Глава 18

Обновления серверов

Данная глава имеет очень специфичную тему: обновление операционной системы отдельного узла. Эта задача, хотя и выглядит обманчиво простой, на самом деле требует большого объема предварительной подготовки и последующего тестирования. Само по себе обновление может осуществляться различными способами. Чем более критичен узел, тем важнее правильность выполнения обновления. Этот метод является строительным блоком. После его детального изучения вы можете перейти к более крупным проектам по обновлению, рассмотренным в главе 20.

Для успешного выполнения этой задачи требуется единственное средство, вне зависимости от типа операционной системы. Это средство – лист бумаги, который будет использоваться для создания контрольного списка. Использование этого средства обязательно.

Некоторые люди предпочитают имитировать лист бумаги при помощи веб-страницы, википедии или электронной таблицы. Такие высокотехнологичные решения имеют преимущества, которые будут рассмотрены позднее. Однако принцип один: обновлять сервер без контрольного списка недопустимо. Возьмите карандаш, давайте начнем.

18.1. Основы

Принципиальная задача любого обновления ОС состоит в том, чтобы как минимум все службы, которые предоставлялись *до* обновления, работали *после* обновления. Обновление может проводиться для того, чтобы *расширить* функциональность или надежность, но оно не должно их снижать. Учитывая это, процесс имеет следующую структуру.

1. Составьте контрольный список служб:
    a. Какие службы предоставлялись сервером?
    b. Кто является пользователем каждой службы?
    c. Какие программы предоставляли каждую службу?
2. Проверьте, чтобы каждая программа работала с новой ОС, или запланируйте путь обновления программного обеспечения.
3. Для каждой службы разработайте тест на проверку работоспособности.
4. Напишите план отмены с конкретными условиями.

5. Выберите технический перерыв.
6. Объявите об обновлении в необходимом порядке.
7. Выполните ранее разработанные тесты, чтобы убедиться, что они действенны.
8. Заблокируйте пользователей.
9. Проведите обновление с наблюдением/помощью (или под руководством) другого человека.
10. Повторите все ранее разработанные тесты. Соблюдайте стандартный процесс отладки.
11. Если тесты будут неудачными или произойдут другие события, которые являются условиями для выполнения плана отмены, выполните план отмены.
12. Разблокируйте пользователей.
13. Сообщите пользователям о завершении/отмене обновления.
14. Проанализируйте, что прошло правильно, а что нет, измените контрольный список в соответствии с приобретенным опытом.

А вы думали, что нужно всего лишь воспользоваться установочным диском, не так ли? Давайте более подробно рассмотрим каждый этап.

18.1.1. Этап 1: составьте контрольный список служб

Контрольный список служб – это средство, которым вы будете пользоваться для проведения всей процедуры. Список должен отражать, *какие* службы предоставляются узлом, *кто* пользуется каждой службой и *какая* программа предоставляет каждую службу.

Электронные таблицы – отличный способ представления такой информации. Самым большим преимуществом представления этой информации в электронном виде является то, что ее смогут легко совместно использовать персонал и пользователи. Лучше предоставить доступ к файлу через Интернет, чем отправлять его каждому человеку, потому что версию в Сети можно быстро обновить. Люди всегда будут видеть самые последние обновления[1]. Однако Сеть предполагает наличие механизмов активной доставки. Люди не будут искать файл сами. Вы можете включать URL в каждое электронное письмо, касающееся проекта, но это не даст гарантии, что его прочтут. Правильная идея – объявлять о любых значительных обновлениях.

Дважды проверьте свои планы, устройте собрание ключевых представителей затрагиваемого сообщества. Покажите им план, шаг за шагом, попросите их проверить ваши предположения. Наиболее эффективно начать процесс с собрания, а затем пользоваться для обновления электронной почтой, устраивая другие личные встречи, возможно, только в ключевые моменты процесса.

---

[1] Явно укажите в контрольном списке номер версии и дату, чтобы каждый мог легко проверить, является ли его версия последней.

Проверка зависимости пользователей

Один системный администратор устроил собрание десяти опытных системных администраторов, каждый из которых ознакомился с планом и сразу подтвердил, что он не возражает. Когда системный администратор начал разбирать его поэтапно, задавая конкретные вопросы, например: «Что будет, если мы это выключим?», они начали говорить: «Ой, нет, если вы это сделаете, биллинговая система не будет работать. Я думаю, нам нужно добавить этап, на котором мы перенесем биллинговую информацию». В результате получился совершенно другой план, в котором было в три раза больше этапов. Если бы не было личной встречи системных администраторов, на которой был рассмотрен каждый этап в отдельности, первоначальный план вызвал бы масштабную катастрофу.

Включение пользователей в процессы принятия решений и планирования дает им ощущение контроля и участия. Пользователи вкладываются в результат и становятся частью команды, что обычно ведет к более позитивным впечатлениям и лучшим отношениям между системными администраторами и подразделениями бизнеса. Совместный с пользователями доступ к информации о зависимости и состоянии через Сеть и по электронной почте позволяет поддерживать рабочие связи.

Машина может быть выделена для предоставления одной службы или предоставлять много служб. В любом случае предоставление службы в целом может обеспечиваться несколькими программами.

Что находится на машине?

Иногда вы точно знаете, для чего используется машина, и первоначальный контрольный список обновления создать легко. Однако со временем на машину добавляются дополнительные службы, функции и программы (Evard 1997). Мы можем дополнительно проконтролировать себя, если проверим сам узел. Вы можете просмотреть программы, установленные под UNIX, в директориях /opt, /usr/local и других местах, общих для таких систем. Операционные системы Майкрософт обычно размещают программы в папке под названием Program Files, хотя некоторые используют собственные правила, например устанавливают по умолчанию папку C:\apps. Вы можете посмотреть, какие процессы запущены в системе. UNIX- и NT-системы выводят все прослушиваемые порты TCP/IP и UDP/IP по команде `netstat -an`. В UNIX есть различные загрузочные скрипты, которые можно проанализировать. В NT есть консоль Службы. В UNIX есть файлы crontab, которые можно просмотреть. В каждой ОС есть по крайней мере один способ перечислить все установленные программы. Некоторые примеры таких средств — это `pkginfo` (Solaris и SVR4), `swlist` (HP-UX 10 и выше) и `'rpmqa'` (Linux).

Обычно каждая служба напрямую связана с одной программой. Иногда служба связана с несколькими программами, например сервер календаря, который использует сервер LDAP. Зафиксируйте все такие взаимозависимости в контрольном списке.

Кроме того, важно определить ключевых пользователей различных служб. Пользователи могут обращаться к службе напрямую или косвенно, взаимодействуя со службами, которые используют другие службы для получения или ввода данных. Людей нужно привлечь к процессу или, по крайней мере, уведомить о том, что происходит обновление. Если от служб зависят другие машины, надо включить в процесс пользователей этих машин.

Часто вы будете находить службы без прямых или косвенных пользователей, и такие службы можно отключить. Это всегда приятно, но будьте осторожны: вы можете найти зависимость тогда, когда службы уже не будет. Оставьте службу в готовом к запуску, но приостановленном состоянии, чтобы при необходимости ее можно было восстановить. Убедитесь, что вы отметили, почему служба существует, но не работает, чтобы в следующий раз ее можно было удалить, если к тому времени она не будет вновь включена. Лучшее место для такой документации – один из файлов конфигурации, который будет редактироваться при возобновлении служб.

18.1.2. Этап 2: проверьте совместимость программ

Следующий этап – убедиться, что каждая программа сможет работать с новой ОС, и запланировать способ обновления для тех программ, которые не будут работать. Используя список, созданный ранее, свяжитесь с разработчиками и узнайте, будет ли работать версия программы после обновления. Часто разработчики предоставляют такую информацию на своих веб-сайтах.

Вы можете захотеть сами проверить достоверность информации или найти другого пользователя, который уже сделал обновление. Представления разработчиков о том, что значит «версия будет работать», часто не включают функции, которые нужны в вашей компании, или точную конфигурацию, которую вы определите. Самостоятельное тестирование может быть дорогим, но, скорее всего, оно окажется дешевле, чем неудачное обновление, и при этом снижается риск неудачи. Целесообразность определяется из соображений управления риском. Если обновляется только одна система и приложение не является критически важным, его персональная проверка может быть пустой тратой времени. Если обновление автоматизировано и будет повторяться тысячи раз и потенциальная ошибка будет сильно заметна, тестирование необходимо.

Если используемая версия программы будет работать с новой версией ОС, зафиксируйте, где вы нашли эту информацию, для справки в будущем. Если программа не поддерживается новой ОС, у вас есть несколько вариантов.

- *Обновление до версии, которая поддерживается обеими ОС*. Если вам повезет, то программу можно обновить до версии, которая работает как в нынешней, так и в будущей системе. Если это возможно, запланируйте обновление до этой версии до обновления ОС. Здесь могут быть полезны тесты, разработанные на этапе 3.
- *Обновление возможно, но работает только в новой ОС*. В данном случае вы должны запланировать обновление программы после завершения обновле-

ния ОС. В зависимости от требований пользователей, это обновление либо может быть элементом обновления ОС, либо можно договориться о перерыве в работе службы, если пользователям не нужен непрерывный доступ. Например, если узел – загруженный веб-сервер, пользователи могут потребовать немедленной установки нового программного обеспечения, потому что это основная функция узла. Однако, если требуется обновить редко используемый компилятор, пользователи могут просто попросить, чтобы он был обновлен на следующей неделе или перед завершением определенного цикла разработки. Это особенно справедливо, если пока для компиляции можно пользоваться другим узлом.

- *Продукт больше не поддерживается*. Иногда мы только при обновлении операционной системы узнаем, что продукт больше не поддерживается разработчиком. Это может заблокировать обновление, либо пользователи могут пожелать сменить поставщиков или отказаться от этого продукта.

18.1.3. Этап 3: тесты для проверки

После того как будет определена каждая служба, нужно разработать тесты, которые будут использоваться для проверки того, что служба правильно работает после обновления. Лучший сценарий – записать все тесты в виде скриптов, которые могут быть запущены автоматически. Можно создать общий скрипт, который выводит сообщение «OK» или «FAIL» (Неудачное завершение) для каждого теста. Затем можно запускать тесты отдельно по мере устранения конкретных проблем. Для более сложных служб пользователи могут писать тесты или, по крайней мере, просматривать их либо предложить, чтобы их вызвали для выполнения вручную их собственных тестов. Некоторые программы имеют средства тестирования установки, которые могут быть запущены для проверки. Иногда эти средства проверки недоступны пользователям, но их можно получить через представителя разработчика.

Процедуры проверки программ

Все программные пакеты должны иметь процедуры проверки, но так редко бывает на самом деле. Иногда такую процедуру лучше написать самому. Тесты могут простыми, такими как проверка компилятора при помощи компиляции программы `Hello, World`. Один тест гораздо лучше, чем их полное отсутствие.

Иногда процедура проверки предоставляется, но на самом деле не работает. Один из поставщиков суперкомпьютеров был известен наличием плохих проверочных баз данных, особенно в бета-версиях ОС.

В обществе программистов для описания определенного способа проверки используется термин **регрессивное тестирование**. Вы сохраняете выходные данные старой системы, вносите изменение, а затем сохраняете выходные данные новой системы. Результаты должны точно соответствовать. Если ожидается, что новый результат будет немного отличаться, вы можете отредактировать базовый вариант вручную, чтобы он отражал ожидаемые изменения, либо воспользоваться алгоритмом *нечеткого соответствия*. Для сравнения результатов можно

воспользоваться простыми средствами. Например, UNIX-программа `diff` – это очень полезное средство, которое сравнивает два текстовых файла и указывает на различия между ними[1]. Программа `diff` имеет ограниченные возможности по оценке нечеткого соответствия, опция `-w` делает одинаковым все незаполненное пространство. Более сложные средства регрессивного тестирования могут программироваться на игнорирование конкретных изменений, обычно основанных на системе стандартных выражений. Однако такая сложность необязательна. Вы можете вручную изменить старый результат – сначала сделайте резервную копию! – чтобы отразить отличия, ожидаемые в новом результате. Например, вы можете изменять номера версий, чтобы они соответствовали новым программам. Прекрасные примеры регрессивного тестирования приведены в книге Кернигана и Пайка *«The Practice of Programming»* (Kernighan and Pike 1999), как и процедура установки `perl` (посмотрите, как реализованы тесты `make tests`).

Иногда тесты могут быть простыми, как при компиляции и запуске программы `Hello, world!` для проверки работы компилятора. Это может быть определенная последовательность команд или щелчков мышью, после которой можно посмотреть, отображается ли ожидаемый результат. Однако будьте внимательны, чтобы такие тесты не оказались поверхностными.

Hello, World!

Однажды Том отвечал за поддержку большого количества компиляторов для нескольких операционных систем. Он создал библиотеку простых программ, большинство из них только выводило `Hello, World!` и завершалось. Он всегда мог проверить, что новый установленный компилятор был, по крайней мере, принципиально правильным, если компилировались и запускались соответствующие программы. При добавлении новых языков программирования Том часто просил программистов написать тестовую программу. Программистам нравилось, что их просят помочь!

Вы должны проверять тесты так же подробно, как любую другую службу. Вы ведь не захотите, чтобы при запуске тестов были какие-то сомнения в том, закончился ли тест неудачно из-за обновления или из-за ошибки в самом тесте.

Заманчиво выполнить эти тесты вручную. Однако помните, что каждый тест будет выполнен как минимум три раза и даже больше, если возникнут проблемы. В этом преимущество автоматизации тестов. Если тесты достаточно общие, ими можно будет снова воспользоваться при дальнейших обновлениях. В конце концов, ими можно пользоваться регулярно просто для устранения проблем или в качестве средств мониторинга для обнаружения сбоев прежде, чем их обнаружат ваши пользователи.

Автоматизированные тесты хорошо подходят для программ, которые выводят предсказуемые текстовые данные, но их гораздо сложнее использовать для графических программ, сетевых служб, например NFS, или для таких физиче-

[1] Несмотря на то что `diff` впервые появилась в UNIX, есть порты практически на каждую существующую ОС.

ских действий, как печать. В случае с NFS вы можете попытаться осуществить доступ к файлу, а не проверять сам протокол. Тестирование сетевых служб, имеющих простые текстовые протоколы, таких как электронная почта (SMTP, POP, IMAP) или веб-службы (HTTP), может быть автоматизировано при помощи простых скриптов, использующих средства типа `netcat` для отправки и получения текста протокола на соответствующий сетевой порт.

Для других программ и служб вы можете найти специализированные тестовые системы, но они обычно очень дорогие. В таких случаях вам придется тестировать их вручную, документируя несколько ключевых функций для проверки или последовательность операций для выполнения. Рано или поздно каждый оказывается в такой ситуации, и все мы должны жаловаться разработчикам, пока они не обеспечат возможность автоматизированного тестирования своих продуктов.

Несмотря на то что автоматизировать все тесты предпочтительно, это не всегда возможно. Некоторые тесты слишком трудно автоматизировать, либо они требуют физического наблюдения. Даже если вы автоматизировали все тесты, то, если вам покажется, что требуется дополнительное тестирование вручную, выполните его. Иногда человеческий глаз видит то, что не может обнаружить лучшая автоматизация.

Разработка через тестирование

TDD (Test-Driven Development) – это относительно новая тенденция в отрасли. Раньше разработчики писали код, а затем создавали тесты, чтобы его проверить (ну, на самом деле это было не так, редко у кого-то было время создавать тесты). TDD – это обратный процесс. Сначала пишутся тесты, а затем – код. Это обеспечивает создание тестов для всего нового кода. Так как тесты выполняются автоматически, вы строите структуру тестов, которая сохраняется с проектом. По мере развития кода снижается риск того, что изменение нарушит функциональность и это не будет замечено. Разработчики свободно могут переписывать, или *перестраивать*, большие либо маленькие элементы кода, зная, что, если они что-то нарушат, это будет сразу замечено. В результате программы содержат меньше ошибок.

Тесты лучше комментариев в коде (документации), потому что комментарии часто устаревают и никто этого не замечает. Тесты, которые охватывают все граничные условия, гораздо более подробны, чем любая документация. Тесты не устаревают, потому что они могут быть включены как элемент процесса разработки, чтобы предупреждать разработчиков об ошибках, которые те внесли в код.

Нам бы хотелось, чтобы и в области системного администрирования изучали TDD и применяли такие методы.

Сохранение тестов для дальнейшего использования

Владелец крупного бизнеса хотел проверить 400 UNIX-серверов сразу после полуночи 1 января 2000 года, чтобы убедиться в том, что основные функции операционной системы и связанная с ними инфраструктура

работают правильно. Был создан ряд бесконтактных тестов, каждый выводил ответ PASS/FAIL: работает ли система, можем ли мы войти в систему, может ли она видеть NIS-серверы, правильно ли время, может ли система разрешать DNS, может ли она подключаться к NFS-серверам и читать файлы, работает ли автоматическое подключение разделов и т. д. При помощи центральной системы администрирования тесты могли одновременно запускаться на нескольких системах, а результаты – централизованно собираться. Все 400 серверов были проверены в течение 20 мин, и персонал смог сообщить о прохождении тестов группе отслеживания проблемы Y2K раньше других, меньших по размеру подразделений. Тесты приобрели такую популярность в группе системных администраторов, что стали элементом ежедневного мониторинга среды.

Тестам нашлось другое применение. Скрытая ошибка в автоматическом подключении разделов в Solaris 2.5.1 и 2.6 могла проявиться после серьезного сбоя сети, но лишь на нескольких случайных машинах. Запуск этого средства тестирования определял затронутые машины после любого сбоя.

18.1.4. Этап 4: напишите план отмены

Как вы вернетесь к предыдущему состоянию, если в ходе обновления что-то идет не так? Как вы сможете «отменить» его? Сколько времени это займет? Очевидно, мелкую неполадку можно попытаться исправить при помощи обычного процесса отладки. Однако вы можете потратить весь технологический перерыв – время, выделенное на отключение системы, – пытаясь сделать что-нибудь, чтобы обновление заработало. Поэтому важно иметь конкретное время, в которое будет задействован план отмены. Возьмите согласованное время окончания и вычтите время, которое потребуется на отмену, а также время, которое потребуется для проверки того, что отмена завершена. Когда вы исчерпаете это время, вы должны либо признать обновление успешным, либо начать свой план отмены. Полезно, чтобы за временем следил кто-то, не входящий в группу, непосредственно выполняющую обновление, например руководитель. План отмены также может быть задействован в случае неудачного выполнения одного или более ключевых тестов либо непредвиденных ситуаций, связанных с обновлением.

В малых или средних системах перед началом обновления можно создать полную резервную копию. Может быть, даже проще сделать точные копии дисков и выполнять обновление на копиях. В случае серьезных проблем можно вновь установить первоначальные диски. Крупные системы гораздо сложнее воспроизвести. В данном случае может быть достаточно создать копии системных дисков и постоянно делать резервные копии.

Обновление копии

Вопрос: Если вы собираетесь сделать точную копию жесткого диска перед обновлением сервера, где нужно выполнять обновление – на копии или на оригинале?

Ответ: Обновляйте копию. Если обновление пройдет неудачно, то вряд ли вам будет приятно обнаружить, что копия была сделана неправильно. Вы просто уничтожите оригинал. Мы видели это много раз.

Точное копирование дисков легко выполнить неправильно. Иногда данные копируются, а с загрузочным сектором возникает проблема и диск не загружается, иногда данные копируются не полностью либо не копируются совсем.

Чтобы избежать такой ситуации, загрузитесь с копии. Убедитесь, что копия работает. Затем выполните обновление на копии.

18.1.5. Этап 5: выберите технический перерыв

Следующий этап – это проверка ваших технических и нетехнических навыков. Вы должны согласовать со своими пользователями технический перерыв, то есть время, когда будет проходить обновление. Для этого вы должны знать, сколько времени займет процесс, и иметь план на случай неудачного обновления. Это больше касается технических вопросов.

- *Когда?* В ваше SLA должны быть включены положения о том, когда можно осуществлять техническое обслуживание. Обычно пользователи хорошо представляют, когда отключение пройдет для них безболезненно. Большинство систем бизнеса не нужны ночью или в выходные. Однако системные администраторы могут не захотеть работать в это время, а в определенное время может быть недоступна поддержка разработчиков. Нужно найти компромисс. Системы, которые должны работать в режиме 24/7, имеют предусмотренный режимом план обслуживания, возможно, содержащий резервные системы.
- *Сколько времени?* Продолжительность технического перерыва равна времени, которое потребуется на обновление, сложенному с интервалами времени, необходимого для устранения проблем, для выполнения плана отмены и для проверки того, что отмена сработала. Изначально лучше умножить ваши оценки на два или на три, чтобы не переоценить свои возможности. Со временем ваши оценки станут более точными.

  Вне зависимости от того, какую продолжительность вы подсчитали, объявите, что перерыв будет гораздо дольше. Иногда вы можете начать работу позже. Иногда она может потребовать больше времени, чем вы предполагали, по техническим (оборудование, программы или несвязанные либо непредвиденные события) или нетехническим (погода либо автомобильные пробки) причинам. Обратная сторона объявления большего перерыва заключается в том, что, если вы раньше закончите обновление и проверку, вам всегда следует сообщать об этом пользователям.
- *Когда нужно задействовать план отмены?* Хорошая идея – в явном виде указать точное время, в которое будет задействован план отмены, в силу причин, рассмотренных на этапе 4.

Скотти всегда преувеличивает

В серии «Relics» сериала *Star Trek: Next Generation* Джеймс Дуэн (James Doohan) появляется в эпизодах в роли Скотти (Scotty) из оригинального сериала. Одной из интересных находок Скотти было то, что он всегда преувеличивал, когда сообщал свои предположения капитану Джеймсу Т. Керку (Captain James T. Kirk). Таким образом, он всегда выглядел чудесным работником, когда проблемы решались быстрее, чем предполагалось. Теперь мы знаем, почему гиперпространственный привод всегда начинал работать раньше, чем предполагалось, а системы жизнеобеспечения функционировали дольше, чем прогнозировалось. Поступайте, как советует Скотти! Преувеличивайте свои оценки! Но не забывайте соблюдать другой принцип Скотти – сразу же сообщать людям, когда работа будет проверена и завершена.

Пример: карт-бланш в понедельник вечером

Когда Том работал в отделении Mentor Graphics, у системных администраторов была такая роскошь, как еженедельный технический перерыв. Вечер понедельника был карт-бланшем системных администраторов. Ожидалось, что пользователи выйдут из системы к 18 часам и системные администраторы смогут использовать этот вечер для выполнения любых видов серьезных обновлений, которые потребуют отключения служб. Каждый понедельник в 16 часов пользователям сообщали о том, какие изменения произойдут и когда системами снова можно будет пользоваться. В конце концов, пользователи выработали привычку планировать на вечер понедельника дела, не связанные с работой. Ходили слухи, что некоторые из них проводили время со своими семьями.

Несмотря на то что для одобрения такой практики руководством требовались серьезные политические вложения, она была важным фактором обеспечения высокой надежности сети отделения. Редко были причины отменять своевременные обновления системы. О проблемах в течение недели можно было позаботиться при помощи временных мер, но долгосрочные меры эффективно принимались вечером в понедельник. В отличие от некоторых организаций, где долгосрочные меры не принимались никогда, в данном случае они выполнялись относительно быстро.

Когда работы было немного, один администратор настаивал на перезагрузке некоторых критически важных серверов в 18 часов, чтобы «подтолкнуть» пользователей пойти вечером домой. Он верил, что это помогало пользователям поддерживать привычку не планировать ничего критически важного на вечер понедельника. Конечно, системные администраторы придерживались гибкого подхода. Когда приближался срок сдачи важного проекта и сотрудники работали круглосуточно, системные администраторы отменяли технический перерыв вечером в понедельник или согласовывали с пользователями, что можно отключить, не мешая их работе.

18.1.6. Этап 6: сообщите об обновлении в соответствии с установленным порядком

Теперь сообщите об обновлении пользователям. Используйте одинаковый формат для всех объявлений, чтобы пользователи к ним привыкли. В зависимости от культуры вашей организации, сообщение может распространяться по электронной или голосовой почте, в виде бумажной записки, записи новостной группы, веб-страницы, объявления на двери или дымовых сигналов. Вне зависимости от формата, сообщение должно быть кратким и по теме. Многие люди читают только строку «Тема», поэтому составьте текст грамотно, как показано на рис. 18.1.

Кому: Всем пользователям

Тема: ПЕРЕЗАГРУЗКА СЕРВЕРА СЕГОДНЯ В 18.00

От: Группа системного администрирования

Обратный адрес: tom@example.com

Дата: Четверг, 16 июня 2001

КОГО ЭТО ЗАТРОНЕТ:

Все узлы на DEVELOPER-NET, TOWNVILLE-NET и BROCCOLI-NET.

ЧТО ПРОИЗОЙДЕТ:

Все серверы будут перезагружены.

КОГДА?

Сегодня с 18 до 20 часов (должно занять 1 час).

ЗАЧЕМ?

Мы распространяем новые параметры настройки ядра по всем серверам.
Это требует перезагрузки. Риск минимален. Более подробную информацию можно найти на веб-странице http://portal.example.com/sa/news0005.

Я ПРОТИВ!

Отправьте сообщение в службу поддержки, и мы попытаемся изменить время. Пожалуйста, назовите имя сервера, который вы хотите нас попросить не перезагружать сегодня.

***Рис. 18.1.** Образец сообщения об обновлении*

Лучше иметь пустой шаблон, который нужно будет каждый раз заполнять, чем редактировать предыдущие объявления для включения новой информации. Это предотвращает форму от изменения со временем. Это также предотвращает распространенную проблему того, что некоторые части текста забывают заменить. Например, при подготовке рис. 18.1 мы первоначально использовали настоящее объявление о перезагрузке маршрутизатора. Мы изменили его на сообщение о серверах, но забыли отредактировать строку «Тема». Пример про-

шел через четыре корректуры, прежде чем кто-то это заметил. Если бы мы начали с пустого шаблона, то этого бы не произошло.

18.1.7. Этап 7: выполните тесты

Прямо перед началом обновления выполните тесты. Такая проверка в последний момент позволит вам убедиться, что вы не будете после обновления отслеживать проблемы, которые существовали еще до него. Представьте ужас от выполнения плана отмены только для того, чтобы узнать, что неудачный тест все равно не выполняется.

18.1.8. Этап 8: заблокируйте пользователей

Обычно лучше позволить пользователям выйти из системы самостоятельно, чем выбросить их путем перезагрузки или отключения службы. В различных службах для этого есть разные способы. Используйте доступные в ОС средства, чтобы предотвратить вход в систему во время технического перерыва. Многие пользователи делают попытку входа в систему или доступа к ресурсу в целях собственной проверки обновления. После успешной попытки пользователь считает, что система доступна для нормальной работы, даже если об этом не было объявлено. Поэтому важно заблокировать пользователей на время технического перерыва.

18.1.9. Этап 9: выполните обновление под чьим-нибудь наблюдением

С этого начинается большинство книг для системных администраторов. Разве вы не рады, что купили именно эту книгу?

Теперь пришел момент, которого вы все ждали: выполните обновление в соответствии с вашими местными процедурами. Вставьте DVD, перезагрузитесь или сделайте что-то еще.

Обновления системы слишком важны, чтобы выполнять их в одиночку. Во-первых, все мы делаем ошибки и вторая пара глаз всегда полезна. Обновления выполняются не каждый день, поэтому кому угодно может не хватать опыта. Во-вторых, когда два человека вместе обновляют систему, происходит уникальное обучение. Системные обновления часто требуют максимального использования наших технических знаний. Мы используем команды, знания и, возможно, даже части нашего мозга, которые не задействованы в другое время. Вы можете многому научиться, если посмотрите и поймете приемы, которые кто-то будет применять в это время. Метод совместной разработки, или так называемого парного программирования, становится все более популярным и предполагает, что разработчики работают в парах и набирают код по очереди. Это еще один метод разработки, использование которого может принести пользу системным администраторам.

Если обновление окажется неудачным, то никогда нелишне обратиться за помощью к коллеге или к старшему сотруднику вашего подразделения. Вторая пара глаз часто творит чудеса, и никому не должно быть стыдно просить помощи.

18.1.10. Этап 10: проверьте свою работу

Теперь повторите все ранее созданные тесты. В случае их невыполнения действуйте в соответствии с обычным процессом отладки. Тесты можно повторять снова и снова по мере проведения отладки. Вполне естественно снова запускать невыполненный тест после каждой попытки устранить проблему. Однако убедитесь, что вы запустили все тесты, прежде чем объявить об успехе обновления, так как многие процессы серверов взаимосвязаны. Исправление неполадки, которая вызывала невыполнение теста, может привести к невыполнению другого теста, который раньше выполнялся.

На данном этапе нужно привлекать пользователей. Как и в случае с моделью службы поддержки в главе 14, работа не может считаться выполненной, пока пользователи не проверили, что все завершено. Это может означать, что нужно позвать пользователей в заранее назначенное время либо пользователи могут согласиться сообщить вам на следующий день после завершения технического перерыва. В этом случае правильная работа автоматизированных тестов даже более важна.

18.1.11. Этап 11: если ничего не получилось, выполните план отмены

Если человек, который следит за временем, сообщает, что наступило время реализации плана отмены, вы должны начать его выполнение. Это может произойти, если обновление занимает больше времени, чем ожидалось, или если оно завершено, но тесты все еще не выполняются. Решение полностью определяется временем, а не вами или вашей группой. Отмена сложного обновления может разочаровывать или раздражать, но поддержание целостности сервера важнее.

Возвращение системы к предыдущему состоянию не должно быть единственным компонентом плана отмены. Пользователи могут согласиться, что, если не будут выполняться только определенные тесты, они смогут прожить 1–2 дня без службы, пока она восстанавливается. Заранее определите план действий на случай каждой потенциальной ошибки.

После выполнения плана отмены службы снова нужно проверить. На этом этапе важно зафиксировать в вашем контрольном списке результаты изменений. Это полезно для сообщения о состоянии дел руководству, усовершенствования процесса в следующий раз или восстановления цепи событий во время разбора. Записывайте такие особенности, как «реализовано по плану», «реализовано, но превысило заданное время», «частично реализовано, требуется больше работы», «не реализовано, изменение отменено», «не реализовано, служба недоступна, ожидается конец света». По возможности зафиксируйте результаты тестов и храните их вместе с информацией о состоянии. Это очень здорово поможет, если вы попытаетесь вспомнить, что произошло, на следующей неделе, в следующем месяце или в следующем году.

18.1.12. Этап 12: восстановите доступ пользователей

Теперь можно снова позволить пользователям начать работать с системой. У различных служб есть разные способы это разрешить. Однако часто сложно осуществить проверку, не разрешив всем пользователям доступ.

Впрочем, есть ряд способов это сделать. Например, при обновлении сервера электронной почты вы можете настроить другие серверы электронной почты не отправлять сообщения на обновляемый сервер. Пока эти серверы удерживают электронную почту, вы можете вручную проверить обновленный сервер, а затем сразу разрешить окружающим серверам отправку, внимательно наблюдая за только что обновленным сервером.

18.1.13. Этап 13: сообщите о завершении/отмене

На этом этапе сотрудникам сообщается о том, что обновление завершено или, если был задействован план отмены, что было выполнено, что не было выполнено и что системами снова можно пользоваться. Здесь выполняются три задачи. Во-первых, людям сообщают, что службы, к которым у них не было доступа, снова работают. Во-вторых, пользователям напоминают, что изменилось. Наконец, если они обнаружат проблемы, которые не были найдены в ходе вашего тестирования, им дают знать, как сообщить об этих проблемах. Если был задействован план отмены, пользователей следует проинформировать о том, что система должна работать идентично тому, как это было до попытки обновления.

Точно так же, как есть много способов объявить о техническом перерыве, существует много способов сообщить о завершении. Здесь возможна противоречивая ситуация. Пользователи не смогут прочесть сообщение электронной почты, если отключение затронуло службу электронной почты. Однако, если вы будете держаться в рамках своего технического перерыва, после него электронная почта возобновит свою работу и пользователи смогут прочесть отправленное по ней объявление. Если пользователи ничего не услышат, они посчитают, что после окончания объявленного технического перерыва все сделано.

Объявления должны быть краткими. Просто укажите, какие системы или службы снова работают, и предоставьте ссылку, по которой люди могут обратиться для получения более подробной информации, и номер телефона, по которому можно позвонить, если служба не заработает и будет невозможно отправлять электронную почту. Одного или двух предложений будет достаточно.

Самый простой способ сделать сообщение коротким – переслать первоначальное сообщение, которое указывало, какие службы будут отключены, и добавить в начале предложение о том, что службы снова включены, и о том, как информировать о проблемах. Это очень эффективно предоставляет людям контекст того, о чем сообщается.

Большие красные знаки

Пользователи склонны игнорировать сообщения от системных администраторов. Джош Саймон (Josh Simon) рассказал о том, что в одной компании, которую он обслуживал, он пытался оставлять записки, приклеенные к мониторам, – черный текст на ярко-красной бумаге, – где крупным шрифтом было написано «НЕ ВХОДИТЕ В СИСТЕМУ – СНАЧАЛА СВЯЖИТЕСЬ СО СВОИМ СИСТЕМНЫМ АДМИНИСТРАТОРОМ ПО ТЕЛЕФОНУ [номер телефона]!». Более 75% пользователей срывали бумагу и пытались входить в систему, а не звонили по указанному номеру. Из этого следует, что часто лучше действительно отключить службу, чем просить сотрудников не пользоваться ею.

18.2. Тонкости

Что вы можете сделать для развития процесса после того, как овладеете основами обновления сервера?

18.2.1. Добавляйте и удаляйте службы одновременно

В ходе обновления вы иногда должны одновременно добавлять и удалять службы. Это усложняет дело, потому что в один момент времени вносится более одного изменения. Отладка системы с двумя изменениями гораздо труднее, потому что требует особых тестов. Для добавления служб характерны те же самые проблемы, что и для установки новой службы на новом узле, но в данном случае вы будете в новой, возможно нестабильной, среде и не сможете подготовиться, написав соответствующие тесты. Однако, если новая служба также доступна на другом узле, можно разработать тесты и запустить их на нем.

Удаление службы может одновременно быть простым и сложным. Оно может быть простым по той же причине, по которой разрушить здание проще, чем его простроить. Однако сначала вы должны убедиться, что в здании нет людей. Иногда мы устанавливаем анализатор трафика для отслеживания пакетов, который показывает, что кто-то пытается воспользоваться службой на узле. Эта информация может быть полезна для поиска отставших.

Мы предпочитаем отключать службу так, чтобы ее можно было быстро снова активировать, если в дальнейшем будут обнаружены забытые зависимости. Обычно справедливо допущение, что можно удалить программу, если в течение следующего месяца или года не будут обнаружены забытые зависимости. Некоторые службы могут использоваться только раз в квартал или раз в год, особенно те или иные средства финансовой отчетности. Не забывайте вернуться, чтобы их убрать! Создайте заявку в вашей системе службы поддержки, отправьте себе сообщение электронной почты или создайте задание для `at`, которое отправит вам по электронной почте напоминание через некоторое время. Если доступ к узлу есть у нескольких групп системных администраторов или привилегированных пользователей, может быть полезным внести в файл конфигурации комментарий или переименовать его в OFF или DISABLED (ОТКЛЮЧЕНО). Иначе другой системный администратор может предположить, что служба должна работать, и снова ее включит.

18.2.2. Полная установка

Иногда гораздо лучше переустановить систему полностью, чем обновить ее. Выполнение одного обновления за другим может привести к тому, что система будет сильно повреждена. При этом могут остаться файлы от предыдущих обновлений, фрагментированные файловые системы и «богатое наследие» эпохи беспорядка.

Ранее мы рассматривали такую роскошь, как точное копирование определенных дисков и выполнение обновления на копии. Осуществить обновление как полную установку на другой системе – еще большая роскошь, потому что это не требует отключения старой системы. Вы можете выполнить полную установку на временной машине в спокойном темпе, убедиться, что все службы работают, а затем

подключить диски к обновляемой машине и соответствующим образом настроить конфигурацию сети. Имейте в виду, что машина, на которой осуществляется установка, должна быть практически идентична обновляемой машине, чтобы на дисках с новой ОС обеспечивалась вся необходимая поддержка и конфигурация оборудования.

18.2.3. Повторное использование тестов

Если тесты хорошо написаны, их можно интегрировать в систему мониторинга реального времени. На самом деле, если ваша система мониторинга уже выполняет все необходимые тесты, то при обновление вам ничего больше не потребуется (см. в главе 22 более подробное описание мониторинга служб).

Редко все тесты можно автоматизировать и внести в систему мониторинга. Например, тестирование нагрузки – определение того, как работает система под нагрузкой определенного объема работы, – часто нельзя выполнить на работающей системе. Однако возможность запустить эти тесты во время низкой интенсивности использования или по требованию при отладке может упростить отслеживание проблем.

18.2.4. Запись изменений системы

Построение контрольного списка служб гораздо проще, если вы ведете лог того, что было добавлено на машину. Например, в UNIX-системе просто записывайте изменения в файле под названием /var/adm/CHANGES. Чем проще редактировать файл, тем более вероятно, что люди будут его обновлять, поэтому создайте псевдоним оболочки или короткий скрипт, который просто открывает этот файл в текстовом редакторе.

Конечно, если машина не будет работать, логи изменений могут быть недоступны. Хранение лога изменений в википедии или на общем файловом сервере решает эту проблему, но может привести к путанице, если кто-то попытается начать новый лог изменений для узла. Установите правила, где должны храниться логи изменений, и соблюдайте их.

18.2.5. Генеральная репетиция

Учитесь у мира театра: повторение – мать учения. Почему бы перед выполнением обновления не провести генеральную репетицию на другой машине. Это может раскрыть неожиданные препятствия, а также показать вам, сколько займет процесс. Генеральная репетиция требует множества ресурсов. Однако, если вы хотите выполнить первое обновление из многих, она может стать ценным средством для оценки времени, которое для этого понадобится. Полностью завершенная генеральная репетиция приводит к появлению новой машины, которая может просто заменить старую машину. Если у вас есть эти ресурсы, почему бы это не сделать?

В театре также бывают так называемые *технические репетиции*, которые больше касаются людей, ответственных за свет и звук, чем актеров. Актеры читают свои роли в нужном порядке, а параллельно их словам делаются указания по свету и звуку. Эквивалент для системных администраторов – рассмотрение задачи всеми заинтересованными сторонами.

Кроме того, мы заимствуем у театра искусство мимики и жеста. Иногда серьезное изменение в системе предполагает физическую замену большого числа кабелей. Почему бы не рассмотреть все этапы, обращая внимание на такие проблемы, как длина кабелей, несоответствие прямых и перекрестных обжатий, несоответствие коннекторов «папа/мама», неправильные коннекторы и конфликтующие планы? Имитируйте замену в точности так, как она должна быть сделана. Лучше, если рядом с вами будет еще один человек, который станет объяснять задания по мере того, как вы имитируете их выполнение. Дайте другому человеку проверить, что каждый коннектор исправен и т. д. Сначала это может показаться глупым и трудоемким, но проблемы, которые вы предотвратите, того стоят.

18.2.6. Установка старых и новых версий на одной машине

Иногда на машине обновляется одна служба, а не вся ОС. В этой ситуации удобно, если разработчик разрешает, чтобы старые версии оставались на машине в отключенном состоянии, пока устанавливаются и сертифицируются новые программы.

Веб-сервер Apache под UNIX – один из таких продуктов. Мы обычно устанавливаем его в директорию /opt/apache-x.y.x, где x.y.z – это номер версии, но символическую ссылку из /opt/apache мы помещаем в ту версию, которой хотим пользоваться. При загрузке новой версии ссылка /opt/apache меняется, чтобы указывать на новую версию. В случае проблем с новой версией мы восстанавливаем символическую ссылку и перезапускаем демон. Это очень простой план отмены (применение символических ссылок в базе программного обеспечения рассмотрено в разделе 28.1.6).

В некоторых ситуациях старая и новая версии программы могут работать одновременно. Если требуется серьезная отладка, мы можем запустить новую версию Apache на другом порте, не трогая старую версию.

18.2.7. Минимальные изменения первоначальной версии

Обновления становятся проще, если сделать нужно мало. При помощи небольшого планирования все пакеты обновлений UNIX можно загружать в отдельный раздел, таким образом оставляя системные разделы в минимально измененном виде. Такие дополнения системы могут документироваться в файле CHANGELOG. Большинство изменений будет располагаться в директории /etc, которая достаточно мала, чтобы можно было скопировать ее перед началом любых обновлений и пользоваться ею для справки. Это предпочтительнее трудоемкого процесса восстановления файлов с магнитной ленты.

В UNIX-среде без данных на всех машинах есть локальная ОС, но остальные данные загружаются с сервера – обычно требуется сохранить между обновлениями только /var, а затем только задачи `crontabs` и `at`, данные электронной почты и, в таких системах как Solaris, файлы менеджера календаря. Для отслеживания изменений в файлах конфигурации удобно пользоваться системами контроля версий, например RCS.

Пример: обновление критического DNS-сервера

В данном примере объединены многие приемы, рассмотренные в этой главе. В спешке исправляя все ошибки Y2K перед 1 января 2000 года, Том нашел критический DNS-сервер, который работал на оборудовании, не защищенном от ошибки Y2K, и поставщик объявил, что не будет его исправлять. Кроме того, ОС не поддерживала Y2K. Это была прекрасная возможность поставить полностью новую ОС на совершенно новом оборудовании.

Том создал контрольный список служб. Хотя он думал, что узел предоставлял только две службы, при помощи `netstat -a` и перечисления всех запущенных процессов он обнаружил на машине много других служб. Он обнаружил, что многие из этих дополнительных служб больше не использовались, и нашел одну службу, которую никто не смог идентифицировать.

Люди знали, что большинство используемых программ будут работать на новой ОС, потому что они функционировали на других машинах с более новой ОС. Однако многие службы были внутренними разработками, и началась паника, когда не смогли найти исходный код из-за того, что автор, написавший эту программу, больше не работал в компании. К счастью, код был найден.

Том собрал новую машину и воспроизвел на ней все службы. На оригинальном узле было много файлов конфигурации, которые регулярно редактировались. Тому требовалось переписать эти файлы на новую систему, чтобы проверить, будут ли скрипты, которые их обрабатывают, работать на новой машине правильно. Однако из-за того, что обновление должно было занять пару недель, эти файлы могли подвергнуться многократному изменению, прежде чем новый узел будет готов. Тесты были выполнены на устаревших данных. Когда новая система была готова, Том остановил все изменения на старом узле, заново скопировал все файлы на новую систему и проверил, принимает ли новая система новые файлы.

Разработанные тесты запускались не один раз перед переходом, а многократно, по мере того как появлялась возможность пользоваться службами на новой системе. Однако Том оставлял большинство служб отключенными, когда они не тестировались, потому что старые и новые машины могли конфликтовать друг с другом.

Переход был организован следующим образом: старая машина была отключена от сети, но оставалась включенной. IP-адрес новой машины был изменен на адрес старой. Через пять минут срок действия ARP-кэшей локальной сети кончился и новый узел был определен. При возникновении проблемы Том мог отключить новую машину от сети и подключить старую. Старая машина осталась включенной, поэтому для ее возвращения в эксплуатацию не требовалось даже перезагрузки, нужно было только остановить новый сервер и подключить сетевой кабель старого.

Фактический технический перерыв мог быть довольно коротким – хватило бы и пяти минут, если бы все прошло хорошо и машину можно было бы сразу подключить. Однако был объявлен 30-минутный перерыв.

Том решил попросить двух человек посмотреть за его работой во время обновления, потому что он не так хорошо знал эту версию UNIX, как другие, и мало спал предыдущей ночью. Оказалось, что дополнительная пара рук помогла в отключении и подключении проводов.

Группа отрепетировала обновление за несколько часов до технического перерыва. Ничего не меняя, они повторили то, что точно было запланировано. Они убедились, что длина каждого кабеля будет достаточной и что все коннекторы были нужного типа. Этот процесс устранил всю потенциальную путаницу, которая могла бы возникнуть.

Обновление прошло хорошо. Некоторые тесты не были выполнены, но группа в короткие сроки смогла устранить проблемы. Неожиданной проблемой стала невозможность выполнения некоторых обновлений баз данных, пока не был исправлен скрипт. Пользователи, которые зависели от этих обновляемых данных, согласились работать с несколько устаревшими данными, пока на следующий день не исправили скрипт.

18.3. Заключение

Мы описали достаточно полный процесс обновления операционной системы компьютера, не упоминая при этом ОС конкретных разработчиков, конкретных команд, которые надо ввести, или кнопок, по которым нужно щелкнуть. Важнейшие элементы процесса связаны не с технологией (это вопрос чтения руководств), а с распространением информации, вниманием к деталям и тестированием.

Основное средство, которым мы пользовались, – это контрольный список. Мы начали с составления контрольного списка, которым затем пользовались для определения того, какие службы требовали обновления, сколько времени займет обновление и когда мы сможем его выполнить. Контрольный список определяет, какие тесты мы разрабатываем, и эти тесты могут использоваться снова и снова. Мы пользуемся этими тестами до и после обновления, чтобы обеспечить должное качество. Если обновление проходит неудачно, мы задействуем планы отмены, включенные в контрольный список. Когда процесс завершен, мы объявляем это заинтересованным пользователям, перечисленным в контрольном списке.

Контрольный список – это простое средство. Это единственное место, где собрана вся информация. Вне зависимости от того, используете ли вы бумагу, электронную таблицу или веб-страницу, контрольный список является точкой сбора. Он, фигурально выражаясь, поддерживает единство группы, позволяет не упускать из вида детали, помогает пользователям понимать процесс, а руководству – следить за происходящим и позволяет новым сотрудникам быстро входить в курс дела.

Как и многие процессы системного администрирования, обновление требует навыков общения. Переговоры – это процесс общения, и мы используем его, когда определяем, когда произойдет обновление, что должно произойти и каковы приоритеты, если что-то пойдет не так. Мы предоставляем пользователям ощущение завершенности, сообщая им об окончании работы. Это улучшает

отношения пользователей и системных администраторов. Мы не можем переоценить важность размещения контрольного списка на веб-странице. Чем больше людей могут ознакомиться с информацией, тем лучше.

Когда тесты автоматизированы, мы можем точно повторять их и обеспечить полноту их выполнения. Эти тесты должны быть достаточно общими, чтобы ими можно было пользоваться для дальнейших обновлений не только на том же узле, но и на похожих. Фактически тесты должны быть интегрированы в ваши системы мониторинга реального времени. Зачем выполнять эти тесты только после обновлений?

Этот простой процесс легко понять и отработать. Это один из основных процессов, которыми системный администратор должен овладеть, прежде чем переходить к более сложным обновлениям. Все примеры из реальной жизни, которые мы показали, требовали некоторого отклонения от основного процесса, по все же содержали главные элементы.

Некоторые дистрибутивы ОС делают обновление практически безопасным и безболезненным, но другие гораздо более рискованны. Хотя и нет полной гарантии успеха, гораздо лучше, когда в операционной системе можно осуществлять обновления надежно, с возможностью повторения и простого возвращения к предыдущему состоянию. Минимальное количество команд или щелчков мышью снижает вероятность человеческой ошибки. Возможность обновить много машин воспроизводимым методом имеет массу преимуществ; особенно важно то, что она помогает поддерживать целостность систем. Любая возможность вернуться к предыдущему состоянию предоставляет уровень отмены, аналогичный политике страхования: вы надеетесь, что это никогда вам не понадобится, ну а если понадобится, вы будете рады, что обеспечили такую возможность.

Задания

1. Выберите сервер в вашей среде и выясните, какие службы он предоставляет. Если вы поддерживаете документированный список служб, какими системными командами вы пользуетесь для проверки списка? Если службы не документированы, какими ресурсами вы можете воспользоваться для построения полного списка?
2. Как вы узнаете в своей среде, кто какими службами пользуется?
3. Выберите место, до которого легко дойти из вашей серверной или офиса, например магазин неподалеку, банк или какое-то место в другом конце вашего здания, если оно очень большое. Попросите трех-четырех студентов, коллег или друзей оценить, сколько времени займет дорога туда и обратно. Теперь вы все вместе должны пойти туда и записать, сколько времени это заняло. (Сделайте это прямо сейчас, до того, как прочтете оставшуюся часть вопроса. Правда!) Сколько времени это заняло? Вы пошли сразу или задержались? Сколько неожиданных событий по пути – случайных встреч с пользователями, с людьми, которые хотели узнать, чем вы занимаетесь, и т. д. – увеличили время дороги? Посчитайте, насколько точны были ваши оценки, их среднее значение и среднеквадратическое отклонение. Чему вы научились в этом эксперименте? Как вы думаете, насколько лучше будут ваши оценки, если вы повторите эксперимент с тем же местом? С другим местом? Повлияет ли на время привлечение большего количества

людей? Свяжите полученный опыт с процессом планирования технического перерыва.

4. В разделе 18.1.3 утверждается, что разработанные тесты должны быть выполнены как минимум три раза, а при наличии проблем – больше. Что это за минимальные три раза? При каких условиях тесты могут быть запущены повторно?
5. В разделе 18.2.7 есть пример, в котором практически невозможно было найти исходный код службы собственной разработки. Как бы вы поступили в такой ситуации, если бы не смогли найти исходный код?
6. Как вы объявляете планируемые отключения и технические перерывы в своей среде? Каковы преимущества и недостатки этого метода? Какая часть ваших пользователей игнорирует эти сообщения?
7. Пользователи часто игнорируют сообщения от системных администраторов. Что можно сделать, чтобы улучшить положение дел?
8. Выберите узел в вашей среде и обновите его (сначала спросите разрешения!).
9. Какие меры вы предприняли бы, если бы вам потребовалось заменить единственный туалет в вашем здании?