Тест Лимончелли: 32 краеугольных принципа хорошей команды системного администрирования

http://ivanpesin.info/translations/other/32LimoncelliTest.html

The Limoncelli Test – 32 bedrock issues of a well-functioning sysadmin teams

URL: http://everythingsysadmin.com/the-test.html

Автор: Том Лимончелли (Tom Limoncelli)

Перевод: Иван Песин (Ivan Pesin)

Sun Nov 6 10:26:30 EST 2011

Меня часто спрашивают, как улучшить работу системных администраторов. Требуется немного времени, чтобы обнаружить фундаментальные пробелы, исправив которые можно улучшить продуктивность и качество работы системных администраторов.

Такие пробелы не просто создают много проблем, они создают много категорий проблем.

Например: использование системы отслеживания запросов (request tracking system или «ticket system») — фундаментальная техника. Она принесет вашей команде много очевидной, и не совсем, пользы. Без неё вы рискуете столкнуться с разными категориями проблем: проблем в результате забытых запросов; проблем из-за того, что пользователи отрывают вас от работы по любому поводу; проблем из-за незнания руководством, чем занимается ваша команда; проблем из-за невозможности обнаружить тенденции; проблем команды, которая не может эффективно передавать задачи.

Исправление фундаментальных пробелов выглядит трудным делом, но если их оставить неисправленными, то это выльется для вас в ещё большее количество работы.

Джоэль Спольски написал блестящий «Тест Джоэла: 12 шагов к лучшему коду”, «…совершенно безответственный и несерьёзный тест для определения качества команды разработчиков», состоящий из 12 вопросов. Я придумал собственный тест для системных администраторов. Он состоит из 32 односложных вопросов. И он такой же безответственный и несерьёзный.

Цель этого теста — упростить общую оценку команды СА. Он будет полезен и руководителю команды, и ведущим специалистам, и рядовым членам команды. Тест также может служить способом оценки потенциального места работы: если вы не хотите присоединиться к кораблю дураков, вы должны выяснить у вашего возможного работодателя, какие из практик они используют, а какие — нет. И не так важно, сколько они набирают очков, как отношение: нежелание или невозможность изменений сигнализирует опасность.

Очевидная проблема этого теста в том, что он в 3.2 раза длиннее теста Джоэла. Он достаточно гениален, чтобы свести свой к 10 вопросам. В свое оправдание, хочу сказать, что изначально в

моем тесте было 40 вопросов.

Разделы, помеченные звёздочкой — самые фундаментальные. Соответствующие практики в моей книге обозначены, как абсолютно необходимые. Остальные тоже важны, но могут быть нерелевантны для маленьких фирм.

Сколько очков у вас?

Тест Лимончелли: 32 вопроса к вашей команде сисадминов

Обновлено: 2011-07-25

A. Подходы по работе с пользователями:

*1. Используете ли вы систему отслеживания запросов?

*2. Есть ли у вас “три регламентирующие политики” и знают ли о них?

3. Ведёт ли команда ежемесячные метрики?

B. Современные подходы в работе:

*4. Есть ли у вас вики с описанием политик и процедур?

5. Есть ли у вас место, где хранятся пароли?

6. Хранится ли ваш код в системе управления версиями?

7. Использует ли ваша команда систему отслеживания ошибок для своего кода?

8. В ваших задачах обеспечение стабильности приоритетнее новых возможностей?

9. Пишет ли ваша команда проектную документацию?

10. Выполняете ли вы анализ причин происшествия?

C. Операционные подходы:

*11. Есть ли у вас документация (OpsDoc) на все ваши сервисы?

*12. Имеет ли каждый сервис необходимый мониторинг?

13. Есть ли у вас график дежурств?

14. Используете ли вы отдельные системы для разработки, тестирования и производства?

15. Используется ли “канареечный процесс” при внесении изменений на большое количество машин?

D. Подходы по автоматизации:

16. Используете ли вы инструменты управления конфигурацией, такие как cfengine/puppet/chef?

17. Выполняются ли автоматизированные задачи под ролевыми учетными записями?

18. Ваши автоматизированные задачи шлют почту только когда есть полезная информация?

E. Подходы по управлению парком машин:

*19. Есть ли у вас база данных всех машин?

20. Автоматизирован ли у вас процесс установки ОС?

*21. Можете ли вы автоматически обновлять ПО для всего парка машин?

22. Есть ли у вас политика обновления компьютеров?

F. Подходы серии “Мы согласны, что техника ломается”:

*23. Будут ли работать ваши сервера, если на них откажет один из дисков?

24. Используется ли схема N+1 в ядре вашей сети?

*25. Автоматизировано ли ваша процедура резервного копирования?

*26. Проверяете ли вы периодически свой план аварийного восстановления?

27. Есть ли в вашем ЦОД системы удаленного управления питанием и доступа к консолям?

G. Подходы к безопасности:

*28. Ваши сервера/ноутбуки/компьютеры защищены автоматически обновляемым антивирусным ПО?

*29. Есть ли у вас политика безопасности в письменном виде?

30. Проводите ли вы регулярный аудит безопасности?

31. Можете ли вы отключить учетную запись пользователя во всех системах за 1 час?

32. Можете ли вы сменить все привилегированные (root) пароли за 1 час?

A. Подходы по работе с пользователями:

*1. Используете ли вы систему отслеживания запросов?

Это настолько фундаментальная практика, что меня гнетет необходимость пояснений.

Люди не обладают такой хорошей памятью, как компьютеры. Ожидать от сисадминов, что они будут помнить все запросы пользователей, это прямой путь к тому, что запросы будут забываться.

Хранение запросов в базе упрощает обмен информацией в команде и позволяет избегать ситуаций, когда два человека одновременно работают над одним и тем же запросом, дает возможность делить работу межу людьми, передавать работу между собой, не теряя информации, и подменять администратора, который недоступен, вышел или в отпуске.

Эта практика делает работу команды прозрачнее. Пользователи видят кто работает над запросом, какой у него статус, и когда он выполнен. Это позволяет пользователю и администратору задавать дополнительные вопросы и следить за ответами.

Использование системы учета запросов позволяет эффективнее использовать рабочее время. Каждый раз, когда сисадмина отрывают от того, что он делает, его отбрасывает на 7 минут назад в задаче, от которой его оторвали. Система учета запросов уменьшает количество прерываний от пользователей, которые хотят сделать новый запрос или узнать статус открытого запроса. Она также позволяет задавать приоритеты в работе администратора, а не решать проблемы пользователя, который жалуется громче всех.

Подход с учетом запросов дает возможность менеджерам стать лучше. Затормозившиеся запросы сразу становятся видны и менеджер имеет возможность вмешаться. Вскрываются тенденции и частые запросы, что позволяет оптимизировать работу с помощью автоматизации или улучшения процесса. Микроменеджеры могут получить информацию из базы запросов, не отрывая администраторов от работы. Жалобы пользователей приобретают обоснованность: если запрос решается слишком долго, менеджер может надлежащим образом решить вопрос; если пользователю только кажется, что запрос долго решается, то у вас будут доказательства противного. Если пользователь заявит, что “эта проблема тянется месяцы, но я только вчера прислал запрос”, то у менеджера будет возможность… пояснить, что путешествие во времени, телепатия и другие сверхъестественные возможности в природе не существуют.

Система учета позволяет обнаружить системные проблемы. Однажды у меня был шеф, который, выполнив запрос в системе учета, обнаружил, что 3 из наших 1000 пользователей создали 10% всех запросов. Проведя небольшое расследование, мы смогли решить некоторые фундаментальные проблемы, которые были причиной этих запросов.

Наконец, такая система позволяет пользователям помогать самим себе. Часто бывает, что когда пользователь формулирует проблему, он сам находит ее решение и запрос так и не отправляется. Если же этого не происходит, то сам процесс помогает пользователю продумать проблему и сформулировать ее яснее, что ускоряет решение проблемы.

Я провел 90-е года в роли радикала, убеждающего людей использовать системы отслеживания запросов. В 2000-х, я с удовольствием наблюдал, как это становится общепринятой практикой. Наступила третья декада и если у вас до сих пор нет системы отслеживания запросов — позор вам.

Подробнее:

```
  TM: p. 26, Chapter 2: Focus Versus Interruptions / Delegate, Record, Do
  P2: p. 28, Chapter 2: Climb Out of the Hole / 2.1.1 Use a Trouble-Ticket
System
  P2: p. 354, Chapter 13: Helpdesks / 13.1.10 Supply Request-Tracking
Software
```

*2. Есть ли у вас “три регламентирующие политики” и знают ли о них?

Если вы хотите, чтобы работа делалась, вам нужно определить три политики, регламентирующих вашу работу с пользователями. Они направлены как на обслуживание клиентов, так и на обеспечение эффективной работы команды сисадминов. Если вы, как менеджер команды сисадминов, считаете, что она ничего не успевает, возможно, это ваша вина, если вы не определили или не выполняете эти три политики:

```
Доступные пользователям способы запроса помощи.
Определение "аварийной ситуации".
```

Объем обслуживания: кто, что и где.

Эти политики помещаются на одной страничке и она должна быть доступна на веб-сайте вашей команды или развешана на стенах, чтобы их знали. Руководство должно поддерживать эти политики; это значит, что менеджмент готов сказать “нет” людям, когда они просят сделать исключение. Процесс получения исключения должен быть похож на пробивание стены, а не на проезд “спящего полицейского”.

Как получить помощь:

Официальный протокол, описывающий как пользователи должны запрашивать помощь, позволяет получить все преимущества использования системы отслеживания запросов, описанные в предыдущем разделе. Без этой политики все преимущества испаряются, потому что люди будут приходить прямо к сисадминам, которые, стараясь помочь, будут постоянно отвлекаться на мелочи, а их эффективность сильно понизится.

У системного администратора должны быть возможность отказать пользователю, когда тот не соблюдает протокол. Без возможности отослать пользователя к официальной политике, администраторы будут работать либо над низкоприоритетными задачами, либо для пользователей, которые громче всех жалуются, либо каждый из них будет пользоваться своей личной политикой, тем самым выставляя всю команду непоследовательной, а в худшем случае будут нездоровым способом демонстрировать свое неудовлетворение. Точнее, нездоровым для пользователей способом.

Определение аварии:

Официальное определение аварийной ситуации позволяет системным администраторам расставлять приоритеты. Без этого определения, всё становится авралом, и снова сисадмины будут постоянно отвлекаться на мелочи, а их эффективность сильно понизится.

Политика — это один из способов, которым руководство коммуницирует приоритеты системным администраторам. В противном случае, сисадмины будут строить догадки о приоритетах, ошибаться, их будут несправедливо за это наказывать; менеджеры будут расстраиваться из-за “дисконнектов”, а пользователи, видя непоследовательность, будут предполагать фаворитизм, халатность и некомпетентность.

Это политика определяет ожидания пользователей. С ее помощью можно корректировать иллюзии пользователей, которые считают любой происшествие аварийной ситуацией.

Каждая организация должна иметь определение аварии или “красного сигнала”. Для газеты — это все, что блокирует печать завтрашнего номера и его погрузку в грузовики в 4 утра. “Красный сигнал” на заводе — это все, что останавливает конвейер. Авария для отдела платежей — все, что делает невозможным проведение платежей. Команды, работающие в сфере образования, знают, что перенести занятие совсем не просто, потому аварийной ситуацией является все, что мешает надлежащему проведению лекции (возможно, только если технологический центр был заблаговременно предупрежден). В университетах “красный сигнал” определен, как любая преграда для своевременной подачи заявок на получение грантов.

“Желтый сигнал” — это все, что приведет к “красному”, если останется без внимания. Например, оплаты проводятся, но не работает подсистема оценки доступных запасов. В таком случае очень рискованно брать новых клиентов. Последняя оценка говорила о запасах,

достаточных на 2 недели. Риск остановки производства растет с каждым днем, пока “желтый сигнал” не будет исправлен.

Все остальное — это рутина. Изобретательные компании разделяют рутинные задачи по приоритетам: высокий, средний, низкий; создание нового сервиса, поддержка существующих, и т.д. Если у вас ничего такого нету, начните с определения, что представляет собой авария.

Что поддерживается:

Официальное определение поддерживаемых сервисов позволяет администраторам говорить “нет”. Должно быть определено когда, где, кто и что поддерживается. Работает ли команда после 6 вечера? На выходных? Ходят ли администраторы по вызову к рабочему месту пользователя? Домой к пользователю? Поддерживаете ли вы кого-либо, кроме сотрудников компании? Какое программное и аппаратное обеспечение поддерживается? Есть ли у поддерживаемого обеспечения жизненный цикл, или вы обречены его поддерживать вечно? Новые технологии включаются в поддержку автоматически или после официального утверждения?

Не имея возможности сказать “нет”, администраторы будут поддерживать всё. Отзывчивый и старательный сисадмин потратит кучу времени, пытаясь заставить работать неподдерживаемую видеокарту, при том, что дешевле будет подарить этому пользователю поддерживаемую карту за счет бюджета СА. Пропавший без вести системный администратор вдруг найдется, после дня, проведенного у пользователя дома, где он ремонтировал Интернет-подключение. Или наоборот, ворчливый сисадмин будет говорить людям, что они это не поддерживают, просто потому, что он занят.

Подробнее:

```
  TM: p. 21, Chapter 2: Focus Versus Interruptions / Directing Interruptions
Away from You
  P2: p. 27, Chapter 2: Climb Out of the Hole
  P2: p. 820, Chapter 33: A Guide for Technical Managers / 33.1.1.1
Priorities and Resources
```

3. Ведёт ли команда ежемесячные метрики?

Когда вы принимаете решения или убеждаете в чём-то высшее руководство, вы должны руководствоваться данными.

Наилучший способ разработки метрик — это создание одной метрики на каждое предложение или пункт устава.

Пример: устав команды развёртывания рабочих мест может звучать как «Обеспечение высококачественным, стандартизированным компьютером каждого сотрудника с первого дня его работы, обновление компьютера на основе 3-х летнего цикла, по оптимальной эксплуатационной стоимости. Тогда метриками могут быть: количество недель с последнего обновления стандартной конфигурации, цена текущей стандартной конфигурации, количество новых сотрудников в этом месяце, капитальные расходы, операционные расходы. Сколько дней новые сотрудники ждали своего компьютера (группировка, когда был установлен компьютер: до прихода, в первый день, на второй день, на 3-й, 4-й, 5-й и больше, чем через 5 дней). Возраст парка (группировка по возрасту: меньше года, от 1 до 2 лет, от 2 до 3 лет, больше 3 лет). Число использующихся машин с нестандартной конфигурацией.

Если у вас нет устава, поговорите со своим менеджером про то, чтобы его написать. В качестве альтернативы, вот список простых «начальных метрик», которые вы можете начать вести немедленно:

```
Сколько у нас системных администраторов?
Сколько у нас пользователей, которым мы предоставляем сервис?
Сколько компьютеров мы поддерживаем?
Сколько у нас совокупного дискового пространства? ОЗУ? Ядер ЦПУ?
Сколько у нас “открытых” тикетов в данный момент?
Сколько новых тикетов было создано за последний месяц?
Кто (или какой отдел) создал большинство тикетов в этом месяце?
Среднее число тикетов на сисадмина за последний месяц?
Выберите 4-5 важных SLA и отметьте насколько точно вы их выполняете.
Статистика использования полосы Internet-подключения за последний месяц.
```

Записывайте эти метрики первого числа каждого месяца. Внесите их в электронную таблицу. Теперь вы сможете использовать эти данные во время бюджетирования или для презентаций, когда нужно объяснить, чем занимается ваша команда.

Вот и всё. Действительно. Учёт этих метрик есть разница между презентацией, начинающейся с графика роста количества машин в вашей сети, и фразой «Привет, меня зовут Джо и мы поддерживаем… э-э-м-м… кучу машин». Во время бюджетирования, способность ответить на эти базовые вопросы, является основой для других вопросов, таких как «Какова средняя стоимость тикета?» (сумма бюджета последнего года / общее количество тикетов за последний год). Если у нас будет 100 пользователей, сколько дискового пространства нам понадобится? (общее дисковой пространство / количество пользователей * 100).

Сбор данных для метрик, в конечном счёте, должен быть автоматизирован. До того, настройте себе напоминание на почту, что это нужно сделать вручную.

Подробнее:

```
  P2: p. 523, Chapter 22: Service Monitoring
  P2: p. 119, Chapter 5: Services / 5.1.13 Monitoring
  P2: p. 765, Chapter 31: Perception and Visibility / The System Status Web
Page
  P2: p. 849, Chapter 33: A Guide for Technical Managers / Sell Your
Department to Senior Management
```

B. Современные подходы в работе:

*4. Есть ли у вас вики с описанием политик и процедур?

Вашей команде нужна вики. В ней можно написать все ваши политики (что должно выполняться) и процедуры (как это должно выполняться).

Автоматизация — это прекрасно, но перед тем, как автоматизировать что-то, вы должны мочь это выполнить вручную. Предусловием автоматизации является документирование ручного процесса. До того, документация дает возможность всей команде выполнять действия однотипно, а вам — делегировать задачи. Если процедура описана, ее может выполнить кто-то еще.

В оглавлении вики должны быть вынесены общие, рутинные задачи. Хорошо начать с процедур по добавлению/изменению/удалению, которые должны уметь делать все в команде, и задач, которые вы не любите выполнять и делегировали бы помощнику, если бы он был.

Список процедур:

```
Приходит новый сотрудник.
Сотрудник уходит из компании.
Сотрудника увольняют.
Установка нового компьютера.
Компьютер выводится из эксплуатации.
Как дать пользователю доступ через VPN.
Как поменять диск в RAID-массиве.
Как поменять root-пароль на всех машинах
```

Здесь включены три категории задач: вещи, в которых вам важно однообразие выполнения; вещи, которые вы редко делаете и не хотите каждый раз тратить время на то, чтобы вспоминать, как сделать правильно; и вещи, которые выполняются в стрессовых ситуациях, чтобы судорожно не думать над тем, что вы делаете.

Все эти три типа задач могут быть описаны простыми пошаговыми списками.

После того, как процедура описана, ее может выполнить каждый. Кроме того, такая документация, по сути, является программой обучения новых сотрудников. А еще ее можно использовать при составлении должностной инструкции для своего помощника, которого вы хотите чтобы компания наняла делать всю вашу работу.

Даже если вы не работаете в команде или есть задачи, которые делаете только вы, документирование имеет преимущества. Вам нужно будет меньше думать при выполнении задачи. Как правдива поговорка “мы автоматизируем, потому что ленивые”, так же правдива и поговорка “мы документируем, потому что нетерпеливые”.

Многие сисадмины не любят писать документацию, но составление пошагового списка не такая трудная задача. Важно держать документацию на вики, чтобы все могли ее исправлять и улучшать.

Для любой из задач можно иметь отдельные документы, описывающие политику и процедуру. Политику определяет руководство: “все новые пользователи должны получать беспроводную мышь”. Процедура определяет как это выполнять: “беспроводные мыши хранятся в 3-м ящике; зарядить и проверить так-то, и т.д.” Политика меняется только с разрешения руководства, процедура меняется техническим персоналом с уведомлением автора или других ответственных лиц.

Подробнее:

```
P2: p. 241, Chapter 9: Documentation
TM: p. 145, Chapter 12: Documentation
```

5. Есть ли у вас место, где хранятся пароли?

Это демонстрирует, что у вас есть зрелый способ управления паролями.

Существует много отличных программных систем хранения паролей. Однако часто достаточно просто держать конверт в обычном закрытом ящике.

Часто возникает проблема проверки паролей. Вы уверены, что злонамеренный сотрудник не вносит неправильные пароли? Если у вас есть основания для таких сомнений, нужно, чтобы кто-то третий проверял все новые пароли.

Подробнее:

```
P2: p. 271, Chapter 11: Security Policy
```

6. Хранится ли ваш код в системе управления версиями?

С появлениям возможности установки новой системы при помощи API-вызова, мы теперь все программисты.

-Лимончелли о облачных вычислениях, devops, и важности навыков программиста в системном администрировании.

Мы теперь все программисты. Программисты используют системы управления версиями.

Что хранить в репозитории: ваши скрипты, программы, конфигурационные файлы, документацию — практически все что угодно. Если вы не уверены, стоит ли что-то держать в репозитории, ответ должен быть “стоит”.

Хранение конфигурационных файлов в системе управления версиями сначала выглядит как роскошь, но позже становиться спасительным средством.

Любая система лучше, чем ничего. Используйте то, чем пользуются ваши разработчики. Нет разработчиков? Выучите Git, Mercurial или даже Subversion. Срочно нужен быстрый способ хранить историю изменения конфигурационных файлов? http://www.nightcoder.com/code/xed (это враппер, который вызывает $EDITOR).

Подробнее:

```
TM: p. 174, Chapter 13: Automation / xed
```

7. Использует ли ваша команда систему отслеживания ошибок для своего кода?

Системы отслеживания ошибок (bug-tracking system) отличаются от систем отслеживания запросов. Если у вас ошибки в коде появляются редко (например, ваша команда не пишет много кода), то достаточно просто открыть запрос для себя в системе отслеживания запросов.

Но если ваша команда пишет серьезный код, запустите отдельную систему отслеживания ошибок. Эти системы используют другой процесс, нежели системы отслеживания запросов. Запросы — это инструмент общения между вами и пользователями. Системы отслеживания ошибок отслеживают жизненный цикл ошибки (сообщение, проверка, назначение, исправление, закрытие, проверка).

8. В ваших задачах обеспечение стабильности приоритетнее новых возможностей?

Добавление функциональности всегда интереснее, чем исправление ошибок. К сожалению, мы не можем делать только то, что интересно.

Зрелые команды приоритезируют свои задачи следующим образом:

```
безопасность (наивысший приоритет)
стабильность
ошибки
производительность
новая функциональность (низший приоритет)
```

Перед тем, как добавлять функциональность, вы должны добиться стабильности. Вопросы безопасности — это вопросы стабильности с наивысшим приоритетом.

Один из принципов, пропагандируемый Марком Бургессом (Mark Burgess), это «стремитесь к стабильности, а не к новой функциональности”. Среди изменений, которые мы делаем, одни — улучшают стабильность, другие — добавляют новые возможности. Последовательность должна быть такой: возможность, стабильность, возможность, стабильность, возможность, стабильность. Но не: возможность, возможность, возможность, ОЙ-ОЙ-ОЙ!, стабильность, стабильность, стабильность. Добейтесь стабильности перед тем, как внедрять очередную замечательную возможность.

Это понимают врачи. В реанимации сначала стабилизируют состояние пациента. Пациенту не лечат грипп, когда у него сильное кровотечение, от которого можно умереть

Приоритет проблем с производительностью варьируется. В некоторых местах производительность эквивалентна стабильности.

Подробнее:

```
  P2: p. 343, Chapter 33: A Guide for Technical Managers / Priorities
  TM: p. 101, Chapter 8: Prioritization / Prioritization
```

9. Пишет ли ваша команда проектную документацию?

Хорошие команды сисадминов сначала думают, потом делают. В больших командах важно коммуницировать, что вы собираетесь сделать или уже сделали.

Стандартным способом предложения новых систем или описанием существующих является проектная документация. Она может быть очень короткой — одна-две страницы, а может, при необходимости — очень детальной и исчерпывающей.

Создайте шаблон и используйте его везде. Он может содержать такие разделы: обзор, цели, ложные цели, предпосылки, решение, рассмотренные альтернативы, безопасность, аварийное восстановление, стоимость.

Этот формат можно использовать для написания 20-страничного плана реструктуризации вашей сети, когда вам нужно участие в этом многих людей. Его же можно использовать для 5-страничного документа, описывающего прототип, чтобы все могли видеть результат вашей работы и дать ему оценку, перед тем как внедрять его в эксплуатацию. И снова, этот формат можно взять для полстраничной заметки про новую структуру каталогов на файловом сервере (в этом случае, вам, вероятно, не понадобятся большинство разделов). Используйте его для документирования системы, которая уже запущена в работу, чтобы ваши коллеги могли использовать это описание, как справочник. Да что там, документируйте в этом формате даже ваши планы на пикник.

Идея в том, чтобы у вашей команды был механизм для размышления перед реализацией, способ обсудить планы, кроме общения в коридорах и митингрумах, система, которая позволяет сохранить информацию для других, кто захочет разобраться почему и как что-то было сделано.

Предложенный формат работает как для получения критических отзывов, так и для сообщения “предупредите меня, если то, что я собираюсь сделать, конфликтует с тем что делаете вы”.

Ваш формат проектной документации может содержать больше или меньше разделов, некоторые могут быть опциональными, другие — обязательными. Вы должны быть гибкими: если нужно полстраницы описания, не раздувайте текст так, чтобы в каждом разделе что-то было. Наличие стандартного шаблона упрощает процесс написания документа и позволяет избежать “синдрома пустой страницы”.

Подробнее:

```
P2: p. 241, Chapter 9: Documentation
```

10. Выполняете ли вы анализ причин происшествия?

Регистрируете ли вы сбои с описанием того, что случилось, чтобы из них можно было что-то вынести, или вы просто надеетесь, что никто ничего не заметит, а проблема не повторится?

Хорошие анализ причин происшествия включает описание в разрезе времени, пострадавших, как была решена проблема, как происшествие повлияло на бизнес и список предложенных решений, для предотвращения повторного возникновения ситуации. Каждое предложение должно быть оформлено в виде тикета или отчета об ошибке, чтобы можно было отследить выполненные шаги.

Анализ причин последовательно формирует более стабильную среду. После каждого сбоя нужно придумывать хотя бы один превентивный ход. Может ли ваша система мониторинга определять проблему так, чтобы вы знали о ней до ваших пользователей? Можете ли вы обнаруживать предпосылки к возникновению проблемы? Во многих системах есть способ их тестировать на новых конфигурациях до внедрения (например, скрипты, выполняемые до добавления кода в репозиторий — pre-submit scripts — в системах управления кодом). Есть ли какие-либо тесты, которые вы могли бы добавить, чтобы обнаружить опечатку, ведущую к простою?

Цель проведения такого анализа не в нахождении виновных и их осуждении. В хорошей среде сисадминов нет ничего зазорного в том, чтобы указать свое имя в разделе “что мы сделали не так”. Этим вы выполняете роль лидера, просвещая людей для того, чтобы они не повторили ту же ошибку.

Если ваш менеджмент использует результаты анализа причин происшествия для того, чтобы наказать виновных, они не понимают, что цель эксплуатации не в том, чтобы работать идеально, а в том, чтобы работать лучше с каждым днем. Любой менеджер, увольняющий человека из-за того, что его действия привели к неумышленному простою, разваливает компанию.

Результаты анализа должны быть доступны всем. Вы можете смущаться и думать, что вы “выносите сор из избы”, но если это делать последовательно, ваши пользователи будут вас уважать больше. Прозрачность рождает доверие.

Ну и конечно, чтобы действительно воспитать уверенность, вы должны работать над всем этими проблемами и тикетами, которые появились в результате проблем.

Подробнее:

```
P2: p. 492, Chapter 20: Maintenance Windows / 20.1.13 Postmortem
```

C. Операционные подходы:

*11. Есть ли у вас документация (OpsDoc) на все ваши сервисы?

Допустим у вас умирает DNS-сервер. Вы его переустанавливаете, ведь вы знаете, как это сделать. Круто, так? Вам нужно скомпилировать и установить последнюю версию BIND, и вы тоже знаете, как это сделать, так? А когда система мониторинга рапортует, что периодически возникает ошибка, вы знаете, как ее исправить, так ведь? Вы знаете, как все это делать. Зачем все это записывать?

Вот зачем:

Будете ли вы помнить все эти мелочи через 6 месяцев? Мне приходится переизобретать процесс с нуля, если я его не выполнял несколько месяцев (а иногда даже несколько дней!). И не только переизобретать, но и повторять старые ошибки и снова делать выводы из них. Время, потраченное в никуда!

А будете ли вы помнить, как все это сделать в стрессовой ситуации? Когда происходит внештатная ситуация, моя память работает хуже.

А что, если вас не будет на месте? Как можно расслабиться в отпуске, когда может случиться, что, кроме вас, никто не сможет справиться с проблемой? Нельзя жаловаться на перегрузку из-за невозможности передать свою работу другим, если вы не сделали это возможным.

Как на счет других людей в команде? Они должны учится, наблюдая, как вы делаете, или они могут учиться самостоятельно? Если они будут учиться самостоятельно, обращаясь за помощью, когда они зашли в глухой угол, это сэкономит вам время, а кроме того, вы не будете выглядеть скрягой, который не хочет делиться информацией — это же не ваша цель? Фактически, если у команды есть такого рода инструкции, то новички будут чувствовать себя комфортнее и быстрее включатся в работу.

Как может ваш менеджер повысить вас или перевести на новый и интересный проект, если вы единственный, кто имеет знания по текущему?

Каждый сервис должен иметь документацию по определенным вещам. Если документация по каждому сервису организована одинаково, люди к этому привыкают и могут быстрее найти нужную им информацию. Я делаю раздел на вики (или мини-сайт, или сайт на Google Sites) для каждого сервиса:

Каждый содержит 7 одних и тех же вкладок (некоторые могут быть пустыми):

1. Обзор: Обзор сервиса — что это такое, зачем оно нам нужно, контактная особа, как сообщить об ошибках, ссылки на документ с дизайном сервиса и другую соответствующую информацию.

2. Сборка: Как собрать программное обеспечение, которое предоставляет сервис. Откуда его

загружать, где находится репозиторий исходного кода, шаги по сборке пакета для дистрибутива. Если вы каким-либо образом модифицируете это ПО (проект с открытым исходным кодом в котором вы участвуете или внутренний проект), добавьте вводную инструкцию для новых разработчиков. В идеале это должен быть установочный пакет, который нужно просто скопировать и установить на машине нового разработчика.

3. Развертывание: Как развернуть программное обеспечение. Как установить сервер с нуля: требования к памяти/диску, версия и конфигурация ОС, какие пакеты нужны, и т. д. Если это автоматизировано с помощью системы управления конфигурацией, такой как cfengine/puppet/chef (как и должно быть), так и укажите.

4. Общие задачи: пошаговые инструкции для таких общих задач, как инициализация (добавление/изменение/удаление), общие проблемы и их решения, и так далее.

5. План нотификаций (Pager Playbook): список всех нотификаций, которые может сгенерировать ваша система мониторинга для этого сервиса и пошаговые инструкции серии “что делать, когда” для каждой их них.

6. DR: План аварийного восстановления (Disaster Recovery). Если сервер с этим сервисом выходит из строя, как перенести сервис на резервный сервер.

7. SLA: Соглашение об уровне обслуживания (Service Level Agreement). Это (социальный или настоящий) договор между вами и вашими пользователями. Обычно, это вещи, вроде целевого аптайма (сколько девяток), RPO (Recovery Point Objective, целевая точка восстановления) и RTO (Recovery Time Objective, целевое время восстановления).

Если сервис разрабатывается вами, восьмая вкладка должна включать информацию для команды: как настроить среду разработки, как выполнять интеграционное тестирование, как готовить релизы и другие советы разработчикам. Например, один из проектов, которыми я занимаюсь, содержит четкий список шагов для добавления в систему нового удаленного вызова процедуры (RPC).

Будьте героем и создайте шаблон для вашей команды. Опишите, для начала, какой-нибудь основной сервис, например DNS. Потом, опишите сервис побольше. Создайте костяк, который другие будут использовать, как основу, и вписывать недостающие части. Возьмите в привычку начинать с сервисной документации каждый новый проект.

Подробнее:

```
P2: p. 241, Chapter 9: Documentation
```

*12. Имеет ли каждый сервис необходимый мониторинг?

Сервис, который не мониторится — это не сервис. Без мониторинга вы просто запустили программу.

-Лимончелли

Мониторинг должен быть основан на SLA, указанном в документации на сервис (OpsDoc). Если у вас нет SLA, то как минимум нужна простая нотификация про доступность/недоступность сервиса.

Не забудьте обновить план нотификаций (Pager Playbook).

Подробнее:

```
  P2: p. 523, Chapter 22: Service Monitoring
  P2: p. 765, Chapter 31: Perception and Visibility / The System Status Web
Page
  P2: p. 119, Chapter 5: Services / 5.1.13 Monitoring
```

13. Есть ли у вас график дежурств?

Есть ли у вас график дежурств или вы простофиля на вечном дежурстве?

График определяет когда и кто “носит пейджер” (или кто реагирует на внештатные ситуации).

Можно, в буквальном смысле, периодически “передавать пейджер” от одного человека к другому. Можно, чтобы у каждого был свой собственный пейджер, а ваша система мониторинга будет сама определять из расписания, кому слать сообщения. Лучше всего, когда есть общий почтовый адрес, который пересылает почту дежурному человеку, скрывая график дежурств от пользователей.

График дежурств может быть как простым, так и сложным. Когда внештатных ситуаций немного, имеет смысл использовать график одна неделя из n (для команды из n человек). Для более сложных ситуаций, имеет смысл разбивать день на три 8-часовые смены. Подход “вслед за солнцем” состоит в организации 8-часовых смен таким образом, чтобы дежурила та часть глобальной команды, у которой сейчас день. Можно дежурить неделю по 8 часов каждые n недель, если у вас в команде 3n человек. Вариации бесконечны.

Расписание приносит пользу многим людям: вам, вашим пользователям, руководству, и отделу кадров.

Оно приносит пользу вам, потому что позволяет строить планы вне работы. Я считаю, что правильный баланс между работой и личной жизнью имеет важнейшее значение. Если ваш баланс нарушен и нет графика дежурств, то это первое, что нужно исправлять.

График дежурств улучшает уровень обслуживания пользователей, потому что, вместо “панических попыток найти сисадмина”, у них есть простой и надежный способ сделать это.

Он приносит пользу руководству, потому что дает уверенность, что когда бы не случилась внештатная ситуация, кто-то всегда будет готов ею заняться.

Этот же график будет полезен и отделу кадров при расчете заработной платы и надбавок. Скрипт может генерировать отчет, основываясь на электронной версии графика, и отправлять его в отдел кадров.

Если вы думаете, что у вас нету графика, то этот график называется «24x7x365», а вы простофиля (и это не означает, что вы можете быть согласны с этим).

14. Используете ли вы отдельные системы для разработки, тестирования и производства?

Программисты ведут разработку на своих серверах, предназначенных для разработки. Когда они решают, что готовы, система собирается и устанавливается на тестовые сервера. Если отдел тестирования (QA, quality assurance) или отдел приема работы (UAT, User Acceptance Testing) подтверждает качество и приемлемость результата, та же сборка устанавливается на производственных серверах.

Это из базового курса по системному администрированию, правильно?

Почему же тогда, я постоянно встречаю администраторов, чье руководство не разрешает им организовать такую схему? Если ваше руководство говорит, что “иметь второй сервер — это слишком дорого”, они безнадежны. QA — не дорогой. Знаете, что дорого? Время простоя.

Эксперименты на производственной системе не просто опасны, они запрещены в среде, которая подчиняется SOX (Sarbanes-Oxley Act, закон США, регулирующий в т.ч. область управления и раскрытия информации - прим. пер.). Таким образом, использование производственных серверов для разработки попросту запрещено!

Тестовая система не обязательно должна быть такой же дорогой, как производственная. Она не должна быть такой же мощной, может иметь меньше оперативной памяти и более простые процессора. Это, вообще, может быть виртуальная машина, запущенная на сервере виртуализации, параллельно с другими виртуальными системами.

Конечно, если масштабируемость и скорость реакции системы являются ключевыми характеристиками, то, вероятно, ваша тестовая система должна будет ближе соответствовать производственной.

Подробнее:

```
P2: p. 435, Chapter 18: Server Upgrades
```

15. Используется ли “канареечный процесс” при внесении изменений на большое количество машин?

Допустим вам нужно выкатить изменения на 500 машин. Быть может, это новое ядро. А может просто маленькое исправление.

Будете ли вы сразу выкатывать изменение на все 500 систем? Нет. Вы выкатите его на небольшое количество машин и посмотрите, вылезут ли какие-то проблемы. Нет проблем? Выкатываем на большее число машин. И так будем продолжать, шаг за шагом увеличивая число машин, на которое выкатываются изменения, пока не внесем изменения на все системы.

Машины, на которые выкатываются изменения вначале, называются “канарейки”.

Классический пример использования индикаторных животных — это канарейка в угольных шахтах. Вплоть до второй половины двадцатого века, шахтеры Англии и Америки брали с собой под землю канареек для раннего обнаружения ядовитых газов, включая метан и угарный газ. Более чувствительные птицы переставали петь или падали от газового отравления до шахтеров, давая им шанс выбраться из шахты или надеть защитные респираторы.

Источник: Wikipedia

Вот некоторые канареечные методы:

Один, несколько, много:

Внесите изменения в одну систему (например, в вашу личную), внесите изменения в несколько машин (например, ваших сотрудников), внесите изменения в большую группу машин (с каждым разом увеличивая группу). Любой сбой означает, что вы останавливаетесь,

откатываете внесенные изменения и повторяете попытку только после того, как проблема решена.

Кластерная канарейка:

Обновляете 1 машину, потом 1% всех машин, потом 1 машину в секунду, пока все не обновятся (обычная практика в Google и других сайтах с большими кластерами)

Эта процедура может выполнятся вручную, но если вы используете систему управления конфигурацией, возможность использовать “канареечный процесс” должна быть в нее вшита.

Подробнее:

```
P2: p. 56, Chapter 3: Workstations / 3.1.2.2 One, Some, Many
```

D. Подходы по автоматизации:

16. Используете ли вы инструменты управления конфигурацией, такие как cfengine/puppet/chef?

Система управления конфигурацией (Config Management, CM) — это инструмент, который координирует настройки машин. Он может управлять операционной системой, программным обеспечением, сервисами, или всем вместе.

До появления CM, администраторы вручную вносили изменения в системы. Если нужно было изменить настройку 100 систем, нужно было сделать это вручную. Те кто был поумнее, такую задачу автоматизировали.

Еще более умные сисадмины поняли, что было бы крайне удобно иметь общий инструмент для такой автоматизации. И они создали автоматизирующие системы, такие как track, cfengine, bcfg2, Puppet, Chef и другие.

Отличительной чертой систем управления конфигураций является то, что вы описываете конечный результат, который нужно получить, а система сама определяет, какие команды для этого нужно выполнить. Конечный результат описывается декларативными утверждениями, например «hostA является веб-сервером» или «на веб серверах установлены такие-то пакеты», а система управления конфигурацией переводит это в команды. Другим важным атрибутом CM является общий характер утверждений (“добавить выполнение скрипта foo.sh в cron”), которые автоматически преобразуются в подходящие для конкретной операционной системы настройки (например, foo.sh будет, в зависимости от ОС, добавлен либо в »/etc/crontab», либо в «/var/spool/cron»).

Использование CM позволяет, вместо ручного внесения изменений на каждой системе, изменить только конфигурационный файл и дать CM внести правки во все необходимые системы вместо вас.

Локальные правки на серверах — это не нормально. Каждый раз, когда вы создаете файл вроде /etc/crontab.bak или /etc/hosts.[сегодняшняя дата], это должно быть для вас индикатором того, что вы что-то делаете неправильно.

Управление конфигурацией — это конечная автоматизация. Из подмастерья чародея вы становитесь повелителем кукол. Это вам не фунт изюму.

17. Выполняются ли автоматизированные задачи под ролевыми учетными записями?

Мы часто создаем автоматизированные процедуры, которые запускаются в определенное время. Например, раз в ночь запускается скрипт, который валидирует базу данных.

В некоторых фирмах такие процессы запускаются от учетной записи одного из системных администраторов. Когда он покидает компанию, эти автоматизированные процессы умирают.

В хороших фирмах, такие скрипты запускаются от ролевой учетной записи, часто «root». Однако, безопаснее их выполнять от учетной записи с меньшими привилегиями.

18. Ваши автоматизированные задачи шлют почту только когда есть полезная информация?

Вы знаете историю о мальчике, который кричал “волк!”? А историю о cron-задаче, на которую не обращали внимания, потому что она дважды в день сыпала почтой и когда, наконец, она начала сообщать об ошибке, этого никто не заметил?

Мои правила простые:

```
Если нужна немедленная реакция: выслать  SMS  или сообщение на пейджер.
Если нужна реакция в течении суток: создать запрос в системе.
Если сообщение информационное: записать в файл.
Ничего не выводить, если нет полезной информации.
```

С помощью почты можно отправить и SMS, и создать тикет в системе запросов, но важно понимать, сама почта — это не лучший механизм оповещения. Вы можете быть включены в копию письма, которое отправляет SMS или создает тикет, но само письмо не должно быть первичным механизмом оповещения.

Наихудший вариант — это система, которая шлет журнальные сообщения всем системным администраторам по почте. Почтовая система — это не архив для логов.

Реальная история: мой друг, который живет в Нью-Йорке, работал в компании, где все автоматизированные задачи слали свой вывод почтой на адрес root@домен-компании. В свою очередь, «root» был списком рассылки, куда были включены все администраторы. Это генерировало бесконечный поток сообщений, а сисадмины в этой фирме буквально не могли прочесть почту, как бы они не фильтровали сообщения. В результате, для коммуникации администраторы использовали личные почтовые аккаунты, даже по рабочим вопросам. Что это была за компания? Большой провайдер сервиса электронной почты, который уже не существует (мне интересно, какие еще плохие решения способствовали развалу фирмы?).

E. Подходы по управлению парком машин:

*19. Есть ли у вас база данных всех машин?

Каждая фирма должна иметь информацию про свои системы. База данных должна хранить, как минимум, такую базовую информацию, как: ОС, ОЗУ, размер диска, IP-адрес, владелец, кого уведомлять о профилактических работах, и т.д.

Наличие базы данных машин, позволяет автоматизировать задачи, которые потенциально могут затрагивать все ваши системы. Такая база реализует ключевое требование для автоматизации многих задач: дает возможность выполнять команды только на машинах с определенной конфигурацией.

Информация о машинах должна собираться автоматически, хотя небольшие фирмы могут обойтись электронной таблицей или вики-страницей.

Инвентарная информация такого типа позволяет вам принимать решения, основываясь на данных, и помогает избежать проблем.

Я знаю одну историю, случившуюся в маленьком университете Нью-Джерси, который мог бы избежать больших проблем, если бы у них была хорошая система сбора инвентарной информации. Они попытались обновить Microsoft Office на всех своих компьютерах до последней версии. Исполнительный совет был преисполнен энтузиазмом: наконец наступит день, когда несовместимость перестанет превращать любою попытку взаимодействия в настоящее испытание. Кроме того, посмотрите на все эти новые возможности! Но как быстро этот энтузиазм сменился негодованием, когда проект провалился. Оказалось, что у трети компьютеров на кампусе не было достаточно памяти или дискового пространства. По всему университету было много случаев, когда обновление прошло неудачно, в результате чего сотрудники не могли делать свою работу. Исполнительный совет был не только разочарован, но и стал избрал принципиальную стратегию на избегание рисков. Пройдет еще не мало времени до следующей попытки провести обновления. Всего этого могло не случиться, если бы использовалась хорошая система сбора инвентарной информации. Она дала бы информацию, необходимую для правильного бюджетирования программы по обновлению программного обеспечения

20. Автоматизирован ли у вас процесс установки ОС?

Автоматизированная установка операционной системы быстрее, последовательнее и позволяет пользователям самим сделать часть ваших задач.

Автоматизированная установка ОС гарантирует, что все машины начинают свой жизненный цикл в одинаковом состоянии. Борьба с энтропией трудна, а если каждая машина сделана вручную, то она становится невозможной.

Ставя ОС вручную, вы тратите свое время впустую дважды: когда ставите систему и когда пытаетесь устранить проблему, которой бы не было, если бы у вас все машины были настроены одинаково.

Когда два человека устанавливают операционные системы вручную, половина из них будет настроена на так как нужно, причем вы не узнаете какая это половина. Оба могут утверждать, что использовали одну и ту же процедуру. Разведите их по двум разным комнатам и скажите чтобы они написали то, что они делали. Теперь покажите каждому из них, что написал другой. Ой, драка будет…

Пользователи воспринимают непоследовательность, как некомпетентность. Когда пользователю приходит новая система с всегда одинаковыми настройками, он знает, где поменять то, что ему не нравится. Когда же настройки постоянно разные, пользователь теряет доверие к системным администраторам. Что за тупицы ставят эти системы?

Если у вас есть возможность автоматически переустановить ОС, то она есть и у пользователей. Это минус одна задача для вас. Автоматизация, которая экономит вам время — это супер. Автоматизация, которая позволяет другим делать то, что им нужно — еще лучше.

Отсутствие возможности легко стереть и переустановить систему приводит к проблемам с безопасностью. Машина должна быть полностью стерта и переустановлена каждый раз, когда она меняет владельца. Когда этот процесс вызывает какие-либо сложности, возникает соблазн

сэкономить время, пропустив этот шаг.

Подробнее:

```
  P2: p. 41, Chapter 3: Workstations
  P2: p. 32, Chapter 2: Climb Out of the Hole / 2.1.4 Start Every New Host
in a Known State
  P2: p. 288, Chapter 11: Security Policy / Case Study: Security Though Good
Infrastructure
```

*21. Можете ли вы автоматически обновлять ПО для всего парка машин?

Если установка операционных систем автоматизирована, все машины начинают свой жизненный цикл в одинаковом состоянии. Если автоматизирована и установка обновлений, то все машины остаются в одинаково-обновленном состоянии. Последовательность — вещь хорошая.

Обновления по безопасности крайне важны, поскольку от них зависит надежность ваших систем. Другие обновления тоже важны, потому что от них тоже зависит надежность ваших систем, но, кроме того, они приносят новые возможности вашим клиентам. Воздержание от установки обновлений — это как воздержание от проявления родительской любви. Кто вас растил?

Обновление приложений также критично, как и обновление операционных систем. Пользователи не делают различий между “ОС” и “приложением”, особенно если это приложение установлено у многих. Противные типы, которые пишут вредоносное ПО, такой разницы тоже не делают.

Я бы очень хотел, чтобы банки были обязаны публиковать описание своих процессов обновления ПО, так, чтобы я мог решить, где хранить деньги.

Обход всех машин, одна за другой, является альтернативой автоматизации установки обновлений. Этот подход раздражает пользователей, тратит их время впустую, да и ваше время тоже. А с распространением ноутбуков, думать, что вы сможете обойти все машины, стало наивным.

Обновление должно происходить незаметно для пользователя всегда, когда это возможно. Если требуется перезагрузка или другие действия, пользователи должны иметь возможность отложить обновление. Но здесь должно быть ограничение, например 2 недели, и это ограничение должно быть настраиваемым, чтобы срочные обновления по безопасности не откладывались на долго.

Подробнее:

```
  P2: p. 41, Chapter 3: Workstations / 3.1.2 Updating the System Software
and Applications
```

22. Есть ли у вас политика обновления компьютеров?

Если у вас нет политики, регламентирующей когда обновляются компьютеры, они обновляться не будут.

[Под “компьютерами” я имею ввиду ноутбуки и рабочие машины людей, а не сервера.]

Обычно мы больше думаем о том, когда должны быть заменены устройства в серверных комнатах. Чтобы оставаться свежей, вашей компьютерной среде необходим повторяемый циклический процесс обновления. Без него, либо техника устаревает и становится неподдерживаемой, либо новая техника становится подтверждением статуса и приобретают политический окрас. При наличии хорошей политики, компьютерный парк остается современным и выгодным.

Определенная часть вашего парка должна быть старой — это просто экономически выгодно. Однако, слишком старые машины требуют больше затрат на поддержание, чем на замену. Поиск решения, как заставить работать новое ПО на слишком слабой машине — пустая трата времени, как и ожидание, пока медленный компьютер, наконец, завершит выполнять операцию. Очень старые машины оказывают негативный эффект как на производительность работы пользователей, так и на эффективность их тайм-менеджмента.

Компании часто попадают в такую ситуацию: стремясь “сэкономить деньги” они не обновляют компьютеры, но сотрудники с плохо работающими инструментами денег не экономят. А бывает, что компания просто не понимает, что компьютеры не вечны.

Вот простая политика для начала, если у вас ее еще нет: срок жизни компьютеров составляет 3 года. Годовой бюджет должен будет включать средства на замену ⅓ парка компьютеров. В первый день каждого квартала должно быть заказано достаточное количество компьютеров для замены 8-9% самых старых машин.

Финансовым директорам нравиться такой подход, потому что им нравится предсказуемость. В одной компании финансовый директор был очень доволен, когда я дал ему возможность решать, когда будет происходить обновление техники. Мы договорились, что ¼ часть всех обновлений должна происходить за квартал, а он решал в какой именно месяц это произойдет. Он даже мог разбить квартальное обновление на несколько частей и закупать их в разные месяцы.

Вместо того, чтобы периодически бегать к финансовому директору и выпрашивать новые компьютеры, мы организовали регулярный и спланированный процесс. Сделали жизнь проще для всех.

Совет профессионалов: в некоторых компаниях срок жизни сервера отличается: они спроектированы, чтобы служить дольше и потому их жизнь длится 4 года. С другой стороны, их цена амортизируется на всех пользователей и потому срок жизни в 2 года может быть вполне оправданным.

Подробнее:

```
P2: p. 41, Chapter 3: Workstations
```

F. Подходы серии “Мы согласны, что техника ломается”:

*23. Будут ли работать ваши сервера, если на них откажет один из дисков?

Раньше как было: если в компьютере что-то умирало, то происходил простой в работе. В сущности, выход из строя одного компонента был эквивалентен простою. Исчезала надежда что-то сделать в этот день и плакали ваши планы съездить на корпоративный пикник. Один

отказавший диск рушил планы на целый день, почти как атомная война.

Сейчас все изменилось. Мы строим “живучие системы”. Диск может отказать, но простоя не будет, если мы используем зеркалирование дисков. Простой возникнет только если диск с зеркальной копией тоже выйдет из строя. По статистике, у нас есть часы и даже дни, чтобы заменить отказавший диск, прежде чем случится простой, видимый для пользователей. Гораздо лучше потратить свое время на то, чтобы быстренько заменить диск, чем провести целый день за восстановлением данных с лент.

Такой подход отделяет “сбой компонента” от “простоя”. Жизнь становится лучше. Раньше системы RAID были дорогими и редкими, что-то вроде роскоши для богатых. Сейчас они стали распространенными, недорогими, а часто и вообще бесплатными (программный вариант). Я сказал распространенными? Хотел сказать обязательными. День, потраченный на восстановление данных с лент, это не просто знак плохого планирования, это знак плохого тайм-менеджмента. Утешение пользователя, потерявшего год, месяц или даже несколько часов работы — это пустая трата времени, а не героизм. Это плохое системное администрирование. И давайте не забывать еще большее расточительство рабочего времени: времени ваших пользователей, которые ждут, пока их данные будут восстановлены с резервной ленты. Отказы дисков — не редкость. Зачем тогда строить системы, считая что диски практически безотказны?

Среднее время наработки на отказ для серверного диска равно примерно 1,5 млн часов. Если у вас 1 тыс. дисков, отказ будет случатся раз в два месяца, если у вас 10 тыс. дисков — то каждую неделю. Вы действительно планируете так часто тратить по дню, восстанавливая данные с лент?

Мой подход следующий: для всех маленьких серверов загрузочный диск должен быть зазеркалирован, и любой диск с пользовательскими данными, должен быть в RAID1 или выше.

Загрузочный диск: я рекомендую зеркалировать загрузочный диск на всех серверах, потому что обычно невозможно полностью восстановить сервер с нуля. На загрузочных дисках часто собирается всевозможное “наследие”: за годы работы ставятся разные пакеты. В глубоких закутках файловой системы могут хранится недокументированные драйвера, патчи и прочие “костыли”. Конфигурация системы часто бывает больше историей компании, нежели продуманным и спланированным дизайном. В идеальном мире, все было бы по-другому: каждую систему можно было бы воспроизвести с помощью автоматизированной системы. Увы, это та цель, который мы еще не достигли. Основным исключением из этого остаются кластеры с гомогенными системами, как, например, высокопроизводительные кластеры (HPC), или Google и Yahoo!. Увы, дорогой читатель, я уверен, что это не ваша ситуация. Вообще говоря, я уверен, что даже в Google и Yahoo! виндовс-сервер, на котором крутится система управления электронными замками дверей здания, представляет собой крайний случай сервера, где требуется зеркалирование загрузочного диска.

Да, вы, вероятно, сможете восстановить такой сервер за день, если повезет, но RAID1-контроллер стоит меньше, чем день работы администратора с минимальной зарплатой.

Пользовательские данные: я рекомендую RAID1 или выше для пользовательских данных потому что это так дешево, что его отсутствие это просто стыд. Кстати… вы же знаете, что для дисков, размером от 2T, RAID6 — это минимум, правда? Использование RAID5 для таких дисков отдает профессиональной халатностью. Еще раз: для таких дисков RAID6 или RAID10 — это минимум. По крайней мере пока, но тут мы уходим в сторону.

Исключением из всего вышесказанного является любая система, которая продолжит предоставлять сервис при сбое отдельного компонента, будь то диск, машина или центр обработки данных. Плюс данные, которые могут быть восстановлены в пределах SLA. Вот некоторые примеры:

 Использование модных избыточных файловых систем, таких как Google File System (GFS). GFS хранит информацию как минимум в 3-х разных местах. Система GPFS Native RAID (GNR) фирмы IBM работает аналогичным способом.
 "Пробное и временное", когда пользователи знают, что этот сервис может исчезнуть в любой момент.
 Видео, либо другая информация только для чтения, которая может быть восстановлена с оригинального носителя в случае утраты.
 Копия только для чтения информации, хранящейся в другом месте. Хотя, если вы реплицируете эту информацию из соображений увеличения скорости доступа, RAID5 может еще больше ее увеличить благодаря большему количеству шпинделей.
 Легкозаменимые машины. Например, веб-сервер со статическим контентом или вторичный кэшируюший DNS сервер могут быстро и автоматически перегенерированы. Если у вас используются сотни серверов такого типа, экономия на RAID-контроллерах может быть существенной.

Подробнее:

```
  P2: p. 83, Chapter 4: Servers / Mirror Boot Disks
```

24. Используется ли схема N+1 в ядре вашей сети?

Сетевой сбой, который влияет на одного сотрудника, это стыд. Сбой, влияющий на много людей, просто неприемлем. Точно также, как и избыточные диски, резервные соединения стали сейчас обязательными.

Да, одно соединение от рабочего компьютера до коммутатора все еще норма, но дальше первого коммутатора все должно быть избыточно по схеме N+1. Как минимум, все магистрали должны быть с двойным подключением. В идеале — отказ любого подключения, сетевой карты, маршрутизатора или коммутатора не должен приводить к потере связности.

Локальные сети обычно проектируются следующим образом: офисные ноутбуки/компьютеры подключаются в настенную розетку. Эти розетки подключены к “коммутаторам доступа” с большим количеством портов. В свою очередь, коммутаторы доступа магистралями подключаются к иерархии центральных коммутаторов, которые обеспечивают доставку пакетов в нужное место, возможно за пределы локальной сети (в другие офисы или в Интернет).

Я использую простое правило: ядро сети должно быть избыточным. Раньше это было роскошью для богатых, теперь это стало необходимым требованием.

Если ваша сеть не использует этот принцип, то вы живете в том времени, когда компьютеры были полезными без сети.

Исключение: маленькие сети в которых нет ядра. И даже в таком случае, магистрали должны быть избыточными, а для каждого типа коммутатора должен быть запасной комплект.

Подробнее:

```
P2: p. 187, Chapter 7: Networks
```

*25. Автоматизировано ли ваша процедура резервного копирования?

Этот вопрос предполагает, что вы делаете резервные копии. Вы ведь делаете их, правда?

Вот четыре причины, почему вам нужны бекапы: (1) Ой, я удалил файл. (2) Опа, железо умерло. (3) Ой-ой-ой, здание сгорело. (4) Архивы. Каждая из этих причин может требовать отдельного похода.

Ситуация (1) решается с помощью снепшотов в краткосрочной перспективе, но не в долгосрочной. Иногда, файл, который нужно восстановить, был удален давно, и тут простые снепшоты не помогут, как и RAID. RAID — это не механизм резервного копирования. Если кто-то удалит файл по ошибке, RAID послушно среплицирует эту ошибку на все зеркала. И у вас получится избыточный массив неправильных данных.

Ситуация (2) вроде бы решается с помощью RAID, но помните, что двойной сбой дисков полностью разрушит RAID1 или RAID5. RAID10 и RAID6 будут разрушены при тройном сбое. И такие вещи случаются. Всего один неуклюжий электрик вас отделяет от сгорания всех дисков одновременно. Серьезно.

Ситуация (3) часто называется «аварийным восстановлением». Единственная надежда — резервные копии либо на дисках, либо на лентах, хранящиеся вне здания.

Ситуация (4) возникает вследствие необходимости соответствовать неким нормам. Технологически это реализуется аналогичным Ситуации 3 образом, но срок хранения информации другой. Если такое резервирование требуется какому-то отделу для соответствия нормам, он должен оплачивать стоимость носителей.

Каждая из перечисленных ситуаций требует автоматизированного решения. Вряд ли вы хотите рассказывать начальству о том, что данные утеряны, потому что “я был в отпуске” или “забыл поменять ленты”, после того, как сгорел ваш офис.

Наконец, автоматизация важна, потому что супернавороченная ленточная библиотека все-равно стоит дешевле, чем вы. Да, можно нанять клерка, чтобы он постоянно менял ленты. А можете купить библиотеку, с достаточным на месяц объемом лент. Библиотека выйдет дешевле. И она должна вмещать в два раза больше лент, чем нужно на ваш самый длинный отпуск.

Подробнее:

```
P2: p. 619, Chapter 26: Backup and Restore
```

*26. Проверяете ли вы периодически свой план аварийного восстановления?

В предыдущем разделе мы немного покривили душой. Там были указаны не 4 причины почему нужно делать резервные копии, а 4 причины почему надо восстанавливать данные.

Людям нет дела до бекапов, им важно иметь возможность восстановить данные. Если вы придумаете, как можно восстанавливать данные, не делая предварительно резервных копий, я пролоббирую в Нобелевском комитете создание номинации для системных администраторов, и вы будете первым, кто получит приз по ней.

Если резервная копия не проверяется, неизвестно действительна ли она. Плохи те системы резервного копирования, которые основаны на вере. Вера придает нам силы, но это не ИТ-стратегия.

Полный тест копии включает симуляцию полного краха и полного восстановления.

Вы не узнаете, сколько понадобиться времени на восстановление, пока его не проведете. Восстановление с лент часто занимает в 10 раз больше времени, чем сам бекап. Если полное резервное копирование вашей системы расчета заработной платы занимает 8 часов, нужно быть готовым, что ее восстановление с нуля займет 80 часов. А это больше трех полных суток.

Если вы вообще не проверяете свои резервные копии, то простенькое тестирование уже лучше, чем ничего. Напишите маленький скрипт, который будет выбирать случайный сервер, на нем — случайный диск, а на диске — случайный файл. Потом скрипт должен создавать тикет на восстановление этого файла (в новое место), каким он был 6 недель назад. Настройте автоматическое выполнение этого скрипта раз в неделю и с большой вероятностью он найдет сервер или диск, который отсутствует в системе резервного копирования. Да, и если вы думаете, что это тестовое восстановление будет занимать много времени, то есть небольшой секрет: оно вообще не будет занимать ваше время, если его выполнит ваш коллега. Пусть ваш скрипт генерирует тикеты с разнообразными описаниями, и тогда коллеги не узнают, что это было просто учебная тревога.

Можно сделать еще один шаг: спланируйте “веселый день”, когда реально проверится аварийное восстановление. Представьте, что некоторые люди “ушли в небытие” и проверьте, что остальные знают, как бороться с отказом сервисов. Напишите описания тестов, которые нужно выполнить. Потом спровоцируйте реальный отказ (выдерните кабель питания или сетевое подключение), либо сымитируйте его, а “человек из небытия” проконтролирует действия по восстановлению. “Так, допустим, вы получили вот такое сообщение. Расскажите мне команды и действия, которые вы будете выполнять”. Еще один способ спровоцировать отказ, это дать возможность генеральному директору выйти в серверную и отключить любой кабель на его усмотрение.

Подробнее:

```
P2: p. 261, Chapter 10: Disaster Recovery and Data Integrity
P2: p. 473, Chapter 20: Maintenance Windows
```

27. Есть ли в вашем ЦОД системы удаленного управления питанием и доступа к консолям?

Здесь мало что нужно объяснять. Коммутаторы удаленного доступа к консоли (IP-based KVM switches) недороги и они встроены в хорошие сервера. Удаленное управление питанием перестает быть роскошью, если компьютер находится более чем в нескольких километрах от вас.

Исключение из этого правила, это грид-системы (grid computing systems) с сотнями и тысячами идентичных машин. Если одна из них сбоит, другая забирает на себя ее функции.

Подробнее:

```
P2: p. 261, Chapter 10: Disaster Recovery and Data Integrity
```

G. Подходы к безопасности:

*28. Ваши сервера/ноутбуки/компьютеры защищены автоматически обновляемым антивирусным ПО?

Вирусы и вредоносное ПО давно уже часть нашего мира. Если вы думаете, что плохие вещи не случаются с хорошими людьми, то тогда мы все плохие люди. Каждый компьютер теперь требует ПО для борьбы с вредоносным ПО.

Каждое вредоносное действие означает работу для вас: чистку машины, восстановление данных, утешение пользователей, потерявших результаты работы. Пустая трата времени для вас, непростительный сбой для ваших пользователей, и плохой тайм-менеджмент.

Противовирусному ПО требуется периодическое обновление. Некоторые из таких программ выводят большое окно, где написано “Есть обновления! Хотите становить?”. Это очень важное окно, потому что в нем есть лого программы. Пользователи запоминают его и потом могут рекомендовать друзьям. Когда много людей знает производителя, это положительно сказывается на цене его акций. И не важно, что 9 раз из 10 пользователь в таком окне нажимает “нет”. Программы, которые молча обновляются, без анимированных информационных окон, теряют эту замечательную возможность. Это очень безответственно со стороны компании не заботиться о своих акционерах.

Если кто-то все-таки сомневается — это сарказм.

Существует большое количество антивирусного ПО. То, которым вы собираетесь пользоваться, должно обновляться молча.

Обеспечить соблюдение политики безопасности — ваша задача, а обновление антивирусного ПО — часть этой политики. Делегирование этой ответственности пользователям неправильно, а, возможно, и неэтично. Вы же не спрашиваете пешеходов: “Мне нажать на тормоза, чтобы вас не задавить?”. Точно также вы не должны давать пользователям возможность отказываться от обновлений. Да, когда-то у нас были модемы на 300 бод и загрузка вирусных баз заняла бы 30 минут. Только вы еще тогда не родились и потому эта отговорка не принимается. А если вы в то время уже работали (как я), то уже много чего повидали, чтобы знать что к чему.

По аналогичным причинам, возможность отключить антивирусное ПО не должна быть легкодоступной пользователям, иначе они будут его отключать по самым нелепым причинам. Обычная причина — чтобы повысить скорость работы. У меня был пользователь, который отключил антивирус, потому что “он привлекал вирусы”. Он объяснял это так: “Понимаете, когда он запущен, постоянно появляются предупреждения о вирусах. Как только я его выключаю — сообщения пропадают”. Да, люди с таким искривленным понятием о причинно-следственной связи существуют.

Раньше антивирусное ПО было “полезно иметь”, но теперь это превратилось в безусловное требование. Вот мои правила:

1) Антивирусное ПО должно быть запущено на всех машинах, включая любой сервер, где находятся пользовательские данные: домашние каталоги, файловые шары, содержимое веб-сайтов, FTP-сервера, и т.д.

2) Сканеры должны обновляться автоматически и скрытно. Никаких подтверждений от пользователей.

3) Должен быть механизм, позволяющий определить, что сканер отключен. Например,

центральный сервер, где все сканеры должны регистрироваться. Тогда вы сможете видеть, какие машины больше не обновляются.

4) Почта должна сканироваться на сервере, не на клиенте (на клиенте тоже может, но в дополнение к серверной проверке). Сообщения с вирусами должны уничтожаться, спам — помещаться в карантин. Вы не можете полагаться, что каждая личная машина будет иметь единый, высококачественный и обновленный фильтр, сравнимый с серверным. Остановите проблему до того, как она доберется до клиента.

Подробнее:

```
  P2: p. 284, Chapter 11: Security Policy / State of Security
  Blog post: EverythingSysadmin Blog: Not being attacked? Your network must
be down.
  Blog post: EverythingSysadmin Blog: Yes, malware scanners on your servers
too!
```

*29. Есть ли у вас политика безопасности в письменном виде?

Просмотрите существующие политики, чтобы набраться идей для своей. Библиотека примеров есть на сайте SANS:

http://www.sans.org/security-resources/policies/

Очень важно иметь политику безопасности в письменном виде до того, как ее реализовывать на практике.

Подробнее:

```
  P2: p. 293, Chapter 11: Security Policy
  P2: p. 293, Chapter 11: Security Policy / 11.1.2 Document the Company's
Security Policy
  P2: p. 293, Chapter 11: Security Policy / 11.1.3.5 Authorization Matrix
```

30. Проводите ли вы регулярный аудит безопасности?

Здесь мало что нужно объяснять. Если вы не проверяете свою безопасность, вы не знаете насколько вы уязвимы.

Подробнее:

```
  P2: p. 298, Chapter 11: Security Policy / 11.1.3.7 Internal Audits
  P2: p. 308, Chapter 11: Security Policy / 11.1.4.3 External Audits
```

31. Можете ли вы отключить учетную запись пользователя во всех системах за 1 час?

Ответ на этот вопрос говорит о вашей команде и среде намного больше, чем может показаться на первый взгляд. Он говорит, используете ли вы централизованную систему управления учетными записями.

Наличие единой системы авторизации для всех систем уже перестало быть “приятным дополнением” и стало обязательным. Если вы думаете, что оно вам не нужно, пока вы не

вырастете, то вскоре обнаружите, что времени на запуск единой системы авторизации, в период интенсивного роста, нету.

Зарекомендовавшим себя подходом является использование систем управления жизненным циклом учетных записей. Использовании таких систем позволяет управлять созданием и модификацией учетных записей начиная с момента подготовки к приему человека на работу, в течении жизненного цикла учетной записи, до завершения сотрудничества и после этого. Что если у пользователя поменяется имя? Что если кто-то возвращается в компанию? А если человек возвратится в компанию и у него изменилось имя? Есть множество подобных “нетипичных случаев” и система управления должна с ними справляться.

Подробнее:

```
P2: p. 223, Chapter 8: Namespaces
P2: p. 899, Chapter 36: Firing System Administrators
```

32. Можете ли вы сменить все привилегированные (root) пароли за 1 час?

Здесь тоже ответ говорит о большем, чем непосредственно спрашивается. Он говорит насколько хорошо контролируется административный доступ.

Если у вас нет такой возможности, создайте список на вики со всеми местами, где нужно сменить пароль. После этого, смените везде пароль по списку, добавляя по ходу пропущенные или забытые системы. Для сложных систем, укажите конкретные команды для смены пароля или опишите процесс смены пароля.

Если такая возможность у вас есть, создайте страничку на вики с описанием, как активировать этот процесс (и укажите все исключения, которые требуют смены пароля вручную).

Подробнее:

```
P2: p. 223, Chapter 8: Namespaces
P2: p. 899, Chapter 36: Firing System Administrators
```

Thanks to all the people that gave feedback on drafts: Strata Chalup, Sabrina Farmer, Aeleen Frisch, Doug Hughes, Duncan Hutty, Ski Kacoroski, Christina Lear, Edward Marczak, Troy Mckee, Adam Moskowitz, Tanya Reilly, Matt Simmons, Josh Simon, Nicole Forsgren Velasquez. (Especially Aeleen and Ski for helping reorganize the list and Josh and Nichole for extensive editing help.)

Since publication, helpful suggestions have come from: Jay Ashworth, Duncan Hutty.

Revision History:

2011-07-25: First public release.

2011-07-25: Many small updates.

2011-11-06: Editing and updates.