свой полосе).

Если ваш веб-сервер время от времени отправляет в ответ не ту страницу – настройте веб-браузер на автоматический запрос страниц. Если в вашем ПО происходят ошибки в момент интенсивного обмена данными по сети, то используйте специальные инструменты генерации трафика, чтобы простимулировать сбой.

4.5. Но не симулируйте её

Есть большая разница между *стимуляцией* появления ошибки (это хороший подход) и её *симуляцией* (этот подход плохой). В предыдущем примере я рекомендовал симулировать нагрузку на сеть, а не саму ошибку. Симулировать условия, в которых воспроизводится ошибка – это нормально. Но старайтесь избегать симуляции самой ошибки.

«А зачем я вообще буду её симулировать?» – спросите вы. Что ж, предположим, вы сталкиваетесь с периодически проявляющейся ошибкой и догадываетесь, какой именно низкоуровневый компонент является причиной сбоя. Вы можете создать тестовую конфигурацию, в которой этот компонент будет использоваться часто, и посмотреть, увеличиться ли частота повторения ошибки. Или вы можете разобраться в причинах ошибки, найденной одним из ваших клиентов, создав аналогичную систему в вашей лаборатории. В любом случае, вы пытаетесь симулировать ошибку – то есть воспроизвести её другим способом или в другом окружении.

В тех ситуациях, когда у вас есть лишь догадки о причинах сбоя, его симуляция часто оказывается безуспешной – обычно потому что ваше предположение было неверным или из-за того, что тестовое окружение не соответствует исходному, и ваша симуляция работает либо вообще без ошибок, либо, что ещё хуже, приводит к другой ошибке, которая отвлекает вас от изначальной. Ваш текстовый редактор (тот самый, с помощью которого вы собираетесь потеснить на рынке Microsoft Word) при наборе текста теряет абзацы, и вы подозреваете, что это как-то связано с записью файла. Тогда вы создаёте тестовое приложение, которое постоянно записывает данные на жёсткий диск, и операционная система зависает. Вы приходите к выводу, что Windows слишком медленная, и приступаете к разработке новой операционной системы, которая потеснит на рынке детище Microsoft.

У вас уже достаточно ошибок; не пытайтесь создавать новые. Используйте свои инструменты отладки, чтобы понять причину ошибку (см. Правило 3: «Не предполагайте, а смотрите»), но не пытайтесь изменить механизм её воспроизведения – это поведёт вас по ложному следу. В примере с текстовым редактором вместо изменения способа записи данных на жёсткий диск лучше было генерировать нажатия на клавиши и следить за тем, что записывается в файл.

Попытка воспроизвести проблему не в том же, а в аналогичном окружении может быть полезна, но в определённых рамках. Если ошибка воспроизводится в разных экземплярах системы (например, на разных экземплярах устройства), то можно сделать вывод, что это ошибка проектирования, а не сбой в конкретной системе. Если ошибка повторяется только в определённых окружениях, то это позволяет сузить круг поиска её возможных причин. Но если вы не можете быстро воссоздать подобное окружение у себя в офисе – не занимайтесь экспериментами, иначе вы увязнете в этом, отвлёкшись от исходного окружения, в котором произошла ошибка. Если у вас есть окружение, в котором ошибка повторяется регулярно (пусть и не постоянно, а с лишь иногда) – то отлаживайте её непосредственно там.

Типичной ситуацией является проблема интеграции на объекте клиента – ваше ПО даёт сбой на конкретном устройстве, которое управляет определённой периферией. Возможно, вы сможете повторить этот сбой у себя в офисе, воспроизведя конфигурацию клиента. Но если у вас нет нужного оборудования или вы не можете повторить внешние условия, которые есть на объекте клиента, и, соответственно, не можете воспроизвести ошибку – то возникает соблазн написать какие-то

программные эмуляторы для отсутствующих устройств, новые тестовые приложения и т. д. Не делайте этого – стисните зубы и либо запросите у клиента доставку нужного оборудования, чтобы передать его вашим инженерам, либо отправьте инженера на объект клиента (снабдив его всеми необходимыми устройствами и инструментами). Если ваш клиент находится на Арубе, то, вероятно, вы сразу откажетесь от варианта «запросить оборудование» и предпочтёте второй вариант – и, к слову, не заинтересованы ли вы сейчас в найме опытного инженера с хорошим опытом в программировании и отладке?

Тревожный звоночек, на который стоит обратить внимание – это замена одного окружения на другое, казалось бы, идентичное. Но они не идентичны. Когда я пытался починить протекающее окно, то мог бы предположить, что проблема в неправильной конструкции самой модели – и использовать для тестирования другое «точное такое же» окно. И не нашёл бы щель в герметике, которая была причиной протечки моего окна.

Учтите, что это не означает, что вам не нужно автоматизировать тестирование или «ужесточать» условия, в которых наблюдается ошибка, чтобы стимулировать её появление. Автоматизация может привести к тому, что ошибка начнёт повторяться существенно чаще – как в моей истории с видеоигрой. «Ужесточение» может сделать проблему более заметной – как в случае протечки окна, когда я смог обнаружить щель с помощью шланга, а не стал ждать очередного сильного ливня. Оба описанных подхода позволяют стимулировать появление ошибки, не симулируя механизм её появления. Вносите изменения в систему на достаточно высоком уровне, чтобы они влияли только на частоту сбоев, а не на природу их появления.

Кроме того, следите за тем, чтобы не переусердствовать и не создать новых ошибок в процессе отладке текущей. Не будьте схемотехником, который предполагает, что причиной проблемы является перегрев и обдувает микросхему горячим воздухом до тех пор, пока она не расплавится, после чего делает вывод, что причина ошибки – вся эта липкая расплавленная пластмасса на печатной плате. Если бы я использовал пожарный шланг для проверки течи в окне – то мог бы сделать вывод, что проблема заключалась в разбитом его струёй стекле.

4.6. А что, если ошибка не систематична?

Воспроизвести ошибку становится гораздо сложнее, если она проявляется только время от времени. Многие сложные проблемы не имеют ярко выраженной систематики в своём появлении – поэтому к ним трудно применить правило, рассматриваемое в этой главе. Вы можете точно знать, в какой момент ошибка проявилась первый раз, но при повторении тех же действий она может повториться только в одном из 5, 10 или (вздох) 450 случаев.

Здесь есть нюанс – вы не знаете точно, из-за чего проявилась ошибка. Вы можете в деталях знать все ваши действия, которые этому предшествовали, но можете не знать всех сопутствующих ей условий. Есть и факторы внешней среды, о которых вы не знаете или не можете контролировать – начальные условия, входные обрабатываемые данные, помехи, температура, вибрация, сетевой трафик, фаза луны и количество промилле алкоголя в крови тестировщика. Если бы вы знали обо всём этом и могли управлять – то сумели бы воспроизводить ошибку на постоянной основе. Но,

естественно, это остаётся за пределами ваших возможностей, и что делать в таком случае – мы обсудим далее.

Что можно сделать для контроля внешних факторов? Прежде всего – попытаться определить хотя бы ряд из них. В рамках ПО – ищите:

- неинициализированные переменные (смотрю на вас с укором, если они есть);
- фрагменты кода, в которых используются произвольные и заранее неизвестные (например, считанные из файла) данные;
- всё, что связано с таймингами;
- всё, что связано с многопоточностью и синхронизацией;
- всё, что связано с внешними устройствами (например, телефонную сеть или 6 тысяч детей, зашедших на ваш веб-сайт).

В рамках аппаратной части устройств обратите внимание на помехи, вибрацию, температуру, тайминги и отличия в аппаратных узлах (например, отличия в моделях микросхем или использование микросхем разных поставщиков). В моём примере с шумом автомобиля проблема могла бы показаться периодической, если бы я не обратил внимание на температуру и скорость.

Байка ветерана. В старом центре обработки данных (где стояли древние мейнфреймы) периодически возникали сбои в программе. Это всегда происходило примерно в одно и то же время – около 15:00 – но во время выполнения разных блоков кода. В конце концов выяснилось, что именно в это время в здании традиционно проходил кофе-брейк, и все торговые автоматы в столовой работали одновременно, что приводило к отключению питания некоторых устройств обработки данных.

Если у вас есть предположения, какие факторы могут быть связаны с интенсивностью воспроизводимости ошибки – попробуйте влиять на них различными способами. Инициализируйте ваши массивы и используйте в качестве входных данных приложения заранее известные наборы значений.

Меняйте тайминги, чтобы проверить, сможете ли вы заставить систему выйти из строя при определённом сочетании настроек. Создавайте вибрацию, нагревайте, охлаждайте, наводите помехи, меняйте тактовую частоту и напряжение питания вашего прибора, пока не увидите какие-то изменения в интенсивности возникновения ошибок.

Иногда вы обнаруживаете, что изменение одного из внешних факторов приводит к исчезновению ошибки. Естественно, вам захочется сосредоточиться на экспериментах, связанных именно с этим фактором, чтобы понять, как именно он влияет на сбой. Если ошибка иногда повторяется со случайным набором входных данных, а с теми, что вы специально сформировали – никогда не проявляется, то просто формируйте ещё как можно больше наборов, пока не определите нужный.

Иногда вы осознаёте, что не в состоянии контролировать воспроизведение ошибки с помощью внешних факторов, но можете сделать её появление менее случайным – например, с

помощью вибрации или помех. Если причина ошибки связана с событием, которое само по себе происходит редко (например, с действием помехи высокой интенсивности), то влияние на этот внешний фактор (путем генерации случайных помех) может увеличить частоту появления ошибки. Возможно, это максимум того, что вы можете сделать, но всё не зря – теперь вы знаете условие возникновения ошибки и сможете наблюдать её чаще. Но будьте осторожны: следите за тем, чтобы воздействие на внешний фактор не привело к новой ошибке. Если на появление ошибки влияет температура, а вы создадите вибрацию, из-за которой посыплются все микросхемы на плате, то увидите ещё больше ошибок, но они не будут иметь никакого отношения к исходной проблеме.

Иногда кажется, что никакие внешние факторы не влияют на проявление ошибки, и вы возвращаетесь к тому, с чего начали. Ошибка всё ещё повторяется время от времени, и не имеет ярко выраженной систематики.

4.7. Что делать, если после всех опытов в появлении ошибки нет систематики?

Помните, что есть три причины, по которым стоит научиться стабильно воспроизводить ошибку:

- потому что вы сможете опять её увидеть;
- потому что это позволит сфокусироваться на её причине;
- потому что в будущем это поможет определить, исправлена ли ошибка.

Ниже описано, как достичь цели, даже если кажется, что у этого чёртового сбоя есть собственный разум. Помните, что это не так – у каждой ошибки есть причина, и её можно найти. Просто иногда она очень хорошо спрятана множеством внешних случайных факторов, на которые вы не можете повлиять.

4.7.1. Пристальный взгляд на неприятный сбой

Вам нужно изучить ошибку. Если она повторяется не при каждом тесте, то вам придётся заниматься этим *каждый раз, когда она возникнет, игнорируя успешные тестовые прогоны*. Ключевым моментом является *сбор информации о каждом тесте, чтобы вам было что анализировать после проявления ошибки*. Для этого заставьте систему выводить как можно больше информации о своей работе и сохраняйте её в лог-файл.

Просматривая этот лог, вы легко сравните данные по успешным и неуспешным тестовым прогонам (см. Правило 5: «Вносите по одному изменению за раз»). Если вы собираете нужную информацию – то сможете увидеть разницу между ними. Анализируйте именно отличия, которые есть у логов неуспешных тестовых прогонов. Именно они помогут понять вам, с чего начать отладку.

Даже если ошибка проявляется без какой-либо систематики – вы можете фиксировать информацию о ней и обрабатывать её так, как будто все эти сбои происходили на регулярной основе.

Байка ветерана. Мы разработали систему для проведения видеоконференций – и у нас иногда возникала ошибка, когда мы пытались организовать звонок в аналогичную систему другого производителя. В одном случае из пяти другая система отключала видео и переходила в режим аудиовызова.

Мы не могли изучить внутренности чужой системы – и стали анализировать свои логи. Мы собрали информацию о двух звонках, которые произошли друг за другом – при этом в первом случае всё было нормально, а во втором случилась упомянутая ошибка. В логе «неудачного» звонка была информация об отправке в другую систему команды, которой мы не ожидали – её не должно было быть. Мы проверили логи «удачных» звонков – в них этого сообщения не было. Мы собрали больше статистики, и когда снова получили ошибку – то, конечно же, в логе было сообщение о той команде.

Оказалось, что источником проблемы является буфер памяти, в котором хранятся команды предыдущего звонка – и в каких-то ситуациях он не очищается при начале нового. Если же буфер был очищен – то всё работает корректно. В противном случае в начале нового звонка происходит отправка некорректной в данном контексте команды – и система другого производителя в результате её обработки переходила в режим аудиовызова.

Мы смогли определить причину сбоя, потому что собирали достаточно информации по каждому звонку. Несмотря на то, что сбой воспроизводился нестабильно, каждый раз при его проявлении в логах появлялось найденное нами сообщение.

4.7.2. Ложь, наглая ложь и статистика

Вторая причина, по которой стоит научиться стабильно воспроизводить ошибку – потому что это позволит сфокусироваться на её причине. Когда ошибка повторятся регулярно – вы начинаете замечать закономерности в выполняемых вами действиях, которые, видимо, влияют на происходящее. Это нормально – но не стоит слишком зацикливаться на них.

Когда ошибка проявляется без какой-либо видимой систематики – вы, вероятно, не сможете собрать достаточно статистики, чтобы определить, имеет ли значение то, пальцем какой руки вы нажимаете на кнопку – левой или правой. Наблюдаемые совпадения заставляют вас думать, что выполнение одного из условий более существенно увеличивает вероятность повторения ошибки, нежели другого. Затем вы начинаете размышлять, в чём же разница между этими двумя условиями, и только напрасно тратите кучу времени, пустившись по ложному следу.

Это не означает, что все совпадения и закономерности, которые вы видите, никак не связаны с вашей ошибкой. Но если они не оказывают на её проявление *прямого* эффекта – то эта связь будет скрыта за пеленой других случайных факторов, и ваши шансы понять её довольно малы. Лучше уж сделать ставку в казино Лас-Вегаса.

Собрав достаточно информации (см. п. 4.7.1) – вы сможете определить, какие условия *всегда* сопутствуют проявлению ошибки, а какие – *никогда*. Именно на первые следует обратить внимание, когда вы приступаете к поискам вероятных причин ошибки.

4.7.3. Ошибка исправлена – или вам просто повезло?

Конечно, если ошибка повторяется нестабильно – то очень сложно быть уверенным, что вы её исправили. Раньше она проявлялась в одном тесте из десяти, а после ваших правок – в одном из тридцати, но вы прекращаете тестирование после 28 успешных прогонов, и считаете, что проблема решена; но это не так.

Если вы используете статистические гипотезы, то чем больше тестов проведёте – тем лучше для вас. Но ещё лучше определить последовательность событий, которая гарантированно приводит к ошибке – даже если эту последовательность не удаётся стабильно повторять каждый раз (например, она зависит от неподвластных вам внешних факторов), по крайней мере, вы знаете, что она точно завершается сбоем. После того, как вы внесёте правки и решите, что ошибка устранена – запускайте тесты до тех пор, пока эта последовательность не повторится. Если последовательность повторилась, а ошибка не возникла – значит, вы успешно её устранили. Не останавливайтесь после 28 успешных тестов, даже если в их рамках последовательность нужных событий не повторится (правда, вы можете остановиться после 28 попыток, потому что курьер привёз пиццу, но тогда после ужина вам всё равно придётся вернуться к работе – ещё один довод за внедрение автоматического тестирования).

Байка ветерана. У нас периодически возникала проблема при видеовызове по шести телефонным линиям. Системам видеоконференцсвязи требуется передавать много битов в секунду, и одной телефонной линии в этом случае недостаточно. Таким образом, наша система совершала шесть звонков и распределяла видеоданные по шести телефонным линиям.

Беда в том, что телефонный провайдер не мог маршрутизировать все шесть вызовов по одному и тому же маршруту, поэтому данные о некоторых вызовах могли поступать немного позже, чем об остальных. Чтобы исправить это, мы использовали метод, называемый «привязкой». Мы добавляли специальный маркер в поток данных каждого вызова, что позволяло принимающей системе определить величины задержек «медленных» вызовов, а потом добавить аналогичные задержки для «быстрых» вызовов, чтобы синхронизировать их (см. рис. 4.2).

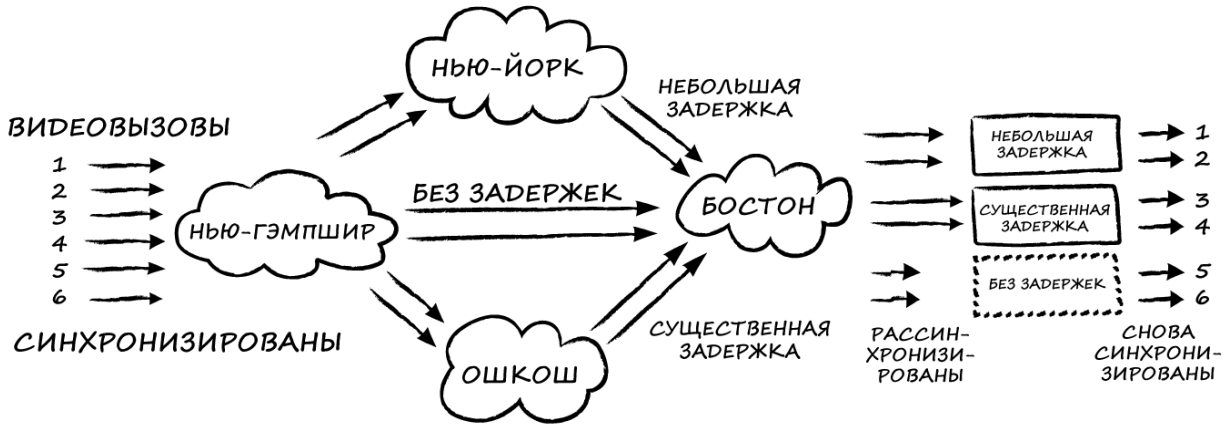


Рис. 4.2. Принцип «связывания» при передаче данных по телефонным линиям

Иногда наши системы получали искажённые данные примерно в одном из пяти вызовов. В других ситуациях у нас бывало более 60 вызовов подряд без единого сбоя. Поскольку причиной подобного искажения могла быть проблема с «привязкой» – мы добавили в систему инструменты для распечатки информации обо всех 6 параллельных вызовах. Это позволило нам обнаружить, что бóльшую часть времени телефонный провайдер коммутировал вызовы в «обычном» порядке – то есть 1–2–3–4–5–6. Но в случаях, когда возникала ошибка, порядок вызовов был «необычным» – например, 1–3–2–4–5–6.

Эта подсказка позволила нам выдвинуть гипотезу, что источник проблемы связан с обработкой внеочередных вызовов. Мы распечатали ещё больше информации о работе системы – и нашли ошибку в коде. Когда мы исправили её – то провели множество тестовых прогонов, не обращая внимания на вызовы с «обычным» порядком. Когда произошла «необычная» последовательность вызовов, а наша система отработала без каких-либо ошибок – мы поняли, что наше исправление помогло.

В этой забавной истории проявление ошибки было связано с трафиком телефонной компании – то есть частота проявления зависела от времени суток и созвонами подростков, живших в окружающих городах. Ожидание повторения ошибки после внесения изменений было ненадёжным вариантом – например, мы могли попасть в период «телефонного молчания». Но как только мы смогли связать проявление ошибки с порядком вызовов – мы больше не зависели от настроения местных подростков.

4.8. «Но этого не может быть»

Если вы какое-то время работали с инженерами – то слышали эту фразу. Тестировщик или наладчик сообщает о проблеме, а инженер поднимает голову, на мгновение задумывается и говорит: «Но этого не может быть».

Иногда инженер прав – тестировщик ошибся. Но обычно тестировщики не ошибаются, и проблема действительно существует. Однако часто даже в этих случаях инженер всё равно прав: «этого» не могло случиться.

Ключевое слово здесь – «этого». Что под ним подразумевается? Скорее всего – механизм проявления ошибки, который, *как предполагают* тестировщик или инженер (или оба), является причиной проблемы; или последовательность событий, которая *кажется* ключом к воспроизведению проблемы. И, на самом деле, возможно, что «этого» действительно не происходит.

Но ошибка всё же возникла. Какая точная последовательность действий была произведена при тестировании и какова причина ошибки – ещё не ясно. Следующий шаг – отбросить любые предположения и заставить ошибку повториться в присутствии инженера. Это докажет, что последовательность действий, позволяющих воспроизвести ошибку, известна, и инженеру останется взять свои слова о невозможности происходящего назад или же попробовать провести другие тесты, которые позволят понять, как *на самом деле* проявляется ошибка.

Байка ветерана. Клик и Клак, ведущие радиопередачи Car Talk, однажды загадали слушателям интересную загадку.

Один из слушателей пожаловался им, что его Volare 1976 года выпуска плохо заводится каждый раз, когда он и его семья покупают мороженое определённого вкуса. Они часто заходили в местное кафе и покупали ванильное, шоколадное или со вкусом тофу и мяты. Когда они покупали ванильное или шоколадное – то машина заводилась отлично. А вот в случае мороженого со вкусом бобового тофу – машина заводилась только после нескольких попыток и ужасно тарахтела во время езды.

Разгадка была вот какой: ванильное и шоколадное мороженое популярны, поэтому их заранее расфасовывали в контейнеры. А вот мороженое вкусом тофу и мяты спросом не пользовалось, поэтому его упаковывали вручную – что требует времени. Этого времени оказывалась достаточно, чтобы старый карбюраторный двигатель Valore, стоявшей на парковке в летнюю вечернюю жару, начинал страдать от паровой пробки.[5]

Вы точно были уверены в том, что вкус мороженого не мог повлиять на работу автомобиля. И вы правы: этого не может быть. Но *покупка мороженого со странным вкусом* сыграла свою роль – и только собрав информацию и изучив ситуацию, вы поймёте, как эти события связаны друг с другом.

4.9. Никогда не выбрасывайте инструменты, разработанные при отладке

Иногда инструменты, разработанные в рамках отладки конкретной проблемы, можно повторно использовать в других ситуациях. Держите это в голове, когда будете проектировать их, и позаботьтесь о том, чтобы они были удобны в обслуживании и доработке. Это подразумевает использование хороших инженерных практик, создание документации и т. д. Добавьте эти инструменты в вашу систему контроля версий. Встраивайте их в своё ПО, чтобы они всегда были под рукой. Не создавайте «одноразовый» инструмент, который будет выброшен после решения конкретной проблемы – вполне возможно, он ещё окажется вам полезен, и решение выбросить его будет ошибочным.

Иногда инструменты, разработанные в процессе отладки, оказываются настолько полезны, что их можно продавать. Многие компании изменили свою модель бизнеса после того, как обнаружили, что разработанные ими для собственных нужд инструменты оказались более востребованными, чем продукты, созданные с помощью этих инструментов. Инструмент может оказаться полезен за счёт факторов, о которых вы даже не подозреваете – как в продолжении моей истории о видеоигре «Пинг-понг»:

[5] Из раздела Car Talk сайта cars.com, цитируется с разрешения авторов. Оригинал: https://web.archive.org/web/20200925141129/https://www.cartalk.com/radio/puzzler/finicky-volare

Байка ветерана. Я и забыл о своём «призрачном игроке», когда несколько месяцев спустя мы с гордостью демонстрировали прототип игры предпринимателю, который обеспечил нам финансовую поддержку (да, он был инвестором, и, как ни странно, проект оказался успешным). В целом, ему понравилось увиденное, но полностью доволен он не был. Инвестор жаловался, что нет возможности активировать сразу две «тренировочные стенки» (он видел более раннюю версию, в которой отключение обоих paddle-контроллеров приводило к отображению двух стенок, между которыми рикошетил шарик).

Мы были в изумлении. Зачем кому-то такой режим? В смысле, вам ведь нужен хотя бы один контроллер для игры, верно? Инвестор сказал: «Ох, вы говорите как настоящие инженеры. Но чего вам не хватает – так это понимания того, как *продать* игру. Я хочу, чтобы приставка выставлялась в магазинах и привлекала посетителей таким режимом – чтобы можно было смотреть на игру, даже если никто в неё сейчас не играет. Шарик должен летать туда-сюда – и если бы у вас было две стенки, то он бы это делал».

Вы знаете, к чему шло дело; я тоже это знал – единственный из всех присутствующих. С трудом скрывая волнение, я совершенно невозмутимо сказал: «Ну, у меня есть идея, которая может сработать. Давайте попробуем». Я спокойно взял приставку и вышел из переговорной. Как только дверь за мной плотно закрылась, я вприпрыжку рванул по коридору. Я добавил свой «отладочный контур» так быстро, как только мог (и добавил переключатель для его активации, чтобы добиться максимального драматического эффекта). Примерно через четыре минуты я вернулся в переговорную с покер-фейсом на лице и поставил прототип на середину стола. Сначала я сыграл партию сам, используя контроллер, а затем щёлкнул переключателем. Пока ракетка сама двигалась вверх и вниз, идеально отбивая шарик, все присутствующие сидели с выпученными глазами – не только наш инвестор, но и мои коллеги-инженеры. Это был один из самых ярких моментов раннего этапа моей карьеры.

Игра вышла как с «тренировочным режимом», в котором игрок сам отбивал шарик от стенки, так и с «автоматическим», который я продемонстрировал на презентации – его использовали на стендах в магазинах. Она продавалась очень хорошо.

4.10. Памятка

Воспроизведите ошибку

Кажется, что это просто, но если у вас не получится – то отладка будет тяжелой.

- **Повторите это**. Вы сможете опять увидеть ошибку, сфокусироваться на её причине и понять, была ли она исправлена после внесения изменений в систему;
- **Начните с самого начала**. Расскажите автомеханику, что были на мойке перед тем, как стёкла в вашей машине примёрзли;
- **Стимулируйте появление ошибки**. Используйте шланг, чтобы проверить наличие течи в окне;
- **Но не симулируйте окно**. Поливайте именно *ваше* окно, а не какое-нибудь «похожее»;
- **Определите внешние неподвластные вам факторы, из-за которых ошибка воспроизводится нестабильно**. Меняйте в приборе всё, что можете – трясите его, катайте и крутите, пока он не задребезжит;
- **Сохраняйте всю информацию о тестовых прогонах и ищите в ней признаки несистематично проявляющихся ошибок**. В нашей системе видеоконференцсвязи сбой возникал только при «необычном» порядке вызовов;
- **Не слишком доверяйте статистике**. Казалось, что проблема проявляется только в определённое время суток, но в действительности она зависела от желаний подростков, массово решивших почесать языком;
- **Держите в голове, что «это» могло случиться**. Даже вкус мороженого может иметь значение;
- **Никогда не выбрасывайте инструменты, разработанные в процессе отладки**. «Автоматический режим» для видеоигры ещё может когда-нибудь вам пригодиться.

Глава 5: Не предполагайте, а смотрите

У меня пока нет никаких данных. Теоретизировать, не имея данных, опасно. Незаметно для себя человек начинает подтасовывать факты, чтобы подогнать их к своей теории, вместо того чтобы обосновывать теорию фактами.

Шерлок Холмс, «Скандал в Богемии»

5.1. Общий обзор

Байка ветерана. Наша компания разрабатывала плату, которая подключалась к ПК и имела на борту собственный процессор со встроенной памятью, работающий по отношению к ПК в режиме slave (см. рис. 5.1). Прежде чем ПК (работающий в режиме host) запускал slave-процессор – он (ПК) должен был загрузить данные в его память; это осуществлялось за счёт специального механизма, поддерживаемого платой, с помощью которого ПК мог обращаться напрямую к памяти slave-процессора. После завершения записи slave проверял память на наличие ошибок (с помощью магии, называемой «контрольной суммой»). Если ошибок не было – то slave-процессор запускался в работу. В случае обнаружения ошибок slave отправлял сообщение ПК, и, учитывая наличие ошибок в памяти, ПК не пробовал запустить программу slave'а (очевидно, что из этого slave-процессора вышел бы плохой политик).

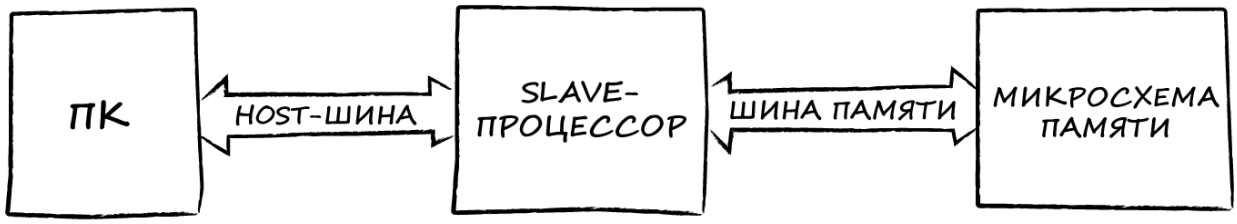


Рис. 5.1. Проблемная система

Проблема заключалась в том, что иногда после загрузки данных в память slave-процессор отправлял сообщение об ошибке и прекращал работу[6]. Это повторялось примерно при каждой десятой загрузке и только в некоторых системах. И, конечно же, для воспроизведения ошибки было достаточно, чтобы лишь один байт из 65 000 имел неправильное значение. Поиск и устранение проблемы поручили паре начинающих инженеров-«железячников».

Сначала они написали тестовую программу для ПК, которая записывала по хост-шине значение в один из регистров slave-процессора, а зачем вычитывала его. Они запускали этот «петлевой» тест миллионы раз, и считанные значения всегда соответствовали записанным. Поэтому они сказали: «Ну, мы проверили обмен между ПК и slave'ом, и он работает без сбоев, так что, видимо, проблема в обмене между slave'ом и его памятью». Они изучили интерфейс памяти (пытаясь понять принцип его работы, что, конечно, было правильным подходом) и поняли, что существующие тайминги

[6] Видимо, речь о том, что загрузка данных производилась в «лабораторных» условиях и всегда должна была проходить корректно (прим. переводчика).

обмена сложно назвать приемлемыми. «Ну и дела» – сказали они, – «Возможно, времени удержания адресов памяти недостаточно, хотя предполагалось, что это будет «прозрачный» (glueless) интерфейс, не требующий дополнительной логики» (младшие инженеры действительно так говорят – правда, обычно используют более крепкие слова, чем «ну и дела»).

Они решили исправить тайминги и приступили к разработке небольшой платы, которая бы подключалась к разъему slave-процессора. Плата должна была содержать исходную микросхему памяти и некоторую логику управления таймингами (см. рис. 5.2). Проектирование и изготовление этой платы заняло много времени: они вручную собирали макетную плату, микросхема памяти была сложной, и в подключении контактов сначала были допущены ошибки. В конце концов, им удалось завершить работу над своей платой – и спустя несколько месяцев после начала отладки они смогли, наконец, проверить, решит ли она проблему. Но проблема никуда не исчезла. Периодически slave-процессор всё равно возвращал ошибку о несоответствии контрольной суммы.

Рис. 5.2. Вот так младшие инженеры попробовали решить проблему

Наш старший инженер был весьма обеспокоен происходящим, потому что никто так и не понял, что именно является причиной проблемы. После неудачи с дополнительной платой он настоял на том, чтобы мы увидели, как именно в памяти оказываются некорректные данные. Он сам взялся за дело, достал продвинутый логический анализатор, аккуратно подключил его к шине памяти и попытался выяснить, почему в памяти оказываются «испорченные» значения. Честно говоря, он не увидел никаких проблем с характеристиками сигналов, но было сложно определить, являются ли записываемые данные корректными, потому что они генерировались программно и выглядели случайными. Поэтому он написал тестовую программу, которая циклически записывала в память определённый паттерн данных – «00 55 AA FF». Старший инженер ожидал увидеть искажение одного или нескольких передаваемых байт – например, «00 54 AA FF». Но в действительности он увидел «00 55 55 AA FF» – то есть дело было не в искажении байтов, а в повторной передаче одного из них.

Старший инженер вернулся к исследованию работы host-шины, подключил осциллограф, проверил несколько сигналов, и обнаружил, что на шину действует импульсная помеха, источником которого является другая микросхема исходной платы. В случайные моменты времени уровень помехи был настолько высок, что slave-процессор воспринимал его как второй импульс записи (см. рис. 5.3.).

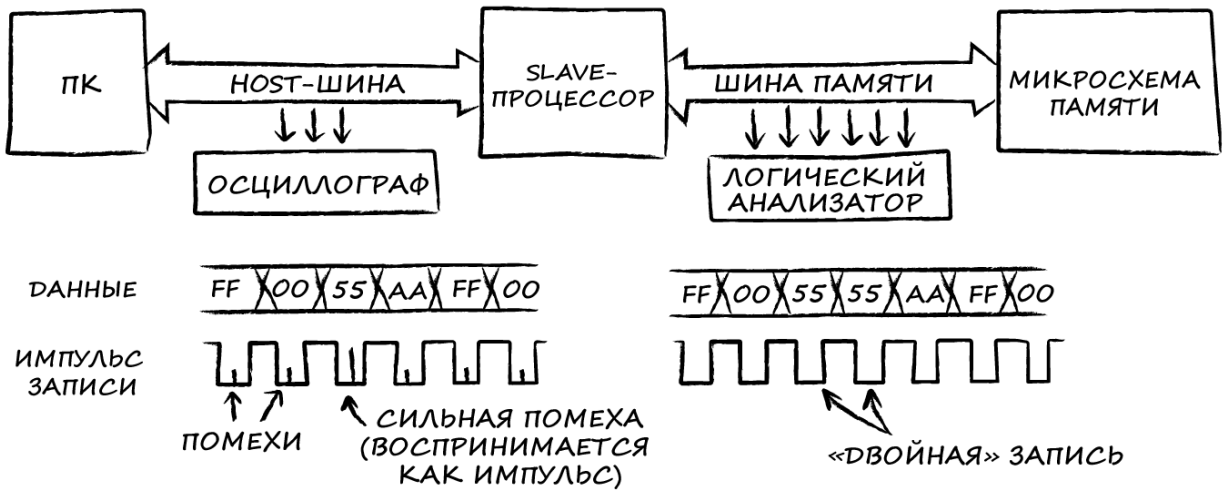


Рис. 5.3. Вот что увидел старший инженер

Когда вы записываете значение в один регистр и после этого его считываете (как в исходном тесте младших инженеров) – то появление двух импульсов записи просто приводит к «двойной» записи одного и того же значения. Само значение остается корректным. Но когда вы загружаете данные в память через slave-процессор, и каждая запись представляет собой обращение к очередной ячейке памяти, то наличие двух импульсов записи вместо одного приводит к тому, что в соседних ячейках оказываются «продублированные» значения – а это, в свою очередь, ведёт к ошибке контрольной суммы. Мы потратили несколько месяцев, гоняясь не за тем, потому что *догадывались* о причинах сбоя, а не *смотрели* на него.

На самом деле, очень важно увидеть, как проявляется сбой на «нижнем» уровне системы. Если вы только *догадываетесь* о его причинах – то часто исправляете то, что не является ошибкой. Это «исправление» не только не работает, но и требует времени и денег на внедрение, а также может сломать что-то ещё. Не занимайтесь подобным.

«Не предполагайте, а смотрите». Этот совет по отладке я даю инженерам чаще всего. Иногда мы дразним того, кто выдвигает гипотезу, которая выглядит очень правдоподобной, но рассыпается в пыль при дальнейших исследованиях, говоря: «Ну, он мыслитель». Все инженеры – мыслители. Инженерам нравится думать. Это очень увлекательное занятие и, определённо, оно лучше физического труда; именно поэтому мы вообще стали инженерами. И ходя мы придумываем всякие изящные вещи, существует столько способов допустить при этом ошибку, сколько не может представить даже самый изобретательный инженер. Так почему же мы считаем, что можем решить проблему, просто обдумав её? Потому что мы инженеры, и нам *легче думать*, чем смотреть.

Смотреть тяжело. Как и старшему инженеру в примере выше, вам нужно подключить логический анализатор и осциллограф, что довольно сложно, особенно если микросхемы на плате расположены так плотно, что нельзя просто закрепить щуп. Вам придётся припаивать провода к крошечным контактам, настраивать логические анализаторы и разбираться с их продвинутыми триггерами (честно говоря, я размышлял о том, не слишком ли длинна эта байка и стоит ли начинать главу с неё, но всё равно решил использовать именно этот пример, потому что он помогает понять суть. Поиск причины ошибки часто (если не всегда) сложнее, чем хотелось бы). Применительно к

ПО поиск причины ошибки означает использование точек останова, добавление вывода в лог отладочных сообщений при достижении программой определённых фрагментов кода, мониторинг значений переменных и анализ дампов памяти. Если говорить о медицине – вы сдаёте анализы крови и делаете рентген. Всё это требует существенных усилий.

А путь, который кажется самым быстрым (особенно если вы выполнили Правило 1 и изучили свою систему) – это просто попытаться понять причину ошибки:

- «Ну, это, должно быть, раммафрам, потому что система выходит из строя только тогда, когда я включаю фробниватор»;
- «Я проверил этот случай в симуляторе, и всё было нормально, так что такой проблемы не может быть»;
- и, в случае проблем с памятью: «Тайминги обмена являются неприемлемыми. Нам лучше переразвести плату. Это исправит ситуацию».

Это всё очень простые (и популярные!) утверждения (ну, может быть, за исключением случая с фробниватором), и кажется, они предлагают более лёгкие и быстрые способы найти причину проблемы. У них есть только один недостаток – обычно они не помогают.

Когда-то я работал вместе с весьма сообразительным парнем, который очень гордился своей способностью логически мыслить и знанием внутреннего устройства продуктов, производимых нашей компанией. Когда он слышал о какой-то ошибке, то обычно говорил: «Спорим, что причина проблемы заключается вот в том-то и том-то». Я всегда соглашался с ним на пари. Мы никогда не использовали в качестве ставок деньги, и очень жаль – я бы выигрывал почти каждый раз. Тот парень был умён и действительно разбирался в продуктах, но не смотрел на ошибку и поэтому не имел достаточно информации, чтобы определить её причины.

Когда вы закончите опровергать свои неверные предположения – вам всё равно придётся искать причины проблемы. Объём работы остался тем же, а времени на неё теперь меньше. Это плохо (только если вы не из тех людей, которые считают, что чем раньше начнёшь срывать сроки – тем больше времени будет наверстать упущенное). Ниже я приведу несколько рекомендаций по поводу того, как смотреть на ошибку, прежде чем начать её обдумывать.

5.2. Смотрите на ошибку

Понятное дело, что если вы хотите обнаружить ошибку – то вам нужно увидеть, что она действительно проявляется. Ведь если вы её не видите, то даже не узнаете, что она есть, верно? Но нет. То, что мы видим, является *последствием* ошибки: я щёлкнул переключателем, а люстра не зажглась. Но в чём же заключается сбой? Ток не смог пройти через сломанный выключатель в щитке или через повреждённую спираль лампы накаливания? (или я щёлкнул не тем выключателем?) Вам нужно внимательно присмотреться к ошибке, что увидеть подробности, которые помогут в отладке. В байке со slave-процессором начинающие инженеры сразу решили, что ошибочные данные в памяти связаны с некорректными таймингами, и не стали проверять это. Но если бы они проверили – то не увидели бы взаимосвязи, потому что её не было.

Многие проблемы легко неправильно истолковать, если вы не можете полностью понять, что же происходит на самом деле. В итоге вы пытаетесь исправить то, что кажется вам причиной ошибки, но в действительности это вообще не имеет к ней никакого отношения. Вы не заметили изменение бита, вызова подпрограммы с некорректными аргументами или переполнение очереди, а вместо это исправили какой-то другой фрагмент кода. Вы не устранили ошибку, но могли снизить частоту её проявления, что заставит вас думать, что она устранена. Что ещё хуже – вы можете добавить новых ошибок. В лучшем случае это будет стоить вам денег и времени – как парню, который покупает новый комплект клюшек для гольфа вместо того, чтобы попросить профессионального игрока поставить ему удар. Новые клюшки не помогут, а уроки гольфа обойдутся дешевле. Он проведет множество раундов с дабл богги, пока, наконец, не смирится и возьмётся за учебу.

Байка ветерана. Мой начальник оказал услугу соседу, который зарабатывал на жизнь продажей и установкой насосов для скважин. Сосед считал, что должен отблагодарить его, и пообещал: «Если вам когда-нибудь понадобится новый насос – я подарю вам самый лучший». Однажды, когда мой начальник был в командировке, его жена услышала звуки из подвала – что-то гудело примерно 10 секунд, а потом останавливалось. Это повторялось каждые несколько часов. Поскольку мужа не было – она позвонила соседу, чтобы посоветоваться с ним. «Проблема в насосе!» – ответил он, и уже на следующий день приехал к ней вместе со своей бригадой. Они вытащили старый насос, установили новый и ушли, хорошо поработав и ответив услугой за услугу. Вот только все эти мероприятия привели к образованию осадка в скважине, так что ещё несколько дней пришлось довольствоваться мутноватой водой, слегка разбавленной хлоркой. Да и гудящий звук никуда не делся.

Вечером того же дня мой начальник позвонил жене и узнал о произошедшем.

- Почему вы решили, что проблема в насосе? Давление снизилось?
- Нет.
- На полу в подвале есть лужи?
- Нет.
- Кто-нибудь стоял рядом с насосом, когда были слышны эти звуки?
- Нет.

Оказалось, что перед отъездом мой начальник решил подкачать шины и воспользовался электрическим компрессором из домашнего гаража. Уходя, он забыл его выключить, и по мере того, как из шланга выходил воздух, двигатель компрессора периодически включался, чтобы поднять давление. Старый насос был в порядке; хотя его замена не стоила моему начальнику ни гроша, ему пришлось иметь дело с мутной водой с привкусом хлорки. И, полагаю, что его жене и соседу пришлось выслушать всё, что он думает об их методах отладки.

Байка ветерана. Мой коллега рассказал о проблеме, которая возникла у его компании с их сервером – каждый вечер он перезагружался примерно в одно и то же время. Они изучали логи, но не нашли ничего, что указывало бы на причину перезагрузки. Они отслеживали процессы, которые запускались автоматически, потому что раз проблема проявлялась примерно в одно и то же время каждый вечер – она должна была быть связана с каким-то периодически запускаемым процессом. Но за несколько недель не нашлось никаких зацепок. И вот мой друг решил задержаться на работе допоздна, чтобы лично присутствовать рядом с сервером в момент перезагрузки. Вскоре после 23:00 сервер выключился. Мой друг огляделся и увидел уборщика, который только что выдернул шнур питания из розетки, чтобы подключить к ней свой пылесос. Уборщик не видел в этом никаких проблем, потому что делал это каждую ночь уже в течение нескольких недель. Причина проблемы стала очевидна, когда кто-то посмотрел на её проявление собственными глазами.

Убедитесь, что вы видите, что именно идёт не так. Взгляд на проблему обычно позволяет решить её гораздо быстрее, чем «простой» путь, связанный с догадками и домыслами.

5.3. Смотрите на детали

В прошлом пункте приведены очень яркие примеры – в каждом из них требовалось приложить минимум усилий, чтобы посмотреть на ошибку (хотя наши герои даже этого не сделали). Обычно каждый раз, когда вы изучаете систему, чтобы понять причину ошибки, вы получаете больше информации о происходящем. Это помогает вам определить, какие исследования проводить дальше, чтобы узнать ещё больше деталей. В конце концов, вы получаете достаточно данных, чтобы сопоставить их с устройством системы и определить причину ошибки.

Байка ветерана. Мы разрабатывали ПО для сжатия видео, которое позволяло передавать его между двумя устройствами, используя для этого минимальное количество бит – что важно для каналов связи, которые не могут обеспечить высокую скорость передачи данных. Если есть возможности для сжатия (примеры будут приведены ниже) – то качество сжатого видео должно быть довольно хорошим; если же нет – то картинка будет выглядеть «блочной» и довольно странной. В нашем случае она была даже более блочной и странной, чем мы предполагали, так что мы решили, что что-то здесь нет так.

При сжатии видео используется множество методов для экономии битов; вместо того, чтобы просто отправлять значения всех пикселей (точек) каждого кадра (с частотой 30 кадров в секунду) – нужно устранить всю избыточную информацию. Например, если фон не изменился, то отправляется бит «без изменений», а другое устройство просто продолжает отображать фон предыдущего кадра. Существуют десятки различных, но взаимосвязанных способов, подобных описанному выше; все они влияют на качество сжатия видео и все очень сложны для анализа и внесения изменений. Если бы мы не поняли, что на самом деле происходит в нашей ситуации – то могли бы месяцами выдвигать гипотезы и менять параметры сжатия, пытаясь улучшить качество видео.

Один из методов сжатия, называемый «оценка движения», позволяет определить, переместилась ли часть изображения (например, моя рука) в новое место в следующем кадре – например, если я ей машу (специалисты по сжатию видео *очень часто* размахивают руками). Если обнаружено движение – то можно представить новое изображение в виде небольшого набора бит, указывающих как часть предыдущего кадра («рука») должна быть смещена по оси X и Y в новом кадре. Если же движение не обнаружено, то приходится передавать все биты «руки» целиком, что приводит к снижению качества видео.

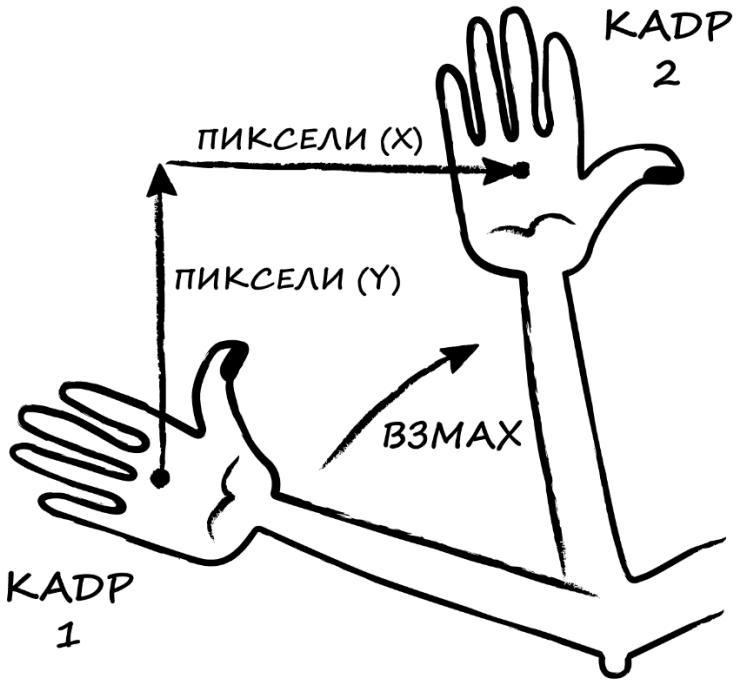


Рис. 5.4 Иллюстрация метода оценки движения

Движущиеся объекты выглядели хуже всего, поэтому мы решили более внимательно изучить код, связанный с оценкой движения. Мы просто хотели посмотреть, определяет ли вообще наше ПО движущиеся объекты. Поэтому мы добавили отладочный код, который отображал обнаруженное движение прямо на экране в виде квадратиков, цвета которых определяли направление движения, а яркость – его «величину». Теперь, когда я махал перед камерой, то видел, что рука покрыта маленькими оранжевыми квадратиками, и чем быстрее я махал – тем более яркими они были. Когда я поднял руку вверх – то квадратики стали фиолетовыми. Когда я махал рукой влево и вправо, то видел зеленые и синие квадратики соответственно, *но их было довольно мало*.

Мы решили разобраться, почему наше ПО так плохо определяет движение объектов влево и вправо. Для этого мы начали выводить подробные «слепки» вычислений движения на отдельный монитор, включая информацию о том, насколько хорошо производится поиск движущихся объектов в кадре. Мы были удивлены, когда увидели, что поиск движения по горизонтали производился лишь в нескольких местах кадра, а остальные пропускались. В алгоритме оценки движения не было ошибок; проблема заключалась в том, что алгоритм не применялся для анализа всех нужных мест кадра. Мы исправили простую ошибку в коде поиска и обнаружили, что движения по горизонтали стали определяться гораздо лучше – и вместе с этим повысилось качество видео.

Насколько далеко следует зайти в изучении ошибки, прежде чем перестать смотреть на неё и снова начать обдумывать? Ответ прост – продолжайте исследования до тех пор, пока не сузите набор возможных причин ошибки до ограниченного (исчислимого) числа вариантов. В примере выше, как только мы поняли, что у нас проблемы с полнотой поиска, то перешли к изучению соответствующего фрагмента кода и обнаружили ошибку. Видимое нами проявление ошибки заключалось в том, что поиск движений происходил не во всех местах кадра. Должны ли мы были пошагово выполнить этот фрагмент кода, чтобы подтвердить ошибку? Вероятно, нет – код поиска был размещён в одной маленькой и простой подпрограмме. Должны ли мы были начать изучать этот код сразу после того, как выясняли, что наше ПО не обнаруживает движение по горизонтали? Определённо, нет – существует множество причин, из-за которых поиск будет работать некорректно даже при анализе каждого фрагмента кадра. Мы бы потратили время на изучение кода, который сравнивает предыдущий и текущий кадр, вместо того чтобы сосредоточиться на коде, связанном с поиском движения. Итак, мы не остановились на достигнутом; мы изучили проблему достаточно глубоко, чтобы понять, что она связана с поиском движения в кадре, а не сопоставлением изображений.

В примере с насосом жена моего коллеги и его сосед приступили к устранению проблемы, даже не зайдя в подвал. Если бы они сделали это – то услышали бы, что звук исходит от компрессора, а не от насоса. Но они сосредоточились на одном предположении, хотя было ещё множество вариантов, из-за которых мог возникнуть подобный звук.

Ваши помощники – опыт и понимание вашей системы. Делая неверные предположения и пытаясь подтвердить их, вы почувствуете, насколько глубоко нужно погрузиться в изучении ошибки в вашем конкретном случае. Вы увидите, когда изучаемый вами сбой затрагивает достаточно малую часть системы. И вы поймёте, что мастерство эксперта по отладке характеризуется не тем, насколько быстро он начинает выдвигать догадки или насколько они оказываются верны, а тем, насколько мало неверных предположений он озвучит.

5.4. То видно, то нет

Наблюдение за ошибкой «на нижнем уровне» имеет ещё одну ценность, значимую для ошибок, возникающих без какой-либо систематики, которую мы уже обсуждали ранее и обсудим ещё раз: если у вас есть детальное представление о сбое и вы думаете, что исправили его, то сможете легко это доказать. Вам не нужно полагаться на статистику; вы сами увидите, что ошибка больше не возникает. Когда наш старший инженер устранил помеху, которая действовала на slave-процессор, он увидел, что импульсы записи больше не удваиваются.

5.5. Добавьте в систему инструменты отладки

Теперь, когда вы решили открыть глаза, вам нужно что-то, что поможет пролить свет на причину проблемы. Вам необходимо разработать инструменты отладки или интегрировать их прямо в вашу систему. Лучше интегрировать; ещё на этапе проектирования позаботьтесь об инструментах, которые помогут вам видеть, что происходит внутри системы. Раз уж вы всё равно добавите в систему ошибки на этапе разработки (это неизбежно) – то добавьте и инструменты отладки. Но поскольку во время разработки системы нельзя предсказать, какая информация будет требоваться в будущем для отладки – то вы обязательно что-то упустите, и вам потребуется либо создавать специальные версии системы с нужными отладочными инструментами, либо разрабатывать отдельные «внешние» инструменты.

5.5.1. Встраивайте инструменты отладки

В контексте аппаратной части устройств это означает контрольные точки, контрольные точки и – ещё раз – контрольные точки. Добавьте сервисный разъём, чтобы обеспечить простое подключение к шине и важным сигналам. В наши дни, когда достаточно часто используются программируемые вентильные матрицы (FPGA) и интегральные схемы специального назначения (ASIC), проблемы часто скрываются внутри тех элементов, к которым невозможно получить доступ с помощью внешних инструментов, поэтому чем больше сигналов вы выведете из микросхемы наружу – тем лучше. Сделайте все регистры доступными для чтения и записи. Добавьте светодиоды и встроенные дисплеи, чтобы видеть, что происходит с вашим прибором. Замечали ли вы, что некоторые ПК могут отображать температуру процессора в окне информации о статусе системы? Это потому что разработчики встроили в них датчик температуры – поскольку процессор обычно полностью корпусирован, то это единственный способ определить степень его нагрева.

В контексте ПО первый уровень инструментов отладки – это та отладочная информация, которая входит в состав разрабатываемого приложения и позволяет наблюдать за его работой с помощью отладки исходного кода. К сожалению, при выпуске релизной версии ПО её часто придётся компилировать без отладочной информации, так что эти инструменты не подойдут для отладки готового продукта. И вы будете вынуждены переходить к следующему варианту (нет, не бросить вашу работу и стать рок-звездой) – он заключается в том, чтобы поместить значения наиболее важных переменных в различные окна мониторинга, которые вы сможете открывать в процессе работы с вашим приложением. В любом случае, добавьте в ваше ПО окно, которое поможет вам при отладке, и пусть в нём отображается информация, позволяющая понять, что происходит в коде приложения. И, конечно же, добавьте возможность сохранения этой информации в виде лог-файлов.

Чем больше информации о состояния приложения вы можете получить – тем лучше, но у вас должен быть способ включать и отключать вывод и логирование выбранных сообщений (или типов сообщений), чтобы вы могли сосредоточить внимание на тех из них, которые нужны вам для отладки конкретной проблемы. Кроме того, вывод отладочных сообщений часто влияет на тайминги приложения, что, в свою очередь, может повлиять на наблюдаемую ошибку (существует распространённая жалоба о том, что включение отладчика замедляет работу системы настолько,

что она перестает давать сбои — «Вот почему они называют это отладчиком». См. чуть дальше пояснение о «Принципе неопределённости Гейзенберга»). Вывод и логирование слишком большого количества отладочных сообщений способны серьёзно нагрузить процессор вашего ПК; когда отклик на каждое нажатие мыши занимает 35 секунд – вы станете раздражительным).

Влиять на вывод отладочных сообщений можно на трёх разных уровнях – на этапе компиляции, на этапе запуска и на этапе выполнения («в рантайме»). Отключение внедрения отладочных символов на этапе компиляции уменьшает размер приложения, но в то же время делает невозможной отладку релизной версии продукта. Возможность включения/отключения вывода информации о запуске приложения легко реализовать, но, скорее всего, она не поможет вам отладить проблему, которая проявляется во время работы системы. Включение/отключение вывода отладочных сообщений, появляющихся в процессе работы системы, реализовать сложнее, но это обеспечивает максимальную гибкость – вы сможете приступить к отладке в любой момент. Если включать/отключать вывод отладочных сообщений можно прямо через графический интерфейс (или конфиг-файлы и т. д.) вашего ПО – вы сможете сообщить клиенту, как это сделать, и производить отладку удалённо (это веская причина проверить корректность текстов ваших отладочных сообщений, а также убедиться, что они не содержат непристойные, политически некорректные или цинично-антименеджерские высказывания).

Формат ваших отладочных сообщений может иметь существенное значение при их последующем анализе. Разбивайте их на столбцы. Один из столбцов должен содержать метку системного времени появления сообщения – достаточно точную, чтобы она могла помочь при отладке проблем, связанных с таймингами. Есть много и других «типовых» столбцов, которые будут полезны:

- название модуля или файла исходного кода, в котором было сформировано сообщение;
- тип сообщения (например, «информация», «ошибка» или «действительно неприятная ошибка»);
- инициалы автора сообщения, чтобы определить, кто работал над кодом, который вывел данное сообщение;
- дополнительная информация, связанная с контекстом сообщения – команды, коды состояний, ожидаемые и действительные значения параметров и т. д. Вся эта информация может вам позже понадобиться.

Используя согласованный формат логов – вы сможете впоследствии фильтровать их, чтобы найти именно те сообщения, которые вас интересуют.

Во встраиваемых системах (где у «компьютера» нет монитора, клавиатуры и мыши) для отладки требуется какой-то способ вывода отладочных сообщений – например, через COM-порт или на подключенный в целях отладки LCD-дисплей. Большинство современных DSP имеют специальные отладочные порты, которые позволяют осуществлять мониторинг работы их RTOS на ПК разработчика. Если микропроцессор встроен в плату, у которой есть экран – используйте его; создайте между ними связь, например, через разделяемую память или регистры сообщений. Чтобы изучать проблемы синхронизации выполнения в устройствах без операционной системы – добавьте несколько аппаратно управляемых пинов, которые вы сможете переключать при входе в подпрограмму и выходе из нее. Вы сможете определить состояния этих пинов с помощью

соответствующих инструментов (например, мультиметра). Эмулятор процессора (ICE) дает вам возможность отслеживать все глупые и вредные действия, совершаемые вашим кодом – даже в тот момент, когда, как он думает, на него никто не смотрит.

Однако будьте осторожны с эмуляторами – хотя они и представляют собой отличный инструмент для отладки ПО, но, как известно, не являются точными копиями процессора, на котором оно будет выполняться. Речь не только о различиях в таймингах и распределении памяти – даже некоторые функции могут не поддерживаться эмулятором. В результате эмулятор нельзя использовать для проверки *аппаратных особенностей устройства.* Однако, когда вы спроектируете плату, которая сможет обрабатывать особенности эмулятора, то получите прекрасный инструмент отладки *ПО для встраиваемых систем*.

Основная мысль в том, что об отладке следует начать размышлять на ранних этапах проектирования вашей системы. Убедитесь, что наличие инструментов отладки является частью требований к продукту. Убедитесь, что отладочные хук-функции являются частью каждой функциональной спецификации и описания API. Сделайте окно с отображением и фильтрацией отладочной информации частью стандартного функционала вашего ПО. И в качестве бонуса: помимо упрощения отладки, размышления об инструментах, которые могут вам понадобиться, позволят вам лучше спроектировать систему и заранее избежать некоторых ошибок.

5.5.2. Добавляйте инструменты отладки по мере необходимости

Независимо от того, сколько времени вы потратили на размышления об инструментах отладки в процессе проектирования, когда вы столкнётесь с реальными ошибками – вам потребуется доступ к информации о работе системы, о которой вы не подумали. Это нормально; вам нужно будет добавить в систему дополнительные инструменты отладки. Но будьте осторожны.

Добавляйте их именно в ту версию продукта (сборку ПО и/или аппаратную ревизию устройства), в которой обнаружена ошибка (чтобы не симулировать её) и только те инструменты, которые действительно вам сейчас нужны. Как только вы добавите инструменты отладки – повторите ошибку, чтобы *убедиться*, что вы использовали нужную версию продукта и что ваши инструменты не повлияли на воспроизводимость ошибки (см. раздел про «Принцип неопределённости Гейзенберга» ниже). Как только вы обнаружите и устраните причину ошибки – уберите все добавленные инструменты отладки, чтобы они не загромождали релизную версию продукта (конечно, вы всегда должны иметь возможность вернуться к «отладочной сборке» – закомментируйте отладочный код или оберните его в директиву #ifdef вместо того, чтобы удалять).

Прелесть таких специально добавленных инструментов в том, что они могут показать всё, что вам нужно.

Байка ветерана. Как-то я работал с FPGA, которая вела себя странно, поэтому я перекомпилировал её, чтобы использовать пару запасных внешних контактов в качестве «отладочных» сигналов. Мне приходилось выполнять компиляцию каждый раз, когда нужно было посмотреть на новый сигнал, но я смог увидеть то, что было нужно, и исправить ошибку.

«Сырые» данные вашей программы часто неудобны для анализа. Это не проблема – добавьте в ваш инструмент отладки обработку данных, чтобы сосредоточиться на важных для вас деталях.

Байка ветерана. У нас была система связи, данные в которой «портились» после нескольких этапов буферизации. Мы не знали, перезаписываются ли данные или удаляются, поэтому добавили операторы отладки для вывода в лог значений указателей буферов памяти. Указатели представляли собой большие шестнадцатеричные числа, означающие адреса памяти. На самом деле, нас интересовали не они, а их разность (разность адресов позволяет определить смещение при обращении к буферу) – поэтому мы добавили соответствующие расчёты и выводили эти значения в лог. После этого легко можно было увидеть, что указатель, используемый для чтения из буфера, периодически спонтанно увеличивался, что приводило к чтению из области памяти, в которую ещё не были записаны данные. Мы проанализировали небольшой фрагмент кода, связанный с этим указателем, и быстро нашли в нём ошибку.

Что следует искать? Ищите данные, которые либо подтвердят ваши предположения, либо укажут на неожиданное поведение, являющееся причиной ошибки. В следующей главе («Разделяйте и властвуйте») вы получите более конкретные советы по стратегии поиска, но на текущем этапе главное понимать, что вам требуется найти нужные детали. Посмотрите на переменные, указатели, содержимое буферов, распределение памяти, тайминги и порядок возникновения событий, флаги семафоров и биты ошибок. Посмотрите на вызовы функций, выходы из функций, на их аргументы и возвращаемые значения. Посмотрите на команды, данные, сообщения в логах, сетевые пакеты. *Узнайте подробности*. В истории с ПО для сжатия видео – мы увидели ошибку в алгоритме поиска движений, выведя информацию о местах кадра, в которых производился поиск.

5.5.3. Не бойтесь углубляться в систему

Я видел совет по отладке уже выпущенного ПО, который гласил: «Поскольку у вас нет доступа к исходному коду, и вы не можете модифицировать приложение, то нужно использовать существующий API и менять местами модули ПО, чтобы определить, в каком из них возникает проблема». Я ненавижу такие советы. Он нарушает правило «не симулировать ошибку». Он написан в предположении, что ваш API и модули имеют продуманную архитектуру. И даже если вы определите «проблемный» модуль, то вы не сможете заглянуть в его код, так что вам придётся предполагать, а не смотреть. Тьфу!

Если в коде есть ошибка, то вам придётся собрать новую версию ПО, чтобы найти её и исправить. Вы должны быть готовы к этому. Соберите отладочную версию – и вы сможете понять, что происходит внутри приложения. Добавьте операторы отладки, чтобы увидеть значения параметров, которые вас действительно интересуют. Используйте правило «Не предполагайте, а смотрите» и затем, когда вы исправите ошибку, оберните все эти отладочные операторы в #ifdef и соберите новую версию приложения с исправленной ошибкой и отключённой отладкой.

В прошлых главах я упоминал, что в некоторых случаях потребуется отправить вашего инженера, снабжённого инструментами отладки, на объект заказчика. Поскольку на объектах обычно используются «релизные» версии ПО, а писать код за пределами офиса сложно, то сервисным инженерам приходится использовать встроенные в приложение или дополнительные инструменты. Но если их оказывается недостаточно – то вам захочется создать симулятор объекта клиента в вашем офисе и попробовать повторить проблему с помощью него. Конечно, вам нужно убедиться, что ошибка повторяется и в таком окружении – чтобы быть уверенным, что ваша симуляция репрезентативна. Если в офисе на симуляторе проблема не воспроизводится – то вам придётся внедрять инструменты отладки в ПО прямо на объекте клиента. Для этого отправьте инженера на Арубу с ноутбуком и модемом. Затем собирайте в офисе тестовые сборки ПО и присылайте ему их по электронной почте.

5.5.4. Разработайте дополнительные инструменты

Если вы не встроили или не смогли встроить инструменты отладки в ваше ПО – то реализуйте их отдельно. В контексте аппаратной части прибора – используйте измерительное оборудование, осциллографы, логические анализаторы, анализаторы спектра, термопары или что-то ещё, что вам потребуется, чтобы получить нужную информацию о работе вашего устройства. Если вы работаете с ПК – вам также потребуются платы расширения. Кроме того, ваши инструменты должны обеспечивать приемлемую для вашей задачи скорость и точность измерений. Низкочастотный осциллограф не поможет вам обнаружить проблемы, проявляющиеся на высоких частотах, а цифровой логический анализатор не покажет помехи и необычные изменения сигналов. С помощью пальца вы можете понять, что микросхема слишком горячая, чтобы к ней прикасаться (и что вы поступили глупо, сделав это), но он не поможет вам определить, превышает ли температура микросхемы допустимую для её корректной работы границу.

Что касается ПО – если вы не можете использовать отладчик, чтобы оказаться «внутри» вашего кода, то в некоторых случаях поможет анализатор, способный подключаться к интерфейсной шине и дизассемблировать инструкции по мере их выполнения. Однако поскольку это приводит вас к языку Ассемблера, то воспринимайте его как последнее средство, если только вы не из той эпохи, когда «корабли были деревянными, а люди – железными» (и испытываете при этом ностальгию по ассемблеру; характерные черты таких людей: помнят наизусть коды таблицы ASCII, могут складывать и вычитать шестнадцатеричные значения и действительно заботятся (to carry) о состоянии флага переноса (carry flag)).

Байка ветерана. Однажды мы использовали видеомагнитофон в качестве дополнительного инструмента отладки для устранения проблемы с отображением видео. Казалось, что система отображала кадры не в порядке их следования, но с помощью видеомагнитофона мы сделали запись, а потом покадрово её просмотрели – и увидели, что на самом деле система отображала дважды один и тот же кадр, а из-за интерлейса казалось, что изображение «прыгало назад».

5.5.5. Инструменты отладки в повседневной жизни

Врачи измеряют температуру градусником и ищут симптомы рака с помощью рентгена. Электрокардиографы (которые измеряют электрические сигналы в сердце) имеют щупы, которые выглядят так же, как щупы логического анализатора; оба устройства можно было бы разместить в одном и том же корпусе. Нам приходится использовать эти дополнительные инструменты, потому что уже слишком поздно встраивать инструменты отладки в человеческое тело. Но некоторые области современной медицинской науки занимаются изучением того, какие инструменты уже в нас «встроены» – например, генетических маркеров наследственных заболеваний или наличия в крови химических веществ, указывающих на рак простаты (в этом случае использование «встроенных» инструментов гораздо приятнее, чем если бы в вас лезли щупом!).

Производители сантехники встраивают в системы контрольно-измерительные приборы – датчики температуры для котлов, манометры для баков и т. д. (в истории с «гудящим» насосом – если бы жена и сосед посмотрели на манометр, то увидели бы, что с насосом всё в порядке. У воздушного компрессора тоже был манометр, стрелка которого медленно опускалась вниз по мере выхода воздуха, а потом резко поднималась в тот момент, когда компрессор начинал работать).

Чтобы найти причину холода в доме – мы обклеиваем утеплителем оконные рамы и электрические розетки, и ищем, откуда дует сквозняк. Если мы можем себе это позволить – то используем тепловизор для поиска холодных мест. Если у вас прокол в велосипедной шине – то вы поливаете её водой с мылом и ищите пузырьки (при условии, что из неё не торчит здоровенный гвоздь). Вы используете мыльные пузыри, чтобы найти утечку в газовом баллоне вашего гриля, который стоит на заднем дворе; встроенный инструмент (включение гриля) использовать не хочется, потому что это приведёт к взрыву, после которого у вас останется очень много поджаренных стейков (извините за мой чёрный юмор). В природный газ добавляют этилмеркаптан, придавая ему запах тухлых яиц, по единственной причине – чтобы проще было обнаружить его утечку. Обойти пляж с металлоискателем будет проще, чем беспорядочно копаться в песке в поисках старых монет и заколок для волос.

«Только ложка знает, что варится в котле».

Сицилийская пословица

5.6. Принцип неопределенности Гейзенберга

Гейзенберг был одним из пионеров атомной физики. Он изучал *очень маленькие* атомные частицы и понял, что можно достаточно точно определить, либо где находится частица, либо куда она движется, но чем точнее вы измеряете один из этим параметров, тем сильнее влияете на другой. Вы не можете провести достаточно точные измерения обоих параметров, потому что ваши датчики тоже становятся частью системы. Иными словами – ваши инструменты тестирования влияют на тестируемую систему.

Ранее уже упоминалось, что работа отладчика влияет на тайминги приложения. Подобное влияние оказывают любые инструменты отладки. Щуп осциллографа добавляет в цепь ёмкость.

Отладочный код изменяет время выполнения и размер программы. Добавление платы расширения на шину PCI изменяет тайминги шины. Даже простое снятие крышки корпуса ПК изменяет температуру его внутренних элементов.

Это неизбежно (я в этом уверен). Вам просто нужно помнить об этом, чтобы такие эффекты не застали вас врасплох. Кроме того, некоторые методы отладки меньше влияют на систему, чем другие. Поиск полезных ископаемых с помощью дистанционного зондирования безопаснее для окружающей среды, чем раскапывание нескольких квадратных миль земли в поисках угля. Рентген или компьютерная томография менее болезненны, чем исследовательская хирургия, но их может быть недостаточно.

Как упоминалось в главе «Воспроизведите ошибку» – даже незначительные изменения в системе могут привести к тому, что ошибка перестанет наблюдаться (но при этом не исчезнет). Добавление в систему отладочных инструментов – это тоже изменение, так что после него вам необходимо воспроизвести ошибку, чтобы убедиться, что вы не столкнулись с проявлением принципа неопределенности Гейзенберга.

5.7. Используйте предположения только для сужения области поиска

Правило «Не предполагайте, а смотрите» не подразумевает, что вам вообще никогда нельзя делать каких-либо предположений о природе ошибки. Предположения – хорошая вещь, особенно если вы разбираетесь в системе. Ваши догадки могут быть даже близки к истине, но вам следует использовать предположения *только для того, чтобы сузить область поиска причин ошибки*. Вам всё равно придётся подтвердить верность своего предположения, воспроизведя ошибку, прежде чем вносить какие-то изменения в систему.

В истории о ПО для сжатия видео мы *предположили*, что в коде оценки движения есть проблема, и поэтому *начали изучать*, как происходит поиск движений. Мы *подтвердили* свою догадку, когда я помахал рукой влево и вправо, а на отладочном мониторе отобразилось всего несколько синих и зелёных квадратиков. После этого мы *предположили*, что оценка движений работает некорректно, и *изучили код*; наша гипотеза не подтвердилась, но мы нашли ошибку в коде поиска – алгоритм оценки движения не применялся ко всем фрагментам кадра.

Так что не слишком доверяйте своим догадкам; часто они поведут вас по ложному следу. Если окажется, что эксперименты не подтвердили конкретное предположение, то придётся вернуться назад и выдвинуть новую гипотезу (или ещё раз достать доску для спиритических сеансов, или бросить дротик в мишень, на которой расписаны возможные причины ошибки – всё зависит от вашей методологии. Я рекомендую использовать Правило 4: «Разделяйте и властвуйте»). Наш старший инженер из истории про slave-процессор начал с того, что попытался увидеть, как в память попадают некорректные данные, но не смог этого сделать. Зато он нашёл дублирование записываемых байтов и сфокусировался на host-шине, которая связывала slave-процессор с основным процессором ПК – и тут он обнаружил дублирование импульсов записи.

Исключение: одна из причин появления предположений – это то, что некоторые проблемы более вероятны, чем другие (или легче внести исправления для их устранения) – поэтому стоит с

них и начать. Одновременное выполнение обоих условий (вероятность и легкость исправления) – это единственный довод в пользу решения выдвинуть какие-то предположения о причине ошибки и попытаться её устранить, не пытаясь выяснить все детали. Ранее я приводил пример с выключателем – если вы щелкнули им, а свет не загорелся, то можно предположить, что проблема либо в выключателе, либо в лампочке. Более вероятно, что перегорела лампочка; заменить её нетрудно. Сделайте это, и если она загорится – то проблема устранена. Но я готов поспорить, что даже в этом случае вы потрясёте старую лампочку – просто чтобы убедиться, что внутри неё бренчит перегоревшая нить накаливания.

Байка ветерана. У моего друга есть кулер со встроенным нагревателем. Однажды он перестал нагревать воду. Мой друг позвонил в техническую поддержку производителя, и ему сказали, что проблема может быть во внутреннем предохранителе. Поэтому он пошёл в магазин, купил новый предохранитель (странного вида шестидюймовый провод) и с трудом сумел заменить старый предохранитель. Но проблему это не решило. Как позже выяснилось – она возникла из-за автоматического выключателя. Его сброс занял несколько секунд (включая вскрытие панели кулера и поиск нужного выключателя). Мой друг не только ошибся с предположением (точнее, ошиблась техническая поддержка), но и взялся за то, которое было сложнее всего проверить. Он никогда не говорил мне, сколько стоил этот предохранитель.

5.8. Памятка

Не предполагайте, а смотрите

Можно выдвинуть тысячи догадок о причинах ошибки. Но когда вы *посмотрите* на неё – то увидите истинную причину.

- **Смотрите на ошибку**. Старший инженер *увидел*, в каком узле системы возникает ошибка, и поэтому понял, с чем она связана. Младшие инженеры *думали, что знают*, в чём проблема, и пытались починить то, что не было сломано;
- **Смотрите на детали**. Не звоните соседу, когда услышите гудение насоса. Спуститесь в подвал и найдите источник шума;
- **Встраивайте инструменты отладки**. Используйте отладчики исходного кода, логи, сообщения о состоянии системы, светодиодные индикаторы и запах тухлых яиц;
- **Добавляйте инструменты отладки**. Используйте цифровые анализаторы, осциллографы, измерительное оборудование, металлоискатели, электрокардиографы и мыльные пузыри;
- **Не бойтесь углубляться в систему**. Да, это релизная версия вашего ПО, и в ней есть ошибка. Вам придётся открыть исходники, чтобы исправить её;
- **Помните о Гейзенберге**. Не позволяйте вашим инструментам отладки влиять на воспроизводимость ошибки;
- **Предполагайте только для сужения области поиска**. Может быть, с таймингами доступа к памяти и правда что-то не то, но удостоверьтесь в этом перед тем, как начать исправлять их.

Глава 6: Разделяйте и властвуйте

Сколько раз я говорил вам: отбросьте всё невозможное; то, что останется, и будет ответом, каким бы невероятным он ни казался.

Шерлок Холмс, «Знак четырёх»

6.1. Общий обзор

Байка ветерана. Представьте себе современную (*на тот момент*) систему бронирования номеров отеля на горнолыжном курорте с терминалами Macintosh на стойке регистрации, подключенными к серверу базы данных с LISP-машиной, размещенному в соседнем помещении (да, LISP-машины – вы знаете, что такое искусственный интеллект? ИИ? Тренд 1985 года, которому мы помахали на прощание в 1989? По крайней мере, Macintosh прижился и покорил мир персональных компьютеров. Ха! Ха! Но я обо всём этом не горюю).

Сотрудники отеля жаловались, что терминалы работали всё медленнее и медленнее, когда пытались получить записи из базы данных. Один из компьютеров работал совсем медленно (и, похоже, это было постоянным эффектом) и иногда не мог завершить поиск в базе данных, выдавая сообщение об ошибке. В отель был отправлен сервисный инженер, и среди ночи (отель работал, и заниматься отладкой днём не было возможности) он приступил к изучению проблемы. Он понимал, как функционирует система: терминалы были подключены к серверу с помощью последовательных линий связи, и при передаче данных производилась проверка на наличие ошибок в пакете. Если происходила ошибка, то терминал повторял запрос до тех пор, пока не получал корректный ответ, или же выдавал сообщение ошибке, если не мог получить ответ в течение длительного времени. Инженер сделал разумное предположение о том, что из-за ошибок обмена терминалы стали работать медленнее, так как им приходилось по нескольку раз повторно отправлять запросы и ожидать ответов от сервера. Он изучил логи и подтвердил свою гипотезу: в них были зафиксированы ошибки при передаче данных в обоих направлениях (от сервера к терминалам и от терминалов к серверу). Поскольку раньше система работала нормально, и ПО с тех пор не изменялось – то инженер предположил, что проблема связана с аппаратными средствами.

Таким аппаратным средством являлась специальная коммуникационная плата в сервере, к которой был подключен плоский кабель. В этом кабеле были проложены 8 последовательных шин связи. Кабель соединялся с распределительной коробкой, вмонтированной в стену, на лицевой панели которой было 8 разъёмов последовательных портов. Без этой коробки было не обойтись, так как в сервере не хватало места для размещения 8 портов. От коробки отходили 8 последовательных шин до 8 терминалов Macintosh (см. рис. 6.1).

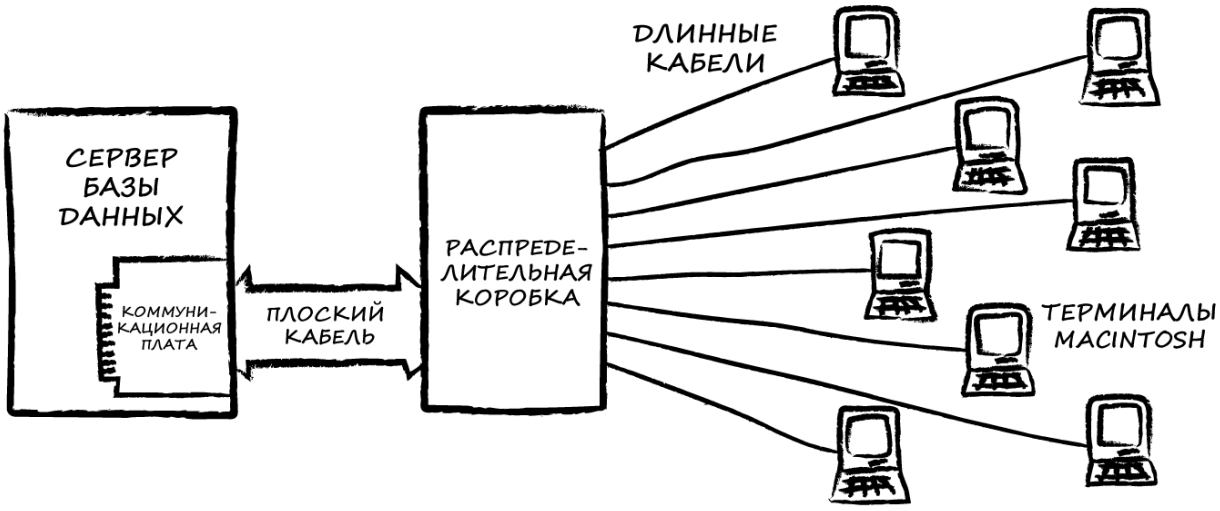


Рис. 6.1. Система бронирования номеров отеля

Инженер не знал, где именно проблема – в коммуникационной плате или проводах, соединяющих распределительную коробку и терминалы. Но поскольку каждый терминал хотя бы периодически успешно связывался с сервером – он предположил, что дело не в проводах. Он подключился осциллографом к точке соединения коммуникационной платы и плоского кабеля (между терминалами и сервером постоянно передавались пакеты типа «Ты на связи?», так что у инженера были сигналы, по которым он мог судить о наличии передачи данных в обоих направлениях). Он обнаружил, что от сервера уходили «хорошие» (с точки зрения физических характеристик) сигналы, а от терминалов приходили «плохие».

С облегчением осознав, что проблема не в сложной и дорогой коммуникационной плате, и удивившись, что она в выглядящем надёжным проводе, инженер подключил осциллограф к разъёмам проводов последовательной шины. На этот раз всё было наоборот: от терминалов в сторону сервера приходили «хорошие» сигналы, а от сервера в сторону терминалов уходили «плохие». Из логов обмена инженер знал, что ошибки проявлялись при передаче данных в обоих направлениях. Теперь он выяснил, что проблема находится где-то между двумя исследованными им точками. Поэтому он стал дальше сужать область поиска, и теперь подключил осциллограф к разъёму плоского кабеля. И снова всё поменялось местами – «хорошие» сигналы от сервера к терминалам и «плохие» от терминалов к серверу. Похоже, что проблема была в распределительной коробке. Чтобы подтвердить это – инженер измерил сопротивление (которое «препятствует» прохождению электрических сигналов) между разъёмами плоского кабеля и разъёмами последовательной шины (т. е. между последними двумя исследованными им точками) и обнаружил, что оно имеет неожиданно высокое значение. После этого он открыл распределительную коробку.

В ней была только плата, соединяющая разъёмы. Измерив сопротивления в различных точках платы, инженер обнаружил высокие значения в тех точках, где к плате были припаяны контакты разъёма COM-порта. Он внимательно изучил это место и увидел микротрещины. С помощью паяльника он расплавил припой и исправил этот дефект. Сопротивление уменьшилось. Инженер повторил этот процесс для всех контактов, подключил всё обратно и осциллографом проверил наличие сигналов. Затем он отключил от коробки все провода, закрыл её, заново подключил их и проверил наличие связи между сервером и терминалами. Это было сделано для того, чтобы учесть

следствие Голдберга из закона Мерфи, которое гласит, что *попытка «собрать систему в единое целое» перед тестированием исправления повышает вероятность, что исправление не устранило проблему, и вам придётся «разбирать» систему заново; при этом вероятность прямо пропорциональна усилиям, необходимым для «сборки».*

Все терминалы теперь работали без задержек – кроме того единственного, который изначально был самым медленным.

Инженер выпил очередную чашку кофе и посмотрел свежие логи по этому терминалу. Теперь ошибки были зафиксированы только в поступающих на терминал сообщениях – а в запросах от терминала их не было. Инженер снова подсоединил осциллограф к точке подключения последовательной шины этого терминала к распределительной коробке и посмотрел на сигналы, уходящие на терминал – они были «хорошими». Он стал спускаться «вниз по реке» и добрался до разъёма COM-порта терминала, с удивлением обнаружив, что один из проводов к нему не подключен.

Инженер изучил электрическую схему, на которой было указано, что кабель последовательной шины является шестижильным – 2 провода нужны для приёма данных, 2 для передачи и 2 ни к чему не подключены (шестижильный кабель был самым дешёвым и доступным, и поэтому проще было использовать его, чем искать четырёхжильный кабель). Монтажник, который прокладывал кабель, вероятно, не посмотрел схему подключения (или делал это при плохом освещении, или был дальтоником) и на одном конце подключил синий провод вместо фиолетового. На другом конце был подключен фиолетовый провод. Коварным образом синий и фиолетовый провода, которые не были соединены друг с другом, но были проложены в одном кабеле длиной 30 метров, сумели наводить друг на друга достаточный сигнал, чтобы терминал работал, хотя и плохо. Как только инженер подключил на обоих концах кабеля фиолетовый провод – терминал заработал идеально.

В этой истории инженер применил ряд правил отладки. Он «изучил свою систему» и использовал полученную информацию, чтобы сосредоточиться на поиске ошибки в её аппаратных узлах. При этом он не просто заменил всё оборудование, а «не предполагал, а смотрел» (конечно, никто не мог предположить, что «медленный» терминал даже не полностью подключен к шине – можно было ожидать, что подобная ошибка вообще сделала бы его работу *невозможной*). Инженер использовал встроенные инструменты отладки (логи) для обнаружения ошибок связи и установления факта, что они происходят при передаче данных в обоих направлениях. Он использовал постоянно передающиеся пакеты («Ты на связи?»), чтобы регулярно «воспроизводить ошибку» и наблюдать её с помощью осциллографа. Он нашёл причину ошибки, измерив сопротивление конкретных точек платы (где к ней были припаяны контакты разъёма COM-порта), размещенной в распределительной коробке, и убедился, что перепайка устранила проблему, повторно проведя измерения. И, когда у него возникли подозрения по поводу кабеля, он начал внимательно изучать его.

Но правило, которое наш инженер особенно хорошо продемонстрировал в этом примере – это «Разделяйте и властвуйте». Он сужал область поиска, деля систему на две части, а потом фокусируясь на той из них, в которой ошибка продолжала проявляться, и повторял эту процедуру до тех пор, пока не добрался до причин проблемы.

Сначала инженер разделил систему на ПО и «железо» и, поскольку он знал, что с течением времени проблема проявлялась всё сильнее, то предположил, что она касается аппаратной части. Затем он изучил характеристики сигналов примерно в середине линий связи. Это укрепило его в мысли, что проблема связана с оборудованием, а не ПО. Он продолжал смотреть на сигналы и, каждый раз, когда видел, что они «плохие» – фокусировался на этой части системы. Когда инженер перестал их видеть (дойдя до коммутационной платы сервера) – то вернулся к предыдущей точке, где они ещё наблюдались. Даже когда он открыл распределительную коробку – то последовательно шёл к цели, измеряя сопротивление в различных её точках (см. рис. 6.2).

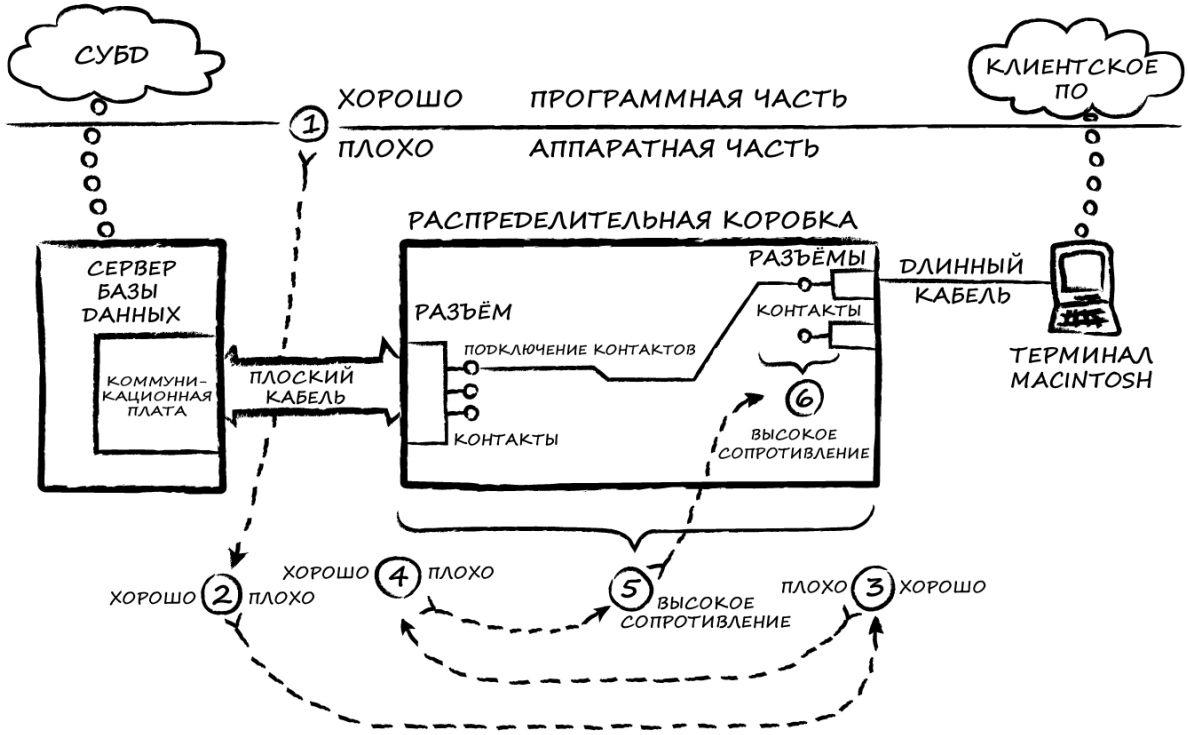


Рис 6.2. Последовательность действий инженера отмечена цифрами

6.2. Сужайте область поиска

Возможно, вы заметили, что «Разделяйте и властвуйте» – это первое из правил, которое касается поиска причины проблемы. Фактически, оно является и *единственным*. Все остальные правила призваны лишь помочь следовать ему. Это правило лежит в основе процедуры отладки, и многие люди интуитивно понимают его. Но многие не понимают – поэтому мне и пришлось написать данную главу.

Сужайте область поиска. Сосредоточьтесь на проблеме. Определите её «дальность» от вас в текущий момент времени. Типовым подходом для поиска заданной цели является метод последовательных приближений: вам нужно найти что-то в известном диапазоне, и вы начинаете с одного из его концов, проходя половину «пути». Если вы не нашли то, что искали – то проходите половину от оставшейся половины – и так далее. На каждом шаге вы определяете «направление

движения» и «перемещаетесь» на половину величины предыдущего «шага». После относительно небольшого числа шагов – вы найдёте то, что ищете.

Пусть ваш друг загадает число в интервале от 1 до 100, и после каждой вашей попытки угадать его говорит, в большую ли вы сторону ошиблись или в меньшую. Вы сможете угадать это число за 7 попыток. Если было загадано «42», то ваши предположения могут быть «50»–«25»–«39»–«44»–«41»–«43»–«42». Если в системе есть 100 мест, где могла возникнуть ошибка, то вы предпочтёте найти её за 7 попыток, а не за 100, 50 или даже 20. В ПО подобный подход используется для быстрого сканирования больших баз данных. Если говорить об аппаратных средствах – то высокоскоростные аналого-цифровые преобразователи (АЦП) калибруются путём проверки соответствия входного напряжения и выходного значения от старшего бита к младшему (каждый следующий бит вносит в значение в два раза меньше «веса», чем предыдущий). Я предполагаю, что канониры прошлых эпох использовали метод последовательного приближения, чтобы как можно быстрее поразить цель: вам хочется поскорее «решить проблему», если она тоже умеет стрелять в ответ.

Байка ветерана. Однажды на летних каникулах я подрабатывал в телефонной компании, на кроссовом узле. Так называется помещение телефонной станции, в которое заводятся телефонные кабели от домов. Кабели подключаются к «вертикали» – линии, состоящей из сотни (или около того) коммутационных стоек, с несколькими сотнями пронумерованных коннекторов на каждой из них. За ними расположена «горизонталь» – это тоже группа стоек с коннекторами, провода от которых идут к коммутационному оборудованию центрального офиса телефонной компании. Чтобы подключить дом к телефонной сети – вы берёте «его» пару проводов из нужной «вертикальной» стойки, протаскиваете их вниз до нужной «горизонтальной» стойки, а дальше подключаете их в нужные разъёмы. Всё в целом это работает как один большой коммутатор, и кабель от любой «вертикальной» стойки можно подключить к любому гнезду любой «горизонтальной» стойки. Это технология древних времён, но она отлично работает.

Конечно, телефонной компании приходится следить за тем, какой дом в какую стойку подключен; иногда кто-то может накосячить и подсоединить провода не в то гнездо (не так просто нанять хорошего работника на лето). И вот вы отсоединили провод от «вертикали», и вам нужно понять, где он подключен в «горизонтали».

Первое, что вы делаете – смотрите, в какую сторону идёт провод по «горизонтальным» стойкам. Затем вы передаёте провод помощнику, и проходите половину его длины вдоль стоек. Вы погружаете свои руки в пучок проводов, идущих вдоль стоек (их тысячи), и пытаетесь ощутить движение вашего провода, пока ваш помощник дёргает его на себя. Если вы ощутили это движение – то найти «ваш» провод будет довольно легко. Если же не ощутили – то провод не был протянут так далеко, и вам нужно отойти на половину пройденного пути обратно. Как только вы почувствовали провод – то хватаете его и начинаете дёргать. Теперь ваш помощник отходит на половину пути. После нескольких перемещений вы находите точку, в которой провод подключается к «горизонтальной» стойке. Это стандартная процедура, хотя, вероятно, использованные мной термины «дёргать» и «ощутить» вызвали бы недовольство профсоюза.

Последовательное приближение основано на двух важных допущениях: вы должны знать «диапазон», в котором производите поиск, и уметь определять, принадлежит ли рассматриваемая «точка» нужной вам «половине». Если вы пытаетесь угадать число в интервале от 1 до 100, а ваш друг загадает 135, или после вашей попытки угадывания откажется говорить вам, в большую или меньшую сторону вы ошиблись, или будет менять задуманное им число после каждой вашей попытки угадывания – то у вас ничего не получится (и вам нужно будет найти нового друга для этой игры).

6.2.1. В зоне обстрела

Определить область поиска проблемы несложно, если вы хорошо представляете себе полное «пространство», которое занимает ваша система. Эта область может быть существенно шире, чем вам хотелось бы, но каждый ваш эксперимент будет сужать её в два раза, что довольно неплохо. Наш инженер начал с рассмотрения всей системы, и на первом этапе разделил её на программную и аппаратные части, после чего выдвинул гипотезу, что проблема связана с аппаратным обеспечением. Если бы после этого он предположил, что проблема в коммуникационной плате сервера – то сузил бы область поиска до плоского кабеля этой платы и не обнаружил бы с ним никаких проблем. Тогда он выругался бы, отчитал себя за такую опрометчивость и снова бы расширил область поиска. Если в игре по угадыванию чисел ваш «друг» загадает 135, то вы довольно быстро дойдёте до 100 и, решив, что этого слишком мало, расширите диапазон.

Далее мы ещё поговорим о том, как область поиска может быть расширена в ходе проверки гипотез.

6.2.2. На какой вы стороне?

На самом деле, нашему инженеру с самого начала крупно повезло – в конкретной точке линии связи он увидел, что проблема возникает при передаче данных в обоих направлениях. Поэтому он мог двигаться в любом из этих направлений.

Однако по мере дальнейшего продвижения он дошёл до точки, где проблемы были только с данными, передаваемыми на терминал. С данными, поступающими от терминала, всё было в порядке. Это типично для большинства ситуаций, в которых требуется отладка:

- в процессе передачи данных из-за бага они портятся;
- программа некоторое время работает нормально, а потом из-за бага вылетает;
- лобовое стекло сначала остаётся чистым, а потом в процессе поездки загрязняется из-за разбившихся об него насекомых[7].

[7] Тут автор обыгрывает различные значения слова «bug» (прим. переводчика).

Вы должны иметь представление об области поиска, уметь делить её на две части и определять, в какой из них проблема проявляется, а в какой нет. Назовём это «вверх по течению» (хорошая, чистая вода) и «вниз по течению» (плохая, грязная розоватая вода с неприятным запахом). Вы ищите канализационную трубу, из которой в реку сбрасывается дурно пахнущая розовая слизь. Каждый раз, когда вы смотрите на какую-то точку реки и видите чистую воду – то делаете вывод, что труба расположена ниже по течению. Если вода розовая и имеет неприятный запах – то нужно двигаться вверх по течению.

Всё это совершенно очевидно, когда речь идёт о поиске канализационной трубы какого-то завода, но как применить это к проблемам программного обеспечения и аппаратных средств? В контексте аппаратных средств и задач по передаче данных «вниз по течению» означает «в следующих точках прохождения сигнала». В контексте ПО это означает «в выполняемых далее фрагментах кода». В данном случае вы можете установить точку останова и дождаться её срабатывания. Если признаков проблемы нет – то ошибка в следующих фрагментах кода. Если признаки есть – то проблема в предыдущих или конкретно в этом фрагменте кода. Если сложный вычислительный алгоритм работает некорректно, то установите точку останова в его середине и проверьте промежуточные результаты вычислений. Если они корректны – то перемещайте точку останова ниже по коду, если нет – то выше.

В моей истории про контроллер управления клапаном (см. главу 3) микропроцессор должен был получить прерывание от интерфейсной микросхемы; «выше по течению» были весы, «ниже по течению» – микропроцессор, а ошибка была связана с интерфейсной микросхемой, расположенной между ними. В истории про сжатие видео (см. главу 5) код оценки движения определял новые места для поиска движений, а потом пытался сопоставить содержимое текущего кадра с предыдущим; мы увидели, что приложение не отслеживает все движения, поэтому проигнорировали логику сопоставления кадров и посмотрели на код, который был расположен выше.

6.3. Использование заранее известных наборов данных

Легко увидеть, как чистая вода становится розовой и дурно пахнущей, но что делать, когда проблема проявляется менее явным образом или используемые в ПО данные настолько случайны, что даже их серьёзное повреждение сложно отследить? Один из способов повысить наблюдаемость проблемы – использовать простые в распознавании наборы данных. В примере с рекой – если в воде грязная и мутная вода, то будет сложно обнаружить в ней розовую слизь. Вы должны устранить грязь и смотреть на чистую воду. В первой истории из главы 5 я рассказывал, как старший инженер не смог сразу обнаружить проблему с повреждением данных, так как они выглядели как произвольный набор байт – и поэтому написал тестовую программу, которая циклически записывала в память определённый паттерн данных («00 55 AA FF»). Это позволило ему легко увидеть, как именно проявляется ошибка. В байке, приведённой в начале следующей главы, я расскажу, как тестовый набор данных, который несколько раз дорабатывали, помог найти ошибку в передаче и обработке аудиопотока.

Во время работы с кодированием видео я часто запускал «шаблонный» видеопоток, в котором цвет изображения плавно менялся, так что ошибки кодирования легко можно увидеть в

виде линий или «гребёнки». Вы видели что-то подобное, если когда-нибудь меняли разрешение экрана своего ПК; когда вы нажимаете кнопку «Тест» – то на экране отображаются шаблонные узоры разных цветов, и все они подписаны, чтобы вы понимали, как они должны выглядеть. Если они выглядят неправильно – то этот видеорежим не поддерживается вашим компьютером.

В истории с оценкой движения я создавал тестовые данные, махая рукой; я знал, в какой части кадра происходит движение и в каком направлении оно производится. Мы могли смотреть на результаты работы алгоритма оценки, зная, что должны получить.

Байка ветерана. В прошлой главе я рассказывал, как мы использовали видеомагнитофон для записи видеопотока с целью его дальнейшего покадрового анализа и обнаружения сбоев в последовательности отображения кадров. Чтобы сделать это – мы направили камеру на вращающийся барабан, состоящий из разноцветных сегментов (он выглядел как кислотная пицца) и совершавший один оборот ровно за 4 секунды. На барабане были пронумерованные отметки, обозначающие 1/30 секунды (время отображения одного кадра). Предполагалось, что в каждом кадре мы будем видеть новую отметку; если порядок отметок нарушался или они повторялись – то мы сразу это замечали.

В другой ситуации нам нужно было понять, почему у нас проблемы с синхронизацией видео и аудио – знаете эту историю, когда речь персонажей не совпадает с движениями их губ? Мы написали программу, которая одновременно делала две вещи: воспроизводила щелчок и меняла цвет изображения с белого на чёрный. Когда мы подали генерируемый ею поток в наш «синхронизатор» - то сразу увидели, что в его работе есть проблемы. Они были заметны даже после сильного сжатия.

В старые времена у микропроцессора Motorola 6800 была инструкция с кодом DD, которая заставляла его переходить его в бесконечный цикл, считывая по порядку данные из каждого адреса памяти (другие инженеры называли эту инструкцию «Halt and Catch Fire» (HCF), но мы обозначили её как «Drop Dead» (потому что DD). Этот режим отлично подходил для определения таймингов «железа» и обнаружения проблем в логике с помощью осциллографа – все адресные шины и тактовые сигналы отображались красивыми циклически повторяющимися прямоугольными «волнами».

Само собой, когда вы начинаете использовать свои наборы данных – то должны быть осторожны, чтобы не изменить условия, от которых зависит ошибка. Если ошибка связана с конкретными данными – то при использовании другого набора вы можете её и не заметить. Используйте правило «Воспроизведите ошибку», прежде чем продолжать отладку.

6.4. Начните с «проблемного» места

Многие системы состоят из существенного числа связанных друг с другом компонентов– подобно тому, как притоки впадают в реку. Вы можете потратить много времени на исследования не имеющих отношения к вашей проблеме «притоков», если начнёте двигаться от «истока». Не делайте этого. Не тратьте время на анализ тех компонентов, которые работают корректно. Начните с «проблемного места» – того, где вода загрязнена дурно пахнущей розовой слизью – и продвигайтесь «вверх по течению». Используйте места впадения «притоков» как контрольные точки – если проблема повторяется и за ними, то вам потребуется проверить каждый «приток», чтобы определить, в каком из них она возникает.

Предположим, ваша печь не включается. Вы можете предположить, что у вас закончилось топливо (это обычное дело) и проверить топливный бак. Топлива достаточно. Если решите убедиться, что у вас нет проблем с подачей топлива – вам придётся проверить, проходит ли оно через все трубы и, в итоге, выходит из форсунки. Вы потратите на это много времени (а ещё испачкаете руки в мазуте). Но вы умны и прочитали эту книгу – поэтому начинаете поиск проблемы с самой печи и быстро убеждаетесь, что топливо есть, а вот с электричеством какие-то проблемы. Вы открываете шкаф автоматики и находите несколько возможных источников проблемы: блок питания, термостат и блок системы контроля пожарной безопасности (СКПБ). Вы могли бы проверить наличие питания – но видите, что вольтметр показывает его наличие, поэтому игнорируете этот «приток». Вы могли бы разобрать термостат, но индикатор температуры показывает корректное значение, поэтому игнорируете этот «приток». Вы могли бы пойти и проверить предохранитель СКПБ, и вы сделаете это, потому что индикатор на блоке СКПБ показывает, что он сработал. Кроме того, датчик температуры, от которого срабатывает предохранитель, прямо над вашей головой – прикручен болтами к самой «жаркой» точке воздуховода. И вот вы вызываете электрика, чтобы он заменил одноразовый предохранитель и перевесил датчик в такое место, температура в котором повысится только при пожаре.

В истории со сжатием видео проблемой было низкое качество сжатого видео. «Выше по реке» был код оценки движения и код сжатия изображения. Мы начали с анализа кода оценки движения и оказались правы – система не обнаруживала всех движений. Если бы во время тестов мы бы видели цветные квадратики в соответствующих направлениях при всех моих взмахах руками, то перешли бы к изучению кода сжатия; мы бы не стали дальше углубляться в код оценки движения. В любом случае, мы не начали с «истока реки» (изучения кода сжатия) – это потребовало бы много времени на проверку корректности преобразований Фурье, кодирования длины серий, кодирования с переменной длиной, квантования и дюжины других сложных в произношении понятий из области информатики, которые использовались для кодирования изображений. Честно говоря, это всё и правда *довольно сложно*, поэтому проверить тот код было бы трудно. Более того, как в итоге выяснялось – этот код работал корректно, так что все эти грандиозные усилия были бы напрасны.

6.6. Исправьте ошибки, о которых вы знаете

Иногда, хотя в это сложно поверить, в системе есть более одной ошибки – как в примере с системой бронирования номеров отеля. Из-за этого становится сложнее использовать правило «Разделяйте и властвуйте» для локализации каждой из ошибок. Поэтому, когда вы обнаружите ошибку – исправьте её до того, как будете искать остальные. Я часто слышал фразу «Ну, тут есть ошибка, но она *не может повлиять* на проблему, которую мы пытаемся обнаружить». Как вы могли догадаться – может, и это происходит часто. Если вы исправите что-то, что работает неправильно, то получите более «чистый» взгляд на другие ошибки. Наш инженер смог обнаружить неподключенный провод после того, как разобрался с высокими сопротивлениями на плате, размещенной внутри распределительной коробки.

Иногда исправление одной ошибки приводит к автоматическому исправлению *другой*; значит, это была одна и та же ошибка.

Более того, если исправление одной из ошибок как-то повлияет на систему – то вам нужно исправить её, прежде чем продолжать отладку. Если в результате исправления что-то сломается – то вы сразу это поймёте, и у вас будет больше времени для решения новой проблемы.

6.7. Устраните «шум»

Следствием предыдущего замечания является тот факт, что определённые виды ошибок *могут вызывать другие ошибки*, поэтому вам следует начать с их устранения. В контексте аппаратных средств – помехи могут вызывать всевозможные трудно обнаружимые и нерегулярно проявляющиеся ошибки. Прежде чем приступать к поиску причин других ошибок – необходимо устранить «дребезг» тактовой частоты, помехи аналоговых сигналов, джиттер и некорректные уровни напряжения; это может привести к исчезновению других наблюдаемых проблем. В контексте ПО – ошибки в синхронизации потоков, случайные повторные вызовы процедур и неинициализированные переменные вносят дополнительные факторы случайности, которые могут превратить вашу работу в ад.

6.8. Памятка

Разделяйте и властвуйте

Насекомому[8] трудно спрятаться, когда его укрытие постоянно разрезают пополам.

- **Сужайте область поиска методом последовательных приближений**. Угадайте число в диапазоне от 1 до 100 за 7 попыток;
- **Определите область поиска**. Если загаданное число – 135, то вам придётся расширить диапазон;
- **Определите, на какой вы стороне**. Если вы видите в реке дурно пахнущую розовую слизь – то канализационная труба выше по течению. Если слизи нет – то труба ниже;
- **Используйте заранее известные наборы данных**. Сделайте воду чистой и прозрачной, чтобы слизь было легко обнаружить;
- **Начните с «проблемного» места**. В вашей системе слишком много нормально функционирующих компонентов, чтобы все их проверять. Начните с того места, в котором проявилась ошибка, и найдите её причину;
- **Исправьте ошибки, о которых вы знаете**. Баги прикрывают друг друга. Как только найдёте какую-либо ошибку – сразу её исправьте;
- **Устраните «шум»**. Следите за вещами, которые могут довести вашу систему до безумия. Но не занимайтесь устранением несущественных проблем или эстетическими изменениями.

[8] Автор опять обыгрывает буквальный перевод слова «bug» (прим. переводчика).

Глава 7: Вносите по одному изменению за раз

Говорят, будто гений – это бесконечная выносливость. Довольно неудачное определение, но к работе сыщика подходит вполне.

Шерлок Холмс, «Этюд в багровых тонах»

7.1. Общий обзор

Байка ветерана. Однажды на выходных мы попросили специалиста по отладке помочь одному из наших инженеров-программистов. Инженер изо всех сил пытался отладить систему, которая пропускала аудиопоток через аппаратные и программные обработчики (часть программных обработчиков была разработана другой компанией), и в итоге отправляла их на колонки компьютера. Естественно, качество звука было плохим – иначе оба инженера занялись бы на выходных более весёлыми делами. По мере передачи аудиоданных по системе, они упаковывались в «кадры», в заголовки которых помещалась информация о начале аудиофрагмента и его типе. В какой-то момент эти заголовки удалялись; одни компоненты системы знали, что данные упаковываются в «кадры», а другие – нет (см. рис. 7.1). Спустя некоторое время инженер догадался, что в одном конкретном месте системы нет заголовка, и добавил его. Но звук от этого лучше не стал.

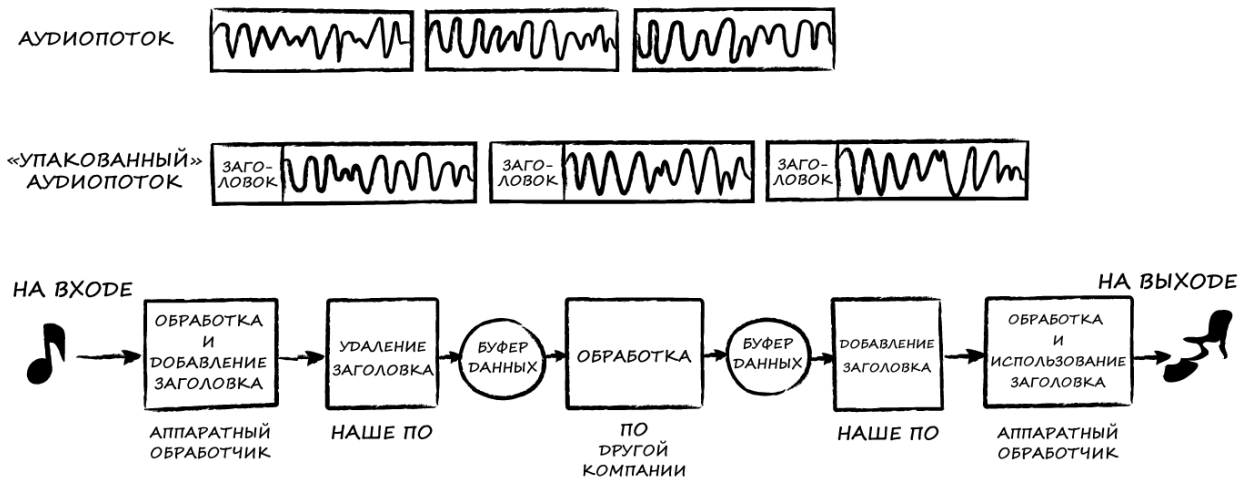


Рис. 7.1. Генерация искажённого звука

На помощь пришёл гений отладки. Он сразу настоял на том, чтобы пропустить по системе заранее известный аудиопоток, и отследить момент его «повреждения», используя инструменты мониторинга. Потребовалось некоторое время, прежде чем наши ребята сгенерировали подходящие тестовые данные и проверили все нужные места, где мог произойти сбой. В итоге они увидели его собственными глазами – в одном из фрагментов кода использовался некорректный указатель на буфер. Они исправили ошибку, проверили, что тестовые данные были корректно

обработаны и воспроизведены с нормальным качеством звука, и с лёгким сердцем повторили тест с реальным аудиопотоком. Он все ещё звучал плохо.

Инженеры растерянно посмотрели друг на друга и стали повторно изучать поток данных, проходящих по системе. Может быть, внесённое ими исправление было некорректным? Может быть, оно не применилось? Они потратили час на подтверждение того, что их исправление действительно было уместным и решило конкретную проблему. Они вернулись к коду, чтобы ещё раз подтвердить, что внедрили своё исправление, и тут инженер-программист хлопнул себя по лбу: «Я изменил обработчик в одном фрагменте кода, чтобы добавить упаковку данных в кадр вместе с заголовком! Это не решило проблему, но я не откатил это изменение!». Оказалось, что следующие обработчики интерпретировали этот заголовок как аудиоданные и воспроизводили его через колонки – поэтому звучание было плохим. Инженер убрал своё первое «исправление», и система заработала идеально. Гений отладки процитировал правило «Вносите по одному изменению за раз» и передал эту историю мне, чтобы я включил её в книгу.

Наш инженер-программист внёс исправление, пытаясь устранить проблему, и когда это не привело ни к каким изменениям – то решил, что оно ни на что не влияет. Это была плохая гипотеза – исправление ухудшало качество звука. Дело в том, что ухудшение звука из-за двух разных ошибок сложно отличить на слух от ухудшения звука из-за одной ошибки. Когда первоначальная ошибка была устранена – то внесённое ранее исправление продолжало ухудшать звук. Инженеру следовало бы убрать это исправление, как только он заметил, что оно не решило проблему.

Предположим, что однажды вы решаете поехать на работу на машине жены (ваша машина, например, на техобслуживании), но она не заводится. Вы замечаете, что она на «паркинге» и догадываетесь, что для того, чтобы завести её – нужно переключиться на «нейтраль». Вы пробуете это сделать, но машина всё равно не заводится. Вы смущённо спрашиваете жену о том, что случилось с машиной, а она отвечает, что ключ зажигания довольно капризный и вам нужно сильнее его поворачивать. Вы поворачиваете его настолько сильно, что едва не ломаете, а машина всё равно не заводится. Но она завелась бы, если бы вы переключились обратно на режим «паркинга», прежде чем тестировать эту мужественную технику поворота ключа.

7.2. Используйте винтовку, а не дробовик

Вносите по одному изменению за раз. Вы слышали о методе дробовика. Забудьте о нём. Купите себе хорошую винтовку. Это позволит гораздо быстрее исправлять ошибки. Я знал инженеров, которые чинили платы путём замены компонентов; они могли сразу поменять 3 или 4 компонента – и обнаружить, что ошибка больше не проявляется. Это, конечно, круто, вот только они так и не могли сказать, в каком именно компоненте была проблема. Хуже всего, что такой подход может добавить новых ошибок.

Кроме того, если бы вы *действительно видели*, что именно в системе происходит не так – то вам было бы достаточно внести одно конкретное исправление. Если вы думаете, что для

попадания в цель вам требуется дробовик – то ваша проблема в том, что вы не можете чётко увидеть свою цель. Что вам действительно нужно – лучшее освещение и новые очки.

Медицинские исследователи часто используют братьев и сестёр (а иногда и близнецов), чтобы локализовать изучаемые факторы. Однояйцевые близнецы генетически идентичны, и любые различия между ними обусловлены факторами окружающей среды. Социологи могут использовать приёмных детей, считая семью «константой» и предполагая, что любые различия между ними и родными детьми обусловлены генетикой. Когда ваш стоматолог пытается найти чувствительный к холоду зуб – он последовательно направляет поток холодного воздуха на каждый из зубов, используя «метод винтовки»; он не запихивает вам в рот кусочки льда. А во времена гирлянд с последовательным подключением лампочек, когда вся цепочка гасла при неисправности любой из них, мы меняли по одной лампочке за раз, пока гирлянда не загоралась снова. Если заменить их все – то осталась бы дорогостоящая куча лампочек, в которой неисправна была только одна.

Это типовой научный подход; чтобы увидеть влияние конкретного фактора на объект наблюдения – учёный пытается контролировать все остальные факторы, которые могут воздействовать на результат. В популярном эксперименте по выращиванию растений из начальной школы вы используете идентичную почву, одинаковый график полива и одни и те же семена, но варьируете количество солнечного света, которое получают растения. Если вы обеспечите им одинаковое количество света, но будете варьировать получаемое количество воды – то разница между растениями будет обусловлена именно этим. Если небольшой размер выросшего растения – это ошибка, которую вы хотите исправить, и вы измените цвет горшка, график полива и количество солнечного света – то не поймёте, что цвет горшка не имеет никакого отношения к вашей проблеме.

Если вы работаете над программой расчёта ипотечного кредита, которая иногда работает некорректно, то задайте фиксированные сумму и срок кредита, и меняйте значение процентной ставки. Если ошибок не будет – то задайте фиксированные срок кредита и значение процентной ставки, и меняйте значение суммы. Если ошибок не будет – то меняйте только срок кредита. Вы либо обнаружите ошибку в своих расчётах, либо наткнётесь на нечто удивительное – например, ошибку математических вычислений в процессоре Pentium («Но этого не может быть»). Подобные сюрпризы обычно проявляются у сложных в отладке ошибок – и именно они делают их сложными. Фиксирование и контроль значений переменных — это что-то вроде помещения известных данных в систему: помогает увидеть «сюрпризы».

Байка ветерана. Я разрабатывал ПО для сторонней платы, которая позволяла захватывать сигнал с VGA-выхода ноутбука, чтобы его можно было передать и отобразить на другом компьютере. Поскольку VGA поддерживает различные разрешения и тайминги, и нет тактового сигнала, с помощью которого можно было бы понять, что сейчас происходит, нужно было сэмплировать поступающее видео, определить его края (чёрные полосы), определить размер изображения в пикселях и измерить, в какие моменты плата захвата обрабатывает конкретный пиксель.

Это было сложно, потому что для каждого измерения нужно было задать предполагаемое значение тайминга, затем сравнить его с реальным, и использовать эту информацию для перерасчёта предполагаемого тайминга следующего измерения. Когда я запустил плату первый раз – то, конечно, меня ждал полный провал (но мы всегда стараемся довести дело до конца, не так ли?).

Необходимость каждый раз рассчитывать тайминги затрудняла анализ полученных мной результатов, но, похоже, проблема была связана с определением того, какой пиксель сейчас обрабатывается.

Я снова запустил свой код, но все измерения были опущены и вместо них использовались значения по умолчанию – за исключением параметра задержки (фазы) момента выборки пикселя. Я менял его вручную и прошёл по всем 8 точкам (см. рис. 7.2). Когда происходил переход от одного пикселя к другому – то моё приложение осуществляло сэмплирование видео на 1 такт раньше. Я ожидал, что выходное изображение сместится на один пиксель влево. Вместо этого оно сместилось вправо, а спустя несколько точек – влево. Я не понимал логики, но поскольку я менял значение только одной переменной – то знал, что наблюдаемое поведение связано именно с ней.

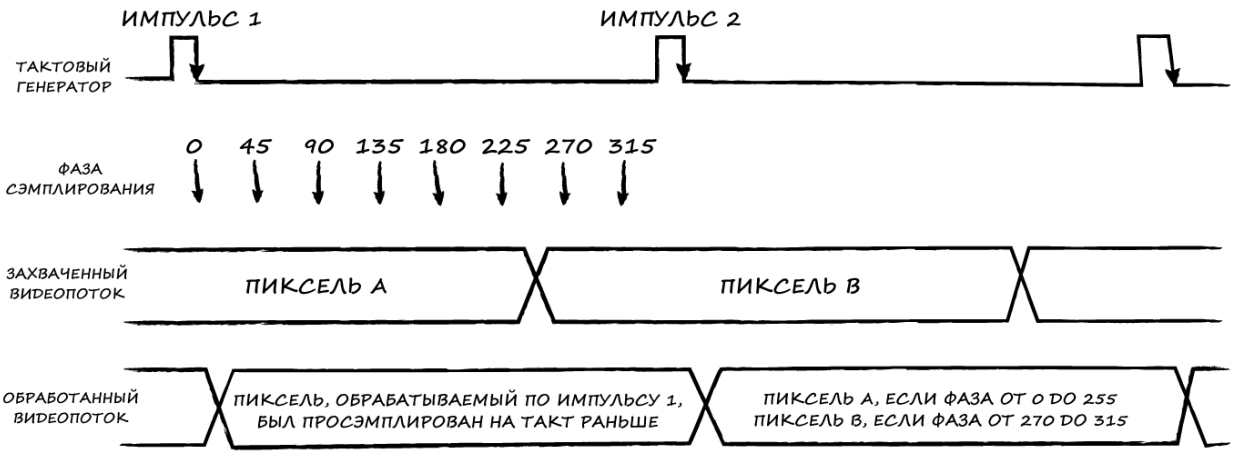


Рис. 7.2. Поиск границ пикселей

Я обратился за консультацией к производителю платы. Он изучил мою проблему и обнаружил в своей документации ошибку в описании параметра фазы. Когда в своём коде я менял его значение с 0 на 359, то в действительности он сначала устанавливался в значение 89, затем в -270, и потом увеличивался вплоть до -1. Изображение сместилось вправо, когда фаза «отскочила назад», а потом сместилось влево, когда позже была пересечена граница пикселя на пути к исходной позиции.

Я бы не поверил, что параметр фазы обрабатывается некорректно, если бы не зафиксировал значения всех остальных переменных, а потом присваивал этому параметру известные мне значения. Поскольку менялось только значение параметра фазы – то именно он *должен был являться* причиной смещения пикселей.

7.3. Держитесь за поручень обеими руками

Во множестве ситуаций вам захочется внести изменения в различные фрагменты системы, чтобы посмотреть, повлияет ли это на проблему. Обычно это признак того, что вы начинаете гадать вместо использования своих инструментов отладки для более детального наблюдения за ошибкой. Вы меняете условия, в которых проявляется проблема, вместо того чтобы наблюдать за её «естественным» проявлением. Это может скрыть исходную ошибку и создать новые. Именно так поступил инженер в истории про кодирование аудиопотока из самого начала главы.

На атомных подлодках перед панелью управления энергоустановкой имеется латунный поручень. Инженеров учат, что при срабатывании сигналов тревоги надо хвататься за этот поручень обеими руками и не отпускать его, пока они не осмотрят индикаторы всех приборов не поймут, что именно происходит в системе. Это помогает им преодолеть искушение сразу приступить к ремонту, начав щёлкать переключателями и открывая клапаны. Такие быстрые действия мешают работе систем автоматического восстановления, скрывают исходную неисправность и могут привести к новым ошибкам, за которыми последует настоящая катастрофа. Гораздо эффективнее помнить о том, что нужно что-то сделать («Держитесь за поручень!»), чем помнить что-то, чего делать не следует («Не трогайте этот вентиль!»). Итак, держитесь за поручень обеими руками!

Байка ветерана. Однажды на рождественской вечеринке в нашем общежитии один из студентов (по прозвищу «Растяпа») установил в гостиной свою стереосистему, чтобы порадовать всех праздничной музыкой. Он протянул провод от правой колонки вдоль стены гостиной, а когда добрался до камина (которым никто никогда не пользовался – но в тот день в нём уже лежали аккуратно порубленные поленья) – то пробросил провод между поленьев. В какой-то момент кто-то разжёг огонь, и вскоре после этого в тёплом свете рождественского пламени правый динамик отключился. Изоляция проводов расплавилась и закоротила правый канал усилителя, из-за чего в нём перегорел предохранитель правого канала. Но студенты, занятые распитием эгг-нога, об этом не подумали; они не собирались воспользоваться мультиметром. Вместо этого они решили выяснить, в чём проблема – в колонке или усилителе – поэтому поменяли местами провода усилителя между правой и левой колонкой. Когда они включили усилитель – то, естественно, левый канал закоротило и его предохранитель тоже перегорел. Теперь ни одна из колонок не работала, и вечеринка прошла без музыки. А ведь обещали! Было бы лучше, если бы они остались у барной стойки со стаканами эгг-нога (и держались бы за них обеими руками).

7.4. Тестируйте одно изменение за раз

Иногда изменение последовательности тестов или значения какого-нибудь рабочего параметра приводит к более частому повторению проблемы; это поможет вам чётче увидеть, что именно происходит, и даст дополнительные подсказки. Но вам всё равно нужно вносить только одно изменение за раз, чтобы точно понимать, какой параметр повлиял на происходящее. И если изменение не оказало никакого эффекта – то сразу уберите его!

7.5. Сравните «успешные» и «неуспешные» тестовые прогоны

Как только у вас появится метод для воспроизведения проблемы (даже если он сводится к ожиданию её проявления в случайный момент времени) – вы получите чудесную возможность стать «машиной различий» (будьте машиной различий! Познавайте мир! Или, хотя бы, ищите отличия!) Проведите несколько тестовых прогонов – в ходе некоторых из них ошибка должна повториться, а в ходе других – нет. Сравните осциллограммы, значения переменных в контрольных точках, отладочный вывод, логи и любую другую информацию, которая вам доступна. Множество раз я находил причины ошибок, построчно сравнивая два журнала логов – один из которых соответствовал «успешному» тестовому прогону, а второй – неуспешному. «Ого! Ну ты посмотри! Во время успешного видеозвонка в поле номера было установлено значение 1-700-VID-TEST, как и ожидалось, а вот при неуспешном в нём каким-то образом оказалось значение 1-700-BAD-JEST».

Если вы внесли множество изменений между двумя тестовыми прогонами или используете в них различные тестовые сценарии, то результаты будет сложно сравнить; вам постоянно придётся учитывать отличия, которые не связаны с ошибкой. Вам нужно сделать так, чтобы в результатах тестов было как можно меньше отличий, и чтобы эти отличия *были связаны с вашей ошибкой*. Поэтому сделайте так, чтобы отличий было минимум. Постарайтесь получить логи «успешного» и «неуспешного» тестовых прогонов, последовательно выполненных на одном и том же ПК; не используйте для них разные компьютеры, окружения, ПО, параметры, вводимые пользователем данные и не проводите их в разные дни. Даже не меняйте свою рубашку; это может скрыть ошибку (не смейтесь; в следующей главе будет байка ветерана, в которой узор на моей рубашке сыграл ключевую роль).

Это не значит, что вам не надо собирать информацию, не связанную с ошибкой. Вы вообще ещё толком не знаете, какая информация вам нужна – поэтому собирайте всё, что можете. Если она не касается ошибки – то в обоих логах будет представлена двумя идентичными фрагментами. Да, вам придётся просмотреть целую кучу ненужных данных, но в обоих логах эти ненужные данные будут одинаковыми; главное, что они у вас будут. В главе 4 («Воспроизведите ошибку») я рассказывал, как анализировал логи звонков системы видеоконференций и нашёл отличия между «удачными» и «неудачными» звонками. Мы тогда точно не знали, что именно ищем, поэтому пришлось просмотреть множество похожих строк, чтобы увидеть неожиданную команду в логе «неудачного» звонка.

Ладно, это не так просто, как кажется. Меня часто просили объяснить, как именно я это делаю – смотрю в логи и определяю, в чём проблема. Меня просили провести обучение для начинающих тестировщиков, чтобы развить у них этот навык. Меня просили написать приложение-фильтр, которое бы автоматически анализировало логи и находило в них все подозрительные строки. Но поиск «странностей» – это не то, чему можно научить новичка или автоматизировать. При каждой отладке вы встречаетесь с чем-то новым – всё уже не так, как в прошлый раз. И требуется изрядное количество знаний и интеллекта, чтобы разобраться в несущественных отличиях, отличиях, вызванных таймингами, и другими факторами. Таких знаний у новичка нет, а ПО не заменит человеческий ум (мы уже помахали ИИ на прощание, помните?) Максимум, что может сделать ПО – помочь вам с фильтрацией и форматированием логов, чтобы когда вы задействовали свой превосходный человеческий мозг (он ведь у вас превосходный, не так ли?) для анализа логов – различия (и, возможно, их причины) сразу бы попали под его прицел.

Когда вы просматриваете большой и сложный для понимания лог – то у вас возникает желание сразу перейти к его потенциально «подозрительным» областям; это нормально и, возможно, вы действительно там сходу что-то найдёте. Но если нет – будьте готовы внимательно прочитать весь лог – вы не знаете, где проявится разница между «удачным» и «неудачным» тестовыми прогонами.

И я должен вас предупредить: вероятно, это самая скучная задача, которую вам когда-либо приходилось выполнять. Возьмите чашку крепкого кофе, подоприте веки зубочистками и сделайте это (хотя если подумать – забудьте о зубочистках. Ими можно выколоть глаз!).

7.6. Что ты поменял перед тем, как система сломалась?

Байка ветерана. В студенческие годы я работал в мебельном магазине, и однажды ко мне обратилась моя коллега: «Эй, ты же из Массачусетского технологического. Наверняка всё знаешь о стереосистемах. Может, ты решишь мою проблему». Честно говоря, я не особо разбирался в стереосистемах, но я ими пользовался и думал, что смогу помочь. Она рассказала мне, что недавно относила проигрыватель в ремонт, и теперь его звучание просто ужасно (поясняю для молодёжи: под «проигрывателем» я имею в виду виниловый проигрыватель. Он воспроизводит большие чёрные виниловые пластинки, на обоих сторонах которых записана музыка). Коллега уточнила, что в ремонтной мастерской ей заменили картридж[9] (который преобразует движение иглы по пластинке в электрический сигнал). Мы пошли к ней домой, чтобы я смог посмотреть на это; она включила пластинку – и звук действительно был просто ужасен. Вообще, это напомнило мне случай, когда я подключил свою портативную магнитофонную деку к усилителю и выкрутил такую громкость, что вход усилителя не мог с ней справиться — звук был очень резким и искажённым. «Слишком громкий входной сигнал» — вот первое, что пришло мне на ум.

Я знал (потому что коллега рассказала), что раньше проблем со звуком не было, но с тех пор произошло изменение: была проведена замена картриджа. По опыту работу со стереосистемами я знал, что существует два типа картриджей – магнитные и керамические – и что для каждого из них используется отдельный вход на задней панели усилителя (см. рис. 7.3).

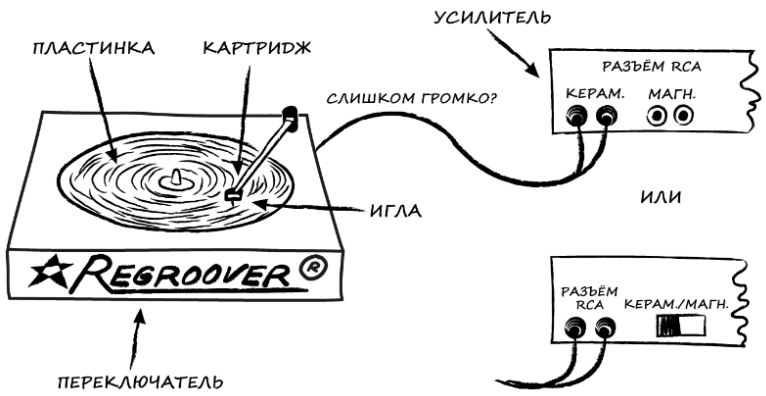


Рис. 7.3. Как слушали музыку в старые времена

[9] В русскоязычной документации этот элемент обычно называется звукоснимателем (прим. переводчика).

Это связано с тем, что разные типы картриджей обеспечивают разную громкость. Проблема коллеги показалась мне очевидной: она заменила картридж одного типа на другой, но подключила его к тому же входу усилителя, что вызвало перегруз.

Я не знал, какой у неё раньше был картридж и какого типа новый, но решил, что подключение кабеля к другому входу усилителя поможет. Я осмотрел заднюю панель усилителя и, к своему удовольствию, обнаружил, что всё ещё проще: там был переключатель. Я изменил его положение, и музыка сразу зазвучала великолепно. Вообще, так как в проигрывателе недавно был заменён картридж и игла – он звучал лучше, чем когда бы то ни было.

- Общее время анализа проблемы: около 30 секунд.
- Общий уровень кажущегося богоподобным моего технического мастерства: огромен.
- (Общее количество последовавших интимных связей: ноль. Ну, в те времена, когда ещё не было доткомов, техническое мастерство было не в моде).

Иногда разница между корректно работающей и «сломанной» системой заключается в изменениях, которые были в неё внесены. Изменения могут быть причиной сбоев. Крайне полезно выяснить, в какой версии системы впервые возник сбой, даже если для этого придётся последовательно тестировать более старые версии. Как только сбой перестанет воспроизводиться – перейдите «на версию выше» и ещё раз проверьте, что в ней он повторяется. Таким образом вы определите, что причина сбоя заключается в изменениях, внесённых в этой версии. Конечно же, у вас есть система управления версиями исходного кода, которая позволяет вам быстро провести сравнение двух версий (ведь так? Если нет – то начните её использовать. См. подробнее об этом в следующей главе «Записывайте всё, что происходит»). Если предположить, что изменения между версиями не касаются концептуальной переделки всей системы, то вы сможете сфокусироваться на конкретных фрагментах, связанных с проблемой.

Обычно что-то новое сначала неисправно; вот почему мы тестируем новые продукты перед поставкой клиенту. Иногда изменения одного компонента системы несовместимы с другим (и этот другой компонент не содержит никаких ошибок). Так было и в рассказанной выше истории про стереосистему: новый картридж был в порядке, но после его установки потребовалось перенастроить усилитель.

В некоторых случаях всё довольно сложно. Иногда проблема существует уже давно, но проявляется только после каких-то изменений в системе – например, таймингов или размера базы данных. Вы можете подумать, что в версии 5.0 появилась новая ошибка, но на самом деле она существовала уже с версии 3.1 – просто раньше не проявлялась так явно. Часто добавление нового фрагмента кода или аппаратного узла создаёт условия, которые приводят к сбою в старой, казавшейся надёжной подсистеме. Но в этой подсистеме уже давно была «дыра» – просто раньше вы не доходили до места, где в неё можно было «провалиться». У вас может возникнуть соблазн пуститься на поиски «ошибки», которая привела вас к этой «дыре» – и в некоторых случаях такая быстрая заплатка может быть разумной – но на самом деле вам нужно устранить саму «дыру».

Байка ветерана. Мы прожили несколько лет в старом доме, и в конце очередной зимы во время ливня обнаружили, что с потолка над лестницей первого этажа капает вода. Я поднялся наверх, пытаясь найти источник протечки, и добрался до чердака, на котором вода капала через обледеневшую крышу. Она стекала по одному из стропил и ударялась о металлической выступ, размером с кредитную карту. Это отклоняло воду так, что она капала на пол чердака и дальше просачивалась сквозь щель, добираясь до первого этажа. Рядом стояла старая пластиковая ёмкость – из тех, которые можно поставить в раковину и использовать для замачивания посуды. Когда мы въехали в дом – ёмкость стояла прямо под выступом, но летом я ползал по чердаку, прокладывая проводку, и сдвинул её, не подозревая о том, зачем она нужна.

Когда я прокладывал проводку – то создал «баг», сдвинув ёмкость с её законного места. Конечно, я воспользовался очевидным «быстрым фиксом», вернув ёмкость обратно, прямо под капли воды. Но это было лишь временное решение. Следующим летом мы исправили настоящий «баг», который выявили зимой – нам пришлось устранить протечку в крыше.

7.7. Памятка

Вносите по одному изменению за раз

Вам нужна определённая степень предсказуемости в вашей жизни. Удалите те изменения, которые не исправили проблему. Они могли привести к последствиям, которые вы не сразу заметите.

- **Выделите ключевой фактор**. Не меняйте график полива, если вам нужно определить влияние солнечного света в эксперименте по выращиванию растений;
- **Держитесь за поручень обеими руками**. Если вы попытаетесь починить ядерный реактор подводной лодки, не выяснив причины неисправности, то можете устроить Чернобыль в океане;
- **Тестируйте одно изменение за раз**. Я понял, что параметр фазы обрабатывается платой VGA-конвертера некорректно, потому что больше ничего не менялось;
- **Сравните «успешные» и «неуспешные» тестовые прогоны**. Если в «неуспешных» вы заметили что-то, чего нет в «успешных» – то вы близки к причине проблемы;
- **Определите, что вы поменяли перед тем, как система сломалась**. Моя коллега заменила картридж в проигрывателе – так что это было хорошей точкой отсчёта в моих размышлениях.

Глава 8: Записывайте всё, что происходит

В сыскном деле нет ничего важнее, чем искусство читать следы, хотя именно ему у нас почти не уделяют внимания.

Шерлок Холмс, «Этюд в багровых тонах»

8.1. Общий обзор

Байка ветерана. Мы отлаживали микросхему для сжатия видеопотока, которая должна была генерировать плавное видео, закодированное небольшим числом бит; она входила в состав системы организации видеоконференцией с передачей сигнала по телефонной линии. Для обеспечения плавности видео требуется, чтобы у него было много кадров в секунду; мы хотели 30. Тестируя ранний прототип микросхемы я заметил, что иногда, без какой-то явной систематики, частота кадров падала с 30 кадров в секунду до примерно 2. Для восстановления частоты требовалось перезагрузить микросхему. Ничего удивительного, что в прототипе новой микросхемы была ошибка; но меня озадачило, что не было ясно, из-за чего возникает падение частоты. Я довольно быстро определил, что этот эффект не связан с временем работы микросхемы – иногда проблема возникала почти сразу, а иногда микросхема работала без сбоев в течение двух часов. Когда я пришёл в офис на следующий день – то ошибка вообще ни разу не повторилась. Я подумал, что температура в комнате повысилась или понизилась, поэтому попробовал нагревать и охлаждать микросхему, но это не привело ни к каким результатам. Я думаю, что вся эта ситуация могла свести меня с ума (а, может, это и произошло), если бы не один момент – я заметил, что ошибка внезапно проявилась, когда я встал со стула. Я сел, перезагрузил микросхему, убедился, что с частотой кадров всё в порядке, и снова встал. Ошибка повторилась. Может быть, микросхеме было одиноко, и она не хотела, чтобы я уходил?

Я понял, что когда встал со стула – то моя рубашка попала в объектив камеры (на самом деле, кусок рубашки попадал в него даже когда я сидел, но когда встал – то в кадре оказалась ещё большая её часть). Я из Нью-Гемпшира, и на мне в тот день была (как и во многие другие дни) фланелевая рубашка в клетку. Но вот вчера я надел простую однотонную синюю рубашку. А ещё за день до этого я был в фланелевой рубашке в клетку, как и сегодня. Ещё несколько экспериментов – и я выяснил, что когда микросхема пыталась обработать крайне трудоемкий для сжатия образец (движущуюся клетчатую рубашку), то довольно быстро «приходила к выводу», что что-то идёт не так и прекращала попытки (см. рис. 8.1).

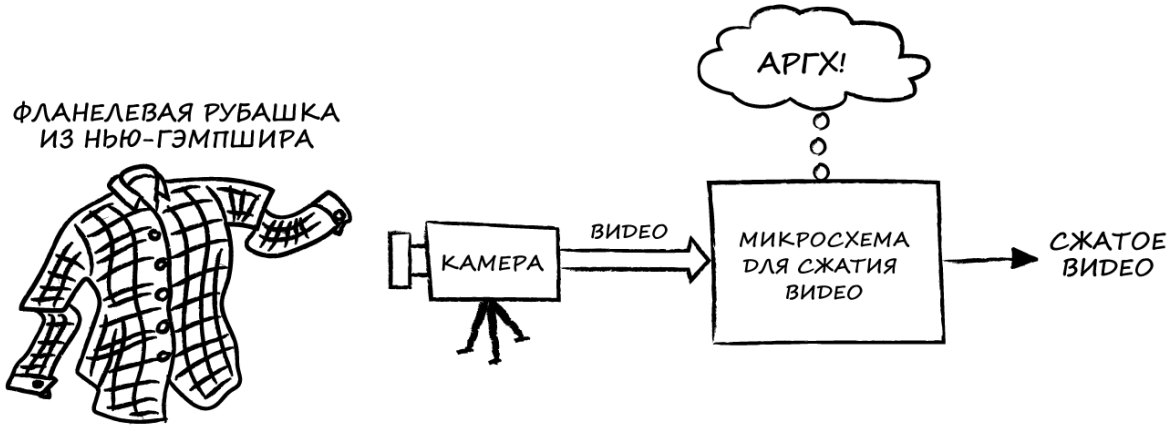


Рис. 8.1. Микросхема для сжатия видеопотока против фланелевой рубашки

Моим коллегам и производителю микросхемы сложно было поверить, что причина сбоя в рубашке, но доказать это было очень легко (впоследствии я часто использовал клетчатые рубашки, чтобы продемонстрировать, что одежда может максимально «нагрузить» алгоритмы сжатия видео – ещё один полезный маркетинговый приём, который я освоил во время отладки).

Должен признаться – я не веду ежедневник с записями, какую рубашку в какой день я надел. Я не записал, что когда встал из-за стола – частота кадров упала. Это было легко запомнить. Но когда я отправлял отчёт производителю микросхемы – я написал, что наблюдаемая мной проблема связана с клетчатой рубашкой, и что она проявилась в тот момент, когда я встал перед камерой, и что после падения частоты кадров требуется перезагрузить микросхему, чтобы всё вернулось в норму. Я даже отправил фото узора своей рубашки, чтобы производитель мог воспроизвести проблему (вообще-то, производитель попросил меня отправить ему мою рубашку, но я бы с ней не расстался). Подумайте о том, насколько эта информация была полезнее для производителя, чем если бы я написал «Частота кадров иногда падает, и требуется перезагрузить микросхему, чтобы вернуть её в норму» или даже «Микросхема иногда сбоит» (я действительно сам получал отчёты об ошибках, в которых, по сути, просто говорилось: «Оно не работает»).

Суть этой истории в том, что иногда самая, казалось бы, незначительная вещь является ключом к пониманию причины ошибки. То, что кажется неважным тестировщику (клетчатая рубашка), может быть важно для человека, который будет устранять проблему. То, что тестировщику кажется очевидным (для возобновления нормальной работы требуется перезагрузить микросхему) – может быть совершенно неочевидно другим людям. Так что вам нужно записать вообще всё – потому что вы не знаете, какая информация окажется важной и неочевидной.

8.2. Записывайте, что вы делали, в какой последовательности, и каков был результат

Записывайте всё, что происходит. В процессе отладки сохраняйте информацию о том, что вы делаете, в какой последовательности и каков результат ваших действий. Делайте это при каждом сеансе отладки. Это похоже на другую информацию, сохраняемую во время тестирования – логи ПО, архивы логического анализатора и т. д. Вы должны понимать, что именно происходило на каждом этапе отладки и каков был результат, чтобы определить, на каких моментах вам следует сосредоточить своё внимание.

Байка ветерана. Клиент регулярно звонил в службу поддержки и сообщал, что дискета сработала один раз, а потом вышла из строя. Служба поддержки уже несколько раз высылала ему новые дискеты, но ситуация раз за разом повторялась. В итоге сотрудник попросил клиента позвонить в момент работы с дискетой и детально и пошагово описать, что же он делает. Как и ожидалось, первый раз при работе с дискетой никаких проблем не возникло, а затем клиент вытащил её из компьютера и прикрепил магнитом к железному шкафу (тем самым размагнитив её).

Если у вас пищевая аллергия, то врач попытается определить конкретный продукт, который её вызывает, попросив вас вспомнить, что вы ели непосредственно перед появлением симптомов. Если это не так просто выяснить – то он может попросить вас записывать в дневник питания всё, что вы ели, и как изменялось ваше самочувствие. Это «лог» – простой и понятный. Он позволит найти связь между употреблением клубники и появлением у вас крапивницы (исправление «бага» в данном случае очевидно: не ешьте клубнику).

Байка ветерана. Каждое воскресенье у меня была мигрень. Я мысленно составил список отличий этого дня от других и понял, что по предшествующим дням – субботам – не пью так много кофе, как в будние дни. Мигрень была «синдромом кофейной отмены» (исправление «бага» было тривиальным – я пошёл и купил кофемашину, чтобы готовить себе по утрам суббот двойной капучино).

Когда вы приходите на приём к психотерапевту – он пытается узнать у вас «лог» вашей жизни (что с вами произошло и что вы чувствовали/чувствуете по этому поводу), чтобы выяснить, почему вам плохо. Представьте, оплату за сколько сеансов вы смогли бы сэкономить, если бы просто передали ему свою подробную биографию! (я знаю, что в реальности это так не работает. Дело в том, что «отладка» человека с ментальными расстройствами практически невозможна, потому что его «отладочные инструменты» неисправны – что-то он забывает, что-то подавляет в себе, про что-то умалчивает и о чём-то лжёт. Психотерапевт не может просто решить проблемы такого человека – но он может помочь ему понять, как решить их самому).

8.3. Дьявол в деталях

К сожалению, хотя в целом все понимают ценность фиксации информации, требуемый уровень её детализации часто не обеспечивается, поэтому большой объём важной информации остается пропущенным. Какой из компонентов системы работал? Какова была последовательность событий, которая привела к ошибке? А иногда даже – «в чём действительного заключается ошибка?» (Да уж!) Зачастую в отчёте об ошибке написано лишь «оно не работает». Там не сказано, что отображаемая графика полностью искажена или что красные области стали зелёными или что третье значение является некорректным. Просто написано, что «не работает».

Ждите. Ситуация будет становиться всё хуже и хуже. Для получения подробной информации – настаивайте на том, чтобы человек, сообщивший об ошибке, прислал вам логи работы системы. Итак, теперь вы получаете отчёт об ошибке («оно не работает») и три лога.

Думаете, вам скажут, какой из них записывался в момент проявления ошибки? Нет. Думаете, вам сообщат о каких-то симптомах ошибки? Нет. «Это всё есть в логе». Ну, в логе есть отладочная информация, а вот того, что видел человек, сообщивший вам об ошибке (и что ему не понравилось) – там нет. Представьте, что вы ведёте дневник питания, но записываете в него лишь то, что вы ели, и не указываете, в какой момент у вас началась крапивница. Врач ничем не сможет вам помочь. Поэтому попросите добавить к логам и отладочным трассировкам информацию о симптомах ошибки, состоянии системы в этот момент и т. д. – всего того, что не попадает в логи. Если для симптомов ошибки можно указать метки времени, в которые они возникали – то это ещё лучше (см. подробности в следующем разделе).

Обращение к тем, кто пишет сообщения об ошибках: будьте конкретны и последовательны в описании того, что происходит. В системах видеоконференцсвязи у нас часто есть системы А и Б (а иногда и система В), которые пытаются связаться друг с другом. Мы получаем отчёт об ошибке: «Нет удалённого (remote) видео». Кто является «удалённой» системой – А или Б? И что именно произошло – в удалённой системе не отображается видео или в данной системе не отображается видео от удалённой системы? Мы не сможем начать разбираться в проблеме, пока не определим основные симптомы.

Ещё одна деталь, на которую следует обратить внимание – важно не только то, где проявляется проблема, но и как интенсивно она проявляется. Например, в системах видеоконференцсвязи используется оборудование различных производителей, и при его подключении и отключении может возникать незначительный шум в динамиках. Поэтому в сообщении об ошибке следует указывать, как долго длится звук и насколько сильно он раздражает. «Полусекундный всплеск едва слышимого жужжания» смело можно игнорировать, а вот «шестисекундный разрывающий уши визг», по всей видимости, требует изучения.

Если вы изучали основы химии в средней школе, то, возможно, помните классическую лабораторную работу «Опишите свечу». Вам нужно было написать 50 уникальных фактов о свече. После того, как вы некоторое время трудились, составляя этот список, вам давали несколько подсказок, и главная из них звучала следующим образом: «Предоставьте читателю достаточно информации, чтобы он мог в точности понять происходящее». Детали! Недостаточно написать, что свеча излучает тепло и свет, когда вы зажигаете фитиль – сколько именно тепла и света? Должен ли человек, решивший повторить ваш эксперимент, спуститься в бункер, чтобы зажечь свечу?

«Свеча так горяча, что вы можете держать руку на высоте шести дюймов над пламенем всего две секунды, прежде чем рефлекторно её отдёрните». Ах, ну и да – спускаться в бункер вам не потребуется.

Байка ветерана. Мой знакомый участвовал в проекте по разработке одного прибора. Однажды он случайно прикоснулся к корпусу блока питания и почувствовал слабый электрический треск (иными словами, получил легкий удар током). Убийство людей электрическим током (наряду с возгоранием) – не очень хороший функционал для прибора, поэтому он начал исследовать возникшую ситуацию. Он не был уверен, что почувствовал именно удар током и что слышал именно треск электрических зарядов (может, это был обычный фоновый шум), поэтому попросил коллегу прикоснуться к блоку питания. Ничего не произошло. Мой друг опять прикоснулся сам и почувствовал, что его кольнуло. Он убедил ещё нескольких коллег повторить этот эксперимент, и никто ничего не ощутил. А вот мой друг чувствовал это каждый раз. Другие сотрудники стояли рядом, уперев руки в бока, и уже были готовы отвезти его в психушку – но тут заметили, что на нём, в отличие от них, нет обуви. Обувь послужила неплохим изолятором и помогла избежать удара током. И хотя, честно говоря, моего друга действительно можно было счесть невменяемым (кто станет работать в лаборатории босиком?) – наблюдаемая им проблема не была галлюцинацией.

Как и в истории с автомобилем, который не заводился, если перед этим было куплено мороженое с определённым вкусом – ключ к проблеме был в детали, о который вы в обычной ситуации даже не подумали бы. Но вы прочитали эту книгу – так что теперь *всё* наматываете на ус и *ко всему* относитесь с подозрением.

8.4. Корреляция

Сопоставление одних симптомов проблемы с другими (или с отладочной информацией) может быть крайне полезным. «Прибор громко зашумел в тот момент, когда я к нему подключился» лучше, чем «Прибор громко зашумел». Но ещё лучше сказать: «Прибор шумел в течение четырёх секунд, начиная с 14:05:23». Имея эту информацию – можно открыть отладочный лог и увидеть несколько команд управления звуком, отправленных в 14:05:23 и 14:05:27 – и сразу предположить, что они как-то связаны с наблюдаемой проблемой.

Снова вернёмся к дневнику питания: предположим, что на одном листе вы записываете, что и когда вы ели, а на другом – что у вас началась крапивница. Но вы не записываете, *когда именно* это произошло. И снова врач ничем не сможет вам помочь.

В системах, в которых происходит обмен данными между несколькими устройствами, нужно сохранять отладочные логи всех устройств с метками времени (и крайне желательно, чтобы время устройств было синхронизировано). Конечно, можно открыть логи двух устройств, которые передавали данные друг другу, и в одном из них мысленно вычитать минуту и 23 секунды, потому что там отстают часы, но это сложно и раздражающе. Так что найдите время, чтобы синхронизировать часы всех устройств системы.

И последний раз проведу аналогию с дневником питания: представьте, что для приёмов пищи вы указываете лондонское время, а для симптомов – время Сан-Франциско. В принципе, ваш врач сможет в итоге их сопоставить, но он рискует стать из-за этого таким же раздражённым, как вы – из-за крапивницы.

Причины многих ошибок можно обнаружить, если сопоставить их симптомы с расписанием человеческих действий:

Байка ветерана. В одной информационной системе периодически появлялись странные сообщения, выглядящие как бессмысленный набор символов. После анализа стало ясно, что эти сообщения появляются только время дежурства Фреда. Как оказалось – его большой живот давил на клавиатуру, когда он тянулся за кофе.

В другой истории ошибка наблюдалась во время дежурства Джорджа. Причиной сбоя было переполнение текстового буфера. Джордж придумал, как отправлять на печать вдвое больше информации, чем предполагалось: он вводил длинную строку на клавиатуре телетайпа, но прежде чем каретка доходила до конца, хватал её и сдвигал влево – а буфера, выделенного в программе, не хватало для обработки таких длинных строк, так как предполагалось, что после печати строки «обычной» длины произойдёт механический возврат каретки.

Сбой в центре обработки данных, о котором я рассказывал в главе 4, был связан с тем, что в 15:00 начинался кофе-брейк, и все сотрудники спешили в столовую, что приводило к одновременной работе всех торговых автоматов, из-за чего пропадало питание у некоторых устройств системы.

8.5. Информация времён этапа разработки полезна при тестировании

В главе 7 («Вносите по одному изменению за раз») я упоминал системы контроля версий. Они включают в себя базы данных вашего исходного кода и инструменты, которые дают возможность воссоздать вам любую версию вашего приложения даже после того, как были выпущены его более новые версии. Эти системы не позволяют разработчикам, трудящимся над одним и тем же приложением, «перетирать» изменения, которые вносит каждый из них (к сожалению, они не могут помешать испортить разработчикам хороший, работающий код). Эти системы также сохраняют историю изменений исходного кода вашего приложения – поэтому благодаря им вы будете знать, когда и какие изменения были внесены, а также при необходимости вернуться к любому состоянию (любой версии) вашего приложения. Это полезно на этапе проектирования, но ещё больше пользы может принести во время отладки. Когда в определённой версии приложения обнаружится ошибка – то вы сможете посмотреть список всех изменений с того момента, когда приложение работало корректно. Если никакие другие внешние факторы не менялись – то пройдясь по этому списку вы сможете оперативно найти причину ошибки и устранить её.

Системы контроля версий теперь входят в состав так называемых «систем контроля конфигурации» (систем сборки), которые не только отслеживают изменения в исходном коде, но и интегрированы с инструментами, использующимися для сборки приложения из этого исходного

кода, и учитывают и их версионность. Это имеет решающее значение в тех случаях, когда нужно заново собрать одну из старых версий приложения. Как будет показано позже – использование при сборке неподходящих версий инструментов может приводить к очень странным эффектам.

8.6. Самый тупой карандаш лучше самой острой памяти

Никогда не доверяйте хранение деталей, связанных с проявлением проблемы, своей памяти – запишите их. Если вы этого не сделаете, то произойдёт следующее:

- вы забудете детали, которые в тот момент не казались вам важными – и, конечно, они окажутся решающими;
- вы забудете детали, которые и правда не являются важными для вас, но они окажутся важными для кого-то ещё, кто будет отлаживать совсем другую ошибку;
- вы не сможете передать информацию кроме как устно – что отнимет время у вас и у собеседника, да ещё вам нужно будет как-то связаться;
- и вы не сможете *в точности* вспомнить, какие события происходили, в каком порядке и как они были связаны друг с другом – а всё это является очень важной информацией.

Запишите всё. Лучше сделать это в электронном виде, чтобы вы могли делать резервные копии ваших записей, прикреплять их к отчётам об ошибках, легко пересылать и, возможно, даже фильтровать с помощью специальных инструментов. Запишите, что вы сделали и к каким последствиям это привело. Сохраняйте логи и журналы трассировки; добавляйте информацию о наблюдаемых событиях и эффектах, которые в них не отражены. Запишите ваши предположения и опишите, как вы пытались исправить проблему (и к чему это привело). Запишите всё.

*- Этой ужасной минуты я не забуду **никогда** в жизни! - сказал Король.*
- Забудешь, - заметила Королева, - если не запишешь в записную книжку.

Льюис Кэролл, «Алиса в Зазеркалье»

8.7. Памятка

Записывайте всё, что происходит

А ещё лучше – не пытайтесь запомнить это правило. Запишите его.

- **Запишите, что вы сделали, в какой последовательности и к чему это привело**. Когда вы последний раз пили кофе? Когда началась мигрень?
- **Поймите, что каждая деталь может быть важна**. Микросхема, использовавшаяся для сжатия видео, зависала из-за клетчатой рубашки;
- **Ищите корреляцию событий**. «Прибор громко шумел в течение четырёх секунд, начиная с 14:05:23» лучше, чем «Прибор шумел»;
- **Помните, что информация времён этапа разработки полезна при тестировании**. Инструменты системы контроля конфигурации предоставят вам список изменений между версией ПО, в которой проявляется ошибка, и исправной версией;
- **Запишите это!** Каким бы ужасным не был момент – запротоколируйте его.

Глава 9: Проверьте кабель

Ничто так не обманчиво, как слишком очевидные факты.

Шерлок Холмс, «Тайна Боскомской долины»

9.1. Общий обзор

Байка ветерана. В июне 1984 года моя семья переехала в дом, построенный 90 лет назад, и быстро познакомилась с мешаниной его инженерных систем. Все системы дублировались: начиная с «двойного» электроснабжения верхнего этажа (с особенностями проводки, о которых я расскажу в другой истории). В системе водяного отопления использовались две печи – одна на дровах, которую предыдущий владелец использовал в качестве основной, а вторая – на мазуте, использовавшаяся в качестве резервной. Я же наоборот решил сделать основной мазутную печь (говорят, что горящие дрова согревают вас трижды: когда вы колете поленья, когда складываете их и когда сжигаете. Максимум сил, который я был готов потратить на обогрев – это поворот ручки термостата). Также в подвале было две скважины и две системы очистки воды.

Водонагреватель представлял собой небольшой теплообменник, который подводил тепло от основной системы отопления к водопроводу с горячей водой, связанному с раковиной и душем (см. рис. 9.1.). Такое решение было неудачным, потому что приходилось топить печь и летом, но меня беспокоило не это. Меня беспокоило, что мне регулярно приходилось не по своей воле принимать холодный душ.

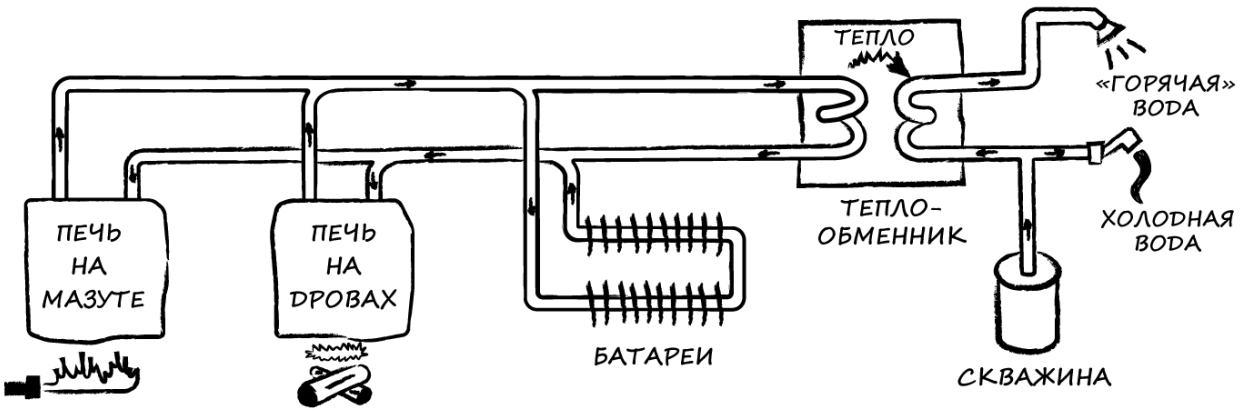


Рис. 9.1. Уникальная система отопления и горячего водоснабжения

Открытие любого крана в любом помещении дома приводило к падению давления горячей воды, из-за чего меня регулярно обливало прохладной водой. Я обдумывал покупку модного смесителя с термостатом, которой мог бы поддерживать стабильную температуру, но он был слишком уж дорогим. Мой друг (инженер-механик) посоветовал мне приобрести редуктор давления – но я обнаружил, что в моей системе уже есть такой. Я начал думать, что в системе просто не хватает нужного мне количества горячей воды.

Система была «проточной», то есть вода нагревалась по мере использования, а не хранилась в баке – так что не могло быть утечки или чего-то подобного. Я предположил, что температура нагрева недостаточно высокая – но нет, термостат на теплообменнике был установлен на 60 °C. Это было достаточно даже для посудомоечной машины. Термометр на теплообменнике показывал, что температура воды действительно равна 60 °C, и каждый раз, когда она опускалась ниже 60 °C, включался насос горячей воды. Проблема заключалась в том, что он не мог поддерживать температуру воды в то время, пока был включен горячий душ, и потом ему требовалось много времени, чтобы подогреть её до прежнего значения. Я подумал, что, возможно, эта система с проточным водонагревателем не так уж хороша, и стал подумывать о её замене.

Наступила поздняя осень, а в Новой Англии она обычно выдаётся холодной. Мы обнаружили, что утром система отопления не в состоянии быстро прогреть дом. Конечно, подогрев воды требует времени (в отличие от систем кондиционирования) – но дом прогревался слишком уж медленно. Я пощупал трубы – и они не были особо горячими. Поэтому я спустился в подвал и посмотрел на термостат мазутной печи. Он был установлен на 75 °C.

Разумная уставка для системы подогрева воды – 90 °C. Я обнаружил, что предыдущий владелец выставил для печи на дровах уставку 90 °C, а для мазутной печи – 75 °C, поэтому мазутная печь включалась только в том случае, если печь на дровах не работала. Это нормально для случая, когда мазутная печь является резервной – но у меня она использовалась в качестве основной. Я выставил для неё уставку в 90 °C, и дом сразу прогрелся.

Мало того – водонагреватель тоже начал работать правильно. Его теплообменник стал получать тепло из печи, в которой была уставка 90 °C вместо прежних 75 °C. Как гласит N-й закон термодинамики: «вы не можете нагреть воду до 60 °C, если температура в вашем теплообменнике 95 °C – или, по крайней мере, делать это достаточно быстро, чтобы можно было спокойно и не спеша постоять под горячим душем».

Моя ошибка заключалась в том, что, дрожа под холодным душем и ломая голову над своей проблемой, я исходил из предположения, что у теплообменника есть хороший источник тепла. Говоря языком правила «Разделяйте и властвуйте» – я излишне сузил диапазон, в котором искал ошибку, и упустил её истинную причину. Обычно так происходит с тем, что я называю «стандартными» или «сопутствующими» факторами. Поскольку все они являются фундаментальными (электричество, тепло, время) – то их часто упускают из виду при отладке конкретных ошибок.

Конечно, дело было в странностях системы водяного отопления, но в мире существует множество странных вещей, и если вы будете игнорировать возможность их наличия – то когда-нибудь они неожиданно заставят вас вздрогнуть – как холодная вода, внезапно полившаяся из душа.

9.2. Сомневайтесь в своих предположениях

Никогда не доверяйте своим предположениям – особенно, если они касаются причин какой-то необъяснимой ошибки. Задайте себе извечный «глупый» вопрос: «Подключён ли кабель питания?» Это кажется глупостью – но часто является источником проблемы. Вы сходите с ума, пытаясь понять, почему программное обеспечение вашего модема не работает – а оказывается, что из него случайно выдернули телефонный провод. Помните моего друга, у которого был кулер со встроенным нагревателем? Он предположил, что проблема в предохранителе, и бессмысленно потратил время на мучения, связанные с его заменой.

Часто причины проблем могут быть ещё более «низкоуровневыми». Вы пытаетесь понять, почему какая-то причудливая микросхема работает неправильно, но даже не проверили, подано ли на неё питание. Есть ли у неё часы реального времени? Ваша плата для обработки графики работает некорректно. Установлен ли в системе нужный драйвер? Эта плата поддерживается используемой операционной системой? Включена ли в реестре поддержка нужной функции? Вы уверены, что анализируемый вами код вообще вызывается? Классическая история: сказать «Пу-пу-пу, этот новый код работает точно так же, как старый» – а потом обнаружить, что хотя вы и загрузили новый код, но исполняется всё ещё старый, потому что вы не перезагрузили ПК или не удалили из системы старый код, и по каким-то причинам был запущен именно он.

Когда мы изучаем проблему – то часто обнаруживаем «область», в которой она проявляется, но причина этой проблемы может выходить за пределы этой области (и лежать «дальше» или «глубже» неё). Если оборудование эксплуатируется в неподходящих условиях – это может приводить к крайне странному поведению. Рано или поздно вы столкнётесь с проблемой, которую нельзя будет объяснить иначе, как происками инопланетян или потусторонних сил. В этот момент остановитесь и проверьте, действительно ли вы находитесь на нужной планете и в нужном измерении.

В прошлой главе я рассказывал историю про микросхему для сжатия видео, которая зависала из-за моей клетчатой рубашки. Я сузил проблему до вызова разработанной вендором функции в коде программы, которая работала не так, как было описано в документации. Я не зацикливался на предположении, что функция работает корректно, так как наблюдал доказательства обратного – поэтому связался с вендором, и он признал, что в ней есть ошибка.

Предположим, вы включили телевизор, и картинка отображается с помехами. Вы не станете сразу бросаться в бой и пытаться починить его; вы задаётесь вопросом – действительно ли должны сейчас видеть хорошее изображение? Может быть, ваш видеомагнитофон включен на запись канала 3, а вы смотрите канал 7? Или ваша антенна направлена на Ист-Сноушу (штат Вермонт), где имеется только одна УВЧ-станция? У вашей кабельной компании опять сбой? Или вы просто пытаетесь смотреть матч Green Bay Packers в середине декабря? В любом случае, вы бьётесь об заклад, что проблема не в телевизоре, и это здорово, потому что внутри него нет деталей, которые смог бы заменить обычный пользователь, и вы не заключили договор на трехлетнее гарантийное обслуживание, которое пытался впихнуть вам продавец из Best Buy.

Ваше суфле не поднимается. Включена ли духовка? Автомобиль не заводится. Прежде чем перебирать карбюратор, проверьте – есть ли у вас бензин?

9.3. Не начинайте с «шага 3»

Продолжая аналогию с автомобилем – следует учитывать, соблюдены ли условия запуска. Возможно, у вашей микросхемы есть питание, но нажали ли вы на кнопку «Включить»? Графический драйвер инициализирован? Производился ли перезапуск микросхемы? Правильно ли запрограммированы регистры? Вы именно 3 раза сжали грушу праймера, прежде чем тянуть за шнур стартера своей газонокосилки? Дроссельная заслонка закрыта? Переключатель ON/OFF в положении ON? (обычно я замечаю, что он в OFF после 6 или 7 бесплодных попыток запуска прибора).

Ещё хуже, если ваша программа зависит от начальных значений переменных, но вы не инициализируете их в своём коде – поскольку содержимое памяти перед запуском программы каждый раз будет произвольным, то *иногда* проблема не будет проявляться. Но не в тот момент, когда вы решите провести демонстрацию для потенциального инвестора.

9.4. Проверяйте свои инструменты

Байка ветерана. На заре эпохи мультимедиа с нами работал высокооплачиваемый консультант, который тестировал невероятно быстрый (на тот момент) процессор 486 с частотой 33 МГц, который мы использовали для чтения и записи видеофайлов. Нам нужна была определённая скорость чтения/записи, и мы хотели выяснить, получится ли её добиться. Тестовые программы консультанта работали не так быстро, как мы надеялись, и, что любопытно, его программа чтения работала медленнее, чем программа записи; это противоречило нашему опыту и ожиданиям. Он работал над ними несколько недель, оптимизировав каждый фрагмент кода настолько, насколько мог. Наконец, признав, что он не понимает, почему чтение происходит медленнее записи, консультант передал нам свои наработки.

Мы изучили его код и обнаружили, что он не указывал в явном виде тип файла (бинарный или текстовый). В нашей среде разработки это приводило к тому, что тип файла автоматически трактовался как тестовый, что заставляло приложение в процессе чтения файла искать в нём символы перевода строки и возврата каретки – и замедляло его. Мы указали в явном виде, что работаем с бинарными файлами, и, как мы и ожидали, чтение стало выполняться существенно быстрее. Когда мы спросили об этом консультанта – он сказал, что думал, что среда разработки по умолчанию трактует файлы как бинарные. Он не проверил своё предположение, а оно оказалось неверным.

Этот консультант, столкнувшись с непонятным поведением программы, потратил много наших времени и денег, пытаясь найти в коде ошибку, которой не было. Но поскольку он не задался вопросом, производит ли компилятор именно ту инициализацию типа файлов, которую он от него ожидал – то так и не нашёл причину проблемы. Больше мы с ним не связывались.

Ваши неправильные предположения могут быть связаны не с продуктом, который вы разрабатываете, а с инструментами, которые используются для его разработки – как в истории

выше. Некорректные настройки по умолчанию – это типичная проблема. Другая проблема – это некорректно настроенное рабочее окружение: если вы используете компилятор от Macintosh, то, очевидно, не сможете запустить свою программу на ПК с процессором Intel, но что насчёт ваших библиотек и других компонентов? Убедитесь, что они подходят вашей платформе и что вы используете нужные версии. Ошибки, возникающие из-за проблем с рабочим окружением, могут проявляться крайне странным образом.

Но даже ваши предположения о некорректных версиях инструментов могут быть неверны. Версии могут быть правильными, но иметь баги (так что предположение о том, что инструмент не содержит ошибок, тоже может быть неверным. Этот инструмент создан инженерами. Почему он должен заслуживать большего доверия чем то, что разрабатываете вы?)

Байка ветерана. Мы разрабатывали специфическую микросхему и столкнулись с аппаратной проблемой – иногда она пропускала сигнал прерывания от периферийного устройства. Поскольку схемотехник не мог «проникнуть» внутрь микросхемы и узнать, что именно происходит – он решил провести симуляцию работы микросхемы на компьютере на языке описания аппаратуры. Естественно, в симуляции никак проблем не обнаружилось. Но симуляция проводилась на уровне регистров, и инженер решил посмотреть, что именно «построил» компилятор микросхемы. Он опустился на уровень ниже – уровень логических вентилей – и обнаружил ошибку в тайминге, из-за которой прерывание могло быть пропущено. В логе было написано, что компиляция прошла корректно – почти так и было, за исключением допущенной компилятором ошибки.

Вы также можете сделать неверные предположения об инструментах отладки. Когда вы используете мультиметр с разряженной батареей для прозвонки провода – то он не пропищит, даже если провод полностью исправен – а вы будете думать, что с ним что-то не так. Поэтому первое, что вам нужно сделать – коснуться щупами друг друга, чтобы убедиться, что мультиметр издает звуковой сигнал и его можно использовать. Прежде чем подключить осциллограф – коснитесь пальцем его щупа, чтобы проверить, что он активен, и подайте на него 5 вольт, чтобы убедиться, что шкала настроена корректно. Если вы используете для отладки вызовы операторов печати (printf и т. п.) при возникновении определённого события – то вы можете ничего не увидеть в логе, если эти операторы не работают. Так что добавьте в лог сообщение, которое будет выводиться всегда, чтобы подтвердить их работоспособность. Если вы измерили температуру вашего ребёнка, и она составляет 24 °C – то измерьте её ещё раз и, желательно, другим градусником (если он покажет такое же значение – то выпустите бедного ребёнка из морозильной камеры).

Байка ветерана. Прошло несколько месяцев после того, как я починил горячий душ в нашем 90-летнем доме – и вот внезапно погасла печь. Поскольку мы теперь использовали печь на мазуте – то я предположил, что, вероятно, нам не хватает топлива. Я проверил датчик топлива и выяснил, что бак заполнен на четверть – так что проблема явно не в этом. Я вызвал специалиста по ремонту печей. Он пришёл в тот же вечер, чтобы выяснить, в чём проблема. В первую очередь он тоже решил проверить уровень топлива в баке. Когда я сказал ему, что уже сделал это, то он ударил по

манометру фонариком, и его стрелка сразу упала в ноль. Он залил мне мазута, и по его взгляду было понятно, что он считает меня одним из глуповатых и беспомощных «городских».

Убеждённость опасней для истины, чем ложь.

Фридрих Ницше

9.5. Памятка

Проверьте кабель

Предположения о чём-то «очевидном» чаще всего ошибочны. И, кстати, ошибки в предположениях обычно устраняются легче всего.

- **Сомневайтесь в своих предположениях**. Вы вызываете нужный код? У вас кончился бензин? Подключен ли кабель?
- **Начните с базовых вещей**. Произведена ли инициализация памяти? Вы сжали грушу праймера, прежде чем тянуть за шнур стартера своей газонокосилки? Была ли нажата кнопка включения?
- **Проверяйте свои инструменты**. Вы используете нужную версию компилятора? Не заклинила ли стрелка манометра? Заряжена ли батарейка мультиметра?

Глава 10: Воспользуйтесь чьим-то свежим взглядом

Лучший способ добраться до сути дела – рассказать все его обстоятельства кому-то другому.

Шерлок Холмс, «Серебряный»

10.1. Общий обзор

Байка ветерана. У меня были проблемы с машиной – настолько странные, что я даже не помню всех их подробностей, но в целом происходило примерно следующее: каждый раз, когда я переключался на заднюю передачу, перегорал предохранитель стоп-сигналов. Я заменил несколько предохранителей, прежде чем понял, что дело не в них, и при случае рассказал о своей ситуации коллеге, который хорошо разбирался в ремонте автомобилей. Он ответил не раздумывая: «Плафон внутреннего освещения салона прижимает провод к крыше машины. Если ты откроешь его, вытащишь провод, обмотаешь изолентой и уберёшь обратно – всё будет в порядке». Я засмеялся; как плафон может иметь какое-то отношение к тормозам? Коллега сказал, что это типичная проблема (см. рис. 10.1).

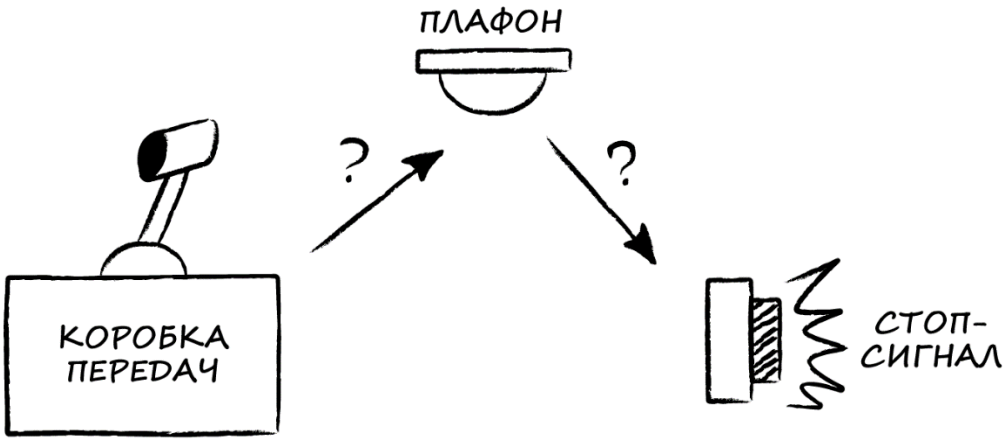


Рис. 10.1. Холистический подход к ремонту автомобиля

Я подумал, что он шутит надо мной, но решил подыграть – поэтому подошёл к машине и открутил плафон. И, действительно, под его кронштейном был провод, прижатый к крыше. Я вытащил его, обмотал чёрной изолентой и включил заднюю передачу. С предохранителем стоп-сигналов ничего не случилось.

Чтобы понять причину проблемы – я мог бы использовать остальные правила книги: прочитать о том, как устроена электрика автомобиля, и проверить его провода мультиметром. Это потребовало бы времени, а мой коллега имел соответствующий опыт и сразу знал ответ. Всё, что мне нужно было сделать – попросить его о помощи. Это заняло всего несколько минут, а поскольку мы с ним были в офисе – то я без труда смог «раздобыть» немного изоленты.

10.2. Попросите о помощи

Есть как минимум три причины обратиться за помощью, не считая желания свалить всю проблему на кого-нибудь другого: получить свежий взгляд, знания и опыт. И обычно люди готовы помочь, потому что это даёт им возможность продемонстрировать, насколько они умны.

10.2.1. Глоток свежих мыслей

Трудно увидеть общую картину со дна колодца. Все мы люди. У всех нас есть свои предубеждения по поводу разных вещей – в том числе, и по поводу причин ошибок. Эти предубеждения могут помешать нам понять, что происходит на самом деле. Человек, который подходит к нашей проблеме со своей непредвзятой (а на самом деле – предвзятой) точкой зрения, может дать нам новую информацию и сподвигнуть нас на новые попытки её решить. В крайнем случае он может согласиться, что вы столкнулись с неприятной проблемой и подставить вам плечо, в которое вы сможете поплакать.

На самом деле, иногда, *рассказывая кому-то* вашу проблему, *вы сами* заново осмысливаете её и внезапно понимаете, как решить. Простая систематизация фактов поможет вам выйти из ступора. Я даже слышал о компании, которая сделала в офисе комнату с манекеном – чтобы сотрудники сначала могли рассказать о своих проблемах ему. Я считаю, что это весьма полезный подход, который способствует быстрому решению ряда проблем (возможно, манекен более отзывчив, чем некоторые ваши коллеги. Также не сомневаюсь, что он снисходителен к вашим ошибочным предположениям, и ни упомянет ни одно из них в обзоре о пересмотре вашей зарплаты на следующий год).

10.2.2. Спросите эксперта

В ряде случаев одна или несколько частей системы остаются для нас загадками; вместо того, чтобы тратить годы на их изучение, вы можете спросить эксперта и быстро получить ответ на интересующий вопрос. Но сначала убедитесь, что собеседник действительно является экспертом в своей области: если он даёт вам расплывчатые и перегруженные модными словечками ответы, то он шарлатан и не сможет вам помочь. Если он говорит, что для подготовки отчёта по вашему вопросу требуются исследования, которые займут 30 часов – то он наёмный консультант, который может вам помочь за соответствующее вознаграждение.

В любом случае, эксперты «изучили свою систему» гораздо лучше, чем вы; они знают её структуру и могут дать вам отличные советы о том, с чего начать поиск причины ошибки. И когда вы их обнаружите – эксперты помогут разработать такое исправление, которое не нарушит работу остальных фрагментов системы.

10.2.3. Голос опыта

Возможно, у вас не так много опыта в какой-то области, но рядом с вами могут быть люди, которые уже сталкивались с такой же проблемой и, получив от вас её краткое описание, могут точно сказать, с чем она связана – как в моей истории с плафоном в машине, который влиял на перегорание предохранителя стоп-сигналов. Как и в случае с экспертами – найти людей с опытом в нужной области бывает сложно и, следовательно, их помощь может дорого стоить.

Байка ветерана. Есть старая история об одном джентльмене, который много лет обслуживал оборудование на большой фабрике, пока не вышел на пенсию. Некоторое время на фабрике всё было нормально, пока однажды один из станков не остановился и больше не запускался, несмотря на все усилия нового обслуживающего персонала. Они позвонили нашему пенсионеру, описали проблему и попросили его о помощи. Он ответил, что, вероятно, сможет всё исправить – но это обойдётся им в 10 000 долларов. Персонал фабрики был в отчаянии, так что согласился на сделку.

Пенсионер пришёл на завод со своим чемоданчиком, полным инструментов. Он подошёл к станку, достал из чемоданчика молоток и один раз ударил по станку где-то сбоку. Станок заработал. Пенсионер убрал молоток в чемоданчик и попросил свои 10 000 долларов. Сотрудники фабрики были в ярости. «10 000 долларов за один удар молотком?»

«Нет», – поправил их пенсионер. – «10 долларов за удар молотком, и 9 990 – за знание, куда именно ударить».

Как упоминалось во вступлении этой книги – существуют руководства по устранению неполадок в конкретных системах. Если для вашей системы есть такое руководство и ваша проблема относится к категории «что-то сломалось», а не связана с ошибками, допущенными в ходе её разработки, прочитайте его – возможно, вы найдёте симптомы своей проблемы в таблице симптомов известных проблем, и сможете быстро её устранить.

Байка ветерана. Когда видеоигра «Пинг-понг», в создании которой я участвовал, была запущена в массовое производство, инженеры, разрабатывавшие её аппаратную «начинку», написали собственное руководство по устранению неполадок. Движение шарика контролировалось уровнем напряжения конденсатора (конденсатор – это такой «резервуар», наполненный электронами). Компания-производитель покупала крайне дешёвые (и, соответственно, не особо качественные) электронные компоненты, поэтому вскоре появилась сообщения от клиентов, у которых в приставке вздулись конденсаторы. Инженеры быстро обнаружили, что если шарик движется вверх быстрее, чем вниз – нужно заменить конденсатор А; если шарик двигался влево быстрее, чем вправо – нужно заменить конденсатор Б. Они добавили эту информацию в своё руководство, и когда со скоростью движения шарика возникала проблема – без колебаний меняли соответствующий конденсатор.

10.3. Где получить помощь

В зависимости от того, хотите ли вы получить свежий взгляд, экспертную оценку, «голос опыта» или какую-то комбинацию вышеперечисленного – вы можете обратиться к разным источникам. Конечно же, к ним относятся и ваши коллеги – они умны, могут быть экспертами в каких-то областях и, возможно, уже встречались с проблемой, которую вы наблюдаете. Некоторые компании разрабатывают «системы управления знаниями», которые агрегируют информацию из документов и электронной почты с указанием авторства – чтобы вы могли узнать, какая информация есть у вашей компании, и кто является её источником. На момент написания книги – это совершенно новый продукт, но скоро он начнёт активно внедряться, так что помните об этом и используйте подобную систему, если она у вас есть. Если же её нет, то вам придётся искать информацию старыми добрыми способами – копаясь на файловых серверах и расспрашивая коллег у кофемашины.

Если вы используете оборудование или ПО стороннего производителя – в случае проблем свяжитесь с ним по электронной почте или по телефону (в любом случае, производитель будет признателен вам за информацию об ошибке). Зачастую оказывается, что вы просто неправильно поняли какой-то фрагмент документации. Помните, что у производителя есть как экспертные знания, так и опыт использования своих продуктов. Иногда, как в моей истории с платой VGA-конвертера, которая некорректно обрабатывала параметр фазы, производитель может узнать от вас что-то новое – я нашёл ошибку, о которой его инженеры не были в курсе. Но у этих инженеров есть опыт, который позволяет выяснить причины ошибки и удержать вас от бесплодных попыток исправить её, если она связана с недоработками производителя. Они могут сделать для вас исправление или, по крайней мере, предложить обходной путь для решения проблемы. В моей истории их пояснения помогли мне обойти допущенную производителем ошибку в документации путём доработки своего кода.

Получение информации от производителя не всегда требует обращения в техническую поддержку. Надо сказать, что от некоторых производителей вообще ничего не удается получить, но всё же большинство предоставляет как минимум документацию по своим продуктам. Сейчас самое время повторить «Прочитайте инструкцию». Точнее, в данной ситуации совет звучит следующим образом: «Если всё остальное не помогло – прочтите инструкцию ещё раз». Да, вы уже прочитали её, потому что добросовестно следовали правилу 1 («Изучите свою систему»). Теперь прочитайте её снова, сосредоточив внимание на возникшей у вас проблеме – вы можете увидеть или понять что-то такое, что раньше упускали.

Производители, которые не могут предоставить качественную техническую поддержку, часто пытаются компенсировать это, размещая информацию в интернете; поищите на их веб-сайте рекомендацию по использованию их продуктов и примеры программ. И помните, что есть и другие пользователи, которые могли столкнуться с такой же проблемой – найдите их и попросите о помощи. Проверьте тематические интернет-форумы и группы рассылки Usenet. В некоторых из них можно даже встретить сотрудников компании-производителя, которые ответят на ваши вопросы. Как уже упоминалось ранее – если вы устраняете ошибки в конкретной системе, то проверьте, нет ли для неё руководства по устранению неполадок.

Наконец, есть ресурсы более общего характера, посвященные приборам, языкам программирования, методам проектирования ПО и даже отладке (правила, изложенные в данной книге, помогут вам более систематично применять советы, почерпнутые из всех этих источников). Сходите в местный книжный магазин (или закажите книги с доставкой), подпишитесь на журналы и почтовые рассылки, связанные с вашей предметной областью. Поищите информацию в сети. Знания экспертов окружают вас со всех сторон.

10.4. Не будьте слишком горды

Может быть, вы боитесь попросить о помощи, потому что думаете, что вас посчитают некомпетентным. Это не так; наоборот, вы покажете истинное стремление как можно скорее исправить ошибку. Если вы воспользуетесь свежим взглядом, экспертностью или опытом других людей – вы сможете исправить её быстрее. Это не говорит о вас ничего плохого; по крайней мере, вы достаточно мудры, чтобы найти подходящую помощь. Верно и обратное: не думайте, что вы дурачок, а эксперт – бог во плоти. Иногда эксперты ошибаются, и вы можете сойти с ума, если будете считать, что ошибка на вашей стороне.

Байка ветерана. Я разрабатывал программу (редактор списков запуска приложений на языке Форт – для тех, кто ностальгирует по эпохе, когда корабли были деревянными, а люди – железными), которая использовала большую схему индексации базы данных, называемую B-деревьями. Она была основана на кодовой базе, созданной моим коллегой, специалистом по компьютерным наукам, который использовал томик Кнута в качестве подушки. Однажды, работая из дома, я просматривал код (в те времена ещё не было домашних ПК, поэтому я не мог запустить и проверить его) и наткнулся на фрагмент программы, который не понимал. Казалось, что при определённых обстоятельствах при перебалансировке дерева будет пропущен целый блок данных. Я часами пытался понять, что упускаю из вида – потенциальная ошибка была слишком очевидной, чтобы ускользнуть от внимания моего коллеги. Что-то в моих рассуждениях было не так. Наконец, я пришёл в офис и разложил листинги кода на его столе. Я показал ему тот фрагмент и сказал, что не понимаю, почему он будет работать корректно, и попросил его дать его какие-то пояснения. Коллега смотрел в код несколько мгновений, а потом сказал: «Хм, а тут ошибка».

10.5. Сообщайте о симптомах ошибки, а не о своих теориях

Независимо от того, какую помощь вы ожидаете, описывая кому-то свою проблему, помните: сообщайте о симптомах ошибки, а не о своих теориях по её поводу. Причина, по которой вам потребовалась помощь, заключается в том, что ваши теории вам не помогли. Если вы обратитесь к другому человеку и изложите ему свою теорию – то он будет ограничен её рамками в своих рассуждениях (в которых оказались и вы). С другой стороны – вы, вероятно, не озвучите некоторые важные детали, которые посчитаете незначимыми. Так что держите себя в руках. Расскажите о том, что вы видели. Опишите условия проявления проблемы, если можете. Обязательно поясните, какие эффекты вы видите при каждом проявлении проблемы, а какие – лишь иногда. Но упоминайте о том, что, по вашему мнению, является причиной проблемы.

Пусть другой человек выскажет *своё* мнение. Оно может совпасть с вашим; или у него будет свежий взгляд на вашу проблему, который вы изначально и хотели получить. Я много раз видел эту ошибку: обращаются за помощью и сразу затуманивают чужой разум уже показавшими свою несостоятельность теориями (если бы эти теории были верны – то помощь бы не потребовалась). В ряде случаев тот, кто возьмётся вам помочь, сможет прорваться через мусор ваших теорий и докопается до фактов, но чаще всего вы просто загоняете в болото своих предположений ещё одного человека.

Врач, к которому вы обращаетесь из-за боли в спине, хочет услышать, что именно вы чувствуете, а не вашу гипотезу о том, что у вас рак позвоночника, которую вы сформировали после гугления ваших симптомов в интернете. Автомеханику было бы полезно узнать, что двигатель вашей машины не заводится холодным утром, а не ваше предположение о том, что на заводе General Motors её собирали в понедельник, и сборщик с похмелья что-то там недокрутил.

Правило из данного пункта работает в обе стороны. Если вас позвали на помощь – то прерывайте поток сознания человека, который начинает излагать вам свои теории. Затыкайте уши и пойте «Ла-Ла-Ла-Ла-Ла». Убегайте. Не позволяйте затянуть вас в болото.

10.5.1. Уверенность не обязательна

Когда вы размышляете о том, что именно стоит рассказать другому человеку о вашей проблеме – то часть информации может лежать в «серой зоне». Иногда вы видите, что что-то в системе выглядит подозрительно, какие-то данные некорректны или что-то в происходящем вроде бы связано с вашей проблемой – но вы не знаете, каким образом. Про такие вещи тоже стоит рассказать; это будет информация, которую вы не ожидали получить или не можете понять. Возможно, она не имеет значения – а может, как раз наоборот. Даже узор рубашки или вкус мороженого могут быть важны.

10.6. Памятка

Воспользуйтесь чьим-то свежим взглядом

В любом случае, вам стоит сделать перерыв и выпить кофе.

- **Попросите кого-нибудь посмотреть на вашу проблему свежим взглядом**. Даже манекен может помочь вам заметить то, чего вы раньше упускали;
- **Привлеките экспертов**. Только производитель платы VGA-конвертера мог подтвердить, что обработка параметра фазы производится некорректно;
- **Прислушайтесь к голосу опыта**. Он объяснит вам, что плафон внутреннего освещения салона прижимает провод к крыше машины;
- **Знайте, что помощь повсюду.** Коллеги, производители используемого вами оборудования и ПО, интернет-форумы и книжные магазины только и ждут, когда вы к ним обратитесь;
- **Не будьте слишком горды**. Все допускают ошибки. Гордитесь тем, что исправляете их, а не тем, что делаете это без посторонней помощи;
- **Сообщайте о симптомах ошибки, а не о своих теориях**. Не втягивайте других людей в ваше болото;
- **Помните, что вам не обязательно понимать, как имеющаяся у вас информация связана с ошибкой**. Расскажите, что в момент проявления ошибки на вас была рубашка в клетку.

Глава 11: Если вы не исправили ошибку – она не исчезла

По-моему, не признавать близкой опасности скорей глупость, чем храбрость.

Шерлок Холмс, «Последнее дело Холмса»

11.1. Общий обзор

Байка ветерана. Я купил подержанную машину незадолго до переезда на Восточное побережье из Северной Калифорнии. По пути я проезжал через Лос-Анджелес; к северу от этого города есть крутой длинный холм, и чтобы преодолеть его – я разогнал машину, как мог. Но прежде чем я добрался до вершины холма – двигатель внезапно заглох. Я выжал сцепление и остановился на аварийной полосе. Я ругался и пытался понять, что же делать; так как идей не было – решил повторить попытку. Когда я повернул ключ зажигания, то двигатель сразу завёлся. Я осторожно доехал до вершины холма, постоянно ожидая, что двигатель снова заглохнет, но этого не случилось.

Моё путешествие продолжалось. В конце декабря я ехал на север через Западную Вирджинию. День был очень холодный, шёл сухой снег; я остановился на маленькой заправке у подножья горы, чтобы пополнить запасы бензина. После этого я поехал в гору, и машина заглохла. Я остановился, вспомнил случай в Лос-Анджелесе, повернул ключ зажигания – и машина снова завелась. Я без проблем доехал до вершины горы. Поскольку я не особо доверял маленькой заправке в глуши – то решил, что в бензине была вода (хотя это не объясняло произошедшее в Лос-Анджелесе). Я добавил в бензобак немного «незамерзайки» и стал надеяться, что проблема больше не повторится.

Но как бы не так. Той же зимой машина снова заглохла на совершенно ровном шоссе (и я не превышал скорость). Я остановился, пришёл в ярость и тут же попытался завести машину, но не смог; ещё некоторое время позлился, затем опять попробовал завести двигатель – и у меня это получилось. Я смог доехать до работы, но обнаружил, что через некоторое время после того, как скорость превысит 70-80 км/час – машина глохнет. Если после этого подождать минуту-две – то её получается завести.

Я не автомеханик, поэтому отвёз машину в местную авторемонтную мастерскую. Её сотрудники сказали: «У вас проблема с проводкой», заменили несколько кабелей и взяли с меня 75 долларов. Разумеется, через пару дней машина снова заглохла (и я понял, что даже если людям платят 50 долларов в час, это не означает, что они умеют отлаживать ошибки – таков был один из первых уроков в моей инженерной карьере).

Я проанализировал все эти происшествия. В каждом из них я ехал в гору, жал на педаль газа и скорость было достаточно высокой. После того, как машина заглохла – нужно было немного подождать, и после этого удавалось снова завести её. Я был немного знаком с устройством двигателя, так что понимал, что двигатель сжигает топливо из резервуара карбюратора, в который оно поступает из бензобака. Я подумал, что если что-то ограничит поток топлива из бензобака в карбюратор, то я могу просто сжечь все топлива карбюратора, когда нажму на педаль газа, и мне

придётся ждать, пока карбюратор снова медленно наполнится топливом из бензобака, после чего двигатель сможет опять работать (см. рис. 11.1).

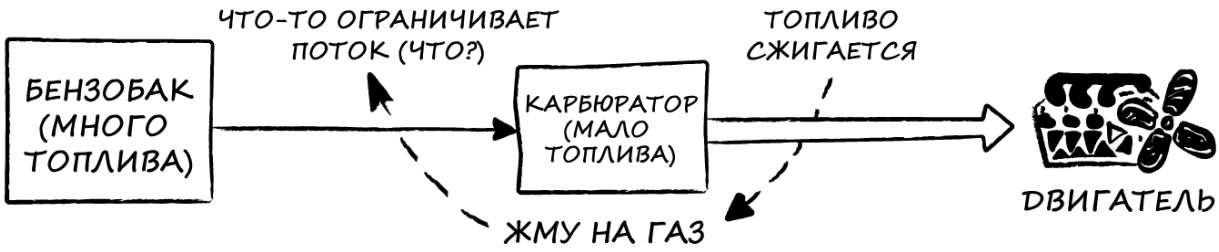


Рис. 11.1 Что-то зажмотило бензин

Затем я применил правило «Воспользуйтесь чьим-то свежим взглядом» и, увидев вокруг кофемашины толпу коллег, спросил у них: «Что может ограничить поток топлива из бензобака в карбюратор?» Один из моих разбирающихся в автомобилях друзей ответил: «Загрязнённый топливный фильтр». Я купил топливный фильтр за 50 центов, самостоятельно заменил его – и это решило проблему.

Механики в авторемонтной мастерской не проверили, устранена ли моя проблема. Они даже не попробовали повторить её (конечно, они заработали 75 долларов. Но это были последние доллары, которые они от меня получили).

11.2. Проверьте, действительно ли исправлена ошибка

Если вы будете следовать правилу «Воспроизведите ошибку» – то сможете доказать, что устранили проблему. Сделайте это! Не считайте, что первое же ваше исправление сработает, как надо; проверьте, что ошибка действительно больше не проявляется. Независимо от того, насколько очевидной вам кажется ошибка и способ её устранения – вы не можете быть в этом уверены, пока не проведёте тестирование. Конечно, вы можете содрать с какого-нибудь молодого дурачка 75 долларов, но он не будет рад, когда проблема снова проявит себя.

11.3. Убедитесь, что именно ваше исправление устранило ошибку

Если вы думаете, что исправили системную ошибку (в ПО, схемотехнике прибора и т. д.) – то уберите ваше исправление. Убедитесь, что ошибка повторится. Верните исправление. Убедитесь, что ошибка не проявится. Только таким образом вы сможете доказать, что именно ваше исправление устранило ошибку.

«И зачем это нужно?» – может спросить читатель. Дело в том, что во время отладки вы часто меняете что-то, что не является частью вашего исправления – например, последовательность тестов

или какие-то компоненты аппаратного или программного обеспечения. Или, возможно, есть какой-то случайный фактор, который вы не замечаете. Это не имеет значения, если ваше исправление не оказывает эффекта, и проблема всё ещё воспроизводится. Но в ряде случаев все эти вещи могут скрыть проблему. Вы этого не осознаёте; вы тестируете своё исправление, и ошибка не проявляется. Ура! Вы с радостью и чувством выполненного долга идёте домой, но ваше исправление не устранило ошибку. Если оно будет отправлено клиентам – то не решит их проблем. И это плохо.

Если после удаления вашего исправления (и без каких-либо других изменений) ошибка повторяется – вы можете быть уверены, что последовательность тестов корректна и именно ваше исправление устранило проблему.

Байка ветерана. В одном из эпизодов «Улицы Сезам» Супер Гровер и Бетти-Лу изо всех сил пытаются включить компьютер. Гровер говорит: «Хм… Может, если я подпрыгну и крикну “Вубба!”, это сработает?». Пока он прыгает, крича «Вубба! Вубба! Вубба!”, Бетти-Лу находит кнопку питания. Она нажимает на неё, и компьютер включается. Гровер, не обративший внимания на действия Бетти, видит, что компьютер включился, и приходит к выводу, что он придумал действенный способ ремонта компьютеров.[10]

Конечно, в тех случаях, когда вы просто ремонтируете конкретный прибор (а не устраняете системную ошибку) – повторная проверка исправления может быть избыточной и неудобной. Мне не требовалось ставить грязный топливный фильтр обратно к себе в машину. И не стоит проводить такую проверку для пациента, которому трансплантировали сердце.

11.4. Ошибка никогда не проходит сама по себе

Если вы не исправили ошибку – то она не исчезла. Всем хочется верить, что ошибка просто растворилась сама по себе. «Кажется, мы больше не можем добиться, чтобы система выпадала в ошибку», «Пару раз случалось что-то странное, но потом система перестала глючить» и, конечно, «логичный» вывод: «Может быть, этого никогда больше не повторится». И, угадайте, что дальше? Это повторится.

Если ранее ошибка проявлялась, а потом перестала – это значит, что вы как-то изменили связанные с ней условия. Если у вас были какие-то догадки о причинах ошибки, и вы использовали метод дробовика в процессе отладки (*звуки осуждения*), то, возможно, вы даже устранили её, но не знаете, как именно и, вероятно, не станете спорить об этом, ставя на кон вашу зарплату. И, конечно, вряд ли вы будете уверены, что следующий продукт вашей компании ждёт успех.

Вернитесь к главе 4 – в ней написано, как сделать бессистемные сбои более регулярными. Верните систему в то состояние, в котором ошибка воспроизводилась, и используйте те же

[10] Цитируется с разрешения Sesame Workshop.

сценарии тестирования. Если в старой версии ПО ошибка проявляется, а в новой нет – то, возможно, она была исправлена. Проанализируйте отличия версий – и выясните, каким именно образом.

Иногда у вас не хватает времени на отладку. Когда вам приходится говорить: «Что ж, эту ошибку слишком сложно воспроизвести. Мы не в состоянии её исправить» – вы не сможете спокойно спать по ночам. И, конечно же, найдётся клиент, который столкнётся с этой проблемой, что докажет – ваши бессонные ночи были неспроста. Так что подготовьтесь к этому заранее. Встройте в систему инструменты для логирования на тот случай, если в процессе её эксплуатации возникнут какие-либо ошибки. Если этого не случится – то и слава богу, но если всё же произойдёт – вы «воспроизвели её», и теперь у вас есть информация для анализа.

Дополнительным преимуществом станет тот факт, что даже если с помощью полученной информации вы не сможете исправить ошибку – ваши клиенты будут знать, как серьёзно вы относитесь к качеству своего ПО. Когда они сообщат о проблеме, то лучше сказать: «Спасибо! Мы уже несколько месяцев пытаемся отловить эту крайне редко проявляющуюся ошибку. Пожалуйста, пришлите ваши логи на наш e-mail», чем «Ух ты, ничего себе. Раньше мы такого не видели».

11.5. Устраните причину ошибки

Когда прибор выходит из строя – не думайте, что это произошло без какой-либо причины. Если существует фактор, который приводит к постепенной деградации одного из элементов прибора – то замена элемента позволит вам лишь выиграть немного времени (если это как-то может вам помочь), прежде чем новый элемент тоже выйдет из строя. Вам нужно найти настоящую причину ошибки. В истории про сломавшиеся на рождественской вечеринке колонки предохранитель правого канала перегорел из-за расплавившейся изоляции проводов. Попытка заменить его предохранителем левого канала привела к тому, что и тот тоже перегорел. К сожалению, это был единственный имевшийся у студентов предохранитель.

Байка ветерана. Одна из моих самых разочаровывающих попыток что-то отладить произошла в те годы, когда я ещё учился в колледже. Друг нашей семьи подарил мне стереосистему со встроенными усилителем, AM/FM-приёмником и восьмидорожечной магнитофонной декой (да, это было очень давно). Но была загвоздка: стереосистема не работала. Я включил питание, и ничего не произошло.

Я отнёс её в школу, взял мультиметр и проверил контур питания. Ни на одном из четырёх или пяти проводов, выходящих из трансформатора, не было напряжения, хотя оно присутствовало на его входе. Я сделал вывод, что проблема заключается в плохом трансформаторе – возможно, он сгорел, когда я включил стереосистему. Естественно, не обладая никакой документацией и не имея исправного трансформатора, я не мог сказать, какое у него должно быть выходное напряжение – поэтому не мог просто так подобрать замену. Я записал part number трансформатора и заказал такой же у его производителя.

Спустя много месяцев он пришёл мне по почте. Я открыл корпус стереосистемы и, с радостью увидев, что цвета проводов совпадают, заменил старый трансформатор на новый. Включил стереосистему, настроил её на WBCN (так называлась рок-радиостанция Бостона) и наслаждался музыкой. Я не измерил напряжения на проводах.

Примерно через час я и пара моих одноклассников решили сходить на студию WBCN (которая находилась на верхнем этаже Prudential Tower, расположенной в паре кварталов от нашего общежития) и попросить, чтобы они поставили какую-то нравившуюся нам песню, подняв табличку с забавной надписью на смотровой площадке рядом с окном студии. Когда мы вернулись – то двое оставшихся в общежитии парней сказали, что диджеи прокомментировали нашу просьбу и, возможно, даже включили запрошенную песню. Но точно они не знали, так как сразу после этого из стереосистемы пошёл дым, и она перестала что-либо воспроизводить.

Я достал сгоревший трансформатор, коря себя за то, что не измерил напряжение на проводах, чтобы в следующий раз заменить его чем-нибудь попроще. На этот раз я измерил напряжение в цепи, к которой подключались эти провода и обнаружил, что в магнитофонной деке было короткое замыкание. Новый трансформатор не мог исправить эту проблему – но он проработал достаточно долго, чтобы я решил, что устранил её.

Я не стал заказывать ещё один трансформатор. Стереосистему я выбросил.

Вполне очевидно, что я нарушил множество правил отладки. Я не изучил свою стереосистему и не приложил достаточно усилий для поиска неисправности. Я думал, что она вышла из строя, когда я подал питание – а она не заработала. На самом деле, я воспроизвёл ошибку, которая была связана с коротким замыканием в магнитофонной деке. Но в тот раз я не измерил напряжение на проводах трансформатора. И это было моей главной ошибкой – я не учёл, что какие-то другие факторы могли привести к выходу трансформатора из строя.

11.6. Внесите изменения в процесс разработки

Ладно, хоть я и обещал, что не буду рассказывать вам про процессы управления качеством, но иногда сложно найти грань между исправлением ошибки и исправлением недостатков процесса разработки, из-за которых была допущена эта ошибка. И переступить эту грань может быть хорошей идеей. Вот пример: по полу цеха течёт машинное масло. Вы можете решить проблему, протерев пол, но это не устранит её причину – труба, входящая в состав производственной установки, подтекает. Итак, вы потуже закручиваете гайку. Проблема устранена? Нет, она снова ослабится – потому что установка сильно вибрирует, а для крепления трубы используются всего два болта, а не четыре, как должно быть. Вот так-то. Наконец-то исходная причина проблемы «пол в масле» обнаружена. Вы воспользовались советами из предыдущих разделов и устранили условия, вызвавшие её возникновение.

Только вот это ещё не всё. Труба следующей установки, которую поставят в цех, тоже будет закреплена лишь двумя болтами – и пол снова будет в масле. Вам необходимо исправить процесс

проектирования установок, чтобы наличие вибрации учитывалось на этапах сбора и постановки требований, проектирования и тестирования.

Размышляя об управлении качеством, я часто думаю об ISO-9000 как о «логе» этапа проектирования. В примере выше ошибка, которую нужно обнаружить, находится на уровне процесса (вибрация), а не продукта (гайки и болты), но вся информация из главы 8 будет актуальна. Как упоминалось во введении – в этой книге рассматриваются универсальные правила отладки.

11.7. Памятка

Если вы не исправили ошибку – она не исчезла

И теперь, когда вы знакомы со всеми правилами отладки, у вас нет оправдания, чтобы её не исправлять.

- **Проверьте, действительно ли исправлена ошибка**. Не думайте, что дело в кабелях и не выезжайте на дорогу с загрязнённым топливным фильтром;
- **Убедитесь, что именно ваше исправление устранило ошибку**. Возможно, выкрик «Вубба!» был ни при чём;
- **Помните, что ошибка *никогда* не проходит сама по себе**. Заставьте её снова проявляться, используя советы из главы 4. А если систему уже пора отправлять на объект – то добавьте в неё инструменты для отладки, чтобы иметь возможность изучать проблемы прямо на объекте;
- **Устраните причину ошибки.** Отключите бесполезную восьмидорожечную деку, прежде чем сжечь ещё один трансформатор;
- **Внесите изменения в процесс разработки**. Не ограничивайтесь простой уборкой. Исправьте процесс проектирования производственных установок.

Глава 12: Все правила в одной истории

Вам известен мой метод. Он базируется на сопоставлении всех незначительных улик.

Шерлок Холмс, «Тайна Боскомской долины»

Байка ветерана. Компания «Т» разработала небольшое устройство, которое иногда не запускалось при подаче питания, потому что микропроцессор не мог считать данные из энергонезависимой памяти. Энергонезависимость обеспечивалась батареей – при пропадании внешнего питания за её счет у устройства оставалось время записать в память текущие данные. На некоторых устройствах проблема проявлялась чаще, чем на остальных; кроме того, при повторной подаче питания устройство могло запуститься. Это означало, что с данными всё в порядке – проблема проявлялась на этапе их чтения.

Инженер А изучил проблему и сделал вывод, что она связана с воздействием какой-то помехи на шину данных – на основании того, что её проявление была связано с операцией чтения. Устройство состояло из основной платы и платы памяти, на которой располагалась микросхема памяти и контроллер памяти с батареей резервного питания. Разъём между платами имел два контакта заземления и контакт, к которому подключалось 5 В, поэтому для повышения помехоустойчивости инженер добавил между платами толстый заземляющий провод. Он также добавил конденсатор в цепь резервного питания. Инженер составил запрос на внесение изменений в спецификацию, получил одобрение и внедрил её на производстве.

После выпуска первой партии плат с внесёнными исправлениями обнаружилось, что проблема до сих пор проявляется. Тогда к решению проблемы привлекли инженера Б.

В 9 утра инженер Б сел за свой стол, подключил осциллограф к шине данных и стал наблюдать, что происходит, когда основная плата пытается прочитать данные из памяти. Когда устройство не запустилось, то данные на шине не были *слегка* повреждены – *все* они представляли собой логические единицы. Инженер решил взглянуть на импульс чтения – и удивился, *не обнаружив его*. Дело было не в помехах, а в том, что микросхема памяти не получала импульс на чтение данных, и это являлось серьёзной проблемой. При этом микропроцессор основной платы формировал этот импульс и отправлял его на контроллер памяти – но тот не пересылал его микросхеме (см. рис. 12.1).

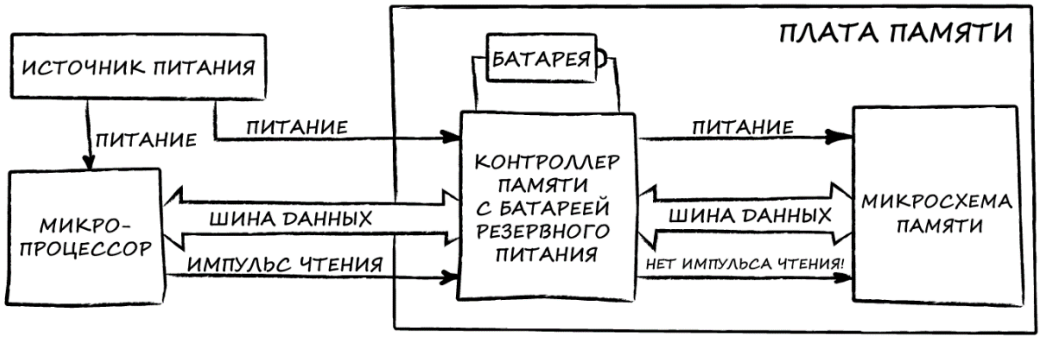


Рис. 12.1. Дело о потерянном импульсе чтения

Инженер Б быстро просмотрел руководство на плату памяти и выяснил, что одна из задач контроллера памяти состояла в предотвращении доступа к ней (и, следовательно, игнорированию импульсов чтения) при плохом питании. Это было похоже на хорошую зацепку, но в тот момент казалось, что с питанием всё в порядке.

В 9:45 инженер Б позвонил производителю платы памяти и пообщался с инженером по применению, который сказал: «О, у вас, вероятно, есть диод между контактом 5 В и цепью питания платы. Если это так – то подключите к контакту 5 В соответствующее напряжение, и контроллер памяти не будет детектировать проблемы с питанием». И действительно, схемотехника устройства была именно такой, как предположил инженер по применению (тут нужно пояснить: изначально устройство проектировалось без платы памяти; плата памяти была добавлена позже, и поэтому в схемотехнике были отступления от рекомендация производителя платы, что в тот момент казалось вполне разумным).

Инженер Б подключил ещё один провод от основной платы к контакту разъёма платы питания, чтобы обеспечить на нём наличие 5 В, как рекомендовал инженер по применению; когда он это сделал – то устройство стало запускаться при каждой подаче питания. Он убрал провод и увидел, что ошибка опять начала воспроизводиться. После этого он вернул провод и провёл полное тестирование устройства. Всё работало нормально. Инженер Б закончил свою работу в 10:15 – и выполнил поставленную перед ним задачу (правда, ему ещё нужно было написать отчёт, и, на самом деле, это заняло больше времени, чем весь сеанс отладки).

Давайте рассмотрим правила, использованные при отладке данной проблемы.

Изучите свою систему. Инженер А никогда не открывал руководство. Инженер Б прочитал его и, хотя он не понимал, почему нет импульса чтения, но предположил, что основным подозреваемым является плата памяти – а информация из руководства помогла ему установить её производителя и связаться с ним. Также он сразу понял, что при отсутствии импульса чтения все данные будут представлены логическими единицами.

Воспроизведите ошибку. То, что ошибка регулярно повторялась, упростило работу инженеру Б (а ещё выставило инженера А не в лучшем свете). Инженер Б мог увидеть отсутствие импульса чтения и логические единицы на шине.

Не предполагайте, а смотрите. Инженер А не смотрел на передаваемые данные (которые состояли только из логических единиц) и на импульсы чтения (которых не было). Если бы он увидел хоть что-то из перечисленного – то сразу бы понял, что проблема не в шуме.

Разделяйте и властвуйте. Инженер Б подключился к шине осциллографом и увидел, что вместо данных по ней передаются только логические единицы. Он попытался найти импульс чтения и понял, что его не было. После этого он двинулся «вверх по реке» и обнаружил, что импульс чтения поступает от микропроцессора в контроллер памяти, но не проходит дальше. Это позволило ему существенно сузить область поиска причины проблемы.

Вносите по одному изменению за раз. Хотя инженер Б подозревал, что исправления инженера А не имели никакого эффекта, он не убирал их во время своих тестов. Ошибка проявлялась и на устройствах с исправлениями – поэтому он отлаживался на них.

Записывайте всё, что происходит. Инженер Б не нашёл информации о том, почему инженер А решил, что причина проблемы в помехе, и отчётов о тестировании, которые бы это подтвердили. Возможно, инженер А вёл какие-то записи, но никому их не показывал. *А вот на производстве всё фиксировалось*; отчёты о сбоях при запуске устройств показывали, что данные в памяти не были повреждены, потому при следующей подаче питания они могли запуститься. Это подтолкнуло инженера Б сосредоточиться на изучении процесса доступа к памяти – и быстро обнаружить сплошные логические единицы на шине и отсутствие импульса чтения. Результаты выпуска первой партии устройств с исправлениями инженера А также свидетельствовали о том, что они не помогли, так как проблема не исчезла. Инженер Б записал всю собранную им информацию, включая имя помогавшего ему инженера по применению из компании-производителя платы памяти.

Проверьте кабель. Контроллер памяти вёл себя странно. Инженер Б не знал, почему это происходит; но он знал, что устройств с такими платами выпускается много, и проблема проявляется на существенном количество устройств – так что, вероятно, проблема не в самом контроллере. Возможно, этот контроллер неправильно используется? Он задался этим вопросом – и, конечно же, дело оказалось в специфической проблеме с питанием.

Воспользуйтесь чьим-то свежим взглядом. Инженер Б *не знал* – действительно ли контроллер памяти используется неправильно? Поэтому он обратился к эксперту (инженеру по применению из компании-производителя платы памяти). Эксперт всё знал о своей плате – и сразу рассказал это инженеру.

Если вы не исправили ошибку – она не исчезла. Инженер А, очевидно, не очень хорошо протестировал свои исправления, потому что они не решили проблему. Эта досадная неудача дала инженеру Б действительно веский повод тщательно проверить *свои* исправления, прежде чем написать запрос на внесение изменений в производственную спецификацию, на котором будет стоять *его* подпись.

Глава 13: Простые упражнения для читателя

То, что изобретено одним человеком, может быть понято другим.

Шерлок Холмс, «Пляшущие человечки»

13.1. Общий обзор

Можете ли вы назвать правила отладки, которые используются или игнорируются в приведённых ниже историях? (подсказка: соответствующие фрагменты текста выделены *курсивом.* Каждый фрагмент отмечен надстрочной цифрой. В конце истории приводится соответствие этих цифр правилам отладки).

13.2. Уборка со светопредставлением

Байка ветерана. Эта история об отладке проблем с проводкой в доме с 90-летней историей, в который моя семья переехала в июне 1984. История действительно была странной, и она дала мне возможность поквитаться с отцом за его комментарий «Когда всё остальное не помогло – читай инструкцию».

Готовясь к визиту отца в вышеупомянутый дом, моя жена включила пылесос в розетку в столовой. Позже супруга сказала мне, что с этой розеткой что-то не так, *потому что при включении пылесоса она увидела вспышку*[(1)]. Конечно, из любопытства *я подключил пылесос к той же розетке, нажал кнопку включения, тоже увидел вспышку и быстро выключил его*[(2)]. Я понял, что вспышка не была похожа на электрический разряд. Скорее – на кратковременное «промаргивание» лампы освещения.

Я снова осторожно *нажал на кнопку включения пылесоса*[(3)] – и, конечно же, люстра над моей головой зажглась. Однако пылесос не включился. Вся эта ситуация показалась мне довольно забавной, *и я решил дождаться отца, чтобы в его присутствии разобраться с проблемой*[(4)]. Он, конечно, не верил, что такое вообще могло случиться (и воскликнул: «*Но этого не может быть*!»[(5)]). Но мы *наглядно продемонстрировали ему*[(6)], что можем управлять нашей люстрой с помощью переключателя питания пылесоса.

Мы на мгновение задумались о том, *как работает люстра*[(7)]. Она управляется двумя выключателями, расположенными в разных концах столовой. Электричество поступает из сети и подводится к одному из выключателей, который может замкнуть любой из двух проводов, идущих ко второму выключателю. Второй выключатель соединяет этот провод с общим контактом, по которому электричество проходит через лампу и далее уходит в контур заземления. Если один и тот же провод замкнут двумя выключателями – то электрическая цепь замкнута, и в столовой горит свет. Если выключатели замыкают разные провода – то цепь разомкнута, и света нет. С помощью любого выключателя можно как включить, так и выключить свет.

Что ж, *мы предположили*[8], что розетка, в которую включался пылесос, вероятно, неправильно подключена к цепи питания между двумя выключателями. И действительно – *когда мы спустились в подвал и проследили за проводкой*[9], то обнаружили две установленные рядом распределительные коробки: одну для главной цепи питания, а вторую – для цепи выключателей. Розетка в столовой была, конечно же, подключена не к той распределительной коробке и оказалась в составе цепи выключателей (см. рис. 13.1). Когда мы включали пылесос – он замыкал цепь между выключателями, тем самым включая свет. Но всё напряжение уходило на лампу люстры, так что двигатель пылесоса не запускался.

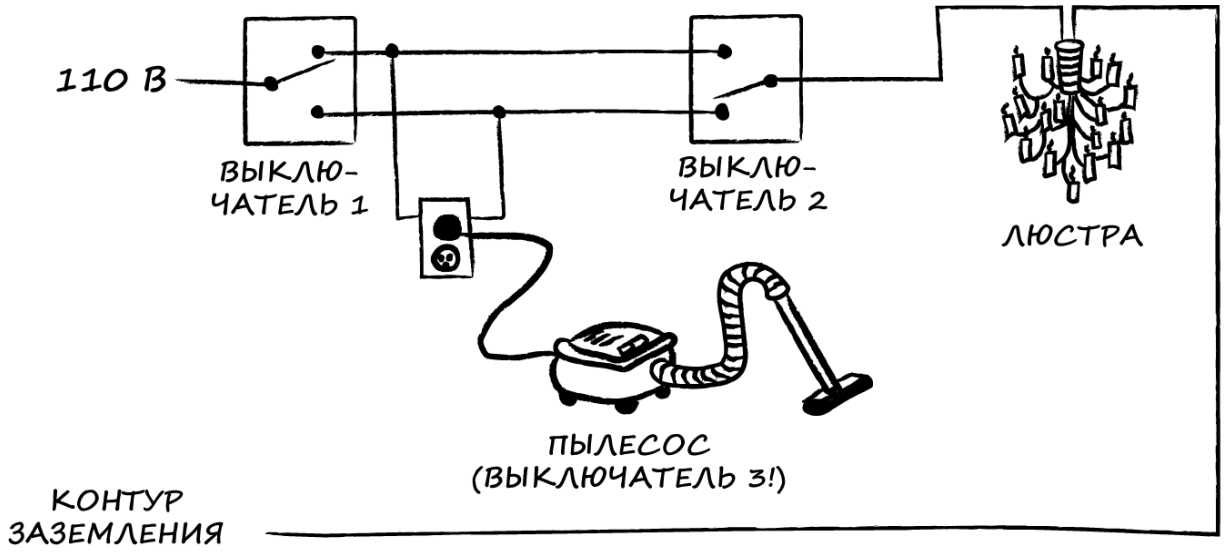


Рис. 13.1. Три выключателя

Прокладка проводки в старых домах может быть довольно занимательным процессом – если, конечно, при этом дом не сгорит дотла.

Правила, использованные и (проигнорированные) в данной истории:

1. **Записывайте всё, что происходит**. Моя жена не записала то, что увидела, но её слова стали поводом для начала отладки. Она *запомнила*, что произошло, и не просто сказала «Розетка не работает», но и уточнила, что проблема с *конкретной* розеткой, а также упомянула про вспышку света в момент включения пылесоса.

2. **Воспроизведите ошибку**. Я сам повторил действия жены, а не кинулся перекладывать проводку.

3. **Воспроизведите ошибку**. Результаты моего эксперимента были довольно странными, так что я повторил его. Два сбоя за две попытки выглядели вполне статически достоверными.

4. **Воспользуйтесь чьим-то свежим взглядом**. Я вовлёк в эту историю отца не столько потому, что был в тупике – мне хотелось увидеть выражение его лица, когда я буду описывать проблему. Часто это хороший повод, чтобы проконсультироваться с другим инженером по поводу странной ошибки – стоит рассказывать заинтересованным людям о всяких интересных происшествиях.

5. (**Не предполагайте, а смотрите; Воспроизведите ошибку**). «Этого не может быть!» – такое заявление может сделать тот, кто думает лишь о том, почему какое-то событие не может произойти, но не видел, как оно на самом деле случается.

6. **Воспроизведите ошибку**. Мы довольно быстро убедили моего отца, что «это» *может* случиться – и, более того, стабильно повторяется.

7. **Изучите свою систему**. Проблема была связана с проводкой, поэтому мы проверили, как устроена электрическая цепь, связанная с люстрой. Понимание принципа работы выключателей дало главную подсказку о том, где искать причину проблемы.

8. **Не предполагайте, а смотрите**. Мы выдвинули предположение – но лишь затем, чтобы сузить область поиска до проводки, связанной с выключателями.

9. **Не предполагайте, а смотрите**. Мы отследили, куда идут провода, и визуально обнаружили причину проблемы.

13.3. Стая багов

Байка ветерана. В 1977 году мы работали над платой, в состав которой входила микросхема памяти с неслыханным (по тем временам) объёмом в четверть мегабайта. Но, к сожалению, данные из памяти не всегда считывались корректно. Один из наших программистов *написал тестовую утилиту*[(1)], которая заполняла всю память ASCII-кодом буквы F (с неё начиналось название нашей компании), а затем считывала из памяти данные и *выводила их на видеотерминал*[(2)].

В те времена мы использовали видеотерминалы, а не ПК. Пояснение для молодёжи: видеотерминал похож на компьютер, но у него нет «мозгов»; он просто выводит строки текстовых символов – слева направо, сверху вниз, строка за строкой. Когда весь экран заполнен – то для отображения новой строки его содержимое «прокручивается» на одну строку вниз и, таким образом, верхняя строка «выталкивается» за границы экрана.

Утилита работала довольно быстро, так что после начала чтения из памяти экран видеотерминала практически мгновенно заполнялся символами F. Изображение на экране выглядело статичным – ведь каждая строка была идентична предыдущей. Но, на самом деле, отображаемые строки быстро прокручивались вверх.

Каждый раз при возникновении ошибки чтения памяти вместо символа F отображался V[(3)]. Он был похож на взлетевшую от испуга птицу (см. рис. 13.2).

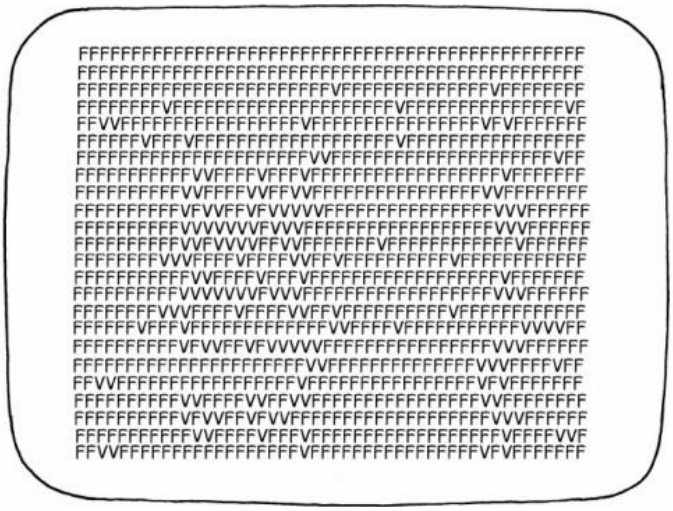


Рис. 13.2. Пугающие символы «V»

Мы подозревали, что причина проблемы связана с помехой[(4)], действующей на микросхему. Один из способов повысить интенсивность помехи – прикоснуться пальцем к проблемной точке устройства. *Я примерно знал, какая «ножка» микросхемы мне нужна*[(5)], потому что ASCII-код символа F в двоичной системе выглядит как 0b01000110, а символа V – 0b01010110. Очевидно, что проблема была в четвёртом[11] бите – в какие-то моменты времени он принимал значение 1, хотя и не должен был. *Когда я прикасался к определённой группе «ножек» на микросхеме памяти – на экране возникали десятки символов V, напоминавшие стаю птиц, испуганно взлетевших от звука выстрела*[(6)] (см. рис. 13.2).

Я убрал палец – и символы V пропали[(7)]. После этого *я решил уменьшить область поиска*[(8)]; поскольку размер моего пальца был существенно больше «ножек» микросхемы, то я взял небольшой кусок проволоки и стал поочередно прикасаться к каждой «ножке» – *и вскоре нашёл «проблемную»*[(9)]. Она отвечала за шину управления памятью. *Я подключился к ней осциллографом*[(10)] – и *увидел*[(11)] «звон» (перерегулирование) сигнала.

Оказалось, что шина была слишком длинной (размер нашей платы составлял 45.7 × 71 см), и на её концах требовалось установить терминаторы (речь о согласующих резисторах, а не роботе, которого играл Шварценеггер) для устранения помех, связанных с отражением сигнала от конца линии связи. *Мы установили терминаторы и убедились, что «птицы» исчезли. Сняли их и проверили, что «птицы» вернулись; опять вернули терминаторы*[(12)]. Это позволило нам проверить наше исправление. Поскольку в возникновении проблемы не было систематики – то в будущем она могла проявиться и в других аналогичных местах, так что *мы также установили терминаторы на всех остальных сигнальных линиях других микросхем нашей платы*[(13)].

[11] Традиционно нумерация бит ведётся с нуля (прим. переводчика).

Правила, использованные и (проигнорированные) в данной истории:

1. **Воспроизведите ошибку**. Тестовая утилита работала очень быстро, поэтому ошибка повторялась примерно каждые 20 секунд, а не раз в час.

2. **Воспроизведите ошибку; Не предполагайте, а смотрите**. Тестовая утилита позволяла не только систематично воспроизводить ошибку, но и отследить момент её появления на видеотерминале.

3. **Воспроизведите ошибку; Не предполагайте, а смотрите**. Видеотерминал позволял не только отследить появление ошибки, но и увидеть, каким именно образом повреждаются данные. Это стало важно позднее.

4. **Изучите свою систему**. Новые аппаратные решения часто имеют проблемы, проявляющиеся периодически и без какой-либо систематики. Обычно их причинами являются помехи (блуждающие токи, которые превращают логические единицы в нули и наоборот).

5. **Изучите свою систему**. Мы знали ASCII-коды символов и какие «ножки» микросхемы памяти за какие биты отвечают (кстати, для молодёжи упомяну – да, в старые времена большие микросхемы памяти действительно имели «однобитную» адресацию).

6. **Воспроизведите ошибку**. Увеличив интенсивность помех путём нажатия пальцем на определённые точки микросхемы – мне удалось заставить ошибку проявляться чаще. Для некоторых микросхем прикосновение пальцем похоже на проверку наличия течи в окне с помощью пожарного шланга – помеха приведёт к выходу из строя совершенно исправных элементов устройства. Иногда бывает и обратное – ёмкость пальца снизит уровень помехи и «успокоит» систему (есть легенда, что в 1970-х в лаборатории аналоговых схем Массачусетского технологического института аспиранты разработали простой прибор, который был электрическим эквивалентом пальца: когда у них возникала проблема с помехами, они прикасались им к различным узлам схемы до тех пор, пока помеха не исчезала. После этого они впаивали этот прибор в цепь, решая таким образом проблему на постоянной основе).

7. **Воспроизведите ошибку**. Я убедился в том, что причиной увеличения интенсивности проявления ошибки стал мой палец, удалив его (из микросхемы, а не совсем!).

8. **Разделяйте и властвуйте**. Я сузил область поиска ошибки до нескольких «ножек» микросхемы. Мне оставалось только найти «ту самую».

9. **Вносите по одному изменению за раз**. Я проверил, что увеличение интенсивности ошибок не было связано с тем, что мой палец замыкал несколько «ножек» микросхемы в единую цепь.

10. **Не предполагайте, а смотрите**. Я использовал осциллограф, чтобы изучить вид сигнала на найденной мной «ножке». Я предполагал, что увижу помеху, но не был в этом уверен. Я просто посмотрел.

11. **Не предполагайте, а смотрите**. И я не увидел помеху. Я увидел перерегулирование – а это другая проблема, которая требует другого решения. Перерегулирование действительно делает систему более подверженной влиянию помех, поэтому в обычных условиях ошибка проявлялась периодически, а при касании моего пальца – существенно чаще. Если бы я предположил, что причиной ошибки является помеха, и принял другие меры, то просто бы сделал проявление ошибки более спорадическим – но она бы не исчезла.

12. **Если вы не исправили ошибку – то она не исчезла**. Мы убедились, что наше исправление решило проблему – ошибка теперь не проявлялась как при касании пальцем микросхемы, так и при его отсутствии. Мы также проверили, что ошибка проявляется после удаления нашего исправления – опять же, независимо от наличия моего пальца.

13. **Если вы не исправили ошибку – то она не исчезла**. Мы выясняли, что причиной ошибки был недочёт в схемотехнике, и что эта же ошибка может проявиться в других сигналах и на других микросхемах. Поэтому мы внесли исправления и для них – чтобы предотвратить потенциально возможные проблемы в будущем.

13.4. Свобода от ограничений

Байка ветерана. Наша компания производила устройства, которые превращали ПК в системы видеоконференцсвязи. Некоторые устройства состояли из двух плат: одна отвечала за сжатие аудио/видео и реализацию протоколов обмена, а другая обеспечивала подключение к телефонной сети. У нас было два типа коммуникационных плат:

- одна с интерфейсом ISDN, которая непосредственно подключалась к телефонной сети;
- вторая с последовательным V.35, которая соединялась с блоком поддержки пользователей (БПП) – и уже он подключался к телефонной сети.

Это было отличным решением – и мы продавали платы обоих типов.

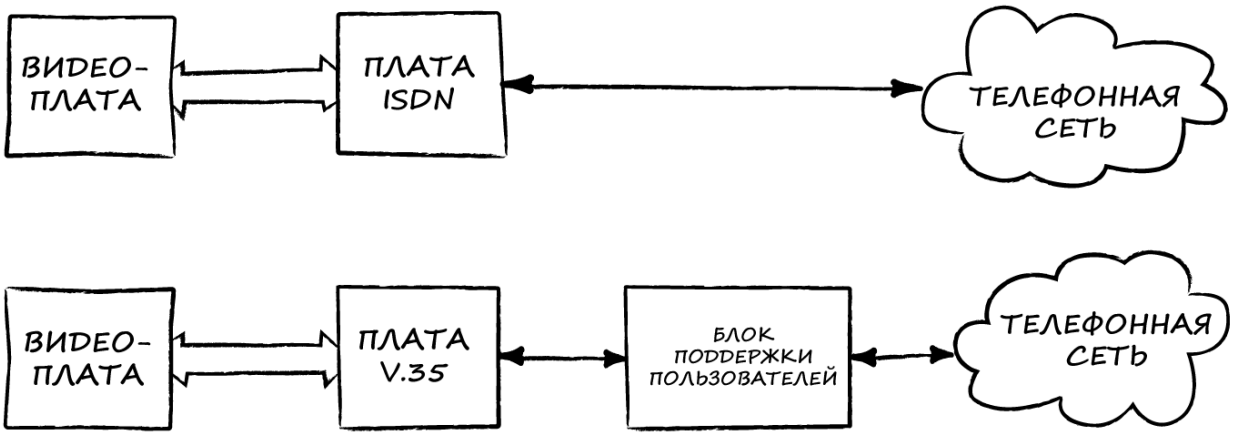


Рис. 13.3. ISDN и V.35

Мы разработали новую версию видеоплаты. Она использовала тот же коммуникационный интерфейс, что и старая, так что должна была работать с обоими версиями коммуникационных плат. Но специалисты из отдела контроля качества обнаружили, что новая плата не может осуществлять вызовы определённого типа через плату V.35. *Каждый раз они завершались неудачно*[1] – ни одного успешного вызова не было. А вот плата ISDN работала отлично.

Упомянутые вызовы относились к «вызовам с ограничениями» – особому типу вызовов, который позволяет телефонным компаниям использовать устаревшее коммутационное оборудование. Ранее телефонные компании использовали 8-битные каналы передачи данных, но один из битов использовался для внутренних целей – так что оконечное оборудование могло использовать только остальные 7 бит для передачи данных. Современные коммутаторы телефонных компаний обеспечивают вызовы по «чистому» каналу, поэтому оконечное оборудование может использовать все 8 бит. В системах видеоконференцсвязи при вызове, если есть возможность, используется «чистый» канал, но иногда с этим возникают проблемы, потому что телефонная компания может коммутировать наши сигналы таким образом, что они пройдут только через устаревшее оборудование. Когда это происходит – системы видеоконференцсвязи выполняют «вызов с ограничениями» и используют только 7 бит данных для передачи одного «символа» (под «символом» подразумевается совокупность стартового бита, стопового бита, бита контроля четности и битов данных).

По интерфейсу между видеоплатой и коммуникационной платой всегда передается 8 бит данных, но в случае «вызова с ограничениями» – только 7 из них являются значимыми. Коммуникационная плата с интерфейсом V.35 отбрасывает незначимый бит и отправляет 7 бит данных блоку поддержки пользователей (БПП; см. рис. 13.4).

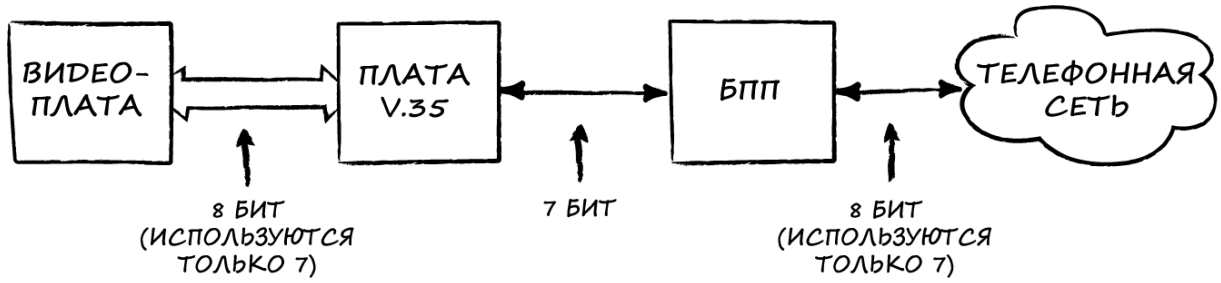


Рис. 13.4. Особенности «вызовов с ограничениями» для V.35

Поскольку с платой ISDN проблема не проявлялась, а для ПО тип коммуникационной платы не имел значения, наши инженеры-программисты *пришли к выводу, что проблема, вероятно, в аппаратной части видеоплаты*[2], в которой учитывались некоторые небольшие различия интерфейсов ISDN и V.35. Наши «железячники» настаивали на том, что, поскольку вызовы по «чистому» каналу для V.35 работают нормально, а для аппаратной части видеоплаты тип канала («чистый» или «с ограничениями») не имеет значения, то, *вероятно, проблема в ПО*[3].

В логах мы нашли информацию о том, что видеоплата не смогла обнаружить признак окончания кадра (который необходим для завершения вызова) в битах, полученных от БПП[4]. Один из разработчиков подтвердил это, *выведя в логи дамп буфера входящих данных*[5]. Затем разработчики ПО привлекли меня, чтобы найти проблему в аппаратной части – так как *я понимал и протоколы обмена, и аппаратную часть платы V.35*[6].

Сначала я *подтвердил отсутствие признака окончания кадра*[7], используя инструменты отладки упомянутого выше разработчика. Затем я *подключился осциллографом к линии связи*[8] между коммуникационной платой и БПП. Признак окончания кадра не передавался. После этого я *посмотрел на биты*[9], которые отправляла видеоплата – и с удивлением обнаружил признак окончания кадра в старшем (7-ом) бите. Предполагалось, что этот бит будет отброшен платой V.35 и обрабатываться не будет. Признак окончания кадра должен быть передаваться в бите 6.

Мы нашли функцию, которая формировала признак окончания кадра, и *добавили вывод сообщения в лог при каждом её вызове*[10]; и увидели, что признак окончания кадра всегда некорректно выставляется в седьмой бит «символа». Мы нашли ошибку в коде и исправили её; плата V.35 стала работать без ошибок. *Конечно, мы убрали исправление, воспроизвели ошибку и заново внедрили своё исправление. Теперь мы знали, что именно наше исправление решило проблему*[11].

Ошибка возникла при разработке ПО для новой видеоплаты – поэтому она не работала с платой V.35, а старая работала. Мы были озадачены – почему же новая видеоплата работает с платой ISDN? Ведь и в данном случае признак окончания кадра устанавливался в не тот бит, в который нужно. *Мы подтвердили это, сняв дамп с линии связи*[12] между видеоплатой и платой ISDN. Но через какое-то время признак окончания кадра начал устанавливаться в «нужный» бит.

Оказалось, что после того, как у платы ISDN не получалось связаться с оконечным оборудованием, она отправляла команду, в которой говорилось: «Это вызов с ограничениями». Оконечное оборудование получало эту команду, и начинало устанавливать признак окончания кадра в бит 6. Таким образом, устройства устанавливали связь и после этого переходили в режим вызова с ограничениями.

«Постойте!» – должны воскликнуть вы. – «Как вообще плата ISDN могла установить признак конца кадра, если это изначально был вызов с ограничениями?» Мы спокойно ответим: *«Почему вы думаете, что это был вызов с ограничениями?»*[13]. Наш отдел контроля качества использует специальный внутренний коммутатор, который позволяет тестировать ISDN, поэтому мы не платим ни цента телефонной компании, когда тестируем тысячи вызовов за ночь. Судя по всему, *этот коммутатор прекрасно эмулирует вызовы с ограничениями, но с одним исключением – он считает, что нет необходимости отбрасывать старший бит входящего «символа»*[14]. Плата ISDN отправляла коммутатору 8-битные «символы» (см. рис. 13.5). Таким образом, эмулятор вызовов с ограничениями позволял плате ISDN обойти ошибку и переходить в режим вызова с ограничениями. Плата V.35, которая передавала только 7 бит данных в «символе», не могла «обмануть» эмулятор.

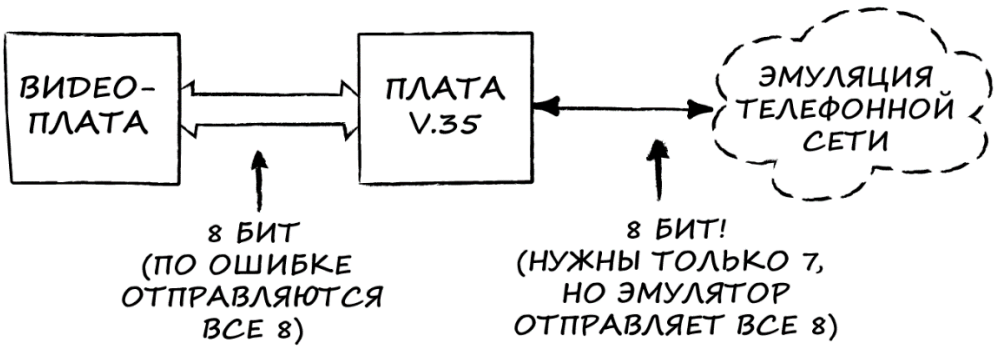


Рис. 13.5. Свобода «вызовов с ограничениями» плат ISDN

Я не знаю, проводилось ли какое-то тестирование наших плат ISDN с реальными телефонными линиями в режиме вызовов с ограничениями. Но либо не проводилось, либо оборудование телефонной компании передавало 8-битные «символы», когда получало их – даже если в этом не было необходимости. В любом случае, ошибка влияла и на платы ISDN – и ждала, пока её кто-нибудь обнаружит. *Если бы наши платы ISDN использовались в реальной сети телефонной компании в режиме вызовов с ограничениями – то мы увидели бы точно такую же ошибку, как и с платами V.35*[(15)].

Правила, использованные и (проигнорированные) в данной истории:

1. **Воспроизведите ошибку**. Нам очень повезло, что наша ошибка повторялась в 100 случаях из 100. Поскольку воспроизводимость была настолько стабильной – с решением проблемы не должно было возникнуть трудностей. Но отсутствие какой-либо систематики усложняло нашу задачу.

2. **(Не предполагайте, а смотрите).** Это предположение, как вскоре выяснялось, было неверным и лишь отвлекало от изучения проблемы. Это было классическое тыканье пальцем в другого.

3. **(Не предполагайте, а смотрите).** Обычное тыканье пальцем в ответ. Оно оказалось верным – но это чистая случайность. Программисты не стали исследовать проблему, потому что считали, что она не в их зоне ответственности. Это очень плохой способ мышления, который и является причиной тыканья пальцем – он только *затрудняет* исследование проблемы.

4. **Не предполагайте, а смотрите**. Этот небольшой тест позволил нам точно определить, когда проявлялась проблема (в момент вызова) и в чём она заключалась (в отсутствии признака окончания кадра). Это помогло нам определиться со следующими действиями.

5. **Не предполагайте, а смотрите**; **(Разделяйте и властвуйте); (Не предполагайте, а смотрите)**. Один из программистов провёл эксперимент – но лишь для того, чтобы доказать, что это «железо» устанавливает признак окончания кадра. Он не стал разбираться, как именно происходит отправка данных, а проблема была связана именно с этим. Инженер *думал*, что в этом нет необходимости, потому что *считал*, что проблема в «железе».

6. **Изучите свою систему**. Я не только разбирался как в аппаратной, так и в программной части системы, но был ещё и непредвзятым участником этой истории – у меня не было причин тыкать в кого-то пальцем (меня всегда удивляло, как при возникновении проблем люди начинают тыкать друг в друга пальцами. Я *предпочитаю*, когда ошибка на моей стороне – это значит, что я могу её исправить. И нужно работать сообща с коллегами – даже если «пробоина на их конце лодки», то тонуть-то в итоге вы будете все вместе).

7. **Не предполагайте, а смотрите.** Я хотел увидеть всё своими глазами, так как знал протокол обмена лучше, чем программист, который первым провёл этот эксперимент.

8. **Разделяйте и властвуйте**. Он смотрел на данные, которые получало ПО. Я пошёл «вверх по течению» и посмотрел, что именно передается по линии связи.

9. **Разделяйте и властвуйте**. В битах, передающихся по линии связи, действительно отсутствовал признак окончания кадра, поэтому я двинулся дальше «вверх по течению». Поскольку в нашей

системе инициатором связи могла быть любая из сторон – то я мог бы двинуться дальше к плате V.35, оттуда к БПП, далее в сеть телефонной компании, вышел бы к БПП другой стороны, прошёл бы через их плату V.35 и оказался бы на линии связи между ней и видеоплатой. Но, признаюсь, я предположил, что проблема будет воспроизводиться на обоих сторонах, и стал изучать ту систему, которая была рядом со мной (и к которой можно было легко подключиться осциллографом). Но если бы я не увидел в ней ничего подозрительного – то действительно перешёл бы к изучению работы аналогичной системы, которая принимала вызов нашей системы.

10. **Разделяйте и властвуйте; Не предполагайте, а смотрите**. Я двинулся дальше и перешёл к ПО, в котором был реализован протокол обмена. Использовав инструменты отладки – я выяснил, что режим вызова с ограничениями обрабатывается некорректно.

11. **Если вы не исправили ошибку – она не исчезла**. Мы проверили, что наше исправление действительно устранило причину проблемы.

12. **Если вы не исправили ошибку – она не исчезла; Не предполагайте, а смотрите**. Терзавший нас вопрос о том, почему проблема не проявлялась при использовании платы ISDN, заставлял сомневаться в действенности нашего исправления. Мы хотели разобраться, что именно происходит, и стали изучать систему с платой ISDN.

13. **(Проверьте кабель)**. Это ошибочное предположение привело к первоначальной ошибочной диагностике проблемы. Мы предполагали, что отсутствие проблемы в случае использованы платы ISDN было подтверждено корректной процедурой тестирования.

14. **(Проверьте кабель)**. Разработчики коммутатора не считали необходимым обрабатывать старший бит «символа». Они предполагали, что отправка этого бита не повлияет на какие-либо тесты – ведь тестируемое оборудование всё равно не использует этот бит, верно?

15. **(Если вы не исправили ошибку – она не исчезла)**. Если бы у клиентов возникла потребность в использовании вызовов с ограничениями через плату ISDN – это привело бы к проявлению исходной проблемы. Нам просто повезло, что этого не случилось.

13.5. Дело раскрыто

Байка ветерана. Мы работали над портативным устройством с аналоговым сенсорным экраном. Сенсорный экран передавал процессору два сигнала напряжения – по одному для координат X и Y точки касания (см. рис. 13.6). Координаты имели некоторую погрешность, потому что сенсор неидеально ровно был приклеен к дисплею. Чтобы учесть это, был разработан механизм калибровки: мы касались сенсорного экрана в заранее определённых точках, измеряли в них значения напряжений и сохраняли эти числа. Затем, во время работы устройства с помощью некоторых математических методов на основании значений напряжений, измеренных при нажатии пользователя на экран, мы вычисляли координаты точки нажатия.

Разрешение дисплея было маленьким – всего он поддерживал 11 · 5 = 55 точек касания (5 «строк» по 11 «столбцов» точек).

1, 1	2, 1	3, 1	4, 1	5, 1	6, 1	7, 1	8, 1	9, 1	10, 1	11, 1
1, 2	2, 2	3, 2	4, 2	5, 2	6, 2	7, 2	8, 2	9, 2	10, 2	11, 2
1, 3	2, 3	3, 3	4, 3	5, 3	6, 3	7, 3	8, 3	9, 3	10, 3	11, 3
1, 4	2, 4	3, 4	4, 4	5, 4	6, 4	7, 4	8, 4	9, 4	10, 4	11, 4
1, 5	2, 5	3, 5	4, 5	5, 5	6, 5	7, 5	8, 5	9, 5	10, 5	11, 5

Рис. 13.6. Значения координат (X, Y) при идеальной калибровке экрана

Мы придумали механическое приспособление для калибровки экрана. Оно представляло собой сетку с отверстиями, каждое из которых соответствовало одной точке нажатия. Наладчик просовывал через каждое отверстие стилус, а ПО устройства фиксировало значения координат X и Y. Работая над прототипом, мы обнаружили, что нажатия неточно обрабатываются в правой части экрана, и совсем некорректно – в его правом нижнем углу. Мы внимательно изучили все расчёты, сделанные в процессе работы, но, казалось, в них не было ошибок.

Мы потратили некоторое время на анализ качества и стабильности работы экранов, и в итоге *я заметил, что проблемы с обработкой нажатий начинались сразу после калибровки*(1) и всегда *проявлялись в одной и той же области экрана*(2). Мне стало ясно, *что я не понимаю, как работает алгоритм калибровки*(3), и *не могу подтвердить его корректность*(4).

Следующее, что мы сделали – *изучили результаты калибровки*(5). Они были сохранены в двух массивах – в одном 55 значений X, в другом 55 значений Y. Мы ожидали, что значения в массиве X возрастают от 1 до 11, потом возвращаются к 1 и возрастают заново – в общем, что в массиве будет 5 последовательно размещенных серий значений (1, 2, 3 … 11), каждая из которых соответствует одной строке экрана. И мы действительно увидели именно эти числа.

В *массиве Y*(6) мы *ожидали увидеть*(7) 11 единиц, за ними 11 двоек и т. д – то есть, опять же, каждая серия значений должна соответствовать одной строке экрана. Однако *лишь 10 первых значений каждой серии оказались верными*(8) – последнее было на единицу больше, чем ожидалось, а 11-е значение для самой нижней (пятой строки) вообще выглядело случайным (см. рис. 13.7.).

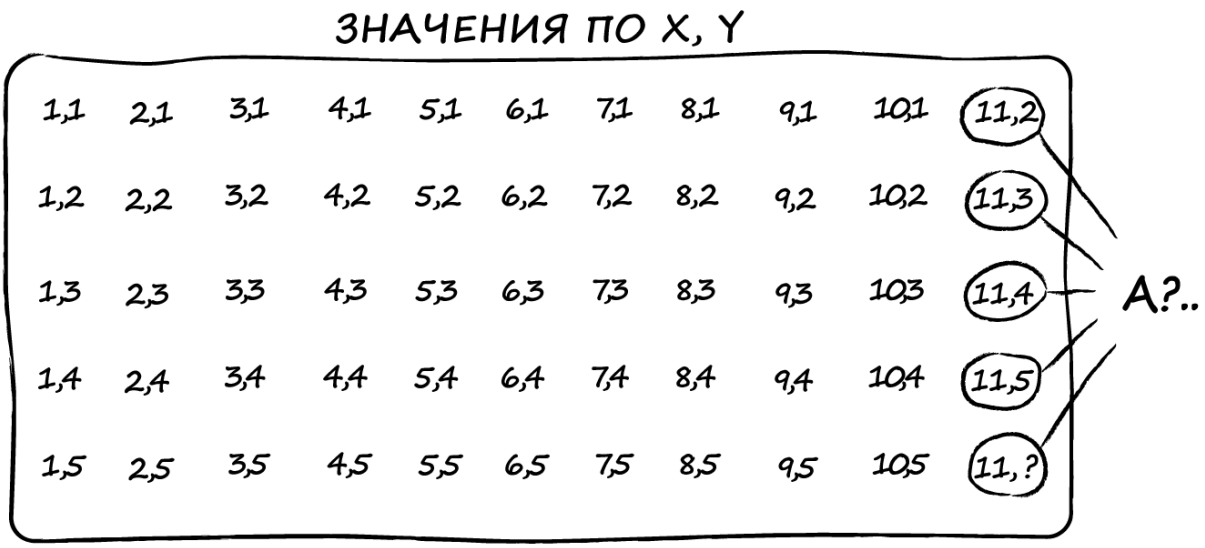


Рис. 13.7. Некорректно откалиброванный сенсорный экран

Мы проследили, как работает программа калибровки[9], и сразу поняли, в чём дело. Её разработчик объявил два массива – один для значений X, другой для Y – и *предположил, что компилятор разместит их в памяти последовательно и без разрывов*[10]. После завершения процедуры калибровки программа копировала значения X по адресу массива X, а значения Y – в (адрес X + 55). Это, несомненно, позволило её автору сократить свой код на одну строку.

Однако компилятор решил расположить массивы по чётным адресам, и поэтому из-за выравнивания памяти между ними возник «разрыв» (см. рис. 13.7). В результате данные массива Y были смещены в памяти на один байт влево – значение Y для 1 столбца 1 строки попадало в «разрыв», значение для 2 столбца 1 строки попадало в начальный элемент массива и т. д.

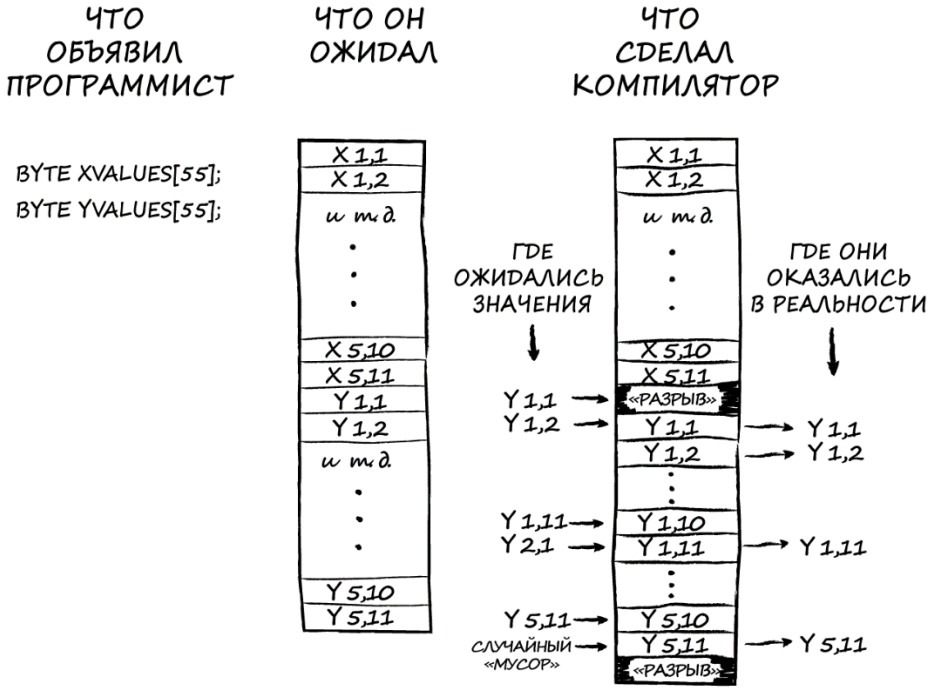


Рис. 13.8. «Разрыв» между теорией и реальностью

И, в целом, это оставалось для нас незаметным, потому что *используемые нами алгоритмы усреднения скрывали ошибку*[11] – за исключением значений последнего столбца. Последнее значение в массиве вообще не было инициализировано, и поэтому в нём оказывался случайный «мусор», что только усугубляло путаницу.

Мы внесли исправление в алгоритм калибровки и убедились, что после этого устройства стали корректно обрабатывать нажатия на любые точки экрана. *Затем мы убрали наше исправление, увидели, что проблема вернулась*[12], и со смущением взяли назад все неприятные слова, которые мы чуть раньше высказали о компании-производителе экранов.

Правила, использованные и (проигнорированные) в данной истории:

1. **Записывайте всё, что происходит**. Мы никогда не отслеживали, не «сбивается» ли калибровка с течением времени, поэтому предположили, что это может быть причиной проблемы. Когда я внимательно изучил ситуацию, то с удивлением обнаружил, что сразу после калибровки нажатия на экран отрабатываются неточно.

2. **Записывайте всё, что происходит**. Я не только зафиксировал факт наличия проблемы, но и локализовал область её проявления, а также определил направление для дальнейших экспериментов.

3. **(Изучите свою систему)**. Я не знал, как работает алгоритм калибровки, поэтому не думал, что в нём может быть ошибка. Забавно, что так случилось.

4. **(Проверьте кабель)**. Мы думали, что калибровка проходит корректно, потому что нажатия в большинстве точек обрабатывались правильно.

5. **Не предполагайте, а смотрите; Разделяйте и властвуйте**. Мы изучили «сырые» значения, полученные в результате калибровки – ещё до момента их обработки в программе, которая находилась «ниже по течению».

6. **Не предполагайте, а смотрите; Разделяйте и властвуйте**. Мы сфокусировались на значении координат Y, потому что погрешность при нажатиях всегда проявлялась в этом направлении.

7. **Изучите свою систему**. Мы знали, какой набор данных должны получить в результате корректной калибровки.

8. **Не предполагайте, а смотрите**. Когда вы смотрите на что-то, и оно выглядит не так, как вы ожидаете – значит, вы столкнулись с проблемой. Если вы будете просто думать об этом, то никогда не столкнётесь с таким «приятным» сюрпризом.

9. **Разделяйте и властвуйте; Не предполагайте, а смотрите**. Мы снова двинулись «вверх по течению» – к программе, которая обрабатывает и усредняет результаты калибровки. Когда мы стали отлаживать её код – то обнаружили причину проблемы.

10. **(Проверьте кабель)**. Вот отличный пример предположения о том, как работают инструменты. Это было очень небрежное предположение – и оно оказалось ошибочным.

11. **(Не предполагайте, а смотрите)**. Попытка установить причину проблемы, изучая обработанные результаты калибровки, не увенчалась успехом, потому что ошибка в этом случае в существенной степени нивелировалась алгоритмами усреднения – и картина происходящего скрывалась от нас в тумане. Не глядя на исходные данные – можно было подумать, что сенсор просто неидеально ровно приклеен к дисплею.

12. **Если вы не исправили ошибку – то она не исчезла**. Мы подтвердили, что причина проблемы действительно находится в коде калибровки, и удостоверились, что наше исправление устранило её.

Глава 14: Взгляд из службы технической поддержки

Неудобно делать дела под чужим именем.

Шерлок Холмс, «Голубой карбункул»

14.1. Общий обзор

Байка ветерана. Я общался с нашим итальянским клиентом по имени Джулио. Он пытался настроить систему видеоконференцсвязи, используя нашу видеоплату и коммуникационную плату ISDN от другого производителя. Джулио объяснил мне, что интерфейс этой платы похож на интерфейс нашей платы ISDN, но требует некоторых изменений в схеме подключения и доработки программной части нашей видеоплаты, отвечающей за передачу данных. Эти доработки касались используемой тактовой частоты, типа импульсов и прочих «аппаратных» коммуникационных настроек, неправильные значения которых приводят к искажению данных.

У Джулио как раз это и наблюдалось. Будучи экспертом по данной системе, я вместе с ним занимался подбором правильного сочетания значений параметров. Он прислал мне по факсу временны́е диаграммы его платы ISDN, что позволило мне определить, как нужно настроить нашу видеоплату. После многочисленных попыток, которые осложнялись тем, что Джулио не очень хорошо владел английским (или, честно сказать – тем, что я вообще не владел итальянским), я наконец понял, что нам удалось установить правильные настройки. Но данные всё равно искажались. Мы снова проверили настройки и протестировали передачу данных. А потом ещё раз. Но проблема не исчезала. Помню, как подумал: «Похоже, я не замечаю чего-то очевидного. Жаль, что видеоконференцсвязь ещё не работает – иначе я мог бы посмотреть на систему собственным глазами, прямо со своего рабочего места».

Джулио прислал мне по факсу снимки с логического анализатора, на котором были видны искажённые данные. Такие анализаторы показывают только логические нули и единицы, но с их помощью нельзя увидеть помеху, действующую на кабель связи – так что я стал подозревать, что проблема именно в ней. Один из способов снизить уровень помех, действующих на кабель – это сделать его очень коротким. И у меня состоялся следующий диалог с Джулио:

Я: Пожалуйста, сделайте кабель как можно более коротким – чтобы его длина была не больше 5 см.
Джулио: Так не получится.
Я: Почему?
Джулио: Мне нужно оставить место для платы, подключённой в разрыв кабеля.
Я: Платы, подключённой в разрыв кабеля?!

Как оказалось, его плата ISDN использовала более высокую тактовую частоту, чем требовалось нашей плате, и Джулио решил эту проблему, поместив между ними свою собственную плату-конвертер (см. рис. 14.1). «Шум» от блока питания этой платы приводил к перегрузке провода заземления линии связи, что создавало сильную помеху. После того, как он подключил блок питания и кабель связи к отдельным несвязанным контурам заземления, проблема исчезла.

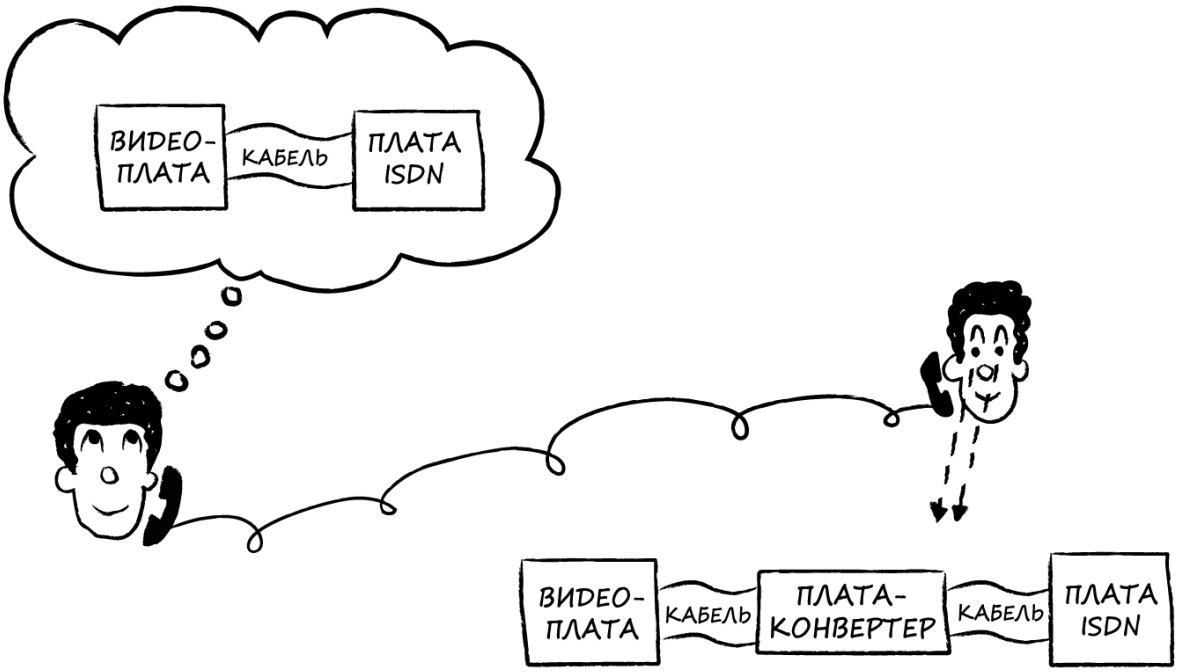


Рис. 14.1. Я и Джулио

Если вы работаете в технической поддержке – то вам знакома эта ситуация. Вы не можете увидеть, что происходит на том конце провода, и за миллион лет не догадаетесь, какие важные детали не рассказал клиент или какую чушь он себе придумал и транслирует вам. Вы наверняка слышали анекдот о пользователе ПК, который жалуется, что у него сломалась «подставка для кофе». Сперва замешкавшись из-за недоумения, техподдержка вскоре выясняет, что привод CD-ROM у клиента больше не выдвигается – вероятно, из-за того, что на него регулярно ставили чашку кофе.

Когда вы занимаетесь отладкой, являясь сотрудником технической поддержки – то возникает ряд специфических проблем, и данная глава поможет вам с ними справиться.

14.2. Ограничения, возникающие в процессе технической поддержки

Прежде чем рассмотреть, как применять правила книги в процессе технической поддержки, давайте определим, что отличает этот процесс от обычной процедуры отладки:

- вы работаете удалённо. Правилам намного легче следовать, когда система, в которой проявилась проблема, находится рядом с вами. Когда вы разговариваете по телефону с человеком, который сейчас присутствует возле неё, то не можете быть уверены, что он корректно описывает то, что происходит, и в точности выполняет ваши указания. Вы также можете столкнуться со странными и новыми для вас конфигурациями вашей системы, пользовательский интерфейс и документация для которых создана на неизвестном вам языке;

- человек на другом конце провода не так хорошо разбирается в этой системе, как вы. Умные люди понимают это – поэтому и звонят вам. В тяжелых случаях пользователь может считать, что разбирается в том, что делает, и успевает наломать дров, прежде чем обратиться в техническую поддержку. Как минимум, такие люди склонны забегать вперёд и делать то, о чём вы их не просили. В любом случае, они, вероятно, не читали эту книгу;
- вы занимаетесь устранением неполадок, а не отладкой. Если дело дошло до обращения в техническую поддержку – значит, проблема действительно есть, и вы уже не сможете по-тихому исправить её до релиза. Часто причиной проблемы является дефект (некорректная конфигурация ПО, выход их строя электронного компонента и т. д.), который *можно* исправить. Если вы действительно столкнулись с ошибкой (которую до этого никто не обнаруживал) – то, в большинстве случаев, устранить на месте её не получится; вам нужно найти обходной путь, который поможет клиенту, а затем сообщить об ошибке инженерам, чтобы они исправили её. Естественно, во время исправления дефекта или поиска обходного пути вы чувствуете серьёзное психологическое давление (сроки горят) – и это приводит к искушению отказаться от правил и пойти коротким путём.

14.3. Правила отладки: адаптация для использования при технической поддержке

В этом пункте я рассмотрю каждое из правил книги и дам вам несколько советов о том, как применять их, даже если человек на другом конце провода думает, что его CD-ROM – это подставка для кофе. Потому что, как бы ни было сложно следовать правилам, это необходимо, а значит, вам нужно найти подходящие для этого способы.

14.3.1. Изучите свою систему

Когда вам поступает звонок – это значит, что у клиента есть основания полагать, что его проблема связана с вашим продуктом; неважно, так это или нет, но ваши знания о продукте – единственное, на что вы можете рассчитывать. Очевидно, что эти знания должны быть очень глубокими и касаться не только всех аспектов продукта и рекомендуемых или основных вариантах его использования, но и предыдущих обращений в техническую поддержку, связанных с ним – то есть уже известных ошибок и обходных путей для их решения. Вероятно, вы знаете продукт лучше, чем инженеры, которые его спроектировали – и это хорошо.

Но, конечно, в системе клиента есть и другие компоненты – связанные с вашим продуктом, работающие поверх него или использующие его как платформу. Они забивают всю память или каким-то другим дьявольским образом делают использование вашего продукта невозможным. Ваша основная задача в рамках правила «Изучите свою систему» – выяснить, что это за компоненты, и как можно детальнее исследовать их.

Когда вы задаёте клиенту какой-то вопрос – то в ответ получаете мнение, которому придётся доверять, если нет других источников информации. Если ваш продукт имеет встроенные инструменты отладки – вы можете получить гораздо более точную информацию. Если у вашего продукта таких инструментов нет – то самое время отправиться в отдел разработки и стучать кулаком по столам менеджеров до тех пор, пока они не внесут добавление инструментов отладки в требования ко всем будущим версиям этого продукта.

Вы можете использовать и сторонние инструменты – например, ПК с Windows позволяет получить множество информации о том, какое «железо» и ПО в нём установлены, а средства мониторинга и диагностики покажут, на что расходуются ресурсы процессора.

Когда вы ничего не знаете о других компонентах системы (например, о плате ISDN, которая была у Джулио), то вам придётся получить эту информацию как можно более эффективным способом – у вас не будет времени ждать, пока служба доставки пришлёт вам руководство пользователя, так что придётся быстро изучить, что этот компонент из себя представляет и какие задачи выполняет. Сначала сосредоточьтесь на том, чтобы определить степень вероятности влияния этого компонента на наблюдаемую проблему; затем, если он вызывает подозрения, углубляйтесь в его изучение. В этот момент легко ошибиться, потому что невозможно глубоко погрузиться во все нюансы и не упустить чего-то важного. В истории про итальянскую плату ISDN я запросил временны́е диаграммы для передаваемых сигналов, но не увидел на них ничего, что указывало бы на некорректное значение тактовой частоты. У меня не было причин предполагать, что в разрыв кабеля связи установлена дополнительная плата-конвертер, так что я не стал дальше исследовать этот вопрос. Я потратил время зря, потому что сосредоточился на передаваемых данных, а не на тактовой частоте. Мораль такова: будьте готовы сместить фокус на другую область системы, если в той, с которой вы начнёте исследование проблемы, не найдётся ничего подозрительного.

Если другие компоненты системы являются аппаратными, то постарайтесь как можно раньше получить структурную схему системы. Если вам могут описать её только устно – нарисуйте такую схему самостоятельно и убедитесь, что вы действительно в точности понимаете, что на ней изображено. Убедитесь, что клиент согласен с используемыми названиями; довольно сложно понять фразу «моё устройство перестало работать после обмена с другим устройством». Даже если вам придётся называть их «устройство А» и «устройство Б» – убедитесь, что у вас с клиентом одинаковое ясное и недвусмысленное представление обо всей системе.

Наконец, неправильно подключённые кабели являются причиной множества странных проблем; если в системе есть кабели – запросите их схемы подключения. Жаль, что я не попросил об этом Джулио.

14.3.2. Воспроизведите ошибку

К сожалению, во многих клиентских обращениях причиной проблемы является череда специфических действий. Клиенты и правда не помнят, что делали, когда «всё сломалось». Изображение на экране ПК застыло, они потеряли терпение и начали кликать повсюду, а потом система сошла с ума. Или они просто пришли утром в офис – а ничего уже не работало. Или похоже, что кто-то пролил кофе на оборудование. Таким образом, вы можете получить лишь клиентскую интерпретацию совершённых ими действий – иногда она может сбить с верного пути или вообще отсутствовать.

Хорошей новостью является то, что, как правило, неисправность является серьёзной. Её легко повторить – просто попробуйте это сделать. Даже если у вас мало подсказок, почему возникла проблема – вы всегда можете заставить её проявиться и посмотреть, что происходит. Тот факт, что пользователь открыл в HEX-редакторе файл реестра, не так уж важен; сообщение об ошибке «Отсутствует запись в реестре» укажет вам, что этот файл был повреждён.

Вам всё равно необходимо получить чёткое представление о последовательности действий, вызывающих появление симптомов проблемы. Начните с самого начала; если необходимо – перезагрузите систему. Тщательно составьте список компонентов, окон, кнопок и полей ввода, с которыми взаимодействует клиент. И убедитесь, что вы точно выясняли, где именно происходит сбой. «Окно на другом компьютере выглядит странно» – не очень полезное сообщение при обращении в техническую поддержку. И когда клиент говорит: «У меня авария» – убедитесь, что он не играет в гоночный симулятор.

14.3.3. Не предполагайте, а смотрите

Когда вы пытаетесь отлаживать проблему путём взаимодействия с клиентом, который её наблюдает, то возникает три трудности:

- клиенты не понимают, на что вы просите их обратить внимание;
- клиенты не могут описать то, что видят;
- клиенты игнорируют ваши просьбы и не смотрят туда, куда вы их просите. Вместо этого они сразу дают вам ответ, который, по их мнению, является верным.

Вы, несомненно, умеете (или должны научиться) терпеливо сопровождать их во время сеанса отладки, просить повторять те действия, в корректности совершения которых не уверены, и сдерживать смешки от того, как они интерпретируют ваши просьбы («Хорошо, теперь – правой кнопкой мыши на поле ввода» – «Вы хотите, чтобы я написал “правой кнопкой мыши” в поле ввода?..». Есть два типа инструментов, которые могут очень помочь в устранении ошибок коммуникации, связанных с человеческим фактором.

Программы для удалённого доступа к ПК (если они у вас есть) могут позволить вам получить контроль над ситуацией. Программы для демонстрации экрана, даже если они не поддерживают возможность управления, позволят вам отслеживать, как клиенты выполняют ваши просьбы, и останавливать их, если они начнут делать что-то не то. Если эти инструменты не входят в ваш набор

корпоративного ПО, то, возможно, вы сможете использовать службу веб-конференции для просмотра экрана ПК клиента через Интернет. Подобные средства доступа могут быть запрещены корпоративными брандмауэрами и другими средствами защиты, установленными нервными системными администраторами, но некоторые из этих служб достаточно прозрачны в своей работе, если вам нужно единовременно наблюдать за работой только одной программы. Имейте в виду, что при таком удалённом подключении скорость обновления изображения экрана не будет приближена к реальному времени; вы не сможете таким образом отладить гоночный симулятор.

Второй тип инструментов, которые позволяют снизить влияние на отладку «человеческого фактора» – это логи. Если ваше ПО умеет генерировать логи с интересной для вас информацией и сохранять их в виде файлов – то клиент может отправить вам эти файлы по электронной почте для изучения (не заставляйте бедного клиента читать их – это сложно, и там часто встречаются бессмысленные с его точки зрения сообщения об ошибках или, по меньшей мере, множество слов с опечатками). Не забывайте следовать правилам отладки – для каждого лога должно быть указано, сохранён ли он в момент, когда проявилась ошибка, или наоборот – это лог, снятый во время корректной работы системы. Пусть клиент фиксирует метки времени, в которые он наблюдал проявление симптомов ошибки, *с подробным описанием этих симптомов*. Системное время всех устройств системы должно быть синхронизировано. Вся эта информация должна быть указана в обращении клиента и, в конечном итоге, в сообщении об ошибке, если потребуется сформировать его для передачи в отдел разработки.

Ещё одна проблема удалённой отладки заключается в том, что ваш набор инструментов ограничен. Мне повезло, что у Джулио был логический анализатор, и он смог присылать мне фото временны́х диаграмм, но такое случается нечасто. Даже если у клиентов есть нужные инструменты, обычно они не могут проникнуть во внутренности системы, а если и могут – то не обладают достаточной информацией, чтобы понять, куда именно эти инструменты надо подключить. С другой стороны, если у вас есть кто-то вроде Джулио с логическим анализатором – дерзайте. Даже обычный мультиметр может подсказать вам, что кабель подключен неверно или что питание отсутствует.

14.3.4. Разделяйте и властвуйте

В зависимости от конкретной системы – применить это правило может быть как очень легко, так и практически невозможно. Проблема в том, что вы занимаетесь анализом уже введённой в эксплуатацию системы – и поэтому не можете разобрать её на части или добавить отладочные инструменты в контрольные точки, чтобы понять, куда двигаться в поисках причины проблемы – «вверх по течению» или «вниз». Если в системе уже есть встроенные инструменты отладки – это отлично. Если промежуточные данные сохраняются в файлы, и их можно просмотреть – это здорово. Если работу системы можно представить в виде отдельных этапов, которые вы в состоянии переключать вручную и анализировать состояние системы между ними – это прекрасно. Если вы можете тестировать различные фрагменты системы независимо друг от друга – это чудесно.

Если система является «монолитной» (с точки зрения клиента), то это не очень здорово. Если проблема связана с неисправностью одного из электронных компонентов – можно просто заменить

всю плату или даже прибор (хотя во многих случаях это не возымеет никакого эффекта, потому что никто так и не доказал, что проблема связана с «железом» – и она не связана. Это особенно раздражает, если вам приходится согласовать замену и отправлять её клиенту; риск меньше, если на объекте есть ЗИП). Если проблема связана с ПО, то, возможно, вам придётся воспроизвести её у себя в офисе, чтобы иметь возможность внести исправления в код и использовать инструменты отладки. Если проблема связана с ошибкой конфигурации – скорее всего, вы не сможете воспроизвести её у себя, но вы можете создать специальную сборку ПО со встроенными инструментами отладки и отправить её клиенту, чтобы получить средства отладки там, где они вам нужны.

Постарайтесь не поддаться искушению начать просто менять аппаратные или программные модули вашей системы; но если это единственный способ разделить вашу проблему на части – то см. следующий пункт.

14.3.5. Вносите по одному изменению за раз

К сожалению, к тому моменту, когда вы получите обращение в техническую поддержку – клиент уже изменил всё, до чего только додумался. Он не откатывал свои изменения и, вероятно, уже не сможет точно вспомнить, что именно творил. Это проблема, но с ней ничего не поделаешь.

Что вы *можете* сделать – так это не усугублять ситуацию, меняя местами схожие компоненты системы. Но, как упоминалось в предыдущем пункте, иногда единственный способ разделить проблему на части – это заменить файл, программный модуль или аппаратный узел – и посмотреть, что изменится. В этом случае обязательно сохраните исходную конфигурацию системы и восстановите её после завершения теста.

Иногда система просто зависает, и единственный способ восстановить её работоспособность – это программная перезагрузка или перезагрузка по питанию, или даже переустановка ПО, в котором произошёл сбой. Иногда вообще приходится переустанавливать всю операционную систему. Это программный эквивалент поставки клиенту нового продукта. Вероятно, этот способ сработает, но все данные клиента будут потеряны. Если в системе есть настоящая ошибка, которая вынудила вас использовать такие печальные меры – то вы потеряете связанные с ней подсказки. Кроме того, переустановка ПО заставит вас плохо выглядеть в глазах клиента – он подумает, что вы хватаетесь за соломинку. И если это не решит проблему, то вы опозоритесь. Единственный хороший момент, который вы получаете после переустановки – теперь у вас есть известное начальное состояние, в котором вы можете вносить по одному изменению за раз.

14.3.6. Записывайте всё, что происходит

Как закалённый ветеран службы технической поддержки и эксперт, знающий правила отладки, вы, конечно, фиксируете всё, что происходит в процессе вашей работы. Единственная проблема заключается в том, что вы не знаете, что на самом деле происходит на объекте клиента. Клиенты часто делают больше или меньше того, что вы просите – или вообще делают что-то иное. Поэтому вы должны общаться с ними особым образом.

Поручая клиентам что-то сделать (или вернуть систему в прежнее состояние), заставьте их сообщить вам, когда они закончит, прежде чем переходить к следующему шагу. А ещё лучше – не просто спрашивайте их, сделали ли они то, о чём вы их просили, а уточняйте, как именно они это сделали; таким образом вы сможете убедиться, что они не совершили ошибок. Многие люди ответят «да» на первый вопрос, даже если они сделали что-то другое: раз они не поняли вашу просьбу в первый раз, то с чего бы им понять её во второй? Пусть своими словами расскажут, что именно они сделали – принудите их к этому.

Логи и другие средства сохранения аудиторского следа гораздо более надёжны, чем слова клиентов, поэтому собирайте и используйте всё, что найдёте. Приложите их к отчёту об ошибке; они могут пригодиться, когда подобная проблема возникнет в следующий раз или когда ваши инженеры займутся её исправлением. Типичной проблемой является поиск нужных файлов логов, поэтому сообщите клиентам, как правильно их назвать. Не пускайте это на самотёк, иначе вместо файлов с названиями «good.log» и «bad.log» вы получите «giulio.log» и «giulio2.log».

И напоследок – не переставайте искать информацию, связанную с проблемой. Неквалифицированные клиенты склонны упускать из виду важные детали, о которых вы никогда не догадаетесь. В прошлой главе я рассказывал про человека, который прикреплял дискеты к железному шкафу магнитом. Есть ещё одна известная байка о пользователе, который скопировал данные на дискету, наклеил на неё стикер и затем засунул в пишущую машинку, чтобы напечатать текст. Вы бы никогда так не сделали. Соответственно, вам и в голову не придёт спрашивать клиентов, не поступили ли они подобным образом. Всё, что вы можете сделать, это спросить у них, с чего всё началось, и что произошло потом, и так далее – пока они не дойдут до момента, когда позвонили вам.

14.3.7. Проверьте кабель

Существует городская легенда (возможно, основанная на реальных событиях) о сотруднике службы технической поддержки компании, которая разработала текстовый редактор. Клиент позвонил ему и пожаловался, что «с экрана исчез весь текст». Выяснив, что на экране вообще ничего не отображается, сотрудник посоветовал клиенту проверить кабель, которым его монитор был подключён к ПК. Клиент сказал, что это было сложно – кругом темнота, и только сквозь окно пробивался свет фонарей. Когда сотрудник техподдержки понял, что у клиента отключилось электричество – то якобы посоветовал ему отнести компьютер обратно в магазин и признать, что он слишком глуп, чтобы использовать его.

Помните, что нельзя считать ваших клиентов глупцами. Даже самые умные из них, вероятно, не знают, как именно устроен ваш продукт и какие условия необходимо соблюсти для его корректной работы. Да, они попытаются установить Windows на Mac. Да, они будут пробовать отправить документ по факсу, поднеся его к экрану компьютера (ладно, это не самые умные клиенты).

Подытожу: не стройте *никаких* гипотез о том, как клиенты используют ваш продукт. Проверьте каждую мелочь. И не позволяйте им услышать ваш смех.

14.3.8. Воспользуйтесь чьим-то свежим взглядом

Как уже упоминалось в главе 10 – руководства по устранению неполадок очень полезны, если вы столкнулись с известной проблемой. Вы занимаетесь устранением неполадок, и подобные руководства – ваши друзья. Вы должны обложиться всеми руководствами, которые сможете достать, и особенно это касается руководств по продуктам вашей компании, а также вашему баг-трекеру и базе знаний.

Пользуйтесь помощью коллег. Они могут знать недокументированные нюансы вашей системы. Они могли участвовать в попытках отладки ошибок, которые не увенчались успехом – но есть шанс, что собранная тогда информация окажется полезной при устранении той ошибки, которой занимаетесь вы. И, конечно же, вы можете получить от них свежий взгляд и мнение, которое поможет вам взглянуть на вашу проблему под другим углом.

Если у вас есть доступ к разработчикам вашей системы – они тоже в состоянии помочь. Как и ваши коллеги из службы технической поддержки, им, вероятно, известны недокументированные нюансы, которые могут быть вам полезны. Возможно, они могут предложить нетривиальные и творческие обходные пути для временного решения проблемы, если вам это потребуется. И, в конечном итоге, не исключено, что этим людям придётся исправлять ошибку, которой вы занимаетесь – так что им будет полезно обдумать вашу информацию.

14.3.9. Если вы не исправили ошибку – она не исчезла

Вы знаете, что не повесите трубку, пока клиент не убедится, что его проблема решена. Но даже если она больше не проявляется – то это не значит, что ошибка, лежащая в её основе, исправлена. И даже если так – когда-нибудь с ней могут столкнуться другие люди. И вы можете им помочь.

Во-первых, добавьте информацию в баг-трекер, базу знаний, руководство по устранению неполадок и т. д. Убедитесь, что вся информация, которую вы собрали, будет доступна следующему человеку, который столкнётся с такой же проблемой. Постарайтесь ёмко сформулировать её симптомы, чтобы ему было проще понять – «Ага! Это то же, что и у меня!». И чётко опишите, какие действия вы предприняли, чтобы устранить проблему – чтобы в следующий раз их можно было легко повторить.

Унция терпения стоит фунта мозгов.

Голландская пословица

Если ваше решение проблемы было лишь обходным путём для устранения проявления существующей ошибки, то ваш конкретный клиент будет счастлив, но другие пользователи наверняка столкнутся с той же самой ошибкой. Составьте отчёт об ошибке, отправьте его руководству и убедите всех, что для компании действительно важно её исправить (если это и правда так). Не соглашайтесь вытирать масло с пола и закручивать гайку; позаботьтесь о том, чтобы в выпущенных в будущем установках для крепления трубы использовалось 4 болта, а не 2.

Наконец, помните, что клиентам проще убедиться в том, что проблема действительно устранена, чем даже самому опытному инженеру технической поддержки. Посоветуйте им пристально наблюдать за системой, в которую вы внесли исправление – возможно, оно повлечёт за собой какие-то побочные эффекты. И если что-то произойдёт – пусть немедленно свяжутся с вами, чтобы вы смогли вернуться к делу до того, как они успеют совершить множество неосознанных действий, изменяющих состояние системы.

14.4. Памятка

Взгляд из службы технической поддержки всегда размыт

Вы находитесь вдали от объекта, клиенты – ваши «глаза» и «уши» – не очень точны в своих высказываниях, а решить проблему надо как можно скорее.

- **Следуйте правилам**. Вам придётся найти способы применять их назло неквалифицированным клиентам;
- **Проверьте, что сделал клиент, и к чему это привело**. Ваши пользователи будут неправильно понимать ваши инструкции и совершать ошибки. Вы должны сразу отслеживать такие моменты, так что проверяйте всё, что они говорят и делают;
- **Используйте инструменты отладки**. Избавьтесь от человеческого фактора с помощью логов и ПО для удалённого подключения и демонстрации экрана ПК;
- **Проверяйте даже самые базовые вещи**. Да, некоторые люди не осознают, что для работы текстового редактора требуется наличие питания у ПК;
- **Используйте доступные вам руководства по устранению неполадок**. Вероятно, вы имеете дело с известными и хорошо спроектированными продуктами; не игнорируйте их историю;
- **Участвуйте в дополнении руководств по устранению неполадок**. Если вы обнаружили неизвестную проблему в существующем продукте, то помогите вашим коллегам, задокументировав всю собранную вами информацию.

Глава 15: Подводя черту

Неужели от меня что-нибудь ускользнуло? Надеюсь, я ничего серьезного не упустил?

Доктор Ватсон, «Собака Баскервилей»

15.1. Общий обзор

Итак, вы выучили правила. Вы помните их наизусть, понимаете, как распознать, когда вы начинаете их нарушать (и останавливаете себя в этот момент) и знаете, как применить их в любой ситуации. Что дальше?

15.2. Веб-сайт, посвященный правилам отладки

Я создал сайт http://www.debuggingrules.com/, чтобы каждый желающий мог улучшить свои навыки отладки. Вам следует посетить его хотя бы по одной причине – скачать модный постер «Правила отладки»; вы можете распечатать его и повесить на стену в своём офисе, как я советовал в главе 2. Также на сайте есть ссылки на различные ресурсы, которые могут быть полезны при обучении искусству отладки. И я всегда рад услышать *ваши* «байки ветеранов» – интересные, забавные и поучительные (желательно, чтобы они имели все три качества сразу); на сайте написано, как их мне отправить. Не забудьте сделать это.

15.3. Если вы инженер…

Если вы инженер, программист, сотрудник технической поддержки или технический специалист другого рода – то теперь вы лучше умеете отлаживать ошибки, чем раньше. Используйте правила книги в своей работе и обучите им своих коллег. Посетите сайт, чтобы узнать о других ресурсах, посвященных отладке; скачайте постер с правилами. И повесьте его на стену, чтобы всегда помнить о них.

Обдумывайте каждый сеанс отладки после его завершения. Был ли он проведён эффективно? Как использование (или игнорирование) правил повлияло на эффективность, и что вы могли бы в следующий раз сделать по-другому? Какое правило следует подчеркнуть на постере?

15.4. Если вы менеджер…

Если вы менеджер, то в вашем отделе есть как минимум несколько человек, которым стоит прочитать эту книгу. Некоторые из них готовы сделать всё, о чём вы их попросите; другие довольно самоуверенны, и им трудно будет поверить, что они могут узнать из неё что-то новое. Вы можете положить экземпляр книги на их рабочее место, но как убедить таких людей прочитать её?

Предположим, им неинтересно, что читает руководитель – и вам нужно заинтересовать их. Загрузите постер с правилами отладки с сайта (см. ссылку на предыдущей странице), распечатайте его и повесьте себе на стену (в любом случае, он круче, чем все эти «мотивирующие» плакаты о командной работе). Попросите их прочитать книгу и высказать своё мнение – притворитесь, что не знаете, что описанные в ней правила действительно работают. После прочтения они либо станут фанатами правил, либо придумают, как их можно улучшить; в любом случае, они обдумают их и пропустят через себя – чего бы не сделали при попытке навязать им прочтение книги (и если они действительно придумают, как улучшить правила – то, пожалуйста, пришлите их идеи на мой сайт).

Вы можете воззвать к их чувству командного духа. Некоторые из людей, рецензировавших черновик этой книги, были руководителями отделов и считали себя специалистами по отладке, но после прочтения они обнаружили, что в правилах чётко сформулирована интуитивно используемая ими последовательность действий – и это помогает при обучении сотрудников. Им было легко подталкивать своих инженеров в нужном направлении с помощью фразы «Не предполагайте, а смотрите» – как я сам уже делаю на протяжении 20 лет.

Слушайте, это короткая и забавная книга. Дайте подчинённым по экземпляру, поместите их на полдня в комнату без телефона и электронной почты – по крайней мере, они приятно проведут время (но чтобы убедиться, что они не потратили его, играя на своих Palm'ax – проведите для них тестирование, когда они закончат. И обязательно спрячьте на это время постер).

И, наконец, помните о том, что как только ваши сотрудники усвоят правила – они будут использовать их в процессе выполнения следующих задач, связанных с отладкой, которые вы им поручите. Не давите на них – иначе они будут пытаться по-быстрому угадать причину ошибки. Дайте им время, чтобы «изучить систему», «воспроизвести ошибку», «не предполагать, а смотреть» и т. д. Будьте терпеливы и доверьтесь правилам – обычно их использование является самым быстрым способом отладки и убережёт вас от бесконечных (и бесплодных) игр в «угадайку», которых вам отчаянно хочется избежать.

15.5. Если вы преподаватель…

Если вы преподаватель технического ВУЗа, то, вероятно, понимаете, что реальный опыт из «баек ветеранов» неоценим для студентов, выросших в стерильном мире школьных проектов. Вы также знаете, что вашим ученикам часто придётся выполнять тяжелую работу, связанную с отладкой – начинающим техническим специалистам и программистам регулярно в довесок к их основной деятельности поручают задачи, связанные с исправлением чужих ошибок. Если они хорошо будут с этим справляться – это повлияет на их карьеру и создаст им репутацию инженеров, умеющих добиваться поставленных целей. Так что заставьте студентов прочитать эту книгу – дайте им такое домашнее задание и закупите экземпляры для школьного магазина. Вероятно, вам не понадобится создавать для этих целей отдельный курс с тремя зачётами, но обязательно используйте её в учебной программе – чем раньше, тем лучше.

15.6. Памятка

Правила этой книги называются «золотыми», потому что они:

- **Универсальны**. Вы можете применить их для отладки любой проблемы в любой системе;
- **Фундаментальны**. Они описывают нужную последовательность действий с рекомендациями по выбору конкретных инструментов и методов отладки, доступных для вашей системы;
- **Необходимы**. Вы не сможете эффективно отладить ошибку, если не будете следовать *всем* перечисленным правилам;
- **Легко запоминаемы**. И я снова напомню их вам:

Правила отладки

- Изучите свою систему;
- Воспроизведите ошибку;
- Не предполагайте, а смотрите;
- Разделяйте и властвуйте;
- Вносите по одному изменению за раз;
- Записывайте всё, что происходит;
- Проверьте кабель;
- Воспользуйтесь чьим-то свежим взглядом;
- Если вы не исправили ошибку – она не исчезла.

Будьте инженером Б и соблюдайте правила. Устраните баги и станьте героем. Вернитесь домой пораньше и хорошенько выспитесь. Или просто потратьте больше времени на отдых и развлечения. Вы это заслужили.