Журнал входит в перечень ВАК Минобрнауки РФ

ISSN 1813-5579

Системный администратор

ежемесячный журнал www.samag.ru

№04(161)
апрель 2016

389 Directory Server
Сервер каталогов в смешанной инфраструктуре предприятия

Старые сценарии на новый лад
Выполнение операций с удаленными системами

Язык Hack
Асинхронное выполнение кода приложения

Распараллеливание
операций в Oracle

Шаблонные строки в ES6
Больше, чем строки

Проводим пентест Wi-Fi
Проникаем в беспроводную сеть

Инструменты | **Хранение данных** | **Угрозы** | 16+

Построение корпоративных VPN
Рассмотрим варианты соединения Linux-Linux с помощью strongSwan, но уже с учетом динамического адреса клиента, а также соединения между Linux и FreeBSD и между Linux и Mikrotik

Системы хранения данных
Начинаем цикл публикаций, посвященных различным видам систем хранения данных. В первой статье цикла рассказывается о системах прямого подключения DAS и особенностях их применения

Анатомия таргетированной атаки
Рассмотрим основные этапы таргетированной атаки. Заглянем внутрь атаки, посмотрим на скелет общей модели и различия применяемых методов проникновения

Новый статус журнала

Системный администратор

ежемесячный журнал www.samag.ru

– ваши новые возможности!

www.samag.ru

Журнал «Системный администратор» вошел в перечень рецензируемых научных изданий Высшей аттестационной комиссии – ВАК!

Высокий уровень авторского контента «Системного администратора», давно ценимый нашими читателями, официально признан государством и российским научным сообществом.

Мы получили статус издания, в котором публикуются основные научные результаты на соискание ученой степени кандидата наук, доктора наук.

Редакция журнала приглашает к сотрудничеству всех тех, чьи профессиональные интересы связаны с научными изысканиями в сфере информационных технологий.

Требования к публикациям научных статей размещены на сайте журнала «Системный администратор»: http://samag.ru/main/part/49

Заявки на публикацию статей в раздел «Наука и технологии» присылайте на e-mail: vak@samag.ru

Dr.WEB®
Сделано в России

Самый легкий несигнатурный антивирус для Windows*

Всего 36 МБ оперативной памяти необходимо Dr.Web Katana для работы

Новинка!

Dr.Web Katana

Kills Active Threats And New Attacks**

Несигнатурный антивирус

для превентивной защиты от новейших активных угроз, целевых атак и попыток проникновения, в том числе через уязвимости «нулевого дня», которые могут быть не известны вашему антивирусу

- не является заменой сигнатурному антивирусу – работает «в связке» с установленным антивирусом
- не конфликтует с антивирусами других производителей
- не требует никакой настройки
- устанавливается на ПК и планшеты с Windows 10/8/8.1/7/Vista SP2/XP SP2+ (32-битные системы), Windows 10/8/8.1/7/Vista SP2 (64-битные системы)

Подробнее

https://products.drweb.com/home/katana

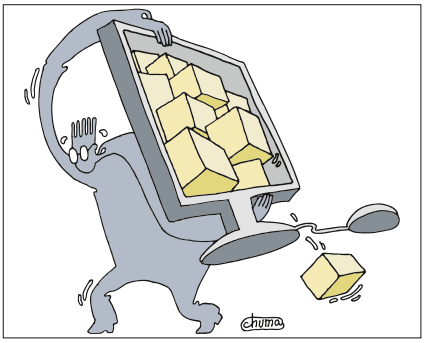
Технологии, используемые в Dr.Web Katana, включены в Dr.Web Security Space и Антивирус Dr.Web версии 11.0 – так что их пользователям новый продукт не потребуется.

* По версии сайта http://www.comss.ru/page.php?id=2752.

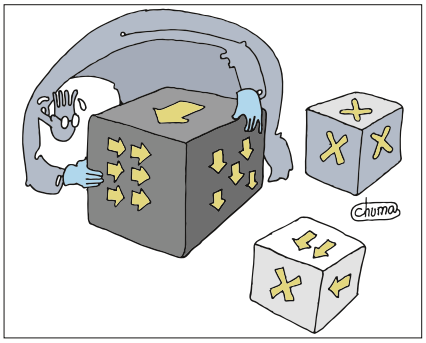
** Противодействует активным угрозам и новым атакам.

«Доктор Веб» — российский производитель антивирусных средств защиты информации под маркой Dr.Web. Продукты Dr.Web разрабатываются с 1992 года. «Доктор Веб» — один из немногих антивирусных вендоров в мире, владеющих собственными уникальными технологиями детектирования и лечения вредоносных программ.

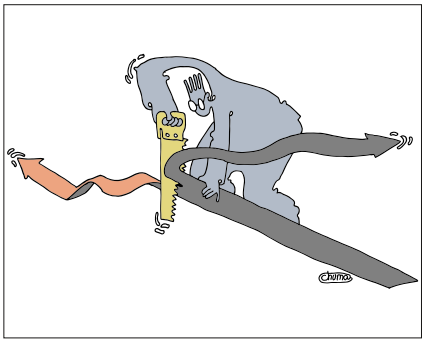
В номере



30



40



48

АДМИНИСТРИРОВАНИЕ

Хранение данных

Служба каталогов

Электронная почта

Продукты и решения

Удаленное управление

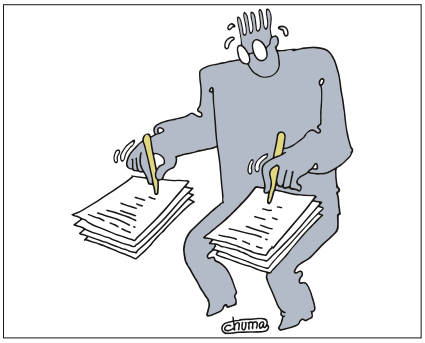
Инструменты

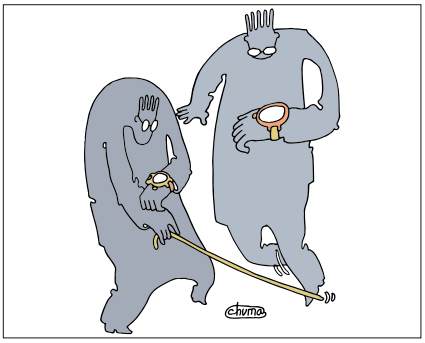
БЕЗОПАСНОСТЬ

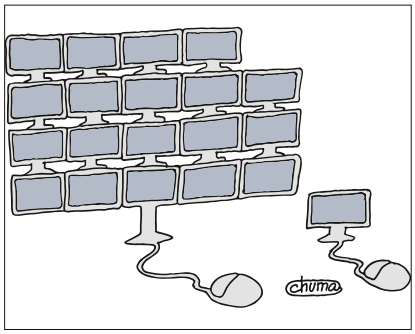
Угрозы

Аудит

Механизмы защиты







Визитка

АЛЕКСЕЙ БЕРЕЖНОЙ,
независимый консультант, системный архитектор, специалист по системам виртуализации и резервного копирования, alexey.berezhnoy@tech-center.com

Системы хранения данных
Часть 1. DAS

Эта статья начинает цикл публикаций, посвященных различным видам систем хранения данных. В ней рассказывается о системах прямого подключения DAS и особенностях их применения

. На сегодня все СХД можно условно разделить на три типа:
- устройства с прямым подключением (DAS),
- сетевые хранилища (Network),
- сети хранения данных (SAN).

Каждое из этих направлений имеет свои преимущества и ограничения, а также определенные особенности при использовании. В этот раз речь пойдет о системах хранения данных на основе DAS.

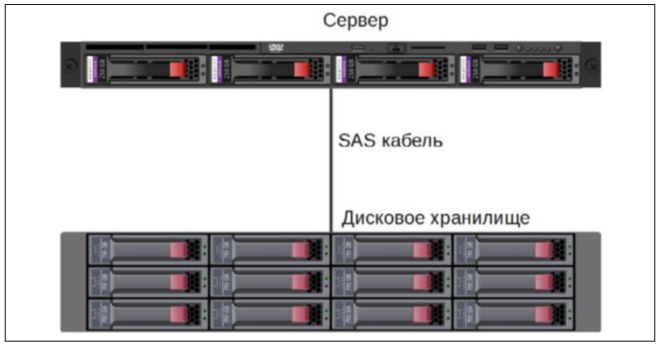
Общая информация о DAS

DAS (direct-attached storage) – устройство долговременной памяти, напрямую подсоединенное к компьютеру. Простейший пример DAS – встроенный жесткий диск.

Принято считать, что конфигурация DAS чаще всего используется в системах начального уровня, нетребовательных к объемам, производительности и надежности систем хранения.

Чаще всего, говоря о системах хранения с прямым доступом, подразумевают сочетание сервер + внешние жесткие диски. На самом деле в качестве внешнего носителя может выступать любое запоминающее устройство. Вариантов использования DAS в нашей жизни превеликое множество. Например, система хранения резервных копий в виде сервера с подключенной к нему ленточной библиотекой также является полноправным членом семейства DAS. Внешний накопитель DVD/CD, подключенный по USB, – тоже своего рода DAS- система.

Рисунок 1. Классическая система DAS



Но, если говорить о наиболее употребительном значении термина DAS, почти всегда подразумевается комплект оборудования из сервера и отдельного устройства для размещения накопителей – дискового хранилища.

Для связи сервера с внешним запоминающим устройством чаще всего используется интерфейс семейства SCSI [1], команды которого позволяют выделить определенный блок данных на специфицированном диске или смонтировать определенный картридж в ленточной библиотеке (см. рис. 1).

На сегодняшний день традиционный метод организации промышленных DAS – объединение сервера и дискового хранилища с помощью интерфейса SAS [2]. Существуют различные способы данного решения. Например, использование дорогостоящих дисковых контроллеров с большим объемом встроенной памяти для кэширования и современных интерфейсов SAS может обеспечить достаточно высокую скорость передачи данных. В то же время на рынке полным-полно простых, хотя и не очень производительных систем на базе дешевых контроллеров и плат расширения.

Примечание. Строго говоря, под понятие DAS подходит не только классическая комбинация сервера с дисковой полкой. Например, хранилище SAN, подключенное одним-единственным кабелем к соответствующему серверному адаптеру, по сути, представляет собой DAS-систему, несмотря на то что для взаимодействия используется протокол Fibre Channel, традиционно применяемый в распределенных сетях хранения данных.

Однако не стоит думать, что сочетание одного сервера и одного дискового хранилища является единственным вариантом использования. На практике встречаются конфигурации, представляющие собой множество дисковых полок, подключенных к одному серверу, при этом могут использоваться несколько контроллеров для подключения различных хранилищ. Например, в одной системе хранения могут

сочетаться старый интерфейс SCSI, более современный SAS и напрямую подключенное дисковое хранилище по протоколу Fibre Channel [3]. С точки зрения клиента это выглядит как один общий узел для размещения информации (см. рис. 2).

Теперь, когда мы познакомились с основными принципами организации DAS, самое время поговорить об особенностях его использования.

Особенности применения DAS

Недорогое решение

При рассмотрении цены стоит различать первоначальное внедрение и стоимость владения (обслуживание, замену комплектующих, модернизацию и так далее).

Начальная точка входа для этого решения достаточно невысока. В минимальном варианте это стоимость одного компьютера с набором жестких дисков требуемого размера.

Однако при росте числа таких систем необходимо учитывать, что на обслуживание каждой из них нужно тратить усилия системного администратора, и чем более ценные данные хранятся на узле, тем больше требования к квалификации специалиста и тем дороже его услуги.

Отдельно стоит сказать о проблеме комплектующих. Если системы хранения приобретались в разные периоды без учета последующего роста требований к объему и быстродействию, то через некоторое время такая разрозненная инфраструктура превращается в настоящий «зоопарк», в котором можно встретить массу редких обитателей: от дисковой полки с PATA [4] (IDE) дисками до древних интерфейсов SCSI-2. Разумеется, при обслуживании этого «хозяйства» возникает масса проблем. Стоимость дефицитных раритетных запчастей может быть в разы больше, чем цена за недорогое сетевое хранилище.

Использовать старое оборудование – удовольствие, прямо скажем, так себе. Но гораздо хуже, если на нем завязаны актуальные бизнес-процессы, затронуть которые – значит, надолго прервать работу организации. В итоге работа персонала ИТ-подразделения по поддержанию работы начинает напоминать колдовские ритуалы для поддержания тлеющего огонька жизни в древних устройствах, которые при обычных обстоятельствах должны были давно помереть.

Простота организации

Если использовать простой комплект оборудования, то организовать классическую DAS действительно не очень сложно. Достаточно приобрести дисковую полку, соответствующий контроллер для установки в сервер и комплект интерфейсных кабелей.

Если операционная система не поддерживает «из коробки» приобретенный контроллер, то еще понадобится дополнительный драйвер для этого устройства.

Выше мы говорили о самых простейших вариантах. Более серьезные системы DAS поставляются с дополнительными интерфейсными портами, позволяющими выполнять сервисные функции. Также применяется специальное программное обеспечение для мониторинга, перезагрузки и других сервисных функций. Системы хранения с прямым доступом уровня Enterprise по своей сложности порой не уступают NAS или SAN.

Высокая скорость передачи данных

Как уже сообщалось выше, для связи сервера с хранилищем чаще всего используются интерфейсы семейства SAS (SCSI). Современные контроллеры SAS для внешних подключений позволяют получить скорость до 12 Gb/s, что является довольно неплохим показателем.

Помимо SAS, для недорогих решений могут использоваться и другие стандарты, например, eSATA (External SATA) [5], позволяющий достичь скорости 3 Gb/s.

Традиционный метод организации промышленных DAS – объединение сервера и дискового хранилища с помощью интерфейса SAS

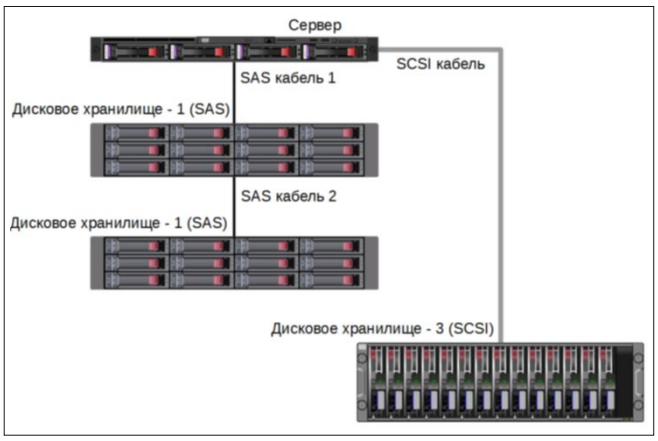
Подключение по USB 3.0 [6] теоретически может достигать скорости 5 Gb/s, что также является неплохим показателем для недорогих систем. На практике тестовая скорость может оказаться существенно ниже, в первую очередь из-за особенностей оборудования и работы программного обеспечения.

К сожалению, мой опыт подсказывает, что подключение по USB не является надежным соединением. Я бы не рекомендовал использование таких устройств в системах с требованием высокой отказоустойчивости.

Сложности масштабирования

При всех достоинствах DAS эти системы хранения представляют собой одиночные устройства. Можно выбрать решение с сервером, вмещающим много жестких дисков, и подключить к нему несколько объемных дисковых хранилищ, но даже такой вариант имеет весьма скромные возможности для наращивания объема. Не стоит забывать, что все операции ввода-вывода будут проходить через один сервер, и любой узел DAS со временем превратится в узкое

Рисунок 2. Пример структуры DAS с параллельным и последовательным подключением хранилищ. Первое и второе хранилища последовательно подключены через интерфейс SAS. Третье хранилище подключено к отдельному SCSI-контроллеру параллельно двум остальным



Из личных наблюдений автора

Пример шаманского ритуала, подсмотренный мной в одном ВЦ довольно крупной компании. (Возможно, поможет одному из тех несчастных, кто вынужден поддерживать старые дряхлые системы на основе SCSI DAS.)

Надо заметить, что все действия, направленные на поддержание жизнедеятельности вверенных систем, ИТ-персонал выполнял старательно и в срок. Оборудование очищалось от пыли, вентиляторы заменялись, электропитание и кондиционирование содержались в норме, и за ними также был установлен соответствующий контроль. Но все равно время от времени случались странные казусы, от которых избавлялись одним из способов, приведенных ниже.

Итак, проблема – сервер перестал видеть дисковое хранилище и «потерял» все подключенные тома.

План «А». Перезагрузить сервер. Если не помогло – перезагрузить еще раз. Если тома на хранилище все еще недоступны, то все выключить (сначала сервер, потом дисковое хранилище). Отсоединить кабели питания. Выждать 15 минут. Снова подсоединить электропитание и включить дисковое хранилище. После его старта выждать 10 минут и включить сервер. Если не помогло, сбросить CMOS Setup в BIOS к настройкам по умолчанию и снова попытаться запустить систему. Если это не помогло, то переходим к плану «Б».

План «Б». Все выключить по вышеописанной схеме. Отсоединить кабели питания. Отсоединить интерфейсный SCSI-кабель. Открыть корпус сервера. Переставить SCSI-контроллер в соседний слот. Подсоединить SCSI кабель. Попытаться запустить систему (сначала хранилище, спустя 5-10 минут сервер). При отсутствии положительного результата переходим к плану «В».

План «В». Снова все выключить (сначала сервер, потом дисковое хранилище). Отсоединить кабели питания. Отсоединить интерфейсный SCSI-кабель. Открыть корпус сервера. Демонтировать SCSI-контроллер. Аккуратно извлечь жесткие диски из корзин дискового хранилища, не забыв пометить каждый из них, из какой по счету корзины его вытащили, чтобы не перепутать. Выждать примерно с полчаса, чтобы все «отлежалось». Установить жесткие диски и SCSI-контроллер на те же места. Подключить кабели. Включить дисковую полку, потом через 5-10 минут – сервер.

И это отнюдь не первоапрельская шутка, а реальная статья из внутренней Knowledge Base, которая использовалась не одним поколением системных администраторов в той организации. Самое удивительное, что это работает! Возможно, кто-то из читателей может поделиться своими шаманскими вариантами реанимации старых дисковых систем с прямым доступом.

Разумеется, реализовать работу в режиме 24/7 с минимальными простоями на таких «динозаврах» не получится.

место для всей ИТ-инфраструктуры. Разумеется, если ничего при этом не предпринимать.

Для того чтобы обойти данное ограничение, используют распределенные файловые системы, самая известная из которых – DFS от компании Microsoft. Но в целом, если DAS используется «в чистом виде», без каких-либо дополнительных средств, облегчающих масштабирование, можно смело утверждать, что рано или поздно такая система достигнет своего предела и придется либо приобретать новую, либо переходить на более универсальные варианты, например SAN.

Отсутствие консолидации ресурсов

Массовое использование DAS приводит к тому, что инфраструктура, по сути, состоит из разрозненных фрагментов долговременной памяти. При этом часто возникает ситуация, когда на каком-либо сервере с DAS-подключением заканчивается место, а на соседнем дисковое пространство практически не используется, и нет возможности компенсировать одно за счет другого.

Ограничения отказоустойчивости

Разумеется, не стоит думать, что большинство DAS-систем ненадежны и начинают сбоить сразу после включения. На практике дело обстоит чаще всего наоборот – современная, хорошо настроенная система хранения, построенная на надежном оборудовании от ведущих производителей, может работать годами без единого инцидента.

Речь идет о том, что в случае проблемы с сервером вся информация, записанная на нем, становится недоступной.

Разумеется, прогресс не стоит на месте, и существуют решения, позволяющие на базе DAS строить достаточно надежные решения, например, на основе отказоустойчивых кластеров Active-Passive и даже Active-Active. Для этого, правда, приходится задействовать дополнительные средства, например, кластерные сервисы, файловые системы общего диска или специальные RAID-контроллеры. Но надо признать, что эти решения являются скорее исключением из правил. Классическая система – сервер с дисковой полкой – не в состоянии обеспечить тот уровень отказоустойчивости, который доступен, например, при использовании SAN.

Сложность разделения ресурсов

Используя DAS, достаточно затруднительно добиться совместного использования одного тома разными клиентами в режиме разделения данных. В отличие от сетевого ресурса, когда к одной и той же директории или к одному и тому же файлу обращаются несколько пользователей с разных устройств, в DAS такой вариант имеет множество ограничений. Даже применение файловой системы общего доступа, например OCFS2 [7], для совместного подключения одного тома (LUN [8]) к нескольким серверам не решает задачу совместного использования одного и того же ресурса несколькими пользователями. Используя DAS, можно реализовать схему отказоустойчивого кластера Active-Active. Но классический вариант создания разделяемого ресурса между несколькими клиентами в DAS практически неосуществим.

Сложность управления

Кроме того, при такой организации хранения нет никакой возможности создать единую точку управления внешней памятью, что неизбежно усложняет процесс резервирования/восстановления данных и создает серьезную проблему защиты информации.

В конце концов если использовать исключительно DAS и ничего более, то стоимость владения подобной системой хранения может оказаться значительно выше, чем внедрение более сложных и более дорогих систем хранения данных: NAS и SAN.

DAS как основа для других систем хранения

Современные системы хранения данных можно условно разделить на два типа:

> Монолитные решения в составе одного корпуса.

- Модульное решение, состоящее из нескольких устройств.

Модульное решение, в свою очередь, состоит из:

- **Controller Enclosure** – главного (управляющего) модуля, используемого для функций обмена данными с внешней средой и управления.
- **Disk Enclosure** – вместилища жестких дисков, соединенные с главным модулем интерфейсным кабелем. Еще одно название таких модулей системы – «дисковая полка». Оба термина обозначают одно и то же – место для размещения HDD-накопителей.

По своей сути, модульное решение представляет собой DAS, заточенную для более узкого применения.

Справедливости ради стоит заметить, что дисковые системы хранения изначально не представляли собой монолитные решения. Первые дисковые хранилища, подключаемые к майнфреймам вычислительных комплексов, строились на базе двух составляющих:

- набора дисковых накопителей в виде напольной тумбы с открывающейся дверцей, через которую можно было извлечь набор съемных пластин;
- контроллера дисковой подсистемы, который занимал отдельный шкаф размером примерно с современную серверную стойку.

Впоследствии, когда габариты жестких дисков значительно уменьшились, это позволило размещать и контроллер и сами диски в небольшой серверный корпус.

Но технический прогресс развивается по спирали. В связи с ростом объемов хранения данных возросли требования к вместимости оборудования и удобству обслуживания. Поэтому вновь произошло разделение, но уже на контроллер и дисковую полку.

Disk Enclosure бывают двух типов:

1) Недорогие реализации, представляющие собой только вместилище дисков с интерфейсной платой для подключения к HBA или RAID-контроллеру. Иногда их еще называют Expander (дословно означает «расширитель»). Данный термин подчеркивает тот факт, что подобные устройства необходимы в основном для увеличения количества подключаемых дисков.

Работа недорогих дисковых «полок» чаще всего строится на основе мультиплексирования, когда из нескольких подключенных дисков в один момент времени доступен только один. Из-за того, что внутренняя скорость обмена обычного HDD значительно меньше быстродействия контроллера, такие переключения не существенно снижают скорость обмена данными. Благодаря специальным функциям интерфейсной платы Expander при «переходе» на другой накопитель операции обмена с текущим HDD не прерываются, а приостанавливаются. Поэтому такие переключения остаются прозрачными для компьютерной системы в целом, что позволяет работать с множеством дисков в одной полке так, как если бы они все были локально установлены в одном сервере.

2) Более сложные Disk Enclosure имеют в своем составе дополнительные устройства, например, кэш с предварительной выборкой, более сложные контроллеры для взаимодействия с жесткими дисками и другие «хитрости», позволяющие избежать недостатков мультиплексирования и сохранить максимально возможный уровень быстродействия. Современные Disk Enclosure от ведущих производителей снабжаются средствами индикации ошибок, мониторинга, замены Firmware и другим полезным функционалом. В то же время стоимость хранилищ значительно выше простых дисковых полок с базовым функционалом.

Установка устройств хранения DAS — более дешевый вариант по сравнению с NAS и SAN

Если говорить о Controller Enclosure, то в современных системах хранения, например устройствах SAN, этот компонент представляет собой мощный компьютер, далеко не всегда построенный на базе x86 совместимых процессоров, обладающий большим объемом памяти для кэширования и собственных вычислительных нужд, снабженный специализированными контроллерами для доступа по протоколам Fibre Channel, сетевыми адаптерами семейства Ethernet и так далее.

В итоге мы фактически получаем сочетание компьютера (пусть даже узкоспециализированного) с внешним запоминающим устройством. Таким образом, модифицированные DAS являются своего рода кирпичиками для построения более сложных промышленных систем на базе сетей хранения данных (SAN) или сетевых хранилищ (NAS).

• • •

Установка устройств хранения DAS – более дешевый вариант по сравнению с двумя оставшимися направлениями: NAS и SAN, при развитии ИТ-инфраструктуры и росте объемов хранимых данных использование только одних серверов с напрямую подключенными к ним дисковыми хранилищами снижает общую эффективность работы ИТ. EOF

[1] Коляденко А. Почти все о SCSI – http://citforum.ru/hardware/pc/scsi.
[2] Зейников А. Интерфейс SAS: история, примеры организации хранения – https://habrahabr.ru/post/175313/.
[3] Касачев К. Основы Fibre Channel – https://habrahabr.ru/post/216369.
[4] PATA, IDE – интерфейс, слот. Для чего нужен, скорость передачи данных и режимы – http://www.xtechx.ru/c40-visokotehnologichni-spravochnik-hitech-book/pata-interface.
[5] SATA – интерфейс. Виды разъемов, скорость передачи данных, ревизии и версии SATA – http://www.xtechx.ru/c40-visokotehnologichni-spravochnik-hitech-book/sata-interface-port.
[6] Сайт, посвященный USB 3.0 – http://www.usb-30.ru/interface.
[7] Описание файловой системы OCFS2 – http://xgu.ru/wiki/OCFS2.
[8] LUN storage: Working with a SAN's logical unit numbers – http://searchstorage.techtarget.com/essentialguide/LUN-storage-Working-with-a-SANs-logical-unit-numbers.

Ключевые слова: устройства хранения, DAS, SAS.

Визитка

АЛЕКСАНДР ТЕТЮШЕВ,
к.т.н., доцент кафедры АВТ Вологодского государственного технического университета (ВоГТУ), tetavv@gmail.com

Сервер каталогов 389 Directory Server
в смешанной инфраструктуре предприятия

В статье описана реализация сервера каталогов на базе 389-DS для централизованного управления пользователями и ресурсами предприятия. а также реализация на его основе перемещаемых профилей для пользователей Linux и Windows

Большинство современных российских предприятий использует Microsoft Active Directory (MS AD) в качестве сервера каталогов, выстраивая свою инфраструктуру на основании его возможностей. Одним из объяснений массового применения MS AD является простота его развертывания и поддержки, а также наличие большого количества документации и руководств. Один из сотрудников компании может легко вести рутинные операции по созданию и сопровождению учетных записей пользователей и ресурсов в MS AD в качестве нагрузки к основной деятельности. С другой стороны, MS AD разрабатывался как продукт корпоративного рынка, и его возможности изначально избыточны для небольших организаций, а стоимость Windows Server неприлично высока. Кроме этого, ремонт MS AD – весьма сложная процедура, требующая специальных знаний и, как правило, заканчивающаяся полной переустановкой сервера с длительным простоем информационных сервисов предприятия. Особо стоит подчеркнуть недружественность MS AD к программным продуктам сторонних разработчиков (Linux, Android, iOS и т.д.).

В этой статье рассматривается альтернатива MS AD на примере сервера каталогов 389 Directory Server [1]. Как будет показано, его использование для рутинных операций по сопровождению учетных пользователей не сложнее MS AD, а настройка, резервирование и восстановление значительно проще. Кроме этого, он абсолютно бесплатен и не требует существенных аппаратных ресурсов.

Рисунок 1. Консоль 389-DS

389-DS является прямым потомком Netscape Directory Server (NDS). Как следствие, сервер каталогов хорошо документирован и локализован. К его особенностям можно отнести хорошую горизонтальную масштабируемость и разграничение контроля доступа по любому из атрибутов. Поскольку изначально 389-DS предназначен для смешанной инфраструктуры, покажем его применение в сетевой среде, где в качестве рабочих станций используются MS Windows XP (Windows) и PCLinuxOS [2] (Linux), а в качестве сервера – CentOS 7 [3].

Дистрибутив PCLinuxOS разрабатывался для устаревших и слабых компьютеров. В его составе небольшое ядро Linux с большим набором драйверов, включая проприетарные, для широкого спектра оборудования. Установка и настройка предельно проста. Скорость работы соизмерима с MS Windows XP на однопроцессорных системах и существенно превосходит ее на многопроцессорных. С другой стороны, MS Windows XP фактически до сих пор остается самой популярной операционной системой в небольших предприятиях, поскольку требования к оборудованию у этой ОС достаточно скромные.

В качестве сервера используется CentOS 7 – новая редакция хорошо известного дистрибутива на основе RedHat Enterprise Linux 7. Дистрибутив активно развивается, реализованы все современные технологии виртуализации, вышел в 2014 году и будет поддерживаться до 2020 года, что делает его очень привлекательным.

Установка сервера каталогов

Рассмотрим установку на «чистую» CentOS 7 x64 в минимальной конфигурации или в конфигурации GUI-сервера. Для организации сервера каталогов нам потребуются службы NTP, DNS, DHCP и LDAP. Ниже кратко показан процесс последовательной установки и настройки этих сервисов для сервера server.local.ru, с IP-адресом 192.168.1.251 в сети 192.168.1.0/24.

Возможности сервера каталогов 389-DS более чем достаточны для управления пользователями и ресурсами

Установка и настройка NTP

Сервер времени для информационной среды с протоколами безопасности имеет первоочередное значение, поскольку каждый сертификат имеет ограниченное время действия, а запросы «из будущего» могут просто не обслуживаться сетевыми сервисами, поэтому начнем настройку именно с него:

```
#yum update -y; yum upgrade -y; yum install ntp
```

Редактируем файл настройки NTP-сервера /etc/ntp.conf:

```
driftfile /var/lib/ntp/drift
# куда раздаем время
restrict 127.0.0.1
restrict ::1
restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap
# откуда берем время
server ntp1.vniiftri.ru
server ntp2.vniiftri.ru
server ntp3.vniiftri.ru
server 0.europe.ru.pool.ntp.org
server 1.europe.ru.pool.ntp.org
server 2.europe.ru.pool.ntp.org
```

Запускаем сервис и прописываем доступ к его портам:

```
#systemctl enable ntpd
#systemctl start ntpd
#firewall-cmd --permanent --add-service=ntp
#firewall-cmd --reload
```

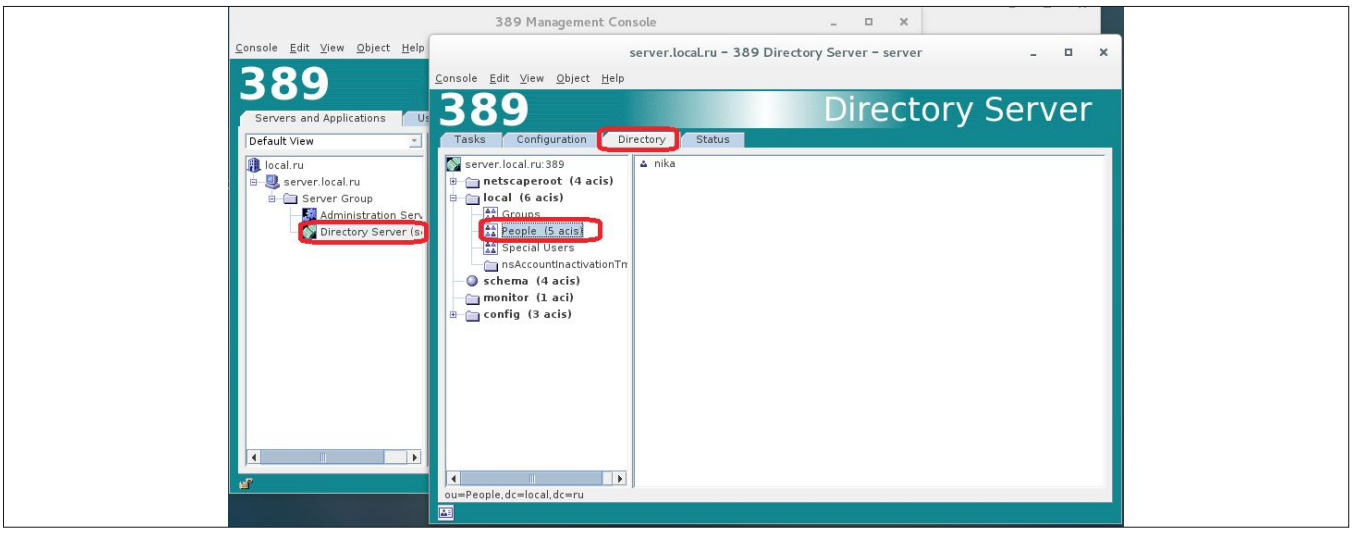
Проверяем работу

```
#ntpq -p 127.0.0.1
```

Установка и настройка DNS и DHCP

Следующие по значимости службы DNS и DHCP. DNS позволят нам разрешать имена рабочих станций в локальной

Рисунок 2. Создание пользователей



сети и интернете, а DHCP позволит донести необходимые настройки для каждой рабочей станции без участия администратора. В нашем случае эти службы развернуты на базе сервера Dnsmasq.

```
#yum install dnsmasq
```

Прописываем полное имя и алиас сервера в /etc/hosts для Dnsmasq:

```
192.168.1.251     server.local.ru    server
```

и добавляем в /etc/resolv.conf строку:

```
nameserver 127.0.0.1
```

Редактируем /etc/dnsmasq.conf:

```
#dns сервера верхнего уровня
server=8.8.8.8
server=192.168.1.1
#сервис dns
domain=local.ru
server=/local.ru/192.168.1.251
server=/0.1.168.192.in-addr.arpa/192.168.1.251
bogus-priv
#dhcp, срок аренды на 7 дней
dhcp-range=192.168.1.50,192.168.1.80,255.255.255.0,7D
# Указываем шлюз по умолчанию
dhcp-option=3,192.168.1.1
# Указываем DNS сервер
dhcp-option=6,192.168.1.251
# Указываем LDAP server и base dn
dhcp-option=95,ldap://192.168.1.251/dc=local,dc=ru
# Указываем сервер времени
dhcp-option=option:ntp-server,192.168.1.251
# расположение файла аренды адресов,
dhcp-leasefile=/var/lib/misc/dnsmasq.leases
# лог файлы
log-queries
log-dhcp
```

Запускаем сервис и прописываем доступ к портам сервисов:

```
#systemctl enable dnsmasq
#systemctl start dnsmasq
#firewall-cmd --permanent --add-service=dhcp
#firewall-cmd --permanent --add-service=dns
#firewall-cmd --reload
```

Проверяем:

Рисунок 3. LDAP-аутентификация

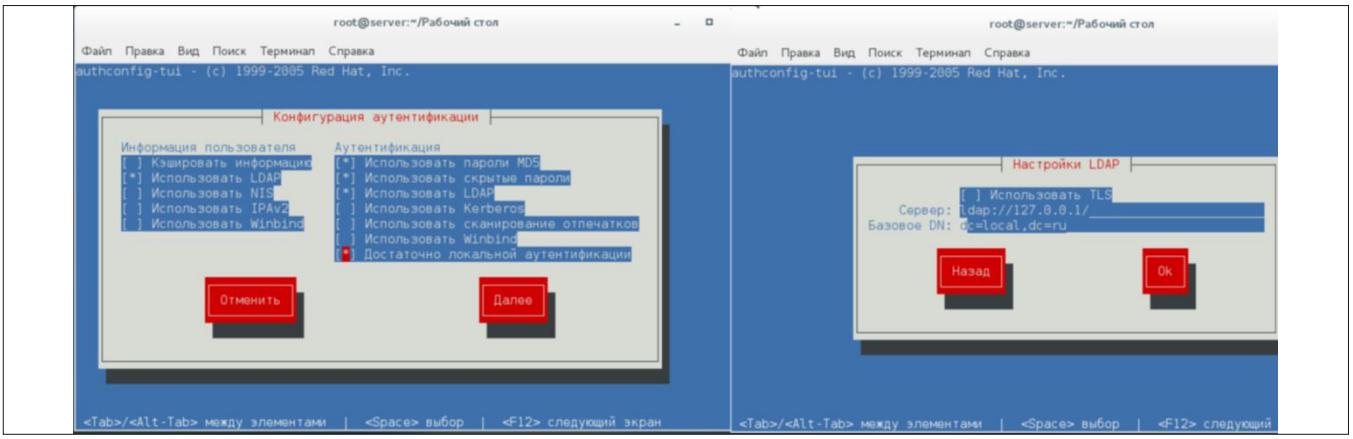
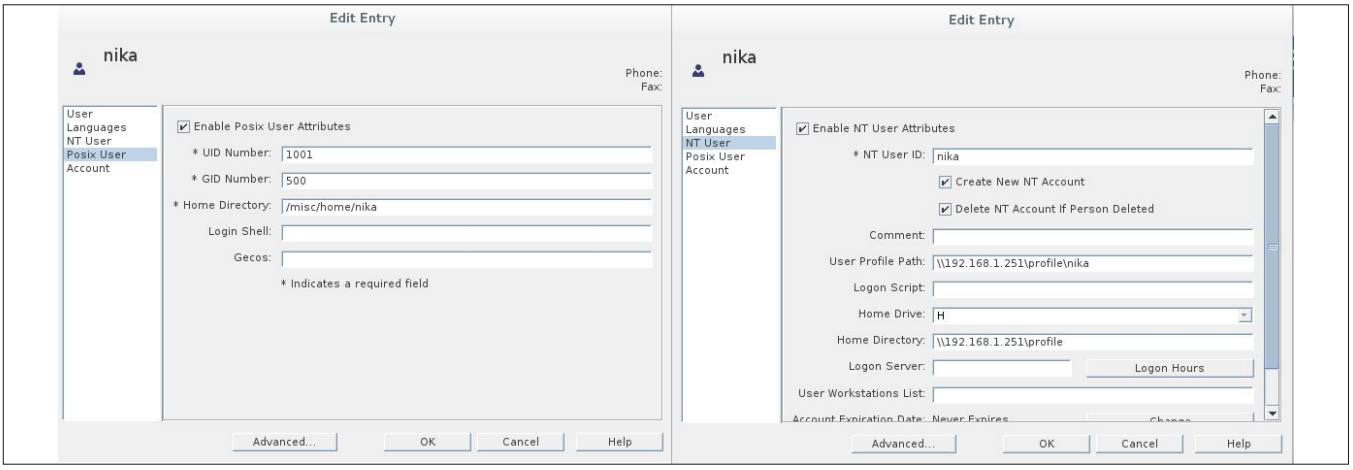


Рисунок 4. Настройка пользователя для смешанной инфраструктуры



```
# dig local.ru
```

Установка и настройка 389-DS

Сервер каталогов 389-DS был переписан командой Red Hat на Java, что, с одной стороны, сделало его кроссплатформенным, а с другой – потребовало дополнительных расширенных системных настроек. Для установки 389-DS на CentOS 7 вносим изменения в системные параметры ядра Linux, добавляя в /etc/sysctl.conf:

```
net.ipv4.tcp_keepalive_time = 300
net.ipv4.ip_local_port_range = 1024 65000
fs.file-max = 64000
```

где переменные:

- **tcp_keepalive_time** – определяет, как быстро закрывать не используемые соединения
- **ip_local_port_range** – расширяет диапазон доступных TCP-портов,
- **file-max** – определяет максимальное количество открытых файловых дескрипторов в системе.

Проверка:

```
#sysctl -p
```

Добавляем в /etc/security/limits.conf:

```
*               soft     nofile          16384
*               hard     nofile          16384
```

Эти параметры определяют максимальное количество файловых дескрипторов (открытых файлов) для отдельного пользователя.

```
#ulimit -n 16384
```

Подключаем репозитории EPEL и REMI, которые содержат готовые пакеты установки 389-DS и Java для его работы:

```
#yum install http://dl.fedoraproject.org/pub/epel/7/ ↵
        x86_64/e/epel-release-7-5.noarch.rpm -y
#yum install http://rpms.famillecollet.com/enterprise/ ↵
        remi-release-7.rpm -y
```

Ставим необходимые пакеты:

- **idm-console-framrwork** – java-платформа для сервера директорий,
- **389-adminutil(-devel)** – набор сервисных скриптов для работы с данными сервера,
- **389-ds-base(-libs,devel)** – сам сервер директорий;
- **389-admin** – сервер администрирования.

```
#yum install idm-console-framrwork 389-adminutil ↵
        389-adminutil-devel 389-admin 389-ds-base ↵
        389-ds-base-libs 389-ds-base-devel -y
```

Непонятно, почему в основной ветке EPEL не размещены пакеты графической консоли. Для их установки придется либо раскомментировать ветку epel-testing в /etc/yum.repos.d, либо установить пакеты по прямой ссылке:

```
#yum localinstall http://ftp.gnome.org/mirror/fedora/epel/ ↵
        testing/7/x86_64/3/ ↵
        389-admin-console-1.1.10-1.el7.noarch.rpm
http://ftp.gnome.org/mirror/fedora/epel/testing/7/x86_64/3/ ↵
        389-admin-console-doc-1.1.10-1.el7.noarch.rpm
http://ftp.gnome.org/mirror/fedora/epel/testing/7/x86_64/3/ ↵
        389-ds-console-doc-1.2.12-1.el7.noarch.rpm
http://ftp.gnome.org/mirror/fedora/epel/testing/7/x86_64/3/ ↵
        389-ds-console-1.2.12-1.el7.noarch.rpm -y
```

Запускаем скрипт первоначальной установки и конфигурирования сервера директорий. Скрипт создаст первоначальную базу с необходимыми атрибутами в /etc/dirsrv.

```
#setup-ds-admin.pl
```

В ходе установки скрипт предложит установить Base dn [dc=local,dc=ru] и задать имя администратора LDAP-сервера [cn=Ldapadmin].

Запуск служб сервера каталогов и допуск к портам сервиса:

```
#systemctl enable dirsrv.target
#systemctl start dirsrv.target
#systemctl enable dirsrv-admin
#systemctl start dirsrv-admin
#firewall-cmd --permanent --add-port=389/tcp
#firewall-cmd --permanent --add-port=636/tcp
#firewall-cmd --permanent --add-port=9830/tcp
#firewall-cmd --reload
```

Запускаем консоль управления (см. рис.1):

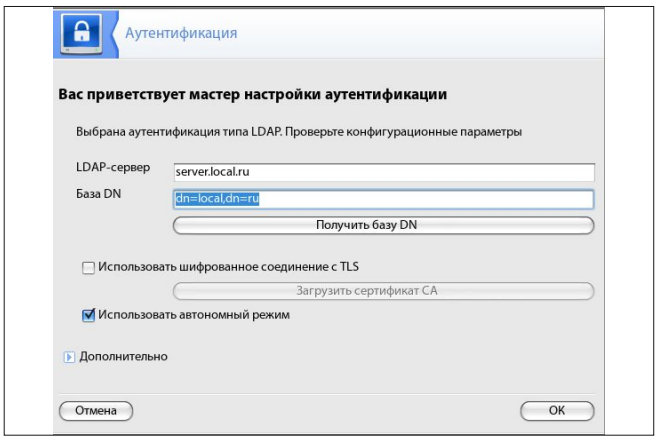
```
#389-console
```

После подключения к серверу можно сразу приступить к созданию учетных записей обычных пользователей в разделе ou=People,dc=local,dc=ru (см. рис. 2) или специальных пользователей в разделе ou=Special Users,dc=local,dc=ru, которые используются для аутентификации сетевых служб или репликации между серверами.

Настройка LDAP-аутентификации на сервере CentOS 7

Для доступа созданных в DS пользователей к ресурсам самого сервера настроим на нем LDAP-аутентификацию.

Рисунок 5. LDAP-аутентификация на PCLinuxOS



Установим пакет pam-ldap и запустим консоль настройки (см. рис. 3):

```
# yum install nss-pam-ldapd
# authconfig-tui
```

Редактируем файл настройки /etc/sysconfig/authconfig, изменяя параметр:

```
FORCELEGACY=yes
```

Этот параметр позволяет авторизовать локальных пользователей без протокола шифрования TLS (в нашем случае мы его не используем).

Настройка LDAP-аутентификации необходима, если мы хотим идентифицировать пользователей DS на самом сервере и разграничить доступ между ними. Теперь при создании объекта от имени одного из пользователей DS система присвоит ему соответствующий идентификатор владельца.

Перемещаемый профиль

Одним из удобств использования сервера каталогов является развертывание перемещаемых профилей пользователей. Благодаря им все личные данные пользователей хранятся на сервере, что упрощает к ним доступ с любого места и резервирование. Поскольку у нас смешанная среда, активируем в профиле пользователя nika закладки Posix User для рабочих станций Linux и NT User для Windows. Домашняя директория для этого пользователя в среде Linux будет /misc/home/nika, а в Windows – /misc/Profiles/nika (см. рис. 4). Безусловно, можно разместить оба профиля в одной директории, но значительно удобнее разнести их по двум независимым каталогам и настроить необходимые ссылки для доступа к данным.

Рисунок 6. Настройка pGina

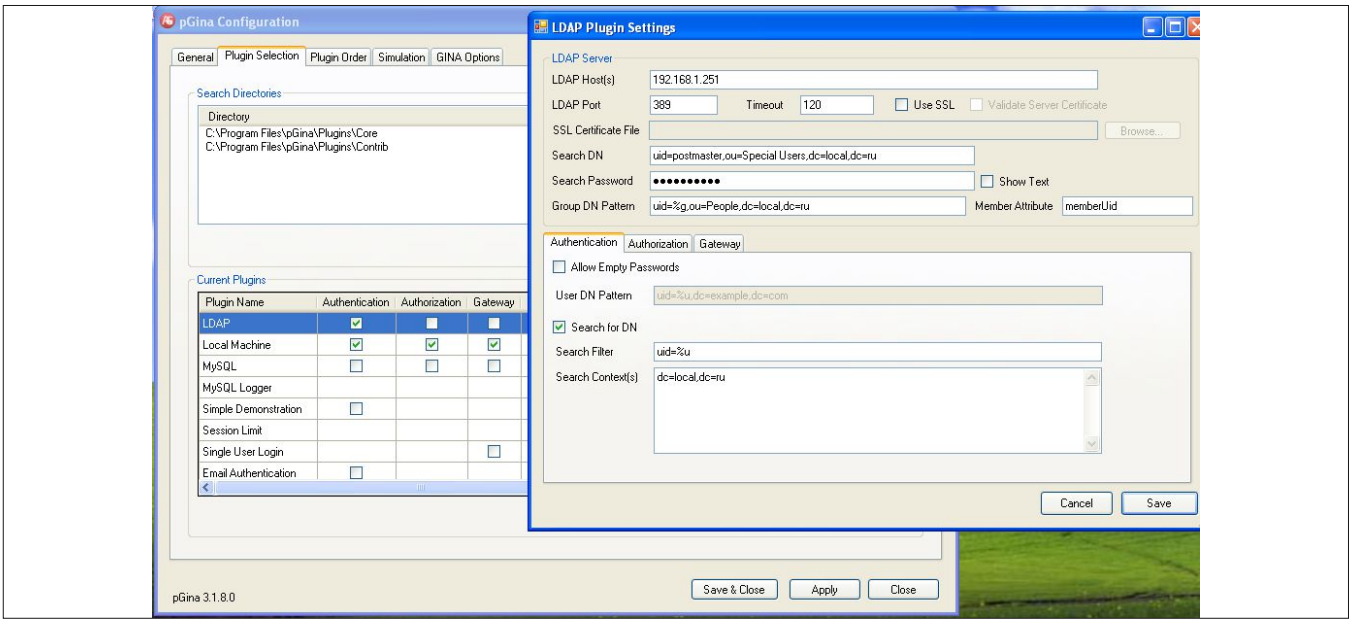
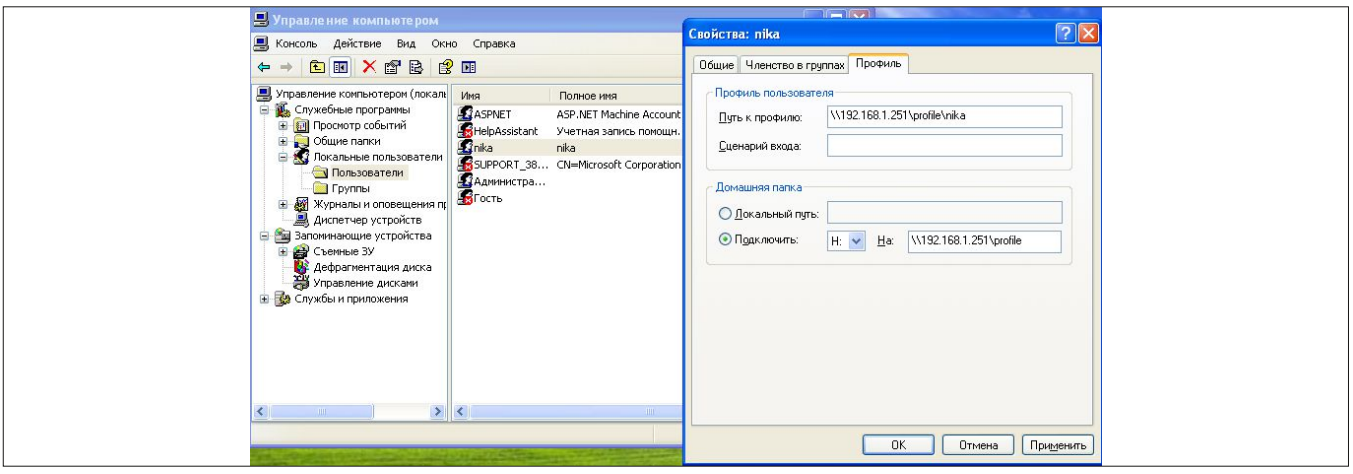


Рисунок 7. Настройки пользователя Windows



Перемещаемый профиль для пользователей Linux

В качестве среды обмена данными между рабочими станциями Linux и сервером используется сетевая файловая система (NFS). Установим службу NFS на сервере:

```
#yum install nfs-server nfs-utils
```

Выполним запуск служб NFS и предоставим необходимый допуск к портам:

```
#systemctl enable rpcbind
#systemctl start rpcbind
#systemctl enable nfs-server
#systemctl start nfs-server
#firewall-cmd --permanent --add-service=nfs
#firewall-cmd --permanent --add-service=rpc-bind
#firewall-cmd --reload
```

Экспортируем каталог misc для хранения профилей пользователей.

Добавим в /etc/exporfs:

```
/misc 192.168.1.0/24(rw,no_root_squash,sync)
#exportfs -a
```

Проверяем:

```
# exportfs
```

На стороне рабочей станции под управлением Linux настроим автоматическое монтирование. Добавим в /etc/autofs/auto.master строку:

```
/misc    /etc/autofs/auto.misc --timeout=10
```

Добавим в /etc/autofs/auto.misc:

```
home      -fstype=nfs4     192.168.1.251:/misc
```

Настраиваем аутентификацию на PCLinuxOS: идем в «Система → Аутентификация». Выбираем LDAP-аутентификацию и задаем параметры LDAP-сервера и Base DN (см. рис. 5).

Теперь при подключении пользователя через 389-DS служба autofs автоматически смонтирует каталог /misc с сервера, в котором будет создан профиль пользователя.

Перемещаемый профиль для пользователей Windows

В качестве среды обмена данными между рабочими станциями Windows XP и сервером удобно использовать протокол SMB. Для этого установим службу SAMBA на сервер CentOS 7:

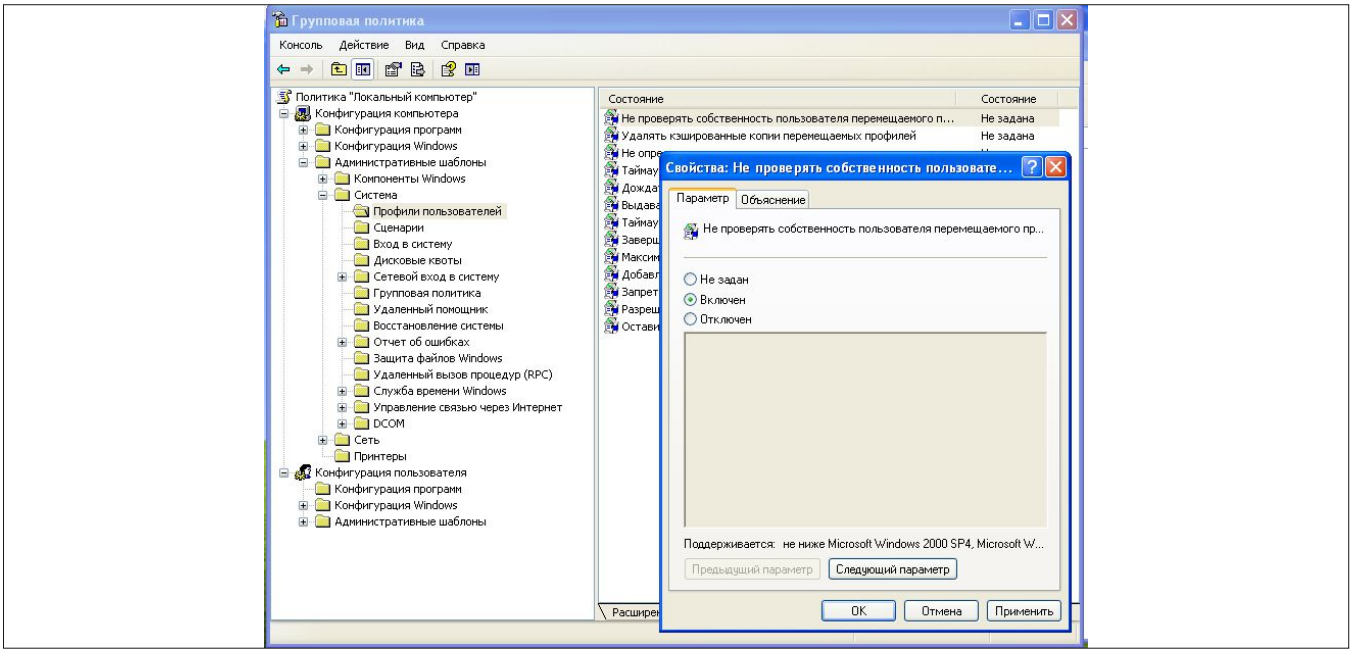
```
#yum install samba
```

Для перемещаемых профилей выберем директорию /misc/profile и расшарим ее для всех пользователей. Редактируем конфигурационный файл /etc/samba/smb.conf:

```
[global]
workgroup = WORKGROUP
server string = Samba Server %v
netbios name = server
security = user
map to guest = bad user
guest account = nobody
log file = /var/log/samba/log.smbd

[profile]
path = /misc/profile
browsable =no
create mask = 0700
writable = yes
guest ok = yes
read only = no
public = yes
```

Рисунок 8. Настройка параметров Windows



Создадим указанный в настройках каталог:

```
#mkdir -p /misc/profile
```

Проверяем конфигурацию Samba:

```
#testparm
```

Запускаем службы и предоставляем необходимый доступ к портам:

```
#systemctl enable smb
#systemctl start smb
#systemctl enable nmb
#systemctl start nmb
#firewall-cmd --permanent --add-service=samba
#firewall-cmd -reload
```

Для аутентификации пользователей Windows в 386-DS, устанавливаем на рабочих станциях Windows pGina [4] и создаем специального пользователя postmaster в разделе ou=Special Users,dc=local,dc=ru. От имени postmaster будет происходить поиск пользователей в дереве DS. После установки и перезапуска рабочей станции Windows заходим в консоль конфигурирования pGina на закладку LDAP Plugin Settings. На этой закладке необходимо указать имя или IP-адрес LDAP-сервера и созданного пользователя uid=postmaster,ou=Special Users,dc=local,dc=ru,. На закладке Group DN Pattern необходимо указать шаблон поиска uid=%g,ou=People,dc=local,dc=ru (см. рис. 6).

При первом входе пользователя на рабочую станцию Windows автоматически будет создана локальная учетная запись с настройками, прописанными на закладке NT User на сервере директорий (см. рис. 7).

Если настройки перемещаемого профиля не сработали при первоначальном входе и сформировался локальный профиль пользователя, измените через консоль безопасности Windows (gpedit.msc) параметр «Не проверять собственность пользователя перемещаемого профиля» (см. рис. 8).

При следующем входе на рабочую станцию Windows на сервере в каталоге /misc/profiles/nika будет создан сетевой профиль пользователя.

Резервирование и восстановление 389-DS

Структура сервера каталогов 389-DS достаточно проста и показана на рис. 9. В качестве системы хранения используется база данных Berkeley DB, работать с которой умеет огромное количество программ.

Для обычного резервирования достаточно воспользоваться командой из состава утилит 389-DS:

```
#/usr/lib64/dirsrv/slapd-ldap/db2bak
```

Бекап будет размещен в каталоге /var/lib/dirsrv/slapd-serv/bak в формате ГГГГ_ММ_ДД_ЧЧ_ММ_СС. Эту же операцию можно выполнить через консоль (см. рис. 10).

Для восстановления базы данных воспользуйтесь командой из состава утилит:

```
#/usr/lib64/dirsrv/slapd-ldap/bak2db ↵
    /var/lib/dirserver/slapd-server/bak/{ГГГГ_ММ_ДД_ЧЧ_ММ_СС}
```

или консолью.

• • •

Возможности сервера каталогов 389-DS достаточны для управления пользователями и ресурсами небольшого предприятия. Его администрированием может заниматься практически любой сотрудник, без отрыва от своей основной деятельности. Возможность синхронизации между серверами позволяет делать распределенные системы практически любой сложности, а возможность интеграция 389-DS с MS Active Directory через плагин Windows Password Synchronization [5] позволяет выполнить незаметный перевод уже существующих пользователей с MS AD на DS. EOF

[1] Официальный сайт 389-DS – http://directory.fedoraproject.org.
[2] Русское сообщество PCLinuxOS – http://pclinuxos.su.
[3] Официальный сайт CentOS – https://www.centos.org.
[4] Официальный сайт pGina – http://pgina.org.
[5] Плагин интеграции с MS Active Directory – http://directory.fedoraproject.org/docs/389ds/download.html#windows-password-synchronization.

Ключевые слова: сервер каталогов, Linux, CentOS, 389-DS.

Рисунок 9. Структура 389-DS

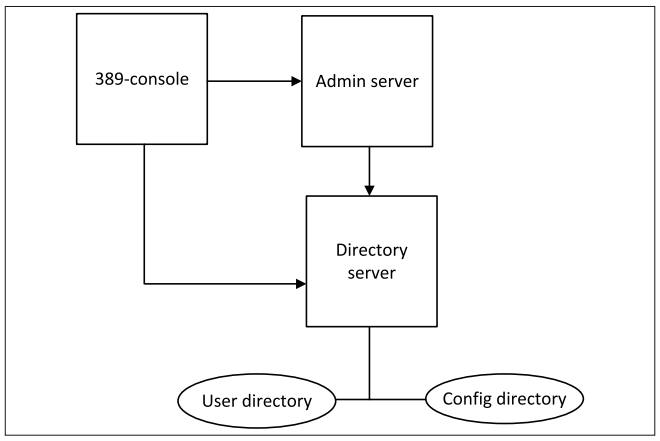
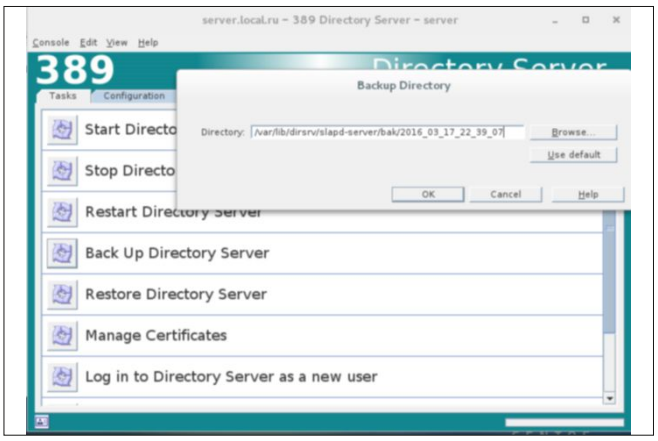


Рисунок 10. Бекап сервера через консоль



Визитка

Александр Пичкасов,
сертифицированный специалист Microsoft, VMware, apichkasov@gmail.com

Расширенная поддержка
почтовых баз Microsoft Exchange 2013

Навыки использования утилит обслуживания и продуманная схема распределения данных определят эффективные действия администраторов в различных аварийных ситуациях

При эксплуатации любой системы возможны сбои, связанные с выходом из строя оборудования. Избежать масштабных катастрофических последствий поможет заложенная на этапе планирования устойчивость системы. Кроме этого, не менее важный фактор минимизации ущерба от сбоя – готовность администраторов к ликвидации последствий, а также регулярность и тщательность выполнения плановых операций по обслуживанию. Для этого ИТ-персонал должен уметь в полном объеме пользоваться инструментами управления и поддержки. К таким инструментам в первую очередь относится штатная утилита eseutil, а также командлеты тестирования и, возможно, сторонние утилиты.

Возможности утилиты eseutil

С помощью eseutil администратор может выполнять операции по обслуживанию почтовых баз Exchange как в онлайновом, так и в оффлайновом режиме. Традиционно утилита используется для управления отключенной базой, в частности для получения информации об ее состоянии.

Для конкретизации примеров и приведения иллюстраций с ожидаемым выводом командлетов определимся с конфигурацией тестовой системы. Предполагается, что представленные фрагменты скриптов выполняются последовательно в едином сеансе командной консоли Exchange, запущенной на сервере с установленными ролями Mailbox и Client Access Exchange 2013 SP1, в среде Windows Server 2012 R2. Учетная запись, от имени которой выполняется сеанс, должна обладать разрешениями Organization Management в системе Exchange и разрешениями локального администратора в операционной системе, командная консоль должна выполняться с административными полномочиями.

Для реализации последующих примеров необходимо подготовить систему аналогично [1] и создать тестовый скрипт доступа к почтовому ящику, представленный в статье [2]. Листинги доступны по ссылкам [3, 4]. В итоге должно быть выполнено следующее:

- установлены компоненты системы WindowsServer Backup, Hyper-V, модуль PowerShell для управления Hyper-V;
- задействованы службы POP3 для клиентского доступа к почтовому ящику;
- сохранен скрипт доступа к почтовому ящику ViewPOP3Mailbox.ps1;
- организована функция создания тома на виртуальном диске Create-VolumeOnVHD.

Завершающие шаги подготовки – создание томов с файловой системой, почтовой базы, почтового ящика с заполнением его тестовыми сообщениями – представлены в листинге 1.

Листинг 1. Подготовка тестовой среды

```
#Создание каталога для файлов виртуальных дисков
mkdir c:\vhds
#Задание имен файлов виртуальных дисков
#(те же имена используются для каталогов монтирования)
$setname = 'diskAdb','diskAlogs'
foreach ($filename in $setname) {
$newpart = Create-VolumeOnVHD -filenamevhd ↵
         "C:\vhds\$filename.vhd" -sizedisk 5000000000
#Создание каталога и монтирование тома
  mkdir c:\$filename | Out-Null
  mountvol c:\$filename $newpart.AccessPaths[1]
#Удаление пути доступа формата DriveLetter для корректной
#обработки резервного копирования
  mountvol $newpart.AccessPaths[0] /d
}
New-MailboxDatabase -Name "MailboxDB" -Server (hostname) ↵
         -EdbFilePath C:\diskAdb\mailboxdb\mailboxdb.edb ↵
         -LogFolderPath C:\diskAlogs\mailboxdblogs
Mount-Database MailboxDB
$ad0=(Get-AcceptedDomain)[0]
$pass = ConvertTo-SecureString 'Pa$$w0rd' -AsPlainText -Force
New-Mailbox -Name "Augusto" -LastName "Cavalli" ↵
         -Database "MailboxDB" -Password $pass ↵
         -ResetPasswordOnNextLogon $false -DisplayName ↵
         "Augusto Cavalli" -Alias augusto ↵
         -UserPrincipalName "augusto@$ad0"
fsutil file createnew c:\attach5mb.bin 5000000
foreach ($i in 1..5) {Send-MailMessage -SmtpServer ↵
         localhost -From "administrator@$ad0" -To ↵
         "augusto@$ad0" -Subject "Test message n.$i with ↵
         5mb attachment" -Attachments c:\attach5mb.bin}
#Проверка содержимого почтового ящика, при необходимости
#повторить, чтобы убедиться в доставке всех сообщений
.\ViewPOP3Mailbox.ps1 -username augusto -password 'Pa$$w0rd'
```

Первое действие, которое обычно выполняют с помощью утилиты eseutil, – это получение информации об ее состоянии, требуемых и примененных журналах транзакций. Для выделения интересующей информации удобно использовать фильтры, например, командлет Select-String (см. листинг 2).

Листинг 2. Оффлайновый режим использования eseutil

```
#Отключение базы для работы eseutil в оффлайновом режиме
Dismount-Database MailboxDB -Confirm:$false
#Получение информации о состоянии базы и журналах
#транзакций из заголовка файла базы
eseutil /mh c:\diskAdb\mailboxdb\mailboxdb.edb | ↵
         Select-String state,log,last
#Получение информации журнала транзакций, префикс E01 -
#частный случай
eseutil /ml c:\diskAlogs\mailboxdblogs\e01.log
#Проверка целостности журналов транзакций
eseutil /ml c:\diskAlogs\mailboxdblogs\e01
#Получение информации контрольных точек
eseutil /mk c:\diskAlogs\mailboxdblogs\e01.chk
#Получение информации о whitespaces
eseutil /ms c:\diskAdb\mailboxdb\mailboxdb.edb
```

На основе информации о свободном пространстве (whitespaces), а конкретнее по соотношению значений Used и Available (см. рис.1), может быть принято решение о дефрагментации базы. Эта операция также выполняется с помощью eseutil в оффлайновом режиме. Для сокращения времени за счет исключения излишнего копирования данных будет разумным использовать отдельный диск для временной базы и последующую замену файлов выполнять перемонтированием диска. Такой подход дает дополнительное преимущество – возможность возврата предыдущего состояния при неудаче дефрагментации за счет обратной подмены диска. Порядок действий и используемые ключи eseutil представлены в листинге 3 (см. рис. 2).

Листинг 3. Оффлайновая дефрагментация базы

```
mkdir c:\tempdb
#Подключение диска для временного файла базы
$newpart = Create-VolumeOnVHD -filenamevhd ↵
         "C:\vhds\diskAtempdb.vhd" -sizedisk 5000000000
#Монтирование тома
mountvol c:\tempdb $newpart.AccessPaths[1]
#Удаление пути доступа
mountvol $newpart.AccessPaths[0] /d
#Создание каталога для временного файла
mkdir c:\tempdb\mailboxdb
#Выполнение дефрагментации с указанием временного файла базы,
#без замены исходного файла
eseutil /d C:\diskAdb\mailboxdb\mailboxdb.edb /t ↵
         C:\tempdb\mailboxdb\mailboxdb.edb /p
#Сравнение размеров файлов базы и использования свободного
#пространства
eseutil /ms c:\diskAdb\mailboxdb\mailboxdb.edb
eseutil /ms c:\tempdb\mailboxdb\mailboxdb.edb
dir C:\diskAdb\mailboxdb\mailboxdb.edb, C:\tempdb\↵
         mailboxdb\mailboxdb.edb | ft fullname,length -a
#Перемонтирование дисков
$volA = Get-Volume -FilePath C:\diskAdb
$volT = Get-Volume -FilePath C:\tempdb
mountvol c:\diskAdb /d
mountvol c:\tempdb /d
mountvol c:\diskAdb $volT.ObjectId
Mount-Database mailboxdb
#Возврат предыдущего состояния
Dismount-Database MailboxDB -Confirm:$false
mountvol c:\diskAdb /d
mountvol c:\diskAdb $volA.ObjectId
```

Другие возможности eseutil позволяют выполнить восстановление базы данных в различных ситуациях. Для этого используются ключи /r – для «мягкого» и /p – для «жесткого» восстановления.

«Мягкое» восстановление используется в случае сохранности как файла базы, так и журналов транзакций, именно поэтому после ключа /r требуется указание трехсимвольного префикса журнала в формате EXX. Отмечу, что в заголовке файла базы префикс не фигурирует, и связаны они через свойства объекта Mailbox Database службы каталогов. Соответственно, значение префикса может быть найдено среди свойств почтовой базы в выводе командлета Get-MailboxDatabase. Запуск eseutil с ключом /r вызывает применение недостающих журналов транзакций, выявленных в результате сопоставления информации в заголовке файла базы, содержимого файла контрольных точек и собственно наличия и целостности журналов. Пути к файлам могут (и должны, в случае отличия!) быть указаны с помощью опций соответственно /d, /s и /l. Использование при этом дополнительных опций (/i, /a) дает возможность игнорировать некоторые ошибки несоответствия.

Использование eseutil с ключом /p подразумевает отсутствие журналов транзакций. В этом случае анализируется только состояние файла базы на низком уровне, а именно наличие и целостность указателей B+ деревьев. Исправление состояния выполняется путем удаления указателей, для которых было выявлено несоответствие, что, естественно, приводит к потере части данных. Нужно отметить, что в Exchange 2013 большинство исправимых состояний почтовых баз может быть скорректировано встроенными процедурами системы, выполняемыми в фоновом режиме или отрабатываемыми при запросе монтирования базы. Необоснованное применение eseutil в режиме «жесткого» восстановления может привести к потере данных, поэтому его следует использовать только как последнее средство для восстановления той части данных, которая не может быть получена из резервной копии. Пример такого использования представлен далее, в сценарии восстановления после сбоя.

Некоторые операции eseutil могут быть выполнены в онлайновом режиме, при этом используется система теневого копирования, что должно быть прямо указано в ключах командной строки. В листинге 4 представлены варианты онлайновых операций: запрос информации, проверка целостности базы (с генерацией отчета) и создание копии базы. Стоит отметить, что такая копия может служить дополнительным экземпляром архива, выполненного в особом режиме, именуемом «копирующей архивацией», и независимого от основной последовательности архивных копий.

Листинг 4. Онлайновый режим работы eseutil

```
#Монтирование базы
Mount-Database mailboxdb
#Получение информации из заголовка подключенной почтовой
#базы, с применением журналов транзакций к теневой копии
eseutil /mh C:\diskAdb\mailboxdb\mailboxdb.edb /vssrec e01 ↵
         C:\diskAlogs\mailboxdblogs
#Создание каталога для временных файлов
mkdir c:\copydb
#Проверка целостности базы в онлайновом режиме
eseutil.exe /g C:\diskAdb\mailboxdb\mailboxdb.edb /t ↵
         C:\copydb\mailboxdb.edb /f mailboxdb_report ↵
```

```
        /vssrec e01 C:\diskAlogs\mailboxdblogs ↵
        /vsssystempath C:\diskAlogs\mailboxdblogs
#Создание копии базы с применением журналов транзакций
eseutil /y C:\diskAdb\mailboxdb\mailboxdb.edb /d C:\copydb\↵
        mailboxdb.edb /vssrec E01 C:\diskAlogs\mailboxdblogs
#Получение информация о копии базы
eseutil /mh C:\copydb\mailboxdb.edb
```

Далее разберем сценарии аварийных ситуаций, которые иллюстрируют обоснованность распределения файлов базы по отдельным физическим дискам. Дело в том, что вероятность одновременного выхода из строя двух дисков чрезвычайно мала, обычно аварии связаны с неисправностью одного из дисков. Естественно, что для реальной конфигурации администратор должен позаботиться о подключении дисков к отдельным контроллерам. Исходя из этого можно выделить два сценария аварийных ситуаций:

- неисправность диска с файлом базы;
- неисправность диска с журналами транзакций.

Для реализации примеров восстановления предварительно создадим резервную копию базы, после чего отправим несколько дополнительных сообщений (см. листинг 5).

Листинг 5. Создание резервной копии

```
#Создание резервной копии
$wbpolicy = New-WBPolicy
$wbbackuptarget = New-WBBackupTarget -VolumePath c:
$wbvolume1 = Get-WBVolume -VolumePath C:\diskAdb
$wbvolume2 = Get-WBVolume -VolumePath C:\diskAlogs
Add-WBBackupTarget -Policy $wbpolicy -Target $wbbackuptarget
Add-WBVolume -Policy $wbpolicy -Volume $wbvolume1
Add-WBVolume -Policy $wbpolicy -Volume $wbvolume2
Set-WBVssBackupOption -Policy $wbpolicy –VssFullBackup
Start-WBBackup -Policy $wbpolicy
#Отправка сообщений после резервного копирования
foreach ($i in 6..10) {Send-MailMessage -SmtpServer localhost ↵
         -From "administrator@$ad0" -To "augusto@$ad0" ↵
         -Subject "Message n.$i with 5mb attachment received ↵
         after backup" -Attachments c:\attach5mb.bin}
#Проверка содержимого почтового ящика, при необходимости
#повторить, чтобы убедиться в доставке всех сообщений
.\ViewPOP3Mailbox.ps1 -username augusto -password 'Pa$$w0rd'
```

Сценарий первый. Выход из строя диска с файлом базы

Этот вариант подразумевает наличие исправной резервной копии, которая содержит данные, актуальные на некоторый момент в прошлом. Оставшийся исправным диск сохраняет в журналах транзакций обновления данных, произошедшие с момента резервного копирования. Таким образом, для восстановления всей информации требуется:

- сохранить существующие файлы журналов транзакций;
- восстановить тома с файлами базы из резервной копии;
- скопировать сохраненные журналы транзакций;
- смонтировать восстановленную базу.

Нужно отметить, что критической ошибкой будет попытка монтирования почтовой базы сразу после восстановления из резервной копии, до копирования журналов транзакций. Такие действия создадут их альтернативную последовательность, что приведет к потере данных, так как сделает сохраненные журналы недействительными. В листинге 6 имитируется выход из строя физического диска и отправляются дополнительные тестовые сообщения. Попытка их доставки вызовет немедленное отключение почтовой базы, потери данных при этом не произойдет, поскольку сообщения сохранятся в очередях транспортной службы.

Листинг 6. Имитация аварии диска с файлом базы

```
#*Имитация* выхода из строя физического диска
Dismount-DiskImage -ImagePath C:\vhds\diskAdb.vhd
#Отправка сообщений после сбоя диска
foreach ($i in 11..15) {Send-MailMessage -SmtpServer localhost ↵
         -From "administrator@$ad0" -To "augusto@$ad0" ↵
         -Subject "Message n.$i with 5mb attachment received ↵
         after crash disk" -Attachments c:\attach5mb.bin}
#Контроль состояния базы, для тестовой среды при
#необходимости повторить до отключения базы
Get-MailboxDatabase mailboxdb -Status | ft name,mounted -a
```

Порядок действий при восстановлении базы описан в листинге 7. Отмечу, что в этой последовательности не предусматривается выполнение «мягкого» восстановления базы (eseutil /r), поскольку в большинстве случаев достаточно встроенных операций, выполняемых при монтировании базы. Последний шаг этого фрагмента позволяет убедиться, что все сообщения успешно восстановлены.

Листинг 7. Восстановление базы для первого сценария

```
#Для исключения нежелательных попыток монтирования базы
#подсистемой управляемой доступности (Managed Availability)
Set-MailboxDatabase mailboxdb -MountAtStartup $false
#Некорректное отключение точек монтирования требует
#повторного создания каталогов
cmd /c rmdir c:\diskAdb
mkdir c:\diskAdb
#Перемонтирование диска с журналами транзакций для сохранения
$vol = Get-Volume -FilePath C:\diskAlogs\
mkdir c:\newdblogs
mountvol C:\newdblogs $vol.ObjectId
```

Рисунок 1. Пример вывода информации о свободном пространстве

```
Enumerated 66 Tables ( 34 Internal Trees, 6 Long Value Trees, 27 Secondary Indices )
Pages 4094 ( 913 Used (22.3%), 3181 Available (77.7%) )
Note: This database is over 20% empty, an offline defragmentation can be used
to shrink the file.

Operation completed successfully in 0.953 seconds.
[PS] C:\vhds>_
```

Рисунок 2. Размеры файлов и состояние базы до и после дефрагментации

```
Enumerated 52 Tables ( 30 Internal Trees, 5 Long Value Trees, 23 Secondary Indices )
Pages 4094 ( 886 Used (21.6%), 3208 Available (78.4%) )
Note: This database is over 20% empty, an offline defragmentation can be used
to shrink the file.

Operation completed successfully in 0.828 seconds.
[PS] C:\vhds>eseutil /ms c:\diskAdb\mailboxdb\mailboxdb.edb_

Enumerated 51 Tables ( 30 Internal Trees, 5 Long Value Trees, 23 Secondary Indices )
Pages 1534 ( 885 Used (57.7%), 649 Available (42.3%) )

Operation completed successfully in 0.500 seconds.
[PS] C:\vhds>eseutil /ms c:\tempdb\mailboxdb\mailboxdb.edb_

FullName                                  Length
--------                                  ------
C:\diskAdb\mailboxdb\mailboxdb.edb 259981312
C:\tempdb\mailboxdb\mailboxdb.edb   58720256

[PS] C:\vhds>dir C:\diskAdb\mailboxdb\mailboxdb.edb, C:\tempdb\mailboxdb\mailboxdb.edb |
     ft fullname,length -a_
```

```
mountvol C:\diskAlogs /d
##Подключение нового диска для файла базы
$newpart = Create-VolumeOnVHD -filenamevhd ↵
         "C:\vhds\diskBdb.vhd" -sizedisk 5000000000
#Монтирование тома
mountvol c:\diskAdb $newpart.AccessPaths[1]
#Удаление пути доступа для корректной обработки системой
#резервного копирования
mountvol $newpart.AccessPaths[0] /d
#Подключение нового диска для журналов транзакций
$newpart = Create-VolumeOnVHD -filenamevhd ↵
         "C:\vhds\diskBlogs.vhd" -sizedisk 5000000000
#Монтирование тома
mountvol c:\diskAlogs $newpart.AccessPaths[1]
#Удаление пути доступа для корректной обработки
#резервного копирования
mountvol $newpart.AccessPaths[0] /d
##Восстановление томов из резервной копии
#Получение набора данных последней резервной копии
$wbbackupset = (Get-WBBackupSet)[(Get-WBBackupSet).count-1]
#Просмотр содержимого резервной копии
$wbbackupset
#Новому тому присвоен отличный от исходного идентификатор,
#поэтому восстановление в исходное местоположение
#невозможно, требуется указание целевого тома
$targetvolume = Get-WBVolume -VolumePath C:\diskAdb
Start-WBVolumeRecovery -BackupSet $wbbackupset ↵
         -VolumeInBackup $wbbackupset.Volume[0] ↵
         -RecoveryTargetVolume $targetvolume -Force
#Так же восстанавливается том с журналами транзакций
$targetvolume = Get-WBVolume -VolumePath C:\diskAlogs
Start-WBVolumeRecovery -BackupSet $wbbackupset ↵
         -VolumeInBackup $wbbackupset.Volume[1] ↵
         -RecoveryTargetVolume $targetvolume -Force
#Копирование сохраненных журналов транзакций,
#префикс e01 - частный случай
Copy-Item C:\newdblogs\mailboxdblogs\e01*.log ↵
         C:\diskAlogs\mailboxdblogs
#Монтирование базы
Mount-Database mailboxdb
#Убедившись в монтировании базы, возвращаем опцию.
Set-MailboxDatabase mailboxdb -MountAtStartup $true
#Проверка содержимого почтового ящика
.\ViewPOP3Mailbox.ps1 -username augusto -password 'Pa$$w0rd'
```

Сценарий второй. Выход из строя диска с журналами транзакций

В такой ситуации можно предположить, что данные завершенных транзакций помещены системой в файл почтовой базы, и для восстановления работоспособности достаточно ее смонтировать. К сожалению, практика показывает, что состояние файла базы и отсутствие актуальных журналов транзакций не всегда позволяют смонтировать базу без потери информации и также приводят к невозможности выполнить резервное копирование на уровне приложения, но будет разумным сохранить в резерве копию базы на файловом уровне. Чаще всего после таких событий потребуется выполнить «жесткое» восстановление с помощью утилиты eseutil. В этом сценарии восстановление базы сложится из следующих этапов:

- сохранение файла обновленной базы;
- восстановление тома с файлами базы из резервной копии;
- монтирование базы;
- создание базы восстановления на основе сохраненного файла;
- создание запроса на восстановление данных из обновленной базы и его выполнение.

В листинге 8 создается новая резервная копия и отправляется дополнительная порция тестовых сообщений. Поскольку в предыдущих фрагментах были заменены диски и тома получили другие идентификаторы, для уверенности в результате стоит заменить объекты WBVolume в политике резервного копирования либо, как в примере, создать новую политику.

Листинг 8. Повторное создание резервной копии

```
#Создание резервной копии
$wbpolicy = New-WBPolicy
$wbbackuptarget = New-WBBackupTarget -VolumePath c:
$wbvolume1 = Get-WBVolume -VolumePath C:\diskAdb
$wbvolume2 = Get-WBVolume -VolumePath C:\diskAlogs
Add-WBBackupTarget -Policy $wbpolicy -Target $wbbackuptarget
Add-WBVolume -Policy $wbpolicy -Volume $wbvolume1
Add-WBVolume -Policy $wbpolicy -Volume $wbvolume2
Set-WBVssBackupOption -Policy $wbpolicy –VssFullBackup
Start-WBBackup -Policy $wbpolicy
#Отправка дополнительной порции сообщений после резервного
#копирования
foreach ($i in 16..20) {Send-MailMessage -SmtpServer localhost ↵
         -From "administrator@$ad0" -To "augusto@$ad0" ↵
         -Subject "Message n.$i with 5mb attachment received ↵
         after second backup" -Attachments c:\attach5mb.bin}
#Проверка содержимого почтового ящика, при необходимости
#повторить, чтобы убедиться в доставке всех сообщений
.\ViewPOP3Mailbox.ps1 -username augusto -password 'Pa$$w0rd'
```

Листинг 9. Имитация аварии диска с журналами транзакций

```
#*Имитация* выхода из строя физического диска
Dismount-DiskImage -ImagePath C:\vhds\diskBlogs.vhd
#Отправка дополнительной порции сообщений после сбоя диска
foreach ($i in 21..25) {Send-MailMessage -SmtpServer localhost ↵
         -From "administrator@$ad0" -To "augusto@$ad0" ↵
         -Subject "Message n.$i with 5mb attachment received ↵
         after crash transactions logs's disk" ↵
         -Attachments c:\attach5mb.bin}
#Контроль состояния базы, в тестовой среде
#при необходимости повторить до отключения базы
Get-MailboxDatabase mailboxdb -Status | ft name,mounted –a
```

Командлеты в листинге 9 выполняют имитацию выхода из строя диска и отправку следующей порции сообщений. Попытка их доставки приведет к отключению базы. В листинге 10 представлены шаги по восстановлению базы в этом варианте. Актуальность данных при этом гарантируется только на момент создания резервной копии, поскольку применение утилиты eseutil с ключом /p обычно приводит к потере части сообщений.

Листинг 10. Восстановление базы для второго сценария

```
#Для исключения нежелательных попыток монтирования базы
#подсистемой управляемой доступности (Managed Availability)
Set-MailboxDatabase mailboxdb -MountAtStartup $false
#Некорректное отключение точек монтирования требует
#повторного создания каталогов
cmd /c rmdir c:\diskAlogs
mkdir c:\diskAlogs
#Перемонтирование диска с обновленной базой для сохранения
$vol = (Get-Volume -FilePath C:\diskAdb\).ObjectId
mkdir c:\newdb
mountvol C:\newdb $vol
mountvol C:\diskAdb /d
#Подключение нового диска для файла базы
$newpart = Create-VolumeOnVHD -filenamevhd ↵
         "C:\vhds\diskCdb.vhd" -sizedisk 5000000000
#Монтирование тома
mountvol c:\diskAdb $newpart.AccessPaths[1]
#Удаление пути доступа
mountvol $newpart.AccessPaths[0] /d
#Подключение нового диска для журналов транзакций
```

```
$newpart = Create-VolumeOnVHD -filenamevhd ↵
        "C:\vhds\diskClogs.vhd" -sizedisk 5000000000
#Монтирование тома
mountvol c:\diskAlogs $newpart.AccessPaths[1]
#Удаление пути доступа
mountvol $newpart.AccessPaths[0] /d
##Восстановление томов из резервной копии
#Получение набора данных последней резервной копии
$wbbackupset = (Get-WBBackupSet)[(Get-WBBackupSet).count-1]
#Просмотр содержимого резервной копии
$wbbackupset
#Новому тому присвоен отличный от исходного идентификатор,
#поэтому восстановление в исходное местоположение
#невозможно, требуется указание целевого тома
$targetvolume = Get-WBVolume -VolumePath C:\diskAdb
Start-WBVolumeRecovery -BackupSet $wbbackupset ↵
        -VolumeInBackup $wbbackupset.Volume[0] ↵
        -RecoveryTargetVolume $targetvolume -Force
#Так же восстанавливается том с журналами транзакций
$targetvolume = Get-WBVolume -VolumePath C:\diskAlogs
Start-WBVolumeRecovery -BackupSet $wbbackupset ↵
        -VolumeInBackup $wbbackupset.Volume[1] ↵
        -RecoveryTargetVolume $targetvolume -Force
#Монтирование базы
Mount-Database mailboxdb
#Возврат опции автоматического монтирования.
Set-MailboxDatabase mailboxdb -MountAtStartup $true
#Создание базы восстановления с существующим файлом базы
New-MailboxDatabase -Name "RecoveryDB" -Server (hostname) ↵
        -EdbFilePath C:\newdb\mailboxdb\mailboxdb.edb ↵
        -LogFolderPath C:\newdb\mailboxdblogs -Recovery
#Исправление базы (требуется подтверждение в графическом
#диалоговом окне)
eseutil /p c:\newdb\mailboxdb\mailboxdb.edb /t ↵
        C:\vhds\tempdb.edb
#Монтирование базы восстановления
Mount-Database RecoveryDB
#Создание запроса на восстановление данных
$mbx = Get-MailboxStatistics -Database RecoveryDB | ? ↵
        {$_.displayname -eq "augusto cavalli"}
New-MailboxRestoreRequest -SourceDatabase RecoveryDB ↵
        -SourceStoreMailbox $mbx ↵
        -TargetMailbox augusto@ arctic.ocean ↵
        -ConflictResolutionOption KeepLatestItem ↵
        -BadItemLimit 5
#Получение информации о состоянии запроса
Get-MailboxRestoreRequest
#Если запрос долгое время остается в состоянии Queued,
#необходимо выполнить процедуру [5]
#Проверка содержимого почтового ящика
.\ViewPOP3Mailbox.ps1 -username augusto -password 'Pa$$w0rd'
```

Сценарий третий. Удаление объекта AD, представляющего почтовую базу

Достаточно редкий, однако вероятный сценарий, при котором файлы базы остаются в исправном состоянии, но удаляется соответствующий им объект в службе каталогов. Такая ситуация может сложиться в результате ошибочных действий (случайного удаления объекта) либо в случае восстановления устаревшей резервной копии службы каталогов.

Для иллюстрации такого варианта прежде всего необходимо установить модуль powershell для управления AD. Это даст возможность имитировать случайное удаление объекта с помощью командлета (см. листинг 11).

Листинг 11. Имитация случайного удаления объекта Mailbox Database

```
#Установка модуля PowerShell для управления AD
dism /online /enable-feature:activedirectory-powershell
#Получение свойств почтовой базы
$mdb=Get-MailboxDatabase MailboxDB
#*Имитация* случайного удаления объекта службы каталогов
Remove-ADObject -Identity $mdb.DistinguishedName ↵
        -Recursive -Confirm:$false
```

Этапы восстановления в данном сценарии следующие:

- создание почтовой базы с произвольными параметрами;
- замена путей файлов для созданной базы в режиме ConfigurationOnly для соответствия имеющимся файлам;
- задание параметра Database для «осиротевших» почтовых ящиков;
- монтирование базы.

Последовательность командлетов представлена в листинге 12. Нужно помнить, что создается новый объект Mailbox Database, поэтому при необходимости все параметры должны быть заданы заново. Кроме того, новый объект получит отличные от прежних идентификаторы, поэтому использующие их операции (например, скрипты обслуживания) становятся недействительными и требуют обновления.

Листинг 12. Восстановление базы для третьего сценария

```
#Перезапуск службы Information Store для сброса блокировок
#файлов
Restart-Service msex*is
#Создание почтовой базы с прежним именем и произвольными
#параметрами
New-MailboxDatabase -Name MailboxDB -EdbFilePath ↵
        c:\fakepath\mailboxdb.edb -LogFolderPath ↵
        c:\fakepath -Server (hostname)
#Задание путей, соответствующих реальным файлам базы
Move-DatabasePath mailboxdb -ConfigurationOnly ↵
        -EdbFilePath C:\diskAdb\mailboxdb\mailboxdb.edb ↵
        -LogFolderPath C:\diskAlogs\mailboxdblogs ↵
        -Force -Confirm:$false
#Задание параметра Database для почтовых ящиков
Get-Mailbox | ? {$_.database -eq $null} | Set-Mailbox ↵
        -Database MailboxDB -Force -Confirm:$false
#Монтирование базы
Mount-Database MailboxDB
#Проверка доступа
.\ViewPOP3Mailbox.ps1 -username augusto -password 'Pa$$w0rd'
```

Необходимо отметить, что, несмотря на разделение файлов базы по отдельным дискам, в одном из рассмотренных аварийных сценариев возможна потеря части данных, и это подтверждается результатами демонстрационных скриптов. Обеспечить полную сохранность данных можно в случае использования групп доступности почтовых баз (DAG) – встроенного средства High Availability системы Exchange. EOF

[1] Пичкасов А. Управление и поддержка почтовых баз Exchange 2013. // «Системный администратор», №1-2, 2016 г. – С. 46-51 (http://samag.ru/archive/article/3118).

[2] Пичкасов А. Управление почтовыми ящиками Exchange 2013. // «Системный администратор», №12, 2015 г. – С. 32-36 (http://samag.ru/archive/article/3091).

[3] Листинги скриптов подготовки системы –http://samag.ru/uploads/artfiles/1455611481source01-02%28158-159%29.txt.

[4] Листинг скрипта доступа к почтовому ящику – http://samag.ru/uploads/artfiles/1450081959source12%28157%29.txt.

[5] Незавершенные запросы перемещения почтового ящика – https://support.microsoft.com/en-us/kb/3016284.

Ключевые слова: Exchange, почтовые базы, PowerShell, восстановление.

Визитка

СЕРГЕЙ БАРАМБА,
заместитель ИТ-директора ООО «БФТ», sbaramba@yandex.ru

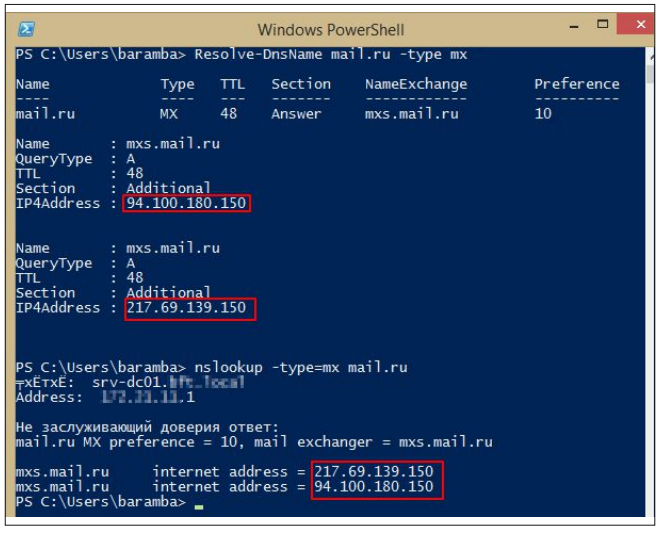
Анализ журналов SMTP-сессий

Что же происходит с электронным письмом после нажатия кнопки «Отправить»?

Зависимость пользователей от скорости, с которой приходят и отправляются письма, уже давно стала навязчивой, и в случае сбоев счет идет буквально на минуты. Ведь для них операция отправить или получить письмо – это только перемещающиеся конвертики в почтовом клиенте. Если важное письмо адресату не пришло через несколько секунд после того, как его отправили, это не обязательно сбой в работе почтовой системы, а может быть, просто особенность взаимодействия почтовых серверов между собой. Чтобы быстро провести диагностику проблем с доставкой сообщений, необходимо понимать, что за информация записывается в журналах и как правильно ее интерпретировать.

Все описываемые в статье операции производятся на Windows Server 2008R2 с установленным SMTP-сервером на базе IIS. Для журналов коннекторов Exchange или почтовых серверов на базе Linux коды откликов серверов и формат будут аналогичными.

Рисунок 1. Вывод команд nslookup -type=mx mail.ru и Resolve-DnsName mail.ru type=mx



Причинами задержки или недоставки электронной почты могут выступать:

- Неправильное написание пользовательской части адреса получателя, он может быть даже уволен, и его учетная запись была заблокирована. Как правило, почтовый сервер сообщает об этом.
- Просроченная оплата домена, повлекшая аннулирование всех записей про домен получателя. Под эту категорию попадают неправильные или еще «не распространившиеся» настройки DNS.
- Неправильная настройка серверов или другие причины, делающие невозможным корректно провести SMTP-сессию.

Эти часто встречающиеся проблемы мы и разберем ниже.

А есть ли адресат?

Некорректно записанный «на слух» адрес электронной почты – популярная проблема недоставки сообщений. Мы все знаем, что почтовый адрес состоит из двух частей, разделенных между собой знаком «коммерческое at (commercial at)», в простонародии «собачкой», – имен пользователя и домена.

Поэтому задача проверки правильности адреса электронной почты делится на две части: проверка домена и проверка имени пользователя. Все проверки необходимо осуществлять на почтовом сервере, т.к. имена DNS-серверов рабочей станции администратора и почтового сервера могут различаться.

Проверка домена, в свою очередь, делится на «существует ли такой домен» и «работает ли хоть один из почтовых серверов», прописанных в MX-записи.

Чтобы проверить, существует ли MX-запись для домена и правильно она разрешается в IP-адрес на почтовом сервере, можно воспользоваться командлетом PowerSpell – Resolve-DnsName или командой nslookup с ключом type. На рис 1. видно, как для домена mail.ru возвращаются IP-адреса серверов.

Следующим этапом требуется проверить, что серверы доступны для приема сообщений.

В консоли почтового сервера вводим команду:

```
telnet msx.mail.ru 25
```

Если произойдет подключение, и вы получите приветствие вроде:

```
220 Mail.Ru ESMTP
```

это значит, почтовый сервер работает и готов принимать команды.

Сторонники Powershell могут воспользоваться командлетом:

```
Test-NetConnection -port 25 mxs.mail.ru
```

Как проверить, существует пользователь или нет, читайте ниже, когда описывается команда VRFY.

Что творится внутри сессии?

Давайте разберемся, как же письмо передается от клиента на сервер или между серверами по протоколу SMTP и какие оставляет следы.

Каждый системный администратор знает, что SMTP-сессия стандартное подключение клиента (MUA) на 25-й порт к серверу (MTA+MSA), в RFC их называют отправитель (sender) и получатель (receiver). Для демонстрации такой сессии можно использовать telnet и отправить небольшое пробное письмо.

В командной строке выполните команду:

```
telnet IP_почтового_сервера 25
```

Получив приглашение сервера, начните аккуратно вводить команды, завершая каждую нажатием на клавишу <Enter> (указав свои адреса).

```
helo mail.ru
mail from: test@yandex.ru
rcpt to: s.baramba@example.com
DATA
From: test@lenta.ru
To: director@another-example.com
Subject: test subject
test letter
.
```

Для завершения письма необходимо набрать точку на пустой строке, чтобы сервер понял, что письмо закончилось, и его требуется обработать. Должно появиться сообщение (см. рис. 2):

```
Message accepted for delivery
```

Протокол SMTP использует модель «команда – отклик». Каждая правильно введенная вами команда должна завершаться ответом сервера «250». Чуть ниже подробно описываются коды ответов сервера на вводимые клиентом команды. В процессе ввода команд нельзя допускать опечатки, и кнопка <Backspace> или <Delete> в Telnet не работает. В случае опечатки при вводе ввода сервер возвратит ошибку:

```
500 5.3.3 Unrecognized command
```

и строку придется начинать набирать сначала.

Хотелось обратить внимание на последовательные три строки после команды DATA. В этом примере я их добавил сознательно.

Почти все письма, которые мы получаем, содержат дополнительные поля, как:

- «Тема» (Subject),
- «Копия» (Cc),
- «Скрытая копия» (Bcc),
- «Ответить» (Reply-To)
- и другие.

Но в стандартных командах (см. врезку «Справочная информация по SMTP») они не передаются. Формат данных полей описывается в другом RFC – 5322 [2]. Они составляют с точки зрения передаваемой информации заголовки письма, которые будут приоритетно отображаться в почтовом клиенте при просмотре сообщения.

Если читать RFC №5321, то в ряде примеров встречается, что первой строкой после команды DATA вводится параметр даты и времени сообщения по часам на клиенте.

Справочная информация по SMTP

Действующее RFC № 321 [1] – Simple Mail Transfer Protocol от октября 2008 года.

Основные команды, передаваемые клиентом серверу:

- **HELO** – представьтесь, указывается имя хоста отправителя, будет зафиксирована в журнале как имя отправителя;
- **EHLO** – представьтесь, указывается имя хоста отправителя и просьба серверу работать с клиентом в режиме расширенных SMTP-команд;
- **MAIL FROM:** – указывается отправитель;
- **RCPT TO:** – указывается получатель;
- **SIZE** – размер сообщения в байтах;
- **DATA** – серверу указывается, что передается тело письма (в первых трех строках должны быть адрес получателя, адрес отправителя и тема письма);
- **RSET** – прервать выполнение текущего процесса с удалением всех сохраненных данных;
- **QUIT** – завершение сессии;
- **HELP** – запрос у сервера полезной помощи по командам;
- **VRFY** – проверить адрес;
- **EXPN** – просит сервер подтвердить, что переданный аргумент - это список почтовой группы;
- **VERB** – подробно.

Рисунок 2. Демонстрации SMTP-сессии с помощью telnet и отправка небольшого пробного письма

Таблица 1. Коды откликов в порядке номеров

211	System status, or system help reply	Отклик с системной справкой или состоянием системы
214	Help message	Информация о работе с сервером или отдельных командах
220	<domain> Service ready	Служба для указанного домена готова
221	<domain> Service closing transmission channel	Закрывается канал передачи для указанного домена
250	Requested mail action okay, completed	Операция благополучно завершена
251	User not local; will forward to <forward-path>	Нелокальный пользователь – почта будет пересылаться по прямому пути
252	Cannot VRFY user, but will accept message and attempt delivery	Не удается проверить почтовый ящик, но сообщение принято, и сервер попытается его доставить
354	Start mail input; end with <CRLF>.<CRLF>	Начало ввода данных. Завершение по <CRLF>.<CRLF>
421	<domain> Service not available, closing transmission channel	Для указанного домена обслуживание невозможно, и канал связи закрывается. Это может быть откликом на любую команду, если известно, что сервис должен быть отключен
450	Requested mail action not taken: mailbox unavailable	Запрошенная операция невозможна – почтовый ящик недоступен (например, занят или временно заблокирован в соответствии с политикой)
451	Requested action aborted: error in processing	Запрошенная операция прервана в результате ошибки
452	Requested action not taken: insufficient system storage	Запрошенная операция не выполнена по причине нехватки пространства (на диске)
455	Server unable to accommodate parameters	Сервер не может принять параметры
500	Syntax error, command unrecognized	Синтаксическая ошибка, команда не распознана (это может говорить о слишком длинной команде)
501	Syntax error in parameters or arguments	Синтаксическая ошибка в параметрах или аргументах
502	Command not implemented	Команда не реализована
503	Bad sequence of commands	Некорректный порядок команд
504	Command parameter not implemented	Параметр команды не реализован
550	Requested action not taken: mailbox unavailable	Запрошенная операция невозможна – почтовый ящик недоступен (например, почтовый ящик не найден, к нему нет доступа или команда отвергнута в соответствии с заданной политикой)
551	User not local; please try <forward-path>	Нелокальный пользователь – попытайтесь использовать прямой путь
552	Requested mail action aborted: exceeded storage allocation	Запрошенная операция прервана по причине превышения выделенного (дискового) пространства
553	Requested action not taken: mailbox name not allowed	Запрошенная операция не выполнена – недопустимый почтовый ящик (например, синтаксическая ошибка в имени ящика)
554	Transaction failed	Отказ транзакции или отсутствие поддержки сервиса SMTP (при попытке соединения)
555	MAIL FROM/RCPT TO parameters not recognized or not implemented	Параметры команды MAIL FROM или RCPT TO не удалось распознать или их поддержка не реализована

В описанной выше telnet-сессии по отправке себя, себя в получателях (поле «Кому» в outlook) я не увижу, там будет стоять director, да еще из другого домена, совершенно нам не знакомого, another-example.com.

После того как письмо было обработано и отравилось, самое время приступить к изучению журнала.

Сервер SMTP ведет журнал в формате W3C (World Wide Web Consortium) (описание формата доступно по ссылке [3], так же как и другие службы IIS. По умолчанию журнал располагается по адресу C:\Windows\System32\LogFiles\SMTPSVC1.

В журнале SMTP-сервера среди записей о других письмах сможете найти похожие на эти строки.

```
2016-03-10 16:41:21 192.168.1.1 mail.ru SMTPSVC1 SRV-ORF
192.168.0.1 0 EHLO - +mail.ru 250 0 192 12 0 SMTP - - - -

2016-03-10 16:41:30 192.168.1.1 mail.ru SMTPSVC1 SRV-ORF
192.168.0.1 0 MAIL - test@yandex.ru 250 0 0 24 0 SMTP - - - -

2016-03-10 16:41:30 192.168.1.1 mail.ru SMTPSVC1 SRV-ORF
192.168.0.1 0 MAIL - +from+test@yandex.ru 250 0 49 24 0 SMTP
- - - -

2016-03-10 16:42:13 192.168.1.1 mail.ru SMTPSVC1 SRV-ORF
192.168.0.1 0 RCPT - +to:+s.baramba@example.com 250 0 33 29
156 SMTP - -

2016-03-10 16:43:56 192.168.1.1 mail.ru SMTPSVC1 SRV-ORF
192.168.0.1 0 DATA - <SRV-ORFDjoYAPNXRPmc000001a0@o.xxx.com>
250 0 127 86 38594 SMTP - - - -

2016-03-10 16:54:34 192.168.1.1 mail.ru SMTPSVC1 SRV-ORF
192.168.0.1 0 TIMEOUT - mail.ru 121 891721006 165 86 665563
SMTP - - - -

2016-03-10 16:54:35 192.168.1.1 mail.ru SMTPSVC1 SRV-ORF
192.168.0.1 0 QUIT - mail.ru 240 26563 63 4 0 SMTP - - - -
```

Как можно увидеть, журнал последовательно фиксирует команды, которые вводились во время telnet-сессии. Место, где записаны коды откликов сервера на введенные команды, выделены жирным. Код «250» обозначает, что все прошло хорошо, и введенная команда корректно обработана сервером.

В этот же журнал также записываются сообщения, не только принимаемые сервером, но и им же отправляемые. Такие сессии отличаются наличием слова «OutboundConnectionResponse»:

```
2016-03-10 19:22:12 192.168.1.1 OutboundConnectionCommand
SMTPSVC1 SRV-O - 25 MAIL - FROM:<xxxxx@xxxx.ru>+SIZE=41667 0
0 4 0 219 SMTP - - - -

2016-03-10 19:22:12 192.168.1.1 OutboundConnectionResponse
SMTPSVC1 SRV-O - 25 - - 250+2.1.0+Sender+OK 0 0 19 0 219 SMTP
```

Для детального анализа для исходящих подключений команды SMTP-сервера и сервера получателя записываются на разных строках, т.к. важными являются все детали.

Коды откликов сервера

Коды ответов SMTP состоят из трех цифр, при этом каждая цифра (первая, вторая и третья) имеет собственное значение. Первая цифра определяет статус ответа, при этом может принимать только четыре значения, четко зафиксированных в RFC 5321 (Раздел 4.2).

- **2XX** – позитивный отклик о завершении, свидетельствует о том, что команда принята и обработана сервером корректно. Подтверждение корректности ввода «250» в примере выше как раз из этой группы.
- **3XX** – позитивный промежуточный отклик, например, сервер просит начать ввод данных (код 354) или передать другую команду, содержащую необходимые данные.
- **4XX** – негативный отклик о временных проблемах, свидетельствует о том, что команда не была принята сервером и запрошенная в ней операция не выполнена. Клиенту необходимо, получив такой отклик сервера, вернуться к началу последовательности команд или попробовать ввести команду заново.
- **5XX** – негативный отклик о постоянной проблеме, связанной с вводом и обработкой команды клиента. Клиенту SMTP не следует просто повторять команду, поскольку она заведомо не будет выполнена. Например, код «550», когда на сервере нет пользователя, которому планируется отправка сообщения.

Различие между временными (коды 4XX) и постоянными проблемами (коды 5XX) сводится к тому, что отклики о временных проблемах обычно возвращаются в тех случаях, когда возможен положительный результат при повторе без изменения формы команды и свойств отправителя или получателя (т.е. команда просто может быть повторена без изменений).

Вторая цифра отклика показывает категорию ошибки:

- **X0X** – отклик сообщает о наличии синтаксической ошибки (сама команда корректна, но она относится к нереализованным командам или излишняя).
- **X1X** – отклик на запрос информации (например, справка или состояние).
- **X2X** – данные отклики относятся к каналу передачи данных внутри сессии.
- **X3X и X4X** – для этих двух кодов описания пока не заданы.
- **X5X** – отклики, возвращающие состояние принимающей почтовой системы по отношению к запрошенной команде.

Вместе с кодами сервер, как правило, возвращает строку текста, которая помогает человеку анализировать происходящее. Часто эта текстовая строка передается программным обеспечением сервера SMTP и отображается как часть ответа сервера. Со всеми возможными вариантами ошибок можно ознакомиться в таблице 1.

Сводные данные таблицы 2 могут оказаться полезными для быстрого просмотра ожидаемых откликов принимающего почтового сервера при попытках отправки почты.

Проверка учетной записи

Мне показалась интересной команда, позволяющая проверить, существует ли запрашиваемый пользователь на почтовом сервере. В этом поможет команда VRFY, переданная почтовому серверу. Но есть важный момент.

Разрешенная на почтовом сервере команда VRFY может использоваться хакером для получения списка логинов, чтобы потом провести брут-форс атаку на пользовательские учетные данные.

Поэтому, если в ответ получите ошибку:

```
502 5.5.1 VRFY command is disabled
```

это означает, что выполнение команды отключено администратором. Корректные варианты ответов сервера на VRFY приведены на рис. 3.

Если сервер не готов принимать почту для этого адреса, вы получите ошибку 550 или 252, если адрес пользователя для данного сервера корректный.

• • •

Анализ журналов нагруженных почтовых систем – достаточно трудоемкая операция, но это верный способ получить четкую картину происходящего с письмом, когда оно должно прийти или быть отправленным.

Для автоматизации и ускорения процесса анализа можно рекомендовать использовать дополнительные утилиты, например, универсальный анализатор Microsoft LogParser [4]. EOF

[1] RFC 5321 – https://tools.ietf.org/html/rfc5321.
[2] RFC 5322 – https://tools.ietf.org/html/rfc4021.
[3] Описание формата журнала SMTP сервера – https://msdn.microsoft.com/ru-ru/library/cc780772(v=ws.10).aspx.
[4] Страница Log Parser – https://technet.microsoft.com/ru-ru/scriptcenter/dd919274.aspx?f=255&MSPPError=-2147217396.

Ключевые слова: smtp, почтовый сервер, диагностика, rfc.

Рисунок 3. Вывод команды VRFY

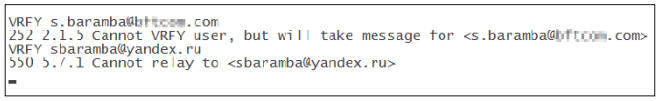


Таблица 2. Сводная таблица возможных откликов сервером на команды клиента

Команда	Успех (S)	Неудача (E)
Организация соединения	220	554
EHLO или HELO	250	504, 550, 502
MAIL	250	552, 451, 452, 550, 553, 503, 455, 555
RCPT	250, 251	550, 551, 552, 553, 450, 451, 452, 503, 455, 555
DATA (промежуточный отклик 354)	250	552, 554, 451, 452, 450, 550 (отказ в соответствии с политикой)
DATA		503,55
RSET	250	
VRFY	250, 251, 252	550, 551, 553, 502, 504
EXPN	250, 252	550, 500, 502, 504
HELP	211, 214	502, 504
NOOP	250	
QUIT	221	

Визитка

ВИКТОР ОСЬМОВ,
технический специалист ЗАО «НПП Родник»

Сравнение решений
VMware vSphere и Stratus everRun Enterprise

Сравниваем решения для обеспечения бесперебойной работы критически важных бизнес-приложений

Сегодня от любой системы или службы требуется постоянная доступность. Говорим мы об информационных сервисах предприятия, управлении производственными циклами, работе систем безопасности, реагировании аварийных служб, выполнении банковских операций – затронут буквально каждый момент повседневной жизни.

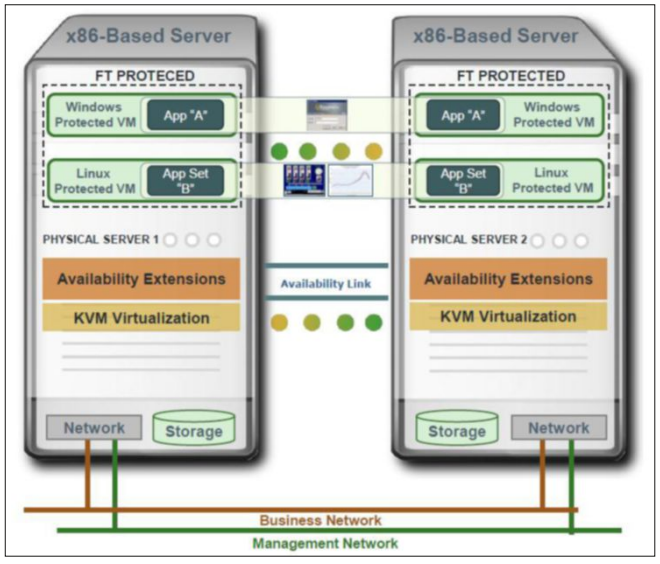
Информационные инфраструктуры, обеспечивающие функционирование многих систем в мире, в настоящее время виртуализируются. Большие и малые предприятия внедряют виртуализацию, чтобы использовать такие ее преимущества над традиционными физическими платформами, как динамичность, эффективность и масштабируемость. По мере того, как все большее число критически важных приложений виртуализируется, главным приоритетом становится обеспечение их постоянной работоспособности.

Стоимость простоя компании в связи с отказом информационных систем увеличивается с каждым годом и для крупных предприятий уже превысила ошеломляющие $650 000 в час. Непредвиденная остановка ключевых приложений, приводящая к потере данных или пропуску операций, становится просто неприемлемой. В ситуации, когда на одном физическом сервере выполняется несколько виртуальных машин, один аппаратный сбой может приводить к масштабным негативным последствиям для бизнеса.

Одним из распространенных подходов для минимизации времени простоя является создание серверного кластера с механизмом аварийного переключения, который перезапускает виртуальную машину на другом узле в случае аппаратного сбоя или ошибки операционной системы. В таком случае процесс восстановления не только занимает много времени, но и подразумевает несение определенного ущерба. Но в идеальном случае, система, исполняющая важное для предприятия виртуализированное приложение, должна исключить перерыв в его работе даже в условиях отказа оборудования.

В этой статье сравниваются два программных продукта для обеспечения доступности приложений: VMware vSphere HA или FT и Stratus everRun.

Рисунок 1. Система Stratus everRun



Внедрение решения: сложность или простота

Создание среды с высоким уровнем доступности, основанном на кластеризации, сложное и затратное дело. В среде VMware для включения системы аварийного переключения вам потребуются кластер из двух узлов, сеть хранения данных и дополнительная сеть VMware vMotion для переноса виртуальной машины с одного узла на другой. Необходимо настроить целый ряд параметров, чтобы определить, как кластер HA или FT будет вести себя в случае сбоя и обеспечить наличие ресурсов для перехода виртуальной машины с одного узла на другой.

Система Stratus everRun строится только на двух x86 серверах, работающих под управлением Centos Linux. Также используются гипервизор с открытым исходным кодом KVM и специальное ПО Availability Extensions, обеспечивающее перенаправление ввода-вывода и синхронизацию узлов системы (см. рис. 1).

ПО everRun устанавливается на оба сервера в несколько шагов, основная настройка происходит автоматически. Виртуальные машины создаются и управляются через встроенный web-интерфейс everRun Availability Console. Установка гостевой ОС и приложений выполняется один раз, после чего everRun создаст ее дубликат и начнет непрерывно зеркалировать диски между физическими узлами, операции ввода-вывода, состояние оперативной памяти и распределять нагрузку для оптимального использования вычислительных ресурсов.

Stratus everRun предлагает выбор двух уровней доступности – отказоустойчивость (fault tolerance, FT) и высокая доступность (high availability, HA).

В случае выбора режима FT, состояние виртуальной машины на обоих физических узлах будет полностью идентичным. Режим FT характеризуется полным отсутствием единой точки отказа. Состояние подсистемы хранения зеркалировано между узлами, и в использовании внешних запоминающих устройства нет необходимости.

При выборе уровня защиты HA, благодаря избыточности на уровне компонентов, система выдерживает сетевые сбои и отказы системы хранения без простоя. При отказе физического узла целиком, виртуальная машина автоматически восстанавливается на другом узле. Как и в FT, нет необходимости в использовании SAN или внешних запоминающих устройства.

И в случае HA, и в случае FT обеспечивается активное наблюдение за состоянием аппаратного обеспечения обоих серверов, автоматическая обработка ошибок и восстановление.

Общая оценка расходов

Кластеру vSphere HA/FT требуется выделенная консоль управления, высокодоступная сетевая инфраструктура и внешняя система хранения данных. Stratus everRun не требует каких-либо дополнительных сетей или систем хранения. К тому же, установка everRun значительно проще, а, следовательно, и менее дорогостоящая по сравнению с vSphere.

Управление и поддержка также являются важными факторами стоимости. Кластерам часто требуется ручное управление, что вызывает затраты на административный персонал. Любые изменения в программном или аппаратном обеспечении в используемом кластере должны аккуратно проверяться перед внедрением.

Администрирование everRun минимально. Автоматический внутренний механизм осуществляет мониторинг за приложениями и устройствами. Обнаружение проблемы вызывает выполнение встроенного скрипта для ее устранения, перезапуск приложения при необходимости и немедленное оповещение через SNMP.

Обработка отказов и ограничения

vSphere HA и everRun предлагают два разных подхода: кластеры vSphere HA полагаются на перезапуск приложения после сбоя сервера, а everRun обеспечивает предотвращение простоев. Из-за этих различий есть факторы, которые следует учитывать при оценке двух решений.

vSphere HA сокращает время простоя за счет автоматического перезапуска виртуальных машин, но не защищает сервер от снижения производительности. При этом данные, не записанные на диск, теряются при отключении электропитания. Повторный запуск приложения зависит от его типа и не может быть мгновенным. Так, среда SAP может находиться в автономном режиме в течение долгого времени после перезапуска (источник: SAP Saber – Carving SAP into separate landscapes for company split. Van Vi and Rick Jones.):

- **ABAP Central/Dialog Instances** – ~ 4–5 минут для перезагрузки ОС и запуска SAP.
- **Java Central/Dialog Instances** – ~ 15–17 минут для перезагрузки ОС и запуска SAP.
- **DB Instance** – ~ 4–5 минут для перезагрузки ОС и восстановления SQL Server.
- **Web-Dispatcher Instance** – ~ 3–4 минуты.

В Stratus everRun приложение находится одновременно на двух виртуальных машинах. Если одна выходит из строя, приложение продолжает работать на другой без перерывов и потери данных. При сбое компонента на сервере приложение использует дубликат компонента из второго сервера.

В настоящее время VMware FT поддерживает до четырех виртуальных процессоров на одну виртуальную машину и максимум четыре защищенные ВМ на одном физическом узле с общим количеством vCPU не более 8. А многие критичные приложения чувствительны к производительности и требуют интенсивной многоядерной симметричной многопроцессорной обработки. Например, Microsoft рекомендует от 4 до 12 процессорных ядер для Exchange Server и от 4 до 8 ядер для SharePoint и SQL Server. Oracle рекомендует от 6 до 12 ядер.

Stratus everRun поддерживает до 8 vCPU на защищенной виртуальной машине и не имеет каких-то дополнительных ограничений, что делает его более удобным для пользователей и экономически эффективным решением.

• • •

При сравнении vSphere HA/FT и everRun различия очевидны: кластеры vSphere требуют сложного внедрения и технического обслуживания. Stratus everRun предлагает простоту установки и администрирования.

В конечном счете, Stratus everRun является более простым и экономичным решением для обеспечения доступности виртуализованных критически важных бизнес-приложений, чем VMware vSphere. ADV

Визитка

ИГОРЬ ОРЕЩЕНКОВ,
инженер-программист, iharsw@tut.by

Старые сценарии на новый лад

Выполнение операций с удаленными системами

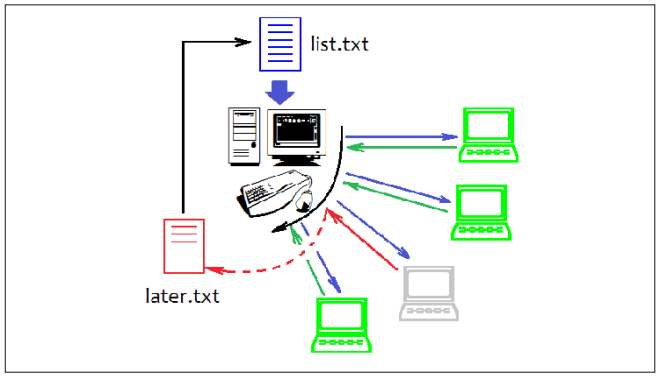
Универсальный сценарий командной строки Windows позволяет выполнять разнообразные действия с подготовленным перечнем сетевых устройств

Системному администратору нередко приходится работать в обстановке, которая навязана внешними обстоятельствами. Далеко не всегда он имеет достаточное влияние на принятие решений о выборе прикладного программного обеспечения, которое используется в организации. Те же, кто приобретает программные продукты, в первую очередь оценивают их полезность с точки зрения бизнеса, отодвигая вопросы сопровождения на второй план. И, чтобы не утонуть в рутине повседневных микропроблем, системный администратор должен переложить их выполнение на автоматизированные системы, находящиеся в его ведении.

Одно дело управлять серверами. Их состояние отображается в системе мониторинга, они ведут себя предсказуемо, изменяют режим работы и перезагружаются по воле администратора. И совсем другое дело – рабочие станции пользователей. Далеко не всегда можно быть даже уверенным, работает пользовательский компьютер или рачительный сотрудник выключил его, отправляясь на обед. А поступившее от разработчика программного пакета обновление требует срочного внесения изменений в параметры конфигурационного файла и добавления ключа в системный реестр.

Решать задачи управления удаленными компьютерами можно множеством способов. В статье рассматривается один из них, заключающийся в программном переборе сетевых узлов по заранее подготовленному списку с выполнением по отношению к ним необходимых действий. Как перебор, так и сами действия реализуются командными файлами, выполняемыми на управляющем компьютере (см. рис. 1). Если на момент выполнения обработки списка сетевой узел недоступен, он фиксируется в специальном списке, который может быть представлен на обработку позже. Таким образом, постепенно будут выполнены запланированные операции над всеми компьютерами.

Рисунок 1. По подготовленному списку выполняется перебор сетевых узлов с выполнением для каждого доступного узла команд, записанных в файле вспомогательного сценария



Сценарий перебора сетевых узлов

Задачу перебора сетевых узлов решает сценарий, текст которого приведен в листинге 1. Рассмотрим подробно его работу.

Листинг 1. Сценарий перебора сетевых узлов

```
<<< Текст сценария main.cmd с пронумерованными строками >>>
0. @ECHO OFF
1. SET DISK=M:
2. IF NOT EXIST %DISK%\NUL GOTO CONT1
3. ECHO ================================
4. ECHO Имеется подключенный диск %DISK%.
5. ECHO Перед выполнением сценария
6. ECHO его нужно отключить!
7. ECHO ================================
8. GOTO :EOF
9. :CONT1
10. IF EXIST later.txt DEL list.txt & REN later.txt list.txt
11. FOR /F "EOL=; TOKENS=1,2" %%a IN (list.txt) ↵
        DO CALL :DOACT %%a %%b
12. ECHO Выполнение сценария завершено.
13. GOTO :EOF
14. REM ============================================
15. REM Выполнение действия для текущего устройства.
16. REM %1 - IP-адрес, %2 - имя устройства
17. REM ============================================
18. :DOACT
19. ECHO Обрабатывается %2 (%1)...
20. IF EXIST %DISK%\NUL net use %DISK% /delete
21. IF ERRORLEVEL 1 GOTO ERR
22. ping -n 1 %1 > NUL
23. IF ERRORLEVEL 1 GOTO ERR
24. net use %DISK% Ошибка! Недопустимый объект гиперссылки.
25. IF ERRORLEVEL 1 GOTO ERR
```

```
26. CALL action.cmd %2 %1
27. IF ERRORLEVEL 1 GOTO ERR
28. ECHO %2 (%1) - ok.
29. IF EXIST %DISK%\NUL net use %DISK% /delete
30. GOTO :EOF
31. :ERR
32. ECHO %1 %2 >> later.txt
33. ECHO %2 (%1) - ERROR!
34. REM Сброс признака ошибки ERRORLEVEL
35. dir >nul
```

Исходными данными для сценария служит перечень сетевых устройств, записанный построчно в файле list.txt следующего формата:

Пример файла list.txt

```
; Бухгалтерия
192.168.1.21      BUCH1
192.168.1.22      BUCH2
192.168.1.21      BUCH3
; Плановый отдел
192.168.1.31      ECON1
192.168.1.32      ECON2
192.168.1.33      ECON3
; Отдел кадров
192.168.1.41      HR1
192.168.1.42      HR2
192.168.1.43      HR3
```

Для управления файлами, размещенными на Windows-компьютерах, удобно в ходе работы сценария отображать диск удаленного компьютера как букву латинского алфавита для того, чтобы в дальнейшем работать с ним как с локальным устройством. В строке 1 задается конкретная буква для такого сетевого диска.

Однако если отображение сетевого ресурса на локальное устройство было выполнено системным администратором до начала работы сценария (например, вручную для выполнения каких-нибудь профилактических работ), то продолжение работы сценария с этим диском может привести к неприятным последствиям (модификации файлов на компьютере, который не предполагалось включать в обработку). Поэтому строки 2-8 предотвращают работу сценария с ранее подключенным сетевым диском, для корректной работы сценария устройство %DISK% должно быть свободно.

Хотя в языке CMD-сценариев имеется средство проверки существования файла с помощью конструкции «IF EXIST ИмяФайла», его нельзя использовать для проверки существования папки в виде «IF EXIST ИмяПапки». Поэтому в строках 2, 20 и 29 используется описанный в [1] трюк, позволяющий обойти это ограничение: «IF EXIST ИмяПапки\NUL».

Строка 11, несмотря на свою лаконичность, выполняет целый комплекс операций:

- построчно читает файл list.txt;
- пропускает строки, начинающиеся с символа «;», что позволяет использовать этот символ в качестве префикса для комментариев;
- осуществляет разбор каждой прочитанной строки на IP-адрес и логическое имя сетевого узла с присваиванием адреса переменной %%a, а имени узла переменной %%b;
- выполняет для каждой пары «IP-адрес», «имя узла» строки 19-35 сценария.

Давайте прервем рассмотрение сценария и подробнее остановимся на мощной команде FOR /L.

Построчная обработка текстовых файлов

Для изучения команды FOR, обрабатывающей списки значений, удобно использовать текстовый файл table.txt следующего содержания:

Файл table.txt

```
r1c1 r1c2 r1c3 r1c4 r1c5 r1c6 r1c7 r1c8 r1c9
r2c1 r2c2 r2c3 r2c4 r2c5 r2c6 r2c7 r2c8 r2c9
r3c1 r3c2 r3c3 r3c4 r3c5 r3c6 r3c7 r3c8 r3c9
r4c1 r4c2 r4c3 r4c4 r4c5 r4c6 r4c7 r4c8 r4c9
r5c1 r5c2 r5c3 r5c4 r5c5 r5c6 r5c7 r5c8 r5c9
```

Он имитирует содержимое таблицы, которая состоит из строк (rows) и столбцов (columns). Обработка подобных файлов в команде FOR происходит построчно. Каждая строка разбивается на элементы, которые в тексте должны быть разделены специальными символами-разделителями. Сначала для простоты в качестве разделителя будем использовать пробел, как это предполагается по умолчанию. Команды будем набирать прямо в окне командного процессора, в интерактивном режиме, поэтому переменные должны записываться с одним префиксом «%», а не двумя «%%», как это требуется в случае выполнения команд из пакетного файла.

Команда:

```
FOR /F %A IN (table.txt) DO @ECHO %A
```

выведет (в столбец) значения первого столбца обрабатываемого файла:

```
r1c1 r2c1 r3c1 r4c1 r5c1
```

Если требуется работать со значениями нескольких столбцов, их номера нужно указать в FOR, а значения извлекаемых элементов будут последовательно присвоены переменным %A, %B... по количеству полей:

```
FOR /F "TOKENS=1,2,4-6,*" %A  IN (table.txt) ↵
        DO ECHO %A,%B,%C,%D,%E,%F
```

В ходе выполнения этой команды при обработке каждой строки файла table.txt переменной %A будет присвоено значение первого элемента, переменной %B – второго элемента, переменным %C, %D и %E – значения четвертого, пятого и шестого элементов соответственно, а переменной %F будет присвоена оставшаяся «неразобранная» часть строки (см. рис. 2).

Если из шаблона исключить символ «*», то переменная %F инициализирована не будет. В заголовке цикла может быть указана только первая последовательность, остальные создаются и инициализируются автоматически в алфавитном порядке. Если вместо %A указать %K, то будут инициализированы переменные %K, %L, %M, %N, %O. Обратите внимание, что запись «4-6» раскрывается в «4,5,6» и приводит к инициализации трех переменных значениями трех элементов, а не «склеивает» три значения с последующим присваиванием одной переменной, как можно было бы подумать.

Когда принцип работы команды FOR с ключом /F станет понятен, воспользуемся дополнительными директивами:

```
FOR /F "EOL=# SKIP=2 DELIMS=,; TOKENS=1,2,4-6,*" %A  ↵
         IN (table.txt) DO ECHO %A,%B,%C,%D,%E,%F
```

При выполнении этой команды из обработки будут исключены первые две строки файла list.txt (директива SKIP=2) и строки, начинающиеся с символа «#» (директива EOL=#, можно указать только один символ), а в качестве разделителей элементов вместо используемых по умолчанию «пробела» и «табуляции» будут подразумеваться «запятая» и «точка с запятой» (директива DELIMS=,;).

Подпрограммы в теле сценария

В качестве тела цикла FOR может быть использована команда CALL, которая в командных процессорах ОС Windows позволяет не только обращаться к вспомогательному сценарию, находящемуся в отдельном BAT или CMD-файле, но и осуществлять переход по метке внутри выполняемого сценария. В этом случае после перехода будут последовательно выполнены все команды, следующие после метки, до конца файла сценария, после чего произойдет переход к следующей итерации цикла. Переход к следующей итерации до достижения конца файла можно осуществить с помощью команды GOTO :EOF (метка :EOF определяется автоматически и ссылается на конец текущего файла сценария).

Рассмотрим еще пример, демонстрирующий использование вложенных циклов. Для этого изменим обрабатываемый файл table.txt, сгруппировав в нем элементы с помощью разделителей «;», а сами элементы разделив символом «,»:

Измененный файл table.txt

```
r1c1,r1c2,r1c3;r1c4,r1c5,r1c6;r1c7,r1c8,r1c9
r2c1,r2c2,r2c3;r2c4,r2c5,r2c6;r2c7,r2c8,r2c9
r3c1,r3c2,r3c3;r3c4,r3c5,r3c6;r3c7,r3c8,r3c9
r4c1,r4c2,r4c3;r4c4,r4c5,r4c6;r4c7,r4c8,r4c9
r5c1,r5c2,r5c3;r5c4,r5c5,r5c6;r5c7,r5c8,r5c9
```

Сценарий, с помощью которого мы будем получать поэлементный доступ к содержимому файла table.txt, выглядит так:

Листинг 2. Пример сценария поэлементной обработки текстового файла

```
@ECHO OFF
FOR /F "DELIMS=; TOKENS=1-3" %%A  IN (table.txt) ↵
         DO CALL :LOOP "%%A" "%%B" "%%C" & ECHO - - - -
PAUSE
GOTO :EOF
:LOOP
FOR /F "DELIMS=, TOKENS=1-3" %%A IN (%1) ↵
         DO ECHO 1: [%%A] [%%B] [%%C]
FOR /F "DELIMS=, TOKENS=1-3" %%A IN (%2) ↵
         DO ECHO 2: [%%A] [%%B] [%%C]
FOR /F "DELIMS=, TOKENS=1-3" %%A IN (%3) ↵
         DO ECHO 3: [%%A] [%%B] [%%C]
GOTO :EOF
```

Первый цикл FOR обеспечивает построчную обработку файла и выделяет в каждой строке группы элементов, разделенные символом «;», присваивая их переменным %%A, %%B, %%C. Команда CALL в теле цикла осуществляет вызов блока строк сценария, начинающегося за меткой :LOOP. Как и при вызове вспомогательного командного файла, в вызываемую подпрограмму можно передавать параметры командной строки, которые будут доступны в ней через обозначения «%1», «%2»... Мы используем этот механизм для работы с выделенными группами элементов, потому что команда CALL создает новый контекст выполнения сценария, в котором значения параметров цикла %%A, %%B, %%C недоступны.

В трех последовательных операторах цикла FOR, размещенных в блоке строк сценария, используется возможность непосредственной обработки строк текста. Для этого в круглых скобках записывается не имя файла, из которого должны извлекаться строки, а сама строка, заключенная в двойные кавычки. Обрамление строк, представляющих группу элементов, разделенных символом «,», двойными кавычками мы обеспечили на этапе вызова блока кода в команде CALL. При выполнении же оператора FOR к единственной строке будет применен шаблон разбора, а для результата разбора будет выполнено тело цикла FOR - оператор ECHO, выводящий на экран извлеченные из строки элементы.

Строго говоря, при таком использовании оператора FOR никакого цикла не организуется (или организуется цикл из одной итерации). Но для разбора текстовых строк на элементы этот синтаксис удобен. А если в круглых скобках оператора FOR в одинарных кавычках записать текст команды, например DIR, то «магия» цикла выполнится для всех строк, которые выведет эта команда в результате своего выполнения.

Код завершения ERRORLEVEL

Вернемся к нашему основному сценарию (см. листинг 1). Строки 20-25 «готовят почву» для выполнения запланированных

Рисунок 2. Цикл FOR с ключом /F выполняет разбор строк текстового файла

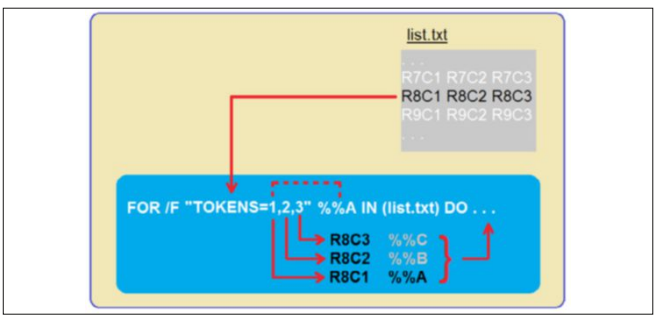
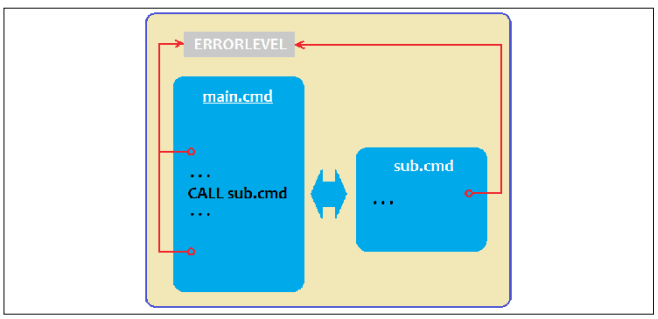


Рисунок 3. Значение кода ошибки ERRORLEVEL является глобальным для сеанса выполнения сценария



действий над текущим сетевым узлом. Строка 22 осуществляет предварительную проверку активности сетевого узла. Дело в том, что команда NET USE из строки 24 надолго «задумывается» при попытке подключить сетевой ресурс выключенного компьютера, прежде чем сообщить об ошибке. А на PING-тест уходит всего 1-2 секунды.

Как правило, признаком успешного выполнения команды является установка в системе кода завершения %ErrorLevel%, равного нулю. Однако из этого правила могут быть исключения, поэтому в общем случае нужно либо обратиться к документации (что предпочтительно), либо (если документация отсутствует) произвести несколько опытов.

Команда IF ERRORLEVEL сравнивает значение кода завершения предыдущей команды и, если он больше или равен аргументу (в нашем сценарии сравнение осуществляется с единицей), то результат считается истинным и выполняется определяемая условием команда (в нашем сценарии – переход на метку ERR для обработки ошибки).

Значение кода ошибки ERRORLEVEL, которое доступно через переменную окружения %ErrorLevel%, является глобальным для всего сеанса выполнения сценария (см. рис. 3). Как в основном сценарии, так и во вспомогательных сценариях, вызванных командой CALL, работа производится с одним и тем же значением, даже если используются команды контекста SETLOCAL/ENDLOCAL (нужно только иметь в виду, что команда SETLOCAL сама изменяет значение ERRORLEVEL).

Если необходимо «зафиксировать» значение кода ошибки в определенный момент, нужно воспользоваться вспомогательной переменной и в дальнейшем анализировать ее значение, например:

```
...
SET EL1=%ErrorLevel%
...
IF %EL1 == 1 GOTO Error1
...
```

Отметим «побочный эффект» проверки с помощью команды PING: ее результат считается «положительным» в случае получения любого ответа, даже «сетевой узел недоступен» от промежуточного маршрутизатора. Для нашего сценария это не принципиально, потому что "ложное срабатывание" устраняется дополнительным контролем в строке 25, но в других применениях это явление надо иметь в виду.

Командой строки 29 основного сценария сетевой ресурс отключается для подготовки к выполнению следующей итерации цикла и обработки очередного устройства или компьютера.

Пара строк 10 и 32, несмотря на их далекое расположение друг от друга, решает общую задачу – составление списка сетевых узлов, которые не были успешно обработаны, для передачи их в обработку при следующем сеансе запуска сценария. Если при обработке сетевого узла произошла ошибка, то в строке 32 его IP-адрес и логическое имя будут записаны в файл later.txt в формате, который используется для формирования файла list.txt. Это позволяет в строке 10 перед началом работы сценария в случае наличия файла later.txt заменить им файл list.txt и дообработать ранее пропущенные компьютеры или сетевые устройства.

Вспомогательный сценарий

Строка 26 вызывает вспомогательный сценарий action.cmd, в котором должны быть описаны действия, выполняемые для каждого сетевого узла. При разработке сценария action.cmd можно использовать следующие предположения:

- сетевой ресурс для обработки отображен на устройство %DISK%;
- логическое имя устройства передано в первом параметре командной строки и доступно через обозначение %1;
- IP-адрес устройства передан во втором параметре командной строки и доступен через обозначение %2.

Текст сценария action.cmd может свободно меняться в зависимости от решаемой задачи. Например, такой сценарий позволяет собрать с Windows-компьютеров конфигурационные файлы autoexec.nt и config.nt:

Листинг 3. Вспомогательный сценарий action.cmd

```
MKDIR WINCOMP\%1
COPY %DISK%\WINDOWS\SYSTEM32\autoexec.nt WINCOMP\%1
COPY %DISK%\WINDOWS\SYSTEM32\config.nt WINCOMP\%1
```

В ходе выполнения сценария перебора сетевых узлов из листинга 1 с таким вспомогательным сценарием action.cmd в папке WINCOMP (которая должна существовать на момент выполнения) будут созданы папки с логическими именами Windows-компьютеров, в которые запишутся их конфигурационные файлы.

Почему удобно подключать сетевой диск %DISK%? Во-первых, этим обеспечивается первичная фильтрация сетевых устройств, способных предоставлять свою файловую систему в общий доступ по протоколу SMB/CIFS (в нашем случае это Windows-компьютеры). Во-вторых, после подключения сетевого диска можно легко осуществить дополнительную фильтрацию по наличию того или иного установленного программного обеспечения, файлов конфигурации и другим параметрам (например, IF EXIST «%DISK%\Program Files\1C\NUL» ...).

Но нужно помнить, что сценарий выполняется на локальном компьютере, из-за чего недоступны некоторые знания о конфигурации управляемого узла, например, значения переменных окружения его текущего сеанса. Поэтому во вспомогательном сценарии явно прописывался путь WINDOWS\SYSTEM32, а не использовалась сеансовая переменная %WinDir% или %SystemRoot%.

Сфера возможных действий вспомогательного сценария не ограничивается файловыми операциями. Благодаря доступности в нем IP-адреса управляемого устройства могут использоваться различные вспомогательные утилиты. Например, команда REG позволяет работать с системным реестром удаленного Windows-компьютера. EOF

[1] Попов А. Командная строка и сценарии Windows – http://www.intuit.ru/studies/courses/1059/225/info.

[2] Орещенков И. Старые сценарии на новый лад. // «Системный администратор», №3, 2016 г. – С. 11-15 (http://samag.ru/archive/article/3142).

Ключевые слова: администрирование, сценарии, cmd, удаленное управление.

Визитка

РАШИД АЧИЛОВ,
главный специалист по защите информации в компании, занимающейся автоматизацией горнодобывающей промышленности, shelton@sheltonsoft.ru

Построение корпоративных VPN. Часть 11.
Linux-Linux, Linux-FreeBSD и Linux-Mikrotik через strongSwan

Рассмотрим варианты соединения Linux-Linux с помощью strongSwan, но уже с учетом динамического адреса клиента, а также соединение между Linux и FreeBSD и между Linux и Mikrotik

Что в адресе тебе моем?

Ситуация, когда в VPN объединяются подсети, подключенные через устройства с постоянным IP, хоть и достаточно характерна для построения основной структуры VPN, все же встречается довольно редко – ведь не каждый же день компания открывает новый филиал или переносит старый в новый офис. Гораздо чаще встречается ситуация, когда к VPN подключается мобильный клиент – один или несколько сотрудников, работающих удаленно.

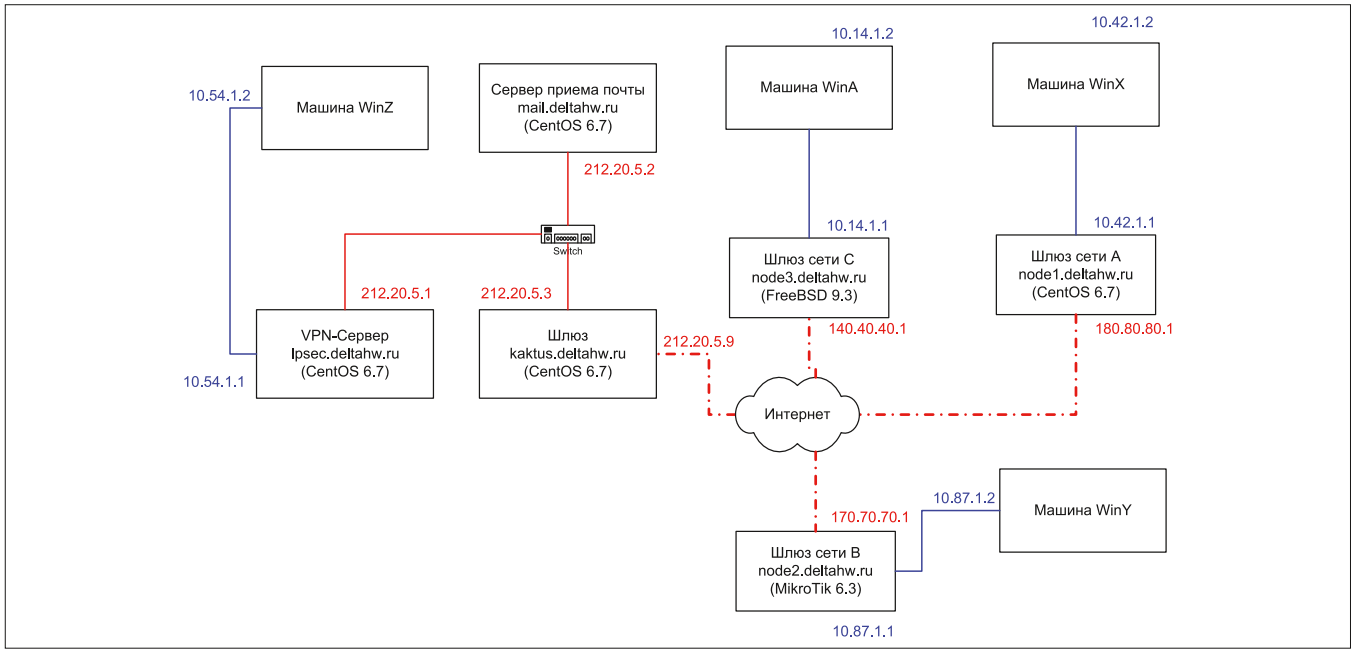
Подключение мобильного клиента, работающего под Windows или Android, мы рассмотрим в следующих частях, сейчас разберем, как организовать работу «на выезде», когда в произвольной точке сети включается роутер (для определенности предположим, что это Mikrotik RB450G) и на портах его локальной сети появляется VPN в локальную сеть компании. Разумеется, при условии, что порт 500 не заблокирован и к сети можно подключать столько устройств, сколько необходимо.

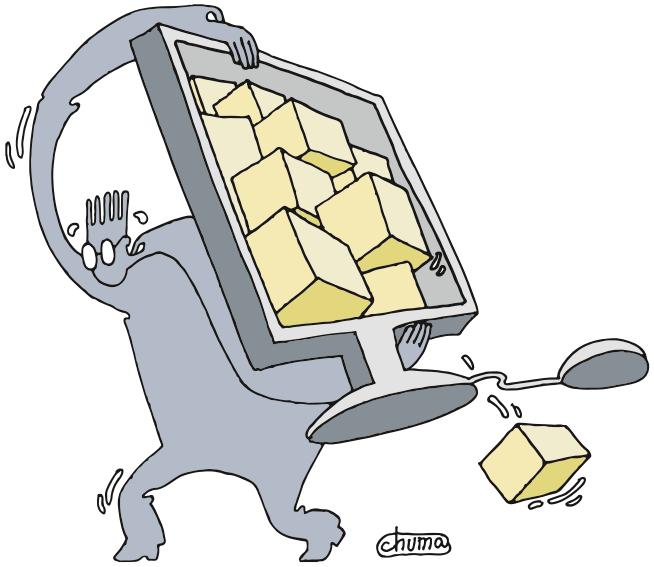
Скажете, искусственная задача? Отнюдь. Она характерна для любого внедрения какого-либо технически сложного решения – приходится все свое везти с собой, потому что на месте добраться до внутренних ресурсов компании, как правило, просто так не получается.

Это же решение можно применить, когда провайдер не в состоянии предоставить выделенный IP и приходится довольствоваться реальным, но динамически выделяемым адресом.

Итак, наша прежняя схема [1] изображена на рис. 1.

Рисунок 1. Схема тестовой модели сети





Подключим к серверу Linux три варианта клиентов — Linux с динамическим IP, FreeBSD и Mikrotik

Будем строить VPN между узлами с IP 212.20.5.1 и Сетью B. Все адреса и имена на схеме вымышленные и в реальности не существуют. Будут использоваться дистрибутив CentOS 6.7, ядро 2.6.32.

С одним нововведением – у Сети B нет фиксированного IP-адреса. Он динамический, то есть от включения к включению роутера он меняется.

Порядок установки strongSwan, порядок настройки конфигурационных файлов, отличных от ipsec.conf, да и порядок настройки общих параметров файла ipsec.conf были описаны в предыдущей части цикла [1]. Здесь мы приведем только блок настройки соединения, описываемый со стороны сервера.

```
conn any-deltahwCA-rsa
    auto=route
    left=212.20.5.1
    leftid="/C=RU/ST=Novosibirsk region/L=Novosibirsk/ ↵
         O=DeltaHardware Ltd/OU=Linux servers/ ↵
         CN=ipsec.deltahw.ru/ ↵
         emailAddress=root@deltahw.ru"
    leftauth=pubkey
    leftcert=ipsec_deltahw.crt
    leftsubnet=10.54.1.0/24
    leftca="/C=RU/ST=Novosibirsk region/L=Novosibirsk/ ↵
         O=DeltaHardware Ltd/OU=Certificate Authority/ ↵
         CN=DeltaHardware Ltd Root CA/ ↵
         emailAddress=certmgr@deltahw.ru"
    right=%any
    rightallowany=yes
    rightid="/C=RU/ST=Novosibirsk region/L=Novosibirsk/ ↵
         O=DeltaHardware Ltd/OU=Routers/ ↵
         CN=rb450g.deltahw.ru/ ↵
         emailAddress=root@deltahw.ru"
    rightcert=rb450g_deltahw.crt
    rightsubnet=10.87.1.0/24
    rightauth=pubkey
    rightca="/C=RU/ST=Novosibirsk region/L=Novosibirsk/ ↵
         O=DeltaHardware Ltd/OU=Certificate Authority/ ↵
         CN=DeltaHardware Ltd Root CA/ ↵
         emailAddress=certmgr@deltahw.ru"
    keyexchange=ikev1
    ike=aes128-sha-modp1024,aes192-sha-modp1024, ↵
         aes256-sha-modp1024!
    esp=aes128-sha-modp1024,aes192-sha-modp1024, ↵
         aes256-sha-modp1024
```

Как мы видим, описание соединения значительно увеличилось в размерах.

Во-первых, больше никаких PSK – только сертификаты. Почему? Предполагается, что IP, через который подключаются, неизвестен, конфигурационные файлы никто исправлять не будет, трафик может проходить через недоверенные узлы. Поэтому только сертификаты.

Во-вторых, не просто какие угодно сертификаты, а только сертификаты, выданные определенным удостоверяющим центром, в данном случае DeltaHardware Root CA. Наличие или отсутствие корневого сертификата в хранилище корневых сертификатов в данном случае не имеет никакого значения – доверие будет только к тем сертификатам, удостоверяющий центр которых совпадает с Subject, указанным в настройках соединения.

- **auto=route** – с этим параметром мы уже встречались. Он задает автоматическое конфигурирование параметров соединения, в том числе создание записей SPD, если это возможно сделать. В нашем случае это невозможно, потому что неизвестен IP-адрес удаленной стороны.
- **left=212.20.5.1** – идентификатор (в данном случае IP-адрес) локальной стороны соединения.
- **leftid="..."** – полный Subject сертификата, используемого для идентификации локальной стороны соединения. Subject можно получить командой:

```
# openssl x509 -in ipsec_deltahw.crt -noout -subject
```

где ipsec_deltahw.crt – имя файла сертификата, Subject которого берется.

- **leftauth=pubkey** – метод аутентификации локальной стороны – SSL-сертификат.
- **leftcert=ipsec_deltahw.crt** – имя файла сертификата, используемого для аутентификации локальной стороны. Файл сертификата должен находиться в каталоге /etc/strongswan/ipsec.d/certs. Не перепутайте: если на сервере установлен (или был когда-либо установлен) openswan, то может присутствовать и каталог /etc/ipsec.d.

- **leftsubnet=10.54.1.0/24** – маршрутизируемая подсеть локальной стороны соединения.
- **leftca="..."** – полный Subject сертификата удостоверяющего центра, который выдал сертификат, используемый для аутентификации локальной стороны соединения. Проверка удостоверяющего центра делается только по данному Subject, никакие обычные приемы типа создания символических ссылок не требуются.
- **right=%any** – идентификатор удаленной стороны соединения. В данном случае записывается ключевое слово, обозначающее, что идентификатор будет заполнен в момент установления соединения.
- **rightallowany=yes** – параметр, который разрешает использование %any в качестве значения параметров left/right соответственно. Без задания данного параметра указание %any в качестве значения параметра left/right будет приводить к ошибке.
- **rightid="..."** – полный Subject сертификата, используемого для идентификации удаленной стороны соединения. Метод получения значения для параметра rightid был описан для параметра leftid.
- **rightcert=rb450g_deltahw.crt** – имя файла сертификата, используемого для аутентификации удаленной стороны. Файл сертификата точно так же должен находиться в каталоге /etc/strongswan/ipsec.d/certs.
- **rightsubnet=10.87.1.0/24** – маршрутизируемая подсеть удаленной стороны соединения. Поскольку она настраивается на нашем маршрутизаторе, она будет фактором постоянным, не изменяемым провайдером.
- **rightauth=pubkey** – метод аутентификации удаленной стороны – SSL-сертификат.
- **rightca="..."** – полный Subject сертификата удостоверяющего центра, который выдал сертификат, используемый для аутентификации удаленной стороны соединения. Проверка удостоверяющего центра делается только по данному Subject, никакие обычные приемы типа создания символических ссылок не требуются. Совсем не обязательно, чтобы сертификаты, используемые для аутентификации разных сторон соединения, были выданы одним и тем же удостоверяющим центром.
- **keyexchange=ikev1** – версия протокола IKE, используемого для установления соединения. Этот параметр не является обязательным и его нужно указывать только в том случае, если удаленная сторона не поддерживает версию IKEv2. Поскольку мы с вами не можем априори предположить, какую версию протокола поддерживает удаленная сторона, поэтому используется протокол меньшей версии.
- **ike="шифронаборы"** – перечень шифронаборов, используемых для шифрования первой фазы соединения в формате «шифр-хеш-группа Диффи-Хеллмана». Восклицательный знак в конце строки шифронаборов отключает дополнение этой строки значениями по умолчанию aes128-sha1-modp2048,3des-sha1-modp1536.
- **esp="шифронаборы"** – перечень шифронаборов, используемых для шифрования второй фазы соединения в том же формате, что и параметр ike. Полный перечень шифров, хешей и групп Диффи-Хеллмана, поддерживаемых strongSwan, можно найти в документации на программу.

Сеть B подключается с использованием роутера Mikrotik RB450G. Настройки его для работы с произвольного IP будут рассмотрены в следующей части. Рассматривается только вариант с использованием сертификатов.

Возьмемся за руки, друзья

Теперь давайте рассмотрим вариант подключения клиента, работающего под управлением FreeBSD, – по приводимой в начале статьи схеме это Сеть C. Описание соединения со стороны сервера (на сервере) будет приведено без комментариев – все используемые там параметры уже рассматривались.

```
conn vmfree-rsa
    auto=route
    left=212.20.5.1
    leftid="/C=RU/ST=Novosibirsk region/L=Novosibirsk/ ↵
         O=DeltaHardware Ltd/OU=Linux servers/ ↵
         CN=ipsec.deltahw.ru/ ↵
         emailAddress=root@deltahw.ru"
    leftauth=pubkey
    leftcert=ipsec_deltahw.crt
    leftsubnet=10.54.1.0/24
    right=140.40.40.1
    rightid="/C=RU/ST=Novosibirsk region/L=Novosibirsk/ ↵
         O=DeltaHwardware Ltd/ ↵
         OU=Web and mail and DNS server/ ↵
         CN=node3.deltahw.ru/ ↵
         emailAddress=root@deltahw.ru"
    rightcert=node3_deltahw.crt
    rightsubnet=10.14.1.0/24
    rightauth=pubkey
    keyexchange=ikev1
```

Это все нам уже знакомо, поэтому останавливаться на описании значений параметров не будем, сразу перейдем к рассмотрению работы strongSwan на FreeBSD.

Ставится strongSwan как обычно, из портов. Security/strongswan, последняя версия 5.4.0. При установке создается каталог /usr/local/etc/ipsec.d, в котором хранятся сертификаты, то есть в нем подкаталоги aacerts, acerts, certs и т.д., а также каталог strongswan.d, содержащий второстепенные файлы конфигурации – charon-logging.conf, charon.conf и т.д., а также подкаталог charon.

Основные файлы – ipsec.conf и strongswan.conf – размещаются прямо в каталоге /usr/local/etc.

Описание соединения делается точно таким же образом. Можно было бы добавить в конфигурацию leftca/rightca, но для инициатора соединения это особого смысла не имеет.

```
conn logsrv-rsa
    auto=start
    right=212.20.5.1
    rightid="/C=RU/ST=Novosibirsk region/L=Novosibirsk/ ↵
         O=DeltaHardware Ltd/OU=Linux servers/ ↵
         CN=ipsec.deltahw.ru/emailAddress=root@deltahw.ru"
    rightauth=pubkey
    rightcert=ipsec_deltahw.crt
    rightsubnet=10.54.1.0/24
    left=140.40.40.1
    leftid="/C=RU/ST=Novosibirsk region/L=Novosibirsk/ ↵
         O=DeltaHardware Ltd/ ↵
         OU=Web and mail and DNS server/ ↵
         CN=node3.deltahw.ru/emailAddress=root@deltahw.ru"
    leftcert=node3_deltahw.crt
    leftsubnet=10.14.1.0/24
    leftauth=pubkey
    keyexchange=ikev1
```

Ну а автоматический запуск strongSwan делается так, как это было всегда во FreeBSD, – добавлением строки в /etc/rc.conf:

```
strongswan_enable="YES"
```

Чем же лебедь, выросший среди снегов, отличается от лебедя, выросшего среди лавовых потоков?

По функционалу ничем – конфиги вполне переносимы и могут использоваться как в версии strongSwan на Linux, так и на FreeBSD. Отличие в основном в управляющих программах – в FreeBSD версии strongSwan отсутствует такая вещь, как swanctl, а вместо нее используется и без того здесь существовавшая программа ipsec:

```
# ipsec --help

ipsec command [arguments]

Commands:
        start|restart [arguments]
        ….
        scepclient|pki
        starter|stroke
        version
```

Ipsec заменяет swanctl во всех ее аспектах управления программой strongSwan, но для получения информации о соединениях она вряд ли годится – уж слишком мало информации предоставляет:

```
# ipsec statusall

mikrotik-rb1100ahx2-rsa{719}:  INSTALLED, TUNNEL, reqid 1,
ESP SPIs: cdbb0dd1_i 05a7d8bf_o
mikrotik-rb1100ahx2-rsa{719}:  AES_CBC_128/HMAC_SHA1_96,
249724 bytes_i (2251 pkts, 1168s ago), 4415664 bytes_o
(3318 pkts, 2s ago), rekeying in 24 minutes
```

Еще нужно отметить, что ipsec status/statusall помнит про все факты обмена ключами фазы 1 вплоть до перезапуска ОС, отчего и возникают числа типа 719.

Если необходима более подробная информация, то обращаются к программе setkey, которая обычно поставляется вместе с полной версией ipsec – она отображает полные данные точно так же, как она делала это всегда:

```
# setkey -D

140.40.40.1 212.20.5.1
    esp mode=tunnel spi=94886079(0x05a7d8bf)
    reqid=1(0x00000001)
    E: aes-cbc  e8df7e9b e6c57d3a 1fd2bc20 296d3b2e
    A: hmac-sha1  3eedd184 b5867eea fd26edd7 d97e9e6d 93a3a82d
    seq=0x00000e8c replay=4 flags=0x00000000 state=mature
    created: Mar 24 22:39:06 2016  current: Mar 24 23:12:22 2016
    diff: 1996(s)  hard: 3600(s)  soft: 2750(s)
    last: Mar 24 23:12:12 2016  hard: 0(s)  soft: 0(s)
    current: 4681904(bytes)  hard: 0(bytes)  soft: 0(bytes)
    allocated: 3724  hard: 0  soft: 0
    sadb_seq=1 pid=62992 refcnt=2

212.20.5.1 140.40.40.1
    esp mode=tunnel spi=3451588049(0xcdbb0dd1)
    reqid=1(0x00000001)
    E: aes-cbc  a16c5e50 98e540bb 9d0be99c 533f0d7a
    A: hmac-sha1  0ef35a89 f38a07b3 4b7379f9 a3da5b88 582e6481
    seq=0x000009c3 replay=4 flags=0x00000000 state=mature
    created: Mar 24 22:39:06 2016  current: Mar 24 23:12:22 2016
    diff: 1996(s)  hard: 3600(s)  soft: 2648(s)
    last: Mar 24 23:12:12 2016  hard: 0(s)  soft: 0(s)
    current: 305893(bytes)  hard: 0(bytes)  soft: 0(bytes)
    allocated: 2499  hard: 0  soft: 0
    sadb_seq=0 pid=62992 refcnt=1
```

Необходимость создавать туннельные интерфейсы давно отпала – хотя и может быть по-прежнему актуально, если использовать Racoon, но по крайней мере при соединении между двумя лебедями в отдельном туннельном интерфейсе нет необходимости.

А вот устройство enc использовать можно. Оно не настраивается, достаточно добавить в /etc/rc.conf:

```
ifconfig_enc0="up"
```

и, как и раньше, можно с помощью tcpdump отображать пакеты, проходящие через «интерфейс»:

```
# tcpdump -i enc0 -n

tcpdump: WARNING: enc0: no IPv4 address assigned
tcpdump: verbose output suppressed, use -v or -vv for full
protocol decode
listening on enc0, link-type ENC (OpenBSD encapsulated IP),
capture size 65535 bytes
23:21:42.816946 (authentic,confidential): SPI 0x05a7d8bf:
IP 10.14.1.1.58898 > 10.54.1.2.902: UDP, length 66
23:21:42.816958 (authentic,confidential): SPI 0x05a7d8bf:
IP 140.40.40.1 > 212.20.5.1: IP 10.14.1.1.58898 >
10.54.1.2.902: UDP, length 66 (ipip-proto-4)
23:21:45.233959 (authentic,confidential): SPI 0x05a7d8bf:
IP 10.14.1.1.63335 > 10.54.1.2.63653: Flags [P.], seq
3486298001:3486298009, ack 3798890528, win 252, length 8
```

Микротик и микротак

Последний вариант подключения, который мы рассмотрим, – это подключение к серверу роутера под управлением Mikrotik.

Настройка VPN с использованием протоколов семейства IPSec на Mikrotik начинается с установки сертификатов. Для их установки файл сертификата и файл ключа сертификата (PKCS#12 архивы не используются) помещаются в файловое хранилище – через ftp или winbox и далее импортируются через команду импорта сертификатов:

```
[admin@MikroTik] > /certificate import file-name ↵
        rb450g_deltahw.crt
```

Импортировать нужно не только сертификат устройства, но и ключ сертификата устройства, сертификат удаленного устройства и сертификат удостоверяющего центра, выдавшего все сертификаты локальных и удаленных устройств. Если сертификаты выданы разными удостоверяющими центрами, то необходимо установить все сертификаты.

После установки всех сертификатов их список в Mikrotik может выглядеть примерно так:

```
[admin@MikroTik] /certificate> print

Flags: K - private-key, D - dsa, L - crl, C - smart-card-key,
A - authority, I - issued, R - revoked, E - expired, T - trusted
```

```
 #    NAME    COMMON-NAME    SUBJECT-ALT-NAME    FINGERPRINT
 0    T DeltaHW Root CA    DeltaHardware Ltd Root CA
email:root@deltahw.ru    1f0ba1956596ec4a9de846a477...
 1    K L    T RB450G cert with key    rb450g.deltahw.ru
email:root@deltahw.ru    02642dbfa04534cf6d4098d834...
 2    L    T RB1100AHx2 cert    ipsec.deltahw.ru
email:root@deltahw.ru    2cf9f870799cd4636597b3f22c..
```

При настройке Mikrotik нам придется сделать значительно больше ручной работы, которую при использовании strongSwan берет на себя лебедь собственной персоной.

Первое – это создание proposal. Я много раз пытался перевести этот термин, но терминология ipsec уже устоялась, и ничего лучше, чем «предложение», в голову не приходит – каждая сторона предлагает другой стороне информацию о том, что она умеет – какие использует алгоритмы шифрования, алгоритмы расчета электронной подписи и Диффи-Хеллмана.

```
[admin@MikroTik] > /ip ipsec proposal
add name="proposal1" auth-algorithms=sha1 ↵
        enc-algorithms=aes-128-cbc,aes-192-cbc, ↵
        aes-256-cbc lifetime=30m pfs-group=modp1024
```

Здесь мы создали «предложение» с именем proposal1, в котором заявляем, что можем использовать указанные алгоритмы шифрования, электронной подписи и Диффи-Хеллмана, а также задаем время жизни Фазы 2, равное 30 минутам. В strongSwan это задавалось параметром lifetime. В устройстве, правда, уже имеется «предложение по умолчанию» с именем default, зачем же нам создавать еще одно?

Дело в том, что в устройствах Mikrotik какая-то странная реализация алгоритма электронной подписи SHA256, из-за чего работает она только при установке соединения микротика с микротиком, поэтому приходится использовать более старый и менее безопасный алгоритм SHA1

Все параметры можно и не указывать – если параметр опустить, то он просто принимается значением по умолчанию. Именно поэтому в скрипте инициализации встречаются иногда довольно странные команды. Например, для создания описанного выше «предложения» достаточно было написать:

```
/ip ipsec proposal
add name=proposal1
```

И оно было бы создано со всеми параметрами по умолчанию. Чем же оно отличается от default? Да тем, что default вовсе на самом деле не default – оно просто автоматически создается при инициализации устройства.

Вторым шагом мы создаем политику шифрования. Все параметры – адреса подсетей и IP-адреса роутеров – нам придется указывать вручную.

```
[admin@MikroTik] > /ip ipsec policy
add src-address=10.87.1.0/24 src-port=any ↵
        dst-address=10.54.1.0/24 dst-port=any ↵
        protocol=all action=encrypt level=unique ↵
        ipsec-protocols=esp tunnel=yes ↵
        sa-src-address=170.70.70.1 ↵
        sa-dst-address=212.20.5.1 proposal=proposal1
```

Как обычно, многие параметры можно не указывать.

Обратите внимание, что именно политика, а вовсе не удаленная сторона, связывается с «предложением», в котором задается длительность фазы.

Ну и последним шагом мы создаем описание удаленной стороны. При его создании используется множество параметров, и большая часть из них должна быть указана:

```
/ip ipsec peer
add address=212.20.5.1/32 passive=no port=500 ↵
        auth-method=rsa-signature certificate=RB450G ↵
        cert with key remote-certificate=IPSec Deltahw ↵
        generate-policy=no exchange-mode=main ↵
        send-initial-contact=yes nat-traversal=yes ↵
        proposal-check=strict hash-algorithm=sha1 ↵
        enc-algorithm=aes-128,aes-192,aes-256 ↵
        dh-group=modp1024 lifetime=1d ↵
        dpd-interval=disable-dpd
```

Большинство параметров здесь напоминает соответствующие параметры в strongSwan, хотя, конечно же, есть и серьезные различия, та же часть, которая их не напоминает, кажется будто взята напрямую из описания по ipsec-tools.

- **passive=no** – указывает на то, что соединение должно активироваться немедленно.
- **lifetime-1d** – задает время жизни Фазы 1 для всех тех условий, которые были согласованы при активации «предложения».
- **exchange-mode=main** – начало обмена между клиентом и сервером будет построено по стандартному алгоритму обмена (в противовес облегченному – aggressive mode).
- **dpd-interval=disable-dpd** – отключение обнаружения неработающих удаленных объектов – при включении этого параметра, если с устройством не было связи больше, чем на N секунд, устройство обнаруживает это.

Поскольку было указано passive=no, то установление соединения к данному удаленному устройству начнется немедленно.

• • •

Итак, мы рассмотрели подключение к серверу на Linux трех вариантов клиентов – Linux с динамическим IP, FreeBSD и Mikrotik, причем FreeBSD и Linux используют одинаковую программу strongSwan, Mikrotik же судя по параметрам использует некоторую модифицированную версию ipsec-tools. Фактически мы заново собрали всю основную сеть, указанную на схеме, осталось рассмотреть только другие, менее популярные, но возможные варианты. EOF

[1] Ачилов Р. Построение корпоративных VPN. Часть 10. Связь Linux-Linux с помощью strongSwan. // «Системный администратор», №1-2, 2016 г. – С. 23-27 (http://samag.ru/archive/article/3113).

[2] Сайт проекта FreeS/WAN – http://www.freeswan.org/download.html.

[3] Документация по Openswan – https://github.com/xelerance/Openswan/wiki.

[4] Сайт проекта strongSwan – https://www.strongswan.org.

[5] Полная схема прохождения пакетов через iptables, включающая XFRM (обработку IPSec-пакетов) – https://commons.wikimedia.org/wiki/File:Netfilter-packet-flow.svg.

Ключевые слова: strongSwan, VPN, Linux, безопасность, сети.

EXPO COMM™ RUSSIA

«Компоненты цифрового бизнеса – Драйверы роста Экономики России»

Международная выставка и конференция перспективных информационных и коммуникационных технологий

19 - 21 апреля 2016

Конгресс-центр ЦМТ Москвы

Expo Comm Russia 2016 это:

- **500 УЧАСТНИКОВ** из **15 СТРАН**
- **12 000 УНИКАЛЬНЫХ ПОСЕТИТЕЛЕЙ** – директора и владельцы бизнесов, CTO, CIO, менеджеры среднего и высшего звена, технические специалисты, эксперты телеком- и IT-рынка
- **МЕЖДУНАРОДНЫЙ ИНФО-МЕДИА КОММУНИКАЦИОННЫЙ ФОРУМ** – самые актуальные темы, острые проблемы отрасли и варианты решений в режиме открытого диалога, курс на импортозамещение, развитие технологий внутри страны, IT-стартапы и инвестиции
- **УЧАСТИЕ И ПОДДЕРЖКА ПРОФИЛЬНЫХ МИНИСТЕРСТВ И ОТРАСЛЕВЫХ АССОЦИАЦИЙ**

Основные тематические разделы:

- **ИНФРАСТРУКТУРА СВЯЗИ**
- **КОММУНИКАЦИИ КАК УСЛУГА**
- **ЦИФРОВЫЕ СЕРВИСЫ**
- **ИКТ – ДОСТУПНОСТЬ**

Сферы деятельности посетителей:

- **Финансовый сектор, банки, консалтинг, страхование**
- **Пищевая промышленность**
- **СМИ, аналитики и эксперты**
- **Телеком, ИТ, медиа**
- **Дистрибуция, оптовая и розничная торговля**
- **Государственное и муниципальное управление**
- **Металлообработка**
- **Автомобилестроение, авиастроение**
- **Медицина**
- **Туризм**
- **Энергетика, нефтегаз**
- **Транспорт и логистика**
- **Строительство**

+7 (495) 649-69-21
expocomm@ejkrause.ru

www.expocomm.ru

Организатор:

EJK® E. J. Krause & Associates, Inc.

Соорганизатор:

ITE

Организатор деловой программы:

TMT Conference
Telecom. Media. Technologies

При поддержке:

МИНПРОМТОРГ РОССИИ

Департамент информационных технологий города Москвы

Ростех Информ

СТРА

RUS•SOFT

ОПКИТ

ГЛОНАСС/ГНСС Форум

HD

AEB Association of European Businesses

Визитка

ВЕНИАМИН ЛЕВЦОВ,
вице-президент, глава корпоративного дивизиона «Лаборатории Касперского»

Визитка

НИКОЛАЙ ДЕМИДОВ,
технический консультант по информационной безопасности «Лаборатории Касперского»

Анатомия таргетированной атаки
Часть 1

С каждым годом организации совершенствуют инструменты ведения бизнеса, внедряя новые решения, одновременно усложняя ИТ-инфраструктуру. Теперь, когда в компании зависает почтовый сервер, с конечных рабочих мест стирается важная информация или нарушается работа автоматизированной системы формирования счетов к оплате, бизнес-процессы просто останавливаются

Осознавая растущую зависимость от автоматизированных систем, бизнес также готов все больше заботиться об обеспечении информационной безопасности. Причем путь создания системы ИБ зависит от ситуации в данной конкретной организации – от имевших место инцидентов, убеждений конкретных сотрудников – и зачастую формируется «снизу», от отдельных подсистем ИБ к общей картине.

В результате создается многоступенчатая единственная в своем роде система, состоящая из различных продуктов и сервисных работ, сложная, как правило, уникальная у каждой компании, где специалисты по ИБ могут:

- проверять файлы с помощью систем безопасности конечных точек;
- фильтровать почтовый и веб-трафик с помощью шлюзовых решений;
- отслеживать целостность и неизменность файлов и системных настроек;
- контролировать поведение пользователей и реагировать на отклонения от обычной модели трафика;
- сканировать периметр и внутреннюю сеть на предмет уязвимостей и слабых конфигураций;
- внедрять системы идентификации и аутентификации, шифровать диски и сетевые соединения;
- инвестировать в SOC для сбора и корреляции логов и событий от упомянутых выше подсистем;
- заказывать тесты на проникновение и иные сервисы для оценки уровня защищенности;
- приводить систему в соответствие с требованиями стандартов и проводить сертификации;
- учить персонал основам компьютерной гигиены и решать еще бесконечное множество подобных задач.

Но, несмотря на все это, количество успешных, т.е. достигающих своей цели атак на ИТ-инфраструктуры не уменьшается, а ущерб от них растет. За счет чего же удается злоумышленникам преодолевать сложные системы безопасности, как правило, уникальные по своему составу и структуре?

Ответ довольно краток: за счет подготовки и проведения сложных атак, учитывающих особенности целевой системы.

Понятие целевой атаки

Самое время дать определение, точно, по нашему мнению, отражающее понятие целевой, или таргетированной, атаки.

Целевая атака – это непрерывный процесс несанкционированной активности в инфраструктуре атакуемой системы, удаленно управляемый в реальном времени вручную.

Во-первых, это именно процесс – деятельность во времени, некая операция, а не просто разовое техническое действие. Проведя анализ подобных атак, эксперты «Лаборатории Касперского» отмечают, что их длительность составляет от 100 дней и больше.

Во-вторых, процесс направлен для работы в условиях конкретной инфраструктуры, призван преодолеть конкретные механизмы безопасности, определенные продукты, вовлечь во взаимодействие конкретных сотрудников. Следует отметить существенную разницу в подходе массовых рассылок стандартного вредоносного ПО, когда злоумышленники преследуют совсем другие цели – по сути, получение контроля над отдельной конечной точкой. В случае целевой атаки она строится под жертву.

В-третьих, эта операция обычно управляется организованной группой профессионалов, порой международной, вооруженной изощренным техническим инструментарием, по сути своей – бандой. Их деятельность действительно бывает очень похожа на многоходовую войсковую операцию. Например, злоумышленниками составляется список сотрудников, которые потенциально могут стать «входными воротами» в компанию, с ними устанавливается связь в социальных сетях, изучаются их профили. После этого решается задача получения контроля над рабочим компьютером жертвы. В результате его компьютер заражен, и злоумышленники переходят к захвату контроля над сетью и непосредственно преступным действиям.

В ситуации целевой атаки не компьютерные системы бьются друг с другом, а люди: одни нападают, другие отражают хорошо подготовленное нападение, учитывающее слабые стороны и особенности систем противодействия.

В настоящее время все большее распространение получает термин APT – Advanced Persistent Threat. Давайте разберемся и с его определением.

APT – это комбинация утилит, вредоносного ПО, механизмов использования уязвимостей «нулевого дня», других компонентов, специально разработанных для реализации данной атаки.

Практика показывает, что APT используются повторно и многократно в дальнейшем для проведения повторных атак, имеющих схожий вектор, против уже других организаций.

Целевая, или таргетированная, атака – это процесс, деятельность. APT – техническое средство, позволяющее реализовать атаку.

Можно смело утверждать, что активное распространение целевых атак обусловлено, в том числе, и сильным сокращением стоимости и трудозатрат в реализации самой атаки. Большое количество ранее разработанных инструментов доступно хакерским группировкам, порой отсутствует острая необходимость создавать экзотические вредоносные программы с нуля. В большинстве своем современные целевые атаки построены на ранее созданных эксплойтах и вредоносном ПО, лишь малая часть использует совершенно новые техники, которые преимущественно относятся к угрозам класса APT. Порой в рамках атаки используются и совершенно легальные, созданные для «мирных» целей утилиты – ниже мы вернемся к этому вопросу.

Стадии целевой атаки

В этом материале нам хочется озвучить основные этапы таргетированной атаки, заглянуть внутрь, показать скелет общей модели и различия применяемых методов проникновения. В экспертном сообществе сложилось представление о том, что целевая атака, как правило, в своем развитии проходит через четыре фазы (см. рис. 1).

На приведенном рисунке отображены четыре фазы целевой атаки, демонстрирующие ее жизненный цикл. Кратко сформулируем основное назначение каждой из них:

- **Подготовка** – основная задача первой фазы – найти цель, собрать о ней достаточно детальной приватной информации, опираясь на которую, выявить слабые места в инфраструктуре. Выстроить стратегию атаки, подобрать ранее созданные инструменты, доступные на черном рынке, либо разработать необходимые самостоятельно. Обычно планируемые шаги проникновения будут тщательно протестированы, в том числе на необнаружение стандартными средствами защиты информации.
- **Проникновение** – активная фаза целевой атаки, использующая различные техники социальной инженерии и уязвимостей нулевого дня для первичного инфицирования цели и проведения внутренней разведки. По окончании разведки и определения принадлежности инфицированного хоста (сервер/рабочая станция) по команде злоумышленника через центр управления может загружаться дополнительный вредоносный код.
- **Распространение** – фаза закрепления внутри инфраструктуры преимущественно на ключевые машины жертвы. Максимально распространяя свой контроль, при необходимости корректируя версии вредоносного кода через центры управления.
- **Достижение цели** – ключевая фаза целевой атаки, в зависимости от выбранной стратегии в ней могут применяться:
  - хищение закрытой информации;
  - умышленное изменение закрытой информации;
  - манипуляции с бизнес-процессами компании.

На всех этапах выполняется обязательное условие по сокрытию следов активности целевой атаки. При завершении атаки часто бывает, что киберпреступники создают для себя «Точку возврата», позволяющую им вернуться в будущем.

> Активное распространение целевых атак обусловлено сильным сокращением стоимости и трудозатрат в реализации самой атаки

Первая фаза целевой атаки – Подготовка

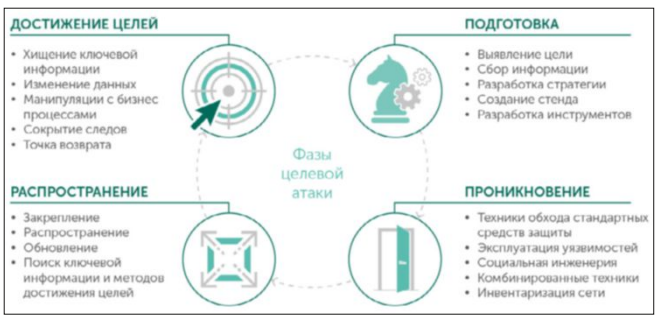
Выявление цели

Целью для атаки может стать любая организация. А начинается все с заказа или общей разведки или, точнее, мониторинга. В ходе продолжительного мониторинга мирового бизнес-ландшафта хакерские группы используют общедоступные инструменты, такие как RSS-рассылки, официальные Twitter-аккаунты компаний, профильные форумы, где обмениваются информацией различные сотрудники. Все это помогает определить жертву и задачи атаки, после чего ресурсы группы переходят к этапу активной разведки.

Сбор информации

По понятным причинам ни одна компания не предоставляет сведения о том, какие технические средства она использует, в том числе защиты информации, внутренний регламент и т.д. Поэтому процесс сбора информации о жертве называется «разведка». Основная задача разведки, сбор целевой

Рисунок 1. Фазы целевой атаки



приватной информации о жертве. Тут важны все мелочи, которые помогут выявить потенциальные слабые места. В работе могут быть использованы самые нетривиальные подходы для получения закрытых первичных данных, например, социальная инженерия. Мы приведем несколько техник социальной инженерии и иных механизмов разведки, применяемых на практике.

Важно отметить рост целевых атак, направленных против компаний самых различных секторов рынка

Способы проведения разведки:

Инсайд. Существует подход с поиском недавно уволенных сотрудников компании. Бывший сотрудник компании получает приглашение на обычное собеседование на очень заманчивую позицию. Мы знаем, что опытный психолог-рекрут в состоянии разговорить почти любого сотрудника, который борется за позицию. От таких людей получают достаточно большой объем информации для подготовки и выбора вектора атаки: от топологии сети и используемых средствах защиты до информации о частной жизни других сотрудников.

Бывает, что киберпреступники прибегают к подкупу нужных им людей в компании, владеющих информацией, либо входят в круг доверия путем дружеского общения в общественных местах.

Открытые источники. В этом примере хакеры используют недобросовестное отношение компаний к бумажным носителям информации, которые выбрасывают на помойку без правильного уничтожения, среди мусора могут быть найдены отчеты и внутренняя информация, или, например, сайты компании, которые содержат реальные имена сотрудников в общем доступе. Полученные данные можно будет комбинировать с другими техниками социальной инженерии.

В результате этой работы организаторы атаки могут иметь достаточно полную информацию о жертве, включая:

- имена сотрудников, email, телефон;
- график работы подразделений компании;
- внутреннюю информацию о процессах в компании;
- информацию о бизнес-партнерах.

Государственные порталы закупок также являются хорошим источником получения информации о решениях, которые внедрены у заказчика, в том числе о системах защиты информации. На первый взгляд приведенный пример может показаться несущественным, но на самом деле это не так. Перечисленная информация с успехом применяется в методах социальной инженерии, позволяя хакеру легко завоевать доверие, оперируя полученной информацией.

Социальная инженерия.

- Телефонные звонки от имени внутренних сотрудников.
- Социальные сети.

Используя социальную инженерию можно добиться значительного успеха в получении закрытой информации компании: например, в случае телефонного звонка злоумышленник может представиться от имени работника информационной службы, задать правильные вопросы или попросить выполнить нужную команду на компьютере. Социальные сети хорошо помогают определить круг друзей и интересы нужного человека, такая информация может помочь киберпреступникам выработать правильную стратегию общения с будущей жертвой.

Разработка стратегии

Стратегия является обязательной в реализации успешной целевой атаки, она учитывает весь план действий на всех стадиях атаки:

- описание этапов атаки: проникновение, развитие, достижение целей;
- методы социальной инженерии, используемые уязвимости, обход стандартных средств безопасности;
- этапы развития атаки с учетом возможных внештатных ситуаций;
- закрепление внутри, повышение привилегий, контроль над ключевыми ресурсами;
- извлечение данных, удаление следов, деструктивные действия.

Создание стенда

Опираясь на собранную информацию, группа злоумышленников приступает к созданию стенда с идентичными версиями эксплуатируемого ПО. Полигон, дающий возможность опробовать этапы проникновения уже на действующей модели. Отработать различные техники скрытого внедрения и обхода стандартных средств защиты информации. По сути, стенд служит главным мостом между пассивной и активной фазами проникновения в инфраструктуру жертвы. Важно отметить, что создание подобного стенда обходится недешево для хакеров. Затраты на выполнение успешной целевой атаки возрастают с каждым этапом.

Разработка набора инструментов

Перед киберпреступниками встает непростой выбор: им важно определиться между финансовыми затратами на покупку уже готовых инструментов на теневом рынке и трудозатратами и временем для создания собственных. Теневой рынок предлагает достаточно широкий выбор различный инструментов, что значительно сокращает время, за исключением уникальных случаев. Это второй шаг, который выделяет целевую атаку как одну из самых ресурсоемких среди кибератак.

Рассмотрим набор инструментов в деталях. Как правило, Toolset состоит из трех основных компонентов:

1. Командный центр, или Command and Control Center (C&C). Основой инфраструктуры атакующих являются командно-контрольные центры C&C, обеспечивающие передачу команд подконтрольным вредоносным модулям, с которых они собирают результаты работы. Центром атаки являются люди, проводящие атаку. Чаще всего центры располагаются в интернете у провайдеров, предоставляющих услуги хостинга, колокации и аренды виртуальных машин. Алгоритм обновления, как и все алгоритмы взаимодействия с «хозяевами», могут меняться динамически вместе с вредоносными модулями.

2. Инструменты проникновения решают задачу «открытия двери» атакуемого удаленного хоста:

- **Эксплойт (Exploit)** – вредоносный код, использующий уязвимости в программном обеспечении.
- **Валидатор** – вредоносный код применяется в случаях первичного инфицирования, способен собрать информацию о хосте, передать ее C&C для дальнейшего принятия решения о развитии атаки либо полной ее отмене на конкретной машине.
- **Загрузчик (Downloader)** модуля доставки Dropper. Загрузчик крайне часто используется в атаках, построенных на методах социальной инженерии, отправляется вложением в почтовых сообщениях.
- **Модуль доставки Dropper** – вредоносная программа (как правило, троян), задачей которой является доставка основного вируса Payload на зараженную машину жертвы, предназначена для:
  - закрепления внутри зараженной машины, скрытой автозагрузки, инжектирования процессов после перезагрузки машины;
  - Inject в легитимный процесс для закачки и активации вируса Payload по шифрованному каналу либо извлечения и запуска зашифрованной копии вируса Payload с диска.

Исполнение кода протекает в инжектированном легитимном процессе с системными правами, такая активность крайне сложно детектируется стандартными средствами безопасности.

3. Тело вируса Payload. Основной вредоносный модуль в целевой атаке, загружаемый на инфицированный хост Dropper, может состоять из нескольких функциональных допмодулей, каждый из которых будет выполнять свою функцию:

- клавиатурный шпион;
- запись экрана;
- удаленный доступ;
- модуль распространения внутри инфраструктуры;
- взаимодействие с C&C и обновление;
- шифрование;
- очистка следов активности, самоуничтожение;
- чтение локальной почты;
- поиск информации на диске.

Как мы видим, потенциал рассмотренного набора инструментов впечатляет, а функционал модулей и используемых техник может сильно отличаться в зависимости от планов целевой атаки. Данный факт подчеркивает уникальность такого рода атак.

• • •

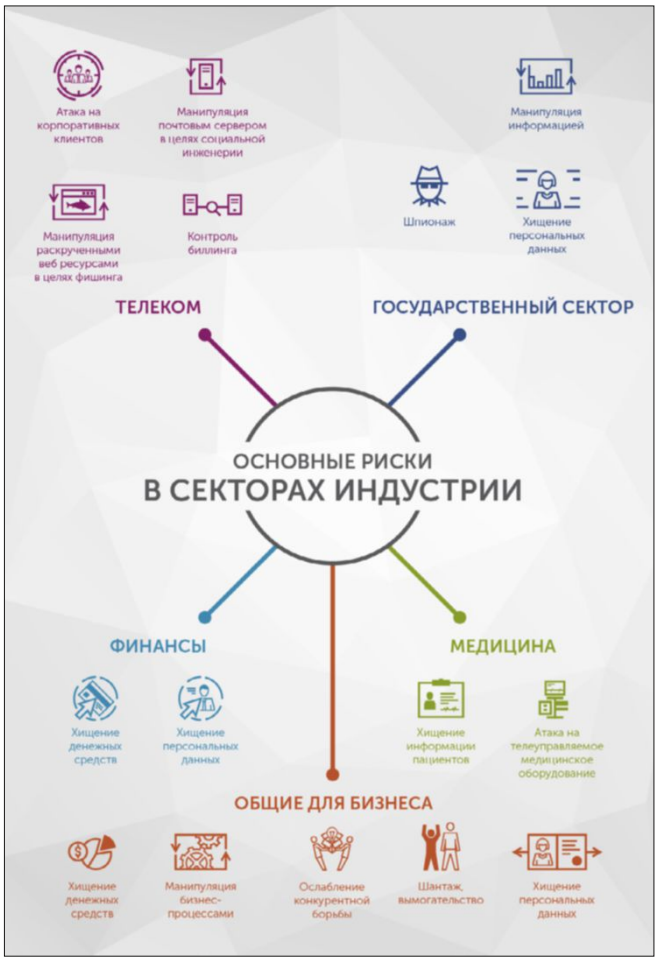
Подводя итог, важно отметить рост целевых атак, направленных против компаний самых различных секторов рынка, высокую сложность их обнаружения и колоссальный урон от их действий, который не гарантированно может быть обнаружен спустя длительный срок. По статистике «Лаборатории Касперского», в среднем обнаружение целевой атаки происходит спустя 200 дней с момента ее активности, это означает, что хакеры не только достигли своих целей, но и контролировали ситуацию на протяжении более чем половины года.

Также организации, выявившие факт присутствия APT в своей инфраструктуре, не способны правильно реагировать и минимизировать риски и нейтрализовать активность: такому просто не обучают персонал, отвечающий за информационную безопасность. Вследствие этого каждая третья компания не на одну неделю приостанавливает свою деятельность в попытках вернуть контроль над собственной инфраструктурой, затем сталкиваясь со сложным процессом расследования инцидентов.

По данным опроса «Лаборатории Касперского», потери в результате крупного инцидента составляют в среднем по миру $551 000 для корпорации: в эту сумму входят упущенные для бизнеса возможности и время простоя систем, а также расходы на профессиональные сервисы для ликвидации последствий. (*Данные исследования «Информационная безопасность бизнеса», проведенного «Лабораторией Касперского» и B2B International в 2015 году. В исследовании приняли участие более 5500 ИТ-специалистов из 26 стран мира, включая Россию.*)

О том, как развивается атака, о методах обхода стандартных средств защиты и эксплуатации угроз нулевого дня, социальной инженерии, распространении и сокрытии следов при хищении ключевой информации и о многом другом – в следующих статьях цикла «Анатомия таргетированной атаки». EOF

Рисунок 2. Основные риски в секторах индустрии



Визитка

АНДРЕЙ БИРЮКОВ,
ЗАО «НИП Информзащита», системный архитектор, mex_inet@rambler.ru

Проводим пентест
Часть 1. Проникаем в беспроводную сеть

Насколько защищена ваша беспроводная сеть? Изучим различные способы проведения тестов на проникновение в нее

Программное обеспечение постоянно развивается, создается множество новых приложений и версий операционных систем. Однако быстрая разработка ПО оставляет немного времени для тестирования программного обеспечения, в первую очередь из-за сложности кода в современных приложениях, а также из-за человеческого фактора. В результате в коде приложений остаются ошибки, которые впоследствии превращаются в уязвимости ПО. Проблема выявления уязвимостей актуальна не только для разработчиков софта и специалистов по информационной безопасности, но и для технических специалистов, внедряющих и обслуживающих бизнес-приложения.

Этой статьей я открываю серию публикаций, посвященных проведению аудитов и тестов на проникновение с помощью свободно распространяемого ПО. В основном речь пойдет об инструментах, входящих в состав дистрибутива Kali Linux [1]. Статьи будут носить практический характер, на реальных примерах разберем особенности реализации тех или иных техник проведения тестов на проникновение (пентестов). Для самостоятельной проработки данного материала я рекомендовал бы читателю развернуть Kali Linux на ноутбуке, так как по крайней мере для этой статьи нам потребуется физический беспроводной адаптер. Для последующих статей рекомендую установить ВМ с ОС Windows XP SP2. Использование столь старой ОС обусловлено тем, что имеющиеся в ней уязвимости широко известны, и их легко можно эксплуатировать в учебных целях. Поиск уязвимостей в более новых системах может повлечь ненужную критику со стороны разработчиков и привести к реальным взломам рабочих бизнес-приложений.

Как будем ломать

Существуют различные подходы к проверке защищенности сети. В некоторых случаях используют методику «черного ящика», когда взломщик ничего не знает об атакуемой сети и ему необходимо произвести «тест на проникновение» (пентест). Как правило, это делается удаленно. В других случаях аудиторам предоставляется возможность проникнуть на территорию предприятия, но не предоставляется никаких прав, только сетевой порт («серый ящик»). Далее взломщику необходимо осуществить тот же пентест. И, наконец, вариант «белого ящика», когда аудитору предоставляется доступ ко всем настройкам сетевых устройств и приложений, и в таком случае необходимо проанализировать правильность настроек безопасности и выдать соответствующие рекомендации.

В своих статьях я буду использовать преимущественно методику «черного и серого ящиков», предполагая, что мы ничего не знаем о целевых системах. Это полезно даже для системных администраторов и специалистов по безопасности – взглянуть на сеть своей организации глазами взломщика, посмотреть на то, как он может попытаться проникнуть в сеть и как от этого защититься.

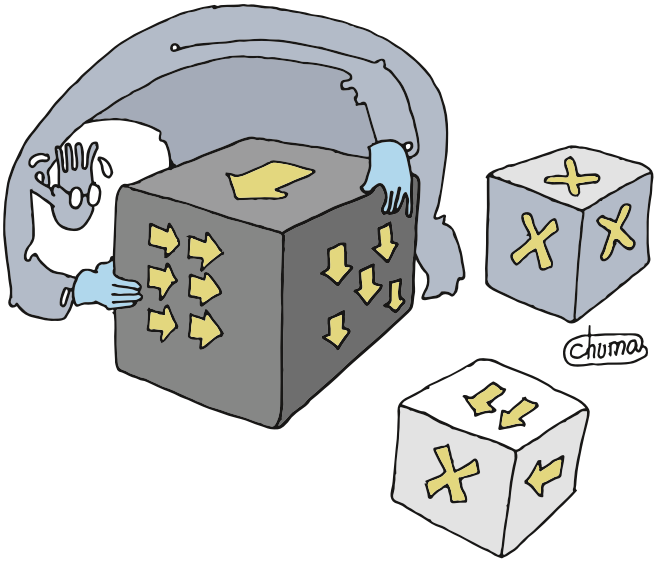
Прежде чем начать, напомню о существовании статьи 272 УК РФ «Неправомерный доступ к компьютерной информации». Все описанное в статье носит исключительно ознакомительный характер и предназначено для проведения аудитов собственных беспроводных сетей.

Ищем вход

На этом, я думаю, с теорией можно закончить и перейти к практической части, а именно к проведению аудита безопасности. Однако прежде чем начать анализ содержимого корпоративной сети атакуемой организации, злоумышленнику необходимо проникнуть в эту сеть. Попытаться сделать это можно несколькими различными способами. Мы не будем рассматривать рассылку троянских программ и другого вредоносного ПО, так как это отдельная тема, весьма скользкая с точки зрения законодательства. А вот беспроводные сети сейчас имеются во множестве организаций, и зачастую с их помощью проникнуть в корпоративную сеть значительно проще.

Начинаем работу с Kali Linux

Для начала аудита нам потребуется дистрибутив Kali Linux, ISO-образ которого необходимо развернуть на флешку или DVD-диск и загрузиться с данного носителя. При этом устанавливать ОС на диск не обязательно,



Всегда полезно взглянуть на сеть своей организации глазами взломщика

хотя если есть свободный ноутбук, то лучше это сделать, так как многим программам требуется дополнительная настройка при запуске. Учетные данные по умолчанию root/toor.

Посмотрим список всех доступных беспроводных интерфейсов:

```
# airmon-ng
```

Выбираем беспроводную сетевую карту и запускаем ее:

```
# airmon-ng start wlan1
```

Далее необходимо перевести ее в режим мониторинга:

```
# ifconfig wlan1mon down
# iwconfig wlan1mon mode monitor
# ifconfig wlan1mon up
# airodump-ng wlan1mon
```

Теперь можем просмотреть доступные беспроводные сети (см. рис. 1).

Из приведенного на рисунке вывода команды нам наиболее интересно содержимое полей BSSID, ESSID и CH (номер канала). Также в поле ENC (шифрование) указан алгоритм, который используется для шифрования. Далее необходимо нажать <Ctrl> + <C> для остановки работы утилиты.

В случае если сеть открытая (OPN), к ней можно пробовать подключиться сразу. Если же используется шифрование WEP, WPA или WPA2, то придется воспользоваться дополнительными утилитами для взлома ключа.

Такой безопасный WPA

Сначала рассмотрим ситуацию, когда используется шифрование WPA/WPA2, так как на сегодняшний день это наиболее распространенная защита Wi-Fi [2].

Для осуществления атаки нам необходимо скопировать нужные BSSID и выполнить следующую команду:

```
# airodump-ng -c [номер_канала] -bssid [bssid] ↵
         -w /root/Desktop/ wlan1mon
```

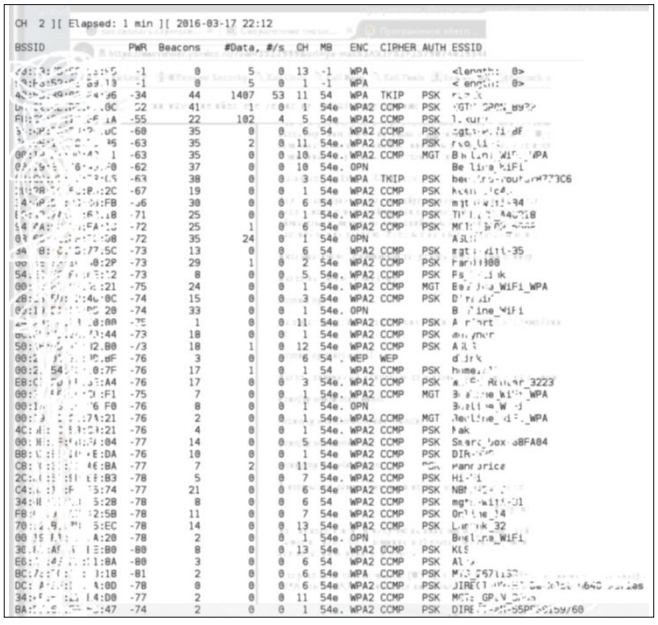
Например:

```
# airodump-ng -c 11 --bssid 01:01:01:01:01:01 ↵
         -w /root/Desktop/ wlan1mon
```

Утилита airodump начала мониторинг выбранного канала. Нам нужно собрать информацию об установке соединения. Для этого можно, конечно, дождаться, что к сети подключится другой пользователь, а можно просто сбросить сессию уже подключившегося пользователя (deauth) для того, чтобы он вынужден был подключиться заново, и в результате нам удалось бы собрать необходимую информацию.

```
# aireplay-ng -0 2 -a [router bssid] ↵
         -c [client bssid] wlan1mon
```

Рисунок 1. Обнаруженные сети



Здесь:
- **параметр -0** – означает переход в режим deauth,
- **2** – число передаваемых для деаутентификации пакетов,
- **параметр -a** – указывает BSSID,
- **параметр -c** – это MAC-адрес клиента.

Например:

```
# aireplay-ng -0 2 -a 01:01:01:01:01:01 ↵
         -c 02:02:02:02:02:02 wlan1mon
```

Если все было выполнено корректно, то в результате выполнения команды мы должны получить примерно следующее (см. рис. 2).

В результате наших манипуляций с пользовательскими сессиями был перехвачен некоторый зашифрованный трафик, который сохранили в /root/Desktop/*.cap файлах. Расшифровка данного трафика позволит узнать ключ сети. Попробовать взломать можно несколькими способами. Сейчас мы попробуем вскрыть шифр с помощью перебора по словарю. О других методах мы поговорим чуть позже.

```
# aircrack-ng -a2 -b [router bssid] -w [path to wordlist] ↵
         /root/Desktop/*.cap
```

Здесь:
- **a2** – это взламываемый алгоритм шифрования (WPA),
- **-b** – DSSID,
- **-w** – это словарь.

На просторах интернета имеется множество готовых словарей, кроме того, генератор словарей при необходимости можно написать самостоятельно.

Итак, если в результате выполнения данной атаки удалось узнать пароль от беспроводной сети, есть повод серьезно задуматься о степени ее защищенности. Однако даже если вы используете шифрование WPA с достаточно сложным и длинным ключом, расслабляться все равно рано, так как существуют другие атаки. Вот пример одной из них.

Черный ход WPS

Вернемся к выводу команды airodump-ng wlan1mon. На рис. 1 можно увидеть, что на некоторых точках доступа включен режим WPS. В случае правильного введения пина точка доступа сама предоставит данные для аутентификации (в т.ч. WPA PSK). Для подбора IPN в состав Kali Linux входит утилита Reaver. Собственно PIN – это восьмизначное число, которое можно вводить в любое время – каких-либо действий на стороне точки доступа не требуется. Для восьмизначных чисел возможно 10^8 (100 000 000) вариантов. Но последняя цифра не является случайно, она рассчитывается по алгоритму, т.е., говоря простым языком, последнюю цифру мы всегда знаем, и количество возможных вариантов сокращается до 10^7 (10 000 000). При этом искомый PIN делится на две половины, и каждая из этих половин проверяется индивидуально. Это означает, что для первой половины 10^4 (10 000) возможных вариантов, а для второй – всего 10^3 (1 000), т.к. последняя цифра не является случайной. Как видно, здесь в отличие от подбора ключа шифрования по словарю шансов на успех гораздо больше.

Утилита Reaver работает следующим образом: сначала подбирается первая половина кода PIN, а затем вторая. Скорость, с которой Reaver тестирует номера пинов, полностью зависит от скорости, с которой точка доступа может обрабатывать запросы. Некоторые достаточно быстрые – можно тестировать по одному пину в секунду, другие – медленнее, они позволяют вводить только один пин в 10 секунд. Но по собственному опыту могу сказать, что некоторые модели точек доступа, используемые провайдерами для доступа в интернет по GPON, при переборе PIN начинают мигать красным сигналом, который при обычной работе не горит. Это может стать дополнительным сигналом о подозрительной активности в сети.

Запустим перебор PIN.

```
# reaver -i wlan1mon -b [BSSID]
```

По умолчанию Reaver имеет задержку в 1 секунду между попытками пина. Для отключения этой задержки необходимо добавить -d 0 к командной строке, но некоторые точки доступа не любят этого:

```
# reaver -i wlan1mon -b [BSSID] -d 0
```

Другая опция, которая может ускорить атаку, это -dh-small. С помощью этой опции Reaver способен несколько ускорить перебор с использованием групп Диффи-Хелманна:

```
# reaver -i wlan1mon -b 01:01:01:01:01:01 --dh-small
```

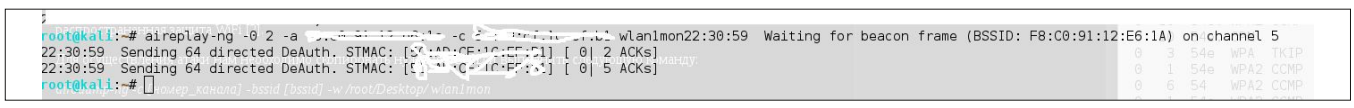
В среднем перебор может занять до трех часов, так что за ночь осуществить такую атаку вполне реально.

Когда WPS нет

Что делать в случае если используется WPA, но пароль не из словаря и WPS не включен? То есть мы собрали пакеты с помощью airodump, и у нас имеется cap-файл.

Здесь нам поможет входящая в состав Kali утилита Hashcat [3], которая позволяет существенно ускорить процесс восстановления паролей с помощью графического процессора. Программа бесплатна, хотя она содержит проприетарную кодовую базу. Доступны версии для Linux, OSX и Windows, есть варианты для использования центрального вычислительного процессора и для использования графического процессора. Hashcat в настоящее время

Рисунок 2. Результат деаутентификации



поддерживает огромное количество алгоритмов хеширования, включая Microsoft LM Hashes, MD4, MD5, семейство SHA, форматы Unix Crypt, MySQL, Cisco PIX и многие другие (их там сотни). Полагаю, что в последующих статьях цикла я еще буду к ней возвращаться при проведении аудита пользовательских паролей.

Но вернемся к WPA. Hashcat имеет несколько режимов атак:

- Атака брутфорсом (перебором)
- Комбинаторная атака
- Атака по словарю
- Атака по отпечаткам
- Гибридная атака
- Атака по маске
- Перестановочная атака
- Атака, основанная на правиле
- Табличная атака
- Атака с переключением раскладки

В рамках решения задачи взлома WPA/WPA2, вероятнее всего, потребуется атака по словарю. Конечно, если что-то известно о том, из чего состоит пароль, можно использовать другие способы, однако в случае полной неизвестности нужен полный перебор.

У нас имеется cap-файл, однако Hashcat имеет собственный формат для расшифровки, поэтому необходимо предварительно очистить cap-файл:

```
# wpaclean <out.cap> <in.cap>
```

Обратите внимание, что сначала идет выходной файл, а потом входной <out.cap> <in.cap>.

Например:

```
# wpaclean /root/Desktop/ wlan1mon_out.cap /root/Desktop/ ↵
        wlan1mon.cap
```

Теперь необходимо конвертировать очищенный файл в формат .hccap с помощью aircrack-ng –J.

```
# aircrack-ng <out.cap> -J <out.hccap>
```

Пример:

```
# aircrack-ng /root/Desktop/ wlan1mon_out.cap ↵
        -J /root/Desktop/ wlan1mon_out.hccap
```

Когда исходные файлы подготовлены, можно приступать непосредственно к взлому. Для взлома файла рукопожатия WPA/WPA2 с Hashcat вводим:

```
# hashcat -m 2500 -a 3 файл.hccap
```

где:

- **-m 2500** – означает атаку на файл рукопожатия WPA2/WPA;
- **-a 3** – означает использование брутфорса (она также совместима с атакой по маске);
- **файл.hccap** – конвертированный файл .cap, после обработки программами wpaclean и aircrack-ng.

Для нашего примера эта команда будет иметь следующий вид:

```
# hashcat -m 2500 -a 3 /root/Desktop/ wlan1mon_out.hccap
```

У Hashcat имеются также редакции для работы с графическими ускорителями cudaHashcat или oclHashcat. При использовании данных утилит синтаксис команд будет несколько отличаться от приведенного в примерах, поэтому перед их использованием необходимо внимательно ознакомиться с документацией. Также не стоит забывать, что предварительно должен быть корректно установлен драйвер видеокарты. По собственному опыту могу сказать, что графический ускоритель позволил увеличить скорость расшифровки в семь-восемь раз по сравнению с использованием центрального процессора.

Еще одним средством взлома WPA являются утилиты Pyrit и coWPAtty. Подробное описание работы с ними приводится в статье [4].

Развитие облачных технологий не обошло стороной и информационную безопасность. В частности, в интернете появился сервис, позволяющий взламывать различные зашифрованные данные [5]. В том числе с его помощью можно попробовать вскрыть WPA/WPA2 cap-файл. Стоимость данной услуги составляет $17. Я не использовал данный сервис, поэтому ничего о его эффективности сказать не могу.

На этом, я думаю, с аудитом WPA можно закончить. Если даже непрерывная работы машины с графическим ускорителем в течение нескольких недель не привела к положительным результатам, то можно сделать вывод о надежности беспроводной сети.

Старый добрый WEP

Теперь рассмотрим случай, когда используется алгоритм WEP. Данный алгоритм шифрования появился гораздо раньше, чем WPA, и в настоящее время является небезопасным [6, 7]. Сам факт использования данного алгоритма является серьезной уязвимостью в корпоративной беспроводной сети, поэтому его настоятельно рекомендуется заменить на WPA2.

Мы уже получили список работающих сетей, нашли в нем шифрование WEP. Теперь нам необходимо собрать некоторое количество шифрованных пакетов для их последующего взлома и получения ключа. Для этого в новом окне терминала указываем:

```
# airodump-ng -c (channel) -w (file name) --bssid (bssid) ↵
        (interface)
```

Здесь:

- **channel** – это канал из столбца CH,
- **file name** – имя файла, в который все будет записываться,
- **bssid** – это идентификатор сети.

Пример использования:

```
# airodump-ng -w wep -c 1 -- bssid 01:01:01:01:01:01 wlan0.
```

Отправляем запросы к атакуемой сети в целях установки соединения (ассоциирования).

```
# aireplay-ng -1 0 -a (bssid) -h 01:01:01:01:01:01 ↵
        -e (essid) (interface)
```

Здесь essid – это имя атакуемой сети. После этого нам необходимо дождаться появления сообщения «Association successful».

Так как взлом шифрования WEP основан на получении статистики от большого числа пакетов, нам нужно собрать необходимое для взлома количество пакетов.

```
# aireplay-ng -3 -b (bssid) -h 01:01:01:01:01:01 (interface)
```

Этот процесс может занять продолжительное время, все зависит от интенсивности работы беспроводной сети. Нам нужно дождаться, пока число в столбце #Data не перейдет отметку в 10 000.

При достижении требуемого количества собранных данных запускаем процесс взлома собранных зашифрованных данных.

```
# aircrack-ng -b (bssid) (file name-01.cap)
```

В качестве имени вводится выбранное вами ранее имя для файла.

```
# aircrack-ng -b (bssid) (wlan0mon.cap)
```

В случае успеха получим строку «KEY FOUND», содержащую ключ.

Делаем выводы

Мы посмотрели на проблему проникновения в корпоративную сеть глазами злоумышленника. Конечно, если в вашей сети Wi-Fi отсутствует в принципе, этот канал проникновения для хакера будет закрыт. (Экзотические варианты с «поднятием» несанкционированных точек доступа внутри контролируемой зоны мы пока рассматривать не будем.) Но сейчас таких организаций становится все меньше. Использование мобильных устройств и концепция BYOD (принеси устройство свое с собой) требуют наличия Wi-Fi на рабочем месте. Поэтому беспроводной доступ нужно сделать максимально защищенным.

Первым делом необходимо правильно настроить мощность сигнала. Дело в том, что слишком «хороший» Wi-Fi на всей территории – это не всегда хорошо. Нужно убедиться, что за пределами контролируемой зоны сигнал не ловится. При необходимости нужно перенести точки доступа подальше от границ контролируемой зоны. Возможно, при этом придется пожертвовать качеством работы беспроводной сети в некоторых помещениях компании. Но слишком рассчитывать на мощность сигнала не стоит. Так, с помощью недорогой внешней антенны ALFA AWUS036NH (которая, кстати, полностью совместима с Kali) мне удалось обнаружить большое количество сетей, которые не видел встроенный адаптер.

Существует весьма спорная рекомендация, скрывать ESSID для того, чтобы сеть не была видна потенциальным злоумышленникам. Однако утилиты, которые мы использовали в этой статье, позволяют обнаруживать и скрытые Wi-Fi, поэтому считать этот способ хоть сколько-нибудь эффективной защитой не стоит. Также не стоит рассчитывать на фильтрацию по MAC-адресам. Во-первых, MAC-адрес можно легко подделать. Подробно тему подмены MAC мы рассмотрим в одной из следующих статей. А во-вторых, в корпоративных сетях, где количество клиентских устройств больше 30, фильтровать по MAC будет крайне сложно. А в гостевых сетях, где устройства постоянно меняются, такая фильтрация вообще невозможна.

Наиболее эффективным средством защиты является использование протокола 802.1x и инфраструктуры открытого ключа. В случае если установить сертификат не представляется возможным, необходимо в дополнение к WPA2-шифрованию использовать также аутентификацию через веб. Тогда злоумышленник даже в случае взлома Wi-Fi не сможет получить доступ к другим ресурсам сети.

Отдельная тема, о которой мы поговорим особо в следующих статьях, – это настройки и пароли по умолчанию. Здесь я лишь хочу напомнить, что после установки оборудования необходимо обязательно сменить заводской пароль.

Также необходимо ограничить доступ к средствам администрирования сетевого оборудования на сетевом уровне. Все интерфейсы управления точками доступа и маршрутизаторами должны быть помещены в отдельный VLAN управления, доступ к которым должен быть только у административного персонала.

Многим данные рекомендации могут показаться очевидными, однако мне приходилось наблюдать в сетях крупных организаций доступные из пользовательской сети веб-интерфейсы управления точками доступа с заводскими паролями.

Также нелишним будет использование специализированных средств обнаружения атак в беспроводных сетях (WIPS). В некоторых моделях оборудования такой функционал встроен. С помощью WIPS можно обнаруживать принудительные разрывы соединений, которые мы делали для сбора шифрованного трафика, а также попытки подбора PIN при атаках на WPS.

• • •

Итак, побывав в роли злоумышленника, мы попробовали подключиться к корпоративной сети с помощью Wi-Fi. Следующим шагом хакера должна стать идентификация корпоративных приложений и устройств, а также перехват передаваемого трафика. В следующей статье мы подробно рассмотрим различные способы выявления версий операционных систем и приложений, а также обнаружение некорректных настроек. EOF

[1] Сайт проекта Kali Linux – https://www.kali.org.

[2] Статья по взлому WPA/WPA2 – http://www.pentestingshop.com/how-to-pentest-your-wpawpa2-wifi-with-kali-linux.

[3] Статья по работе с Hashcat – https://webware.biz/?p=3799.

[4] Использование Pyrit и coWPAtty – https://webware.biz/?p=2984.

[5] Облачный сервис по взлому шифров – https://www.wpacracker.com.

[6] Подробная инструкция по взлому WEP – http://goo.gl/1X3tBy.

[7] Статья по WEP – http://lifehacker.ru/2012/10/27/kak-vzlomat-wi-fi-set-s-wep-shifrovaniem.

Ключевые слова: безопасность, аудит, Wi-Fi, Kali.

Визитка

СЕРГЕЙ ЯРЕМЧУК,
автор более 800 статей и шести книг. С «СА» с первого номера.
Интересы: сетевые технологии, защита информации, свободные ОС, grinder@samag.ru

Использование Web Application Proxy в Windows Server 2012 R2/2016

Знакомимся с возможностями роли WAP, появившейся в Windows Server 2012 R2, которая позволяет обеспечить безопасный доступ к приложениям

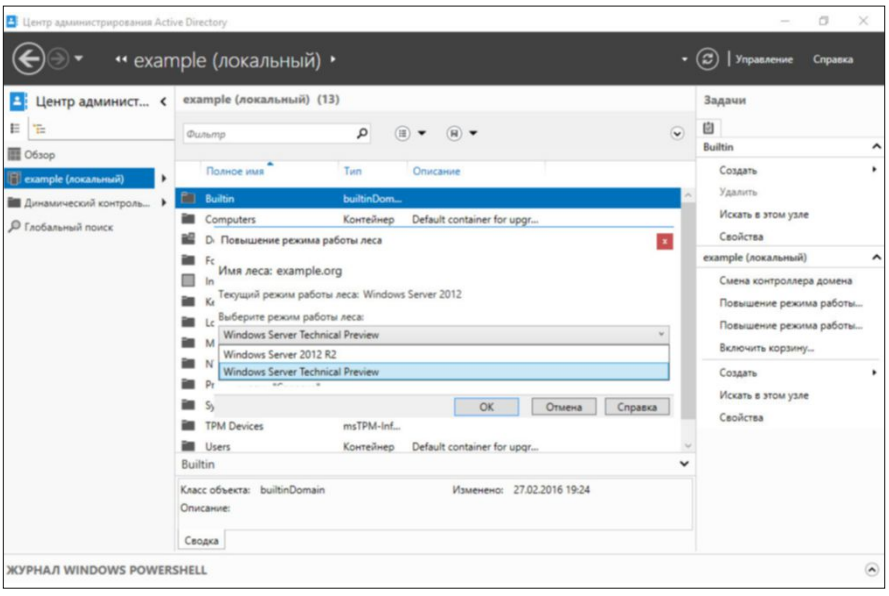
Возможности Web Application Proxy

Роль Web Application Proxy (прокси-сервер веб-приложений) пришла на замену продукту Forefront Unified Access Gateway, поддержка которого была завершена в 2013 году. Сервер WAP размещается в DMZ и позволяет публиковать приложения для доступа извне, обеспечивая требуемый уровень безопасности. Для этого он выступает как прокси, принимая HTTPS-запрос на внешний адрес и транслируя его на сервис, работающий по протоколу HTTP или HTTPS. В отличие от традиционно используемых для доступа к внутренним ресурсам VPN пользователю будут доступны только разрешенные ему приложения. Такой прокси обеспечивает дополнительную защиту, т.к. любой нецелевой трафик и известные сетевые атаки отбрасываются, не достигая приложения. Примечательно, что приложения могут использовать один IP-адрес или порт, это не вызовет проблем, т.к. WAP определяет нужное по имени. WAP также поддерживает функцию Workplace Join, дающую возможность пользователям получать доступ к корпоративным ресурсам с мобильных устройств. Поддержка технологии Single-Sign-On (SSO) позволяет после подключения к домену больше не вводить пароль для доступа к разрешенным приложениям.

В своей работе WAP опирается на Active Directory Federation Service (AD FS), которому передает все запросы на подключение, для аутентификации пользователя средствами Active Directory и контроля доступа на основе заявок (Claims Based Access). В случае удачи AD FS выдает SSO-маркер безопасности, содержащий идентификатор пользователя и ресурса, к которому запрашивались доступ и срок. WAP сохраняет всю информацию в Cookies и инициирует соединение к приложению. Приложение после проверки маркера допускает пользователя без ввода пароля.

Использование AD FS упрощает масштабирование, т.к. теперь можно легко подключить к AD FS любое количество серверов с ролью WAP, но, естественно, требует обязательного наличия домена. Сервер с ролью AD FS традиционно находится во внутренней сети, WAP при этом повышает его защищенность, не позволяя обращаться к нему напрямую. AD FS обеспечивает все современные технологии безопасности: многофакторная аутентификация (Multifactor authentication) и контроль доступа (Multifactor Access control) с использованием дополнительных ступеней – одноразовый пароль или смарт-карта. Для старых приложений, не поддерживающих аутентификацию через AD FS, предусмотрен режим Pass-through preauthentication, при котором соединение просто пробрасывается дальше, а аутентификация

Рисунок 1. Перед установкой WAP схему нужно повысить до Windows Server 2012 R2 или Windows Server Technical Preview



производится самим приложением. Естественно, в этом случае о MFA и MAC речи не идет.

Кроме аутентификации пользователя, WAP совместно со службой Device Registration Service может проверять сертификат устройства пользователя, разрешая доступ к корпоративным ресурсам только с одобренных устройств. WAP не требует дополнительных лицензий клиентского доступа (CAL), необходима лишь лицензия на сам сервер.

На основании отзывов пользователей в Windows Server 2016 WAP был доработан и получил новые возможности. Теперь автоматически происходит перенаправление на защищенный сайт, т.е. HTTP к HTTPS. Раньше это требовало дополнительных настроек и установки IIS, теперь же достаточно отметить флажок. Возможна публикация HTTP-приложений со сквозной проверкой подлинности, ранее это не разрешалось по причинам безопасности, но HTTP, как оказалось, все-таки нужен. WAP совместим с Lync Server, поддерживает Exchange Active Sync (EAS) и Remote Desktop Gateway. Поддерживается возможность преаутентификации HTTP Basic, используемого многими приложениями, в том числе и Exchange ActiveSync. Логин и пароль будут автоматически извлечены из URL и на их основании выдан маркер. Стала доступной публикация нескольких приложений внутри одного домена с помощью шаблона URL, вроде https://*.example.org/.

Рисунок 2. Установка роли Web Application Proxy

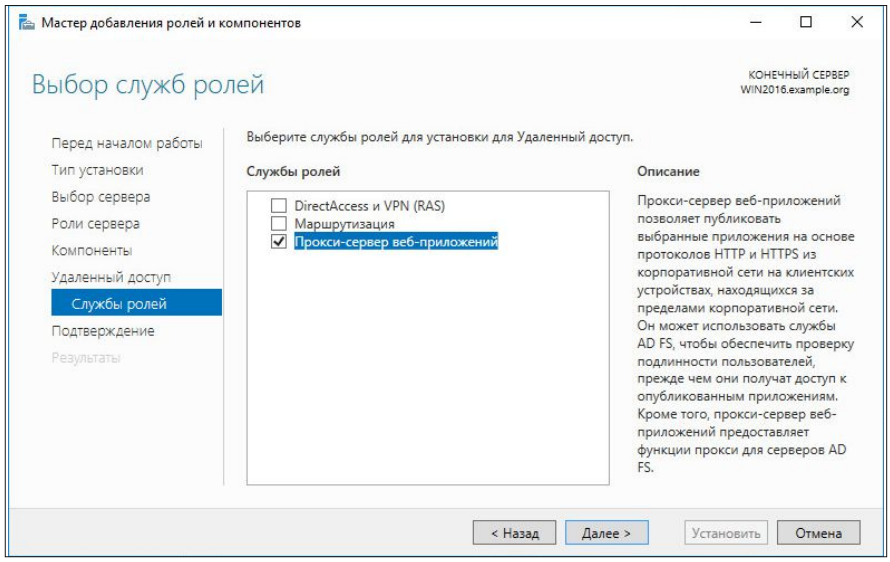


Рисунок 3. Работа мастера настройки WAP

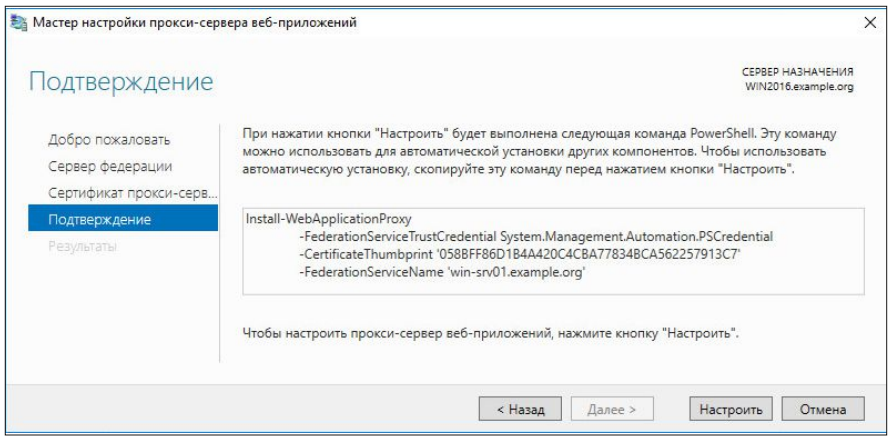
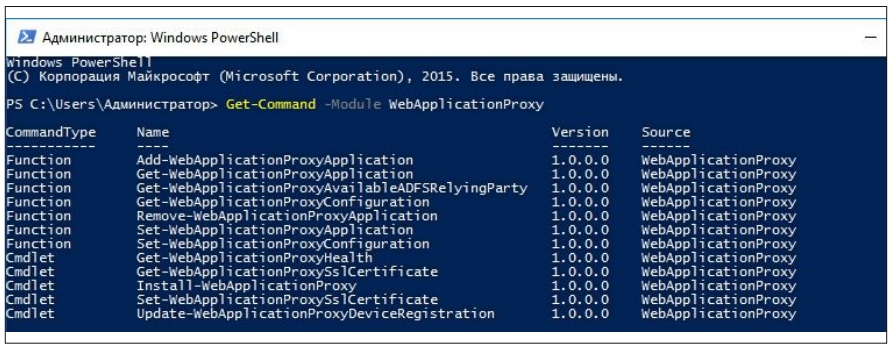


Рисунок 4. Список командлетов модуля WebApplicationProxy



Требования к установке WAP

Установка и последующая конфигурация Forefront UAG не была простым делом, в большинстве случаев приходилось разбираться в многочисленных зависимостях и с полученными ошибками. В этом смысле WAP, являющаяся ролью сервера, выигрывает. Мастер установки WAP, обнаружив роль AD FS на этом же сервере, прерывает работу, что весьма логично.

Поэтому для тестирования понадобится как минимум два сервера: контроллер домена + AD FS и WAP. Учитывая назначение и расположение, желательно, чтобы WAP была на сервере единственной ролью.

Мастера помогают просто настроить роль, поэтому сервер должен быть установлен в варианте с GUI. Хотя все операции можно произвести и с помощью PowerShell. Сервер WAP необходимо подключить к домену, сгенерировать сертификаты, в поле Subject Alternative Name которых содержится DNS-имя публикуемой службы. Сертификаты понадобятся для каждого сервера и по одному для каждого приложения. Схему нужно обязательно повысить до Windows Server 2012 R2 или Windows Server

Technical Preview (см. рис. 1), иначе мастер установки роли при выборе пункта WAP просто откажется работать.

Особенных требований к железу не предъявляется, удовлетворят минимально необходимые для работы ОС.

Развертывание WAP

Если нет AD FS, его необходимо развернуть первым. Для этого понадобятся права администратора и сертификат. При их наличии установка проста.

Вызываем диспетчер сервера и в мастере добавления ролей указываем Active Directory Federation Services или используя PowerShell:

```
PS> Install-WindowsFeature ADFS-Federation ↵
        -IncludeManagementTools
```

Необходимо добавить корневой ключ KDS, иначе при работе мастера получим ошибку о недоступности групповых учетных записей службы (он обновляется только через 10 часов, и в реальной ситуации лучше дождаться, когда он реплицируется).

```
PS> Add-KdsRootKey -EffectiveTime (Get-Date).AddHours(-10)
```

По окончании запускаем мастер настройки AD FS.

Далее указываем «Создать первый сервер федерации в новой ферме», учетную запись для подключения, выбираем сертификат (возможен экспорт из файлов формата PFX). При этом автоматически заполняется DNS-имя службы федерации, которое будет использоваться клиентами при подключении. Хранение конфигурации AD FS возможно во внутренней базе данных (Windows Internal Database, до пяти серверов) или SQL-сервере.

Для настройки с помощью PowerShell следует указать отпечаток сертификата:

```
PS> Install-AdfsFarm -CertificateThumbprint 'abcd...890' ↵
        -FederationServiceName ad.example.org ↵
        -GroupServiceAccountIdentifier EXAMPLE\adfs$
```

В дальнейшем все настройки также можно производить в графической консоли или же с помощью командлетов модуля ADFS:

```
PS> Get-Command -Module ADFS
```

Установка роли WAP также не отличается разнообразием: вызываем мастера, отмечаем роль Remote Access (удаленный доступ) и на этапе выбора служб ролей отмечаем Web Application Proxy (прокси-сервер веб-приложений) (см. рис. 2), подтверждаем выбор дополнительных компонентов и устанавливаем.

```
PS> Install-WindowsFeature Web-Application-Proxy ↵
        -IncludeManagementTools
```

По окончании следует настроить службу с помощью мастера настройки. Открываем консоль Remote Access Management (управление удаленным доступом), выбираем «Настройка → Прокси-служба веб-приложений» и по ссылке запускаем мастер настройки. Указываем данные сервера AD FS и учетную запись администратора, затем сертификат для WAP. Это же можно сделать и с помощью PowerShell (см. рис. 3).

```
PS> Install-WebApplicationProxy ↵
    -CertificateThumprint '098...cab' ↵
    -FederationServiceName ad.example.org
```

Публикация сервиса производится с помощью пошагового мастера. В процессе выбираем метод преаутентификации (AD FS или Pass-through), задать имя публикуемого приложения, указать сертификат, внешний URL, который будут использовать для подключения клиенты, URL-приложения (при необходимости порт), на который будут пересылаться запросы.

В случае выбора аутентификации средствами AD FS появляется дополнительный шаг, на котором следует указать механизм доверия – Device Registration Service, WS-Fed, SAML, Oauth.

По окончании можно это протестировать, зайдя с помощью браузера по внешнему адресу. Также в этом случае нужно создать доверие в консоли AD FS. Для этого просто переходим в Trust Relationships (отношения доверия), нажимаем на Add Non-Claims-Aware Relaying Party Trust (добавить отношения доверия с проверяющей стороной, не поддерживающей утверждения), затем указываем имя, добавляем идентификатор доверия (обычно внешний URL) и окончательно настраиваем нужную нам многофакторную аутентификацию.

По окончании редактируем правила авторизации (Authorization Rules). В Edit Claim Rules for ... нажимаем Add Rule и указываем шаблон правила и при необходимости его редактируем.

Настройками WAP также можно управлять с помощью PowerShell. Причем многие операции удобнее производить именно с помощью 12 командлетов модуля WebApplicationProxy PowerShell (их количество одинаково в Windows Server 2012 R2 и 2016) (см. рис. 4):

```
PS> Get-Command -Module WebApplicationProxy
```

Командлеты Get-WebApplicationProxy* позволяют получить информацию по сервису. Публикация приложения производится с помощью командлета Add-WebApplication ProxyApplication:

```
*PS> Add-WebApplicationProxyApplication ↵
        -BackendServerURL 'https://srv.example.org:8080/' ↵
        -ExternalCertificateThumbprint '098...cab' ↵
        -ExternalURL 'https://service.org/' ↵
        -Name 'Service' ↵
        -ExternalPreAuthentication ADFS
```

• • •

Web Application Proxy позволяет решить проблему безопасного доступа к приложению из внешней сети. Поддержка популярных приложений и наличие большого числа мастеров делает его внедрение очень простым. EOF

Ключевые слова: Web Application Proxy, Active Directory, Windows Server 2012 R2/2016.

Визитка

ВЛАДИМИР ТИХОМИРОВ,
директор по информационным технологиям
СПАО «Ингосстрах», Vladimir.Tikhomirov@ingos.ru

Визитка

ВАЛЕРИЙ МИХЕИЧЕВ,
эксперт Oracle, ОСАО «Ингосстрах»,
Valery.Mikheitchev@ingos.ru

Распараллеливание операций в Oracle
Часть 1

Практика распараллеливания операций показала, что оно дает порой весомые результаты при выполнении DML, DDL и других действий. Но имеются особенности и ограничения

В процессе работы с Oracle специалистам приходится сталкиваться с задачами по ускорению ввода данных и модификации данных в таблицах, с задачами по сокращению времени извлечения данных из таблиц SQL-запросами, а также с задачами по ускорению создания и перестройке индексов, сбору статистики таблиц и т.д.

Существуют различные методы решения данных задач, среди которых эффективными являются распараллеливание процессов, секционирование таблиц, использование более эффективных планов выполнения запросов путем их оптимизации и др.

В данной статье рассматриваются вопросы распараллеливания процессов как один их эффективных методов ускорения обработки данных. В первой части статьи будут рассматривается распараллеливание DML-операции, во второй части будут изложены вопросы распараллеливания DDL и других операций в Oracle.

Oracle поддерживает распараллеливание следующих операций:

- распараллеливание выполнения SQL-запросов (parallel query),
- DML-операций ввода,
- модификации строк таблиц (Insert, Update, Delete, Merge) и DDL-операций (таких, как Create tables, Create index и Rebuild index).

Кроме того, поддаются распараллеливанию также следующие конструкции внутри одной команды:

- Select distinct
- Group by
- Order by
- Not in
- Union и Union all
- Cube и Rollup
- агрегатные функции, например Sum и Max
- соединения типа Nested loop, Sort и Merge

Для секционированных таблиц распараллеливание поддерживается для операций с секциями, такими как Move, Split и Coalesce. Кроме того, распараллеливание работает для сбора статистики таблиц процедурой dbms_stats.gather_table_stats.

Поддерживается также параллелизм в границах секции секционированной таблицы, т.е. данные, хранящиеся в одной секции, могут одновременно обрабатываться несколькими серверными процессами.

На степень параллелизма существенное влияние оказывает ряд таких факторов, как число процессоров на сервере, значения параметров инициализации экземпляра базы данных, параметры, установленные на уровне сессии, а также другие факторы, например, предельное значение степени параллелизма, разрешенное пользователю.

В данной статье речь пойдет о распараллеливании DML и DDL-операций, ускорении создания и перестройке индексов, сборе статистики таблиц и индексов, основанных на распараллеливании процессов, и другие вопросы. Рассматриваются в том числе факторы, влияющие на параллелизм, и особенности параллелизма.

Давайте рассмотрим вопросы распараллеливания DML-операций, таких как:

- Insert
- Update
- Delete
- Merge

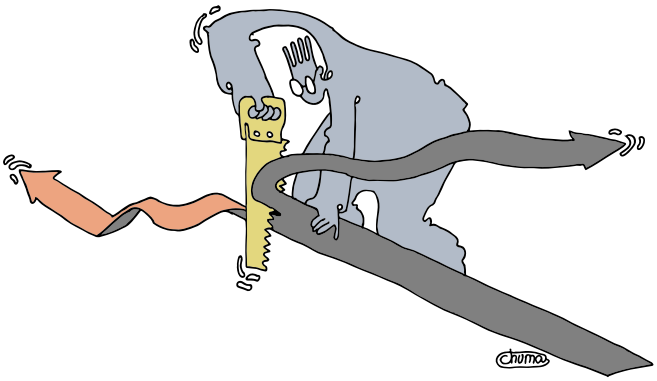
Одними из важных факторов, влияющих на параллелизм, являются:

- параметры инициализации, действующие на уровне всей системы,
- параметры, действующие на уровне сессии (сессионные параметры).

Параметры инициализации, отвечающие за параллелизм во всех сессиях

К ним относятся документированные параметры инициализации, отвечающие за параллелизм, которые можно увидеть по запросу:

```
Select name, value, description from v$parameter ↵
        where name like '%parallel%' order by 1;
```



Рассмотрим факторы, влияющие на параллелизм, и особенности параллелизма

Кроме того, существуют еще не документированные параметры инициализации, которые можно найти по запросу:

```
Select a.ksppinm,b.ksppstvl,a.ksppdesc ↵
        from x$ksppi a,x$ksppcv b,x$ksppsv c ↵
        where a.indx = b.indx and a.indx = c.indx ↵
        and upper(a.ksppinm) like upper('_parall%') ↵
        order by 1;
```

Среди документированных параметров, отвечающих за параллелизм, особый интерес для пользователя Oracle представляют следующие параметры:

- **parallel_adaptive_multiuser** – ограничивает степень параллелизма пользователя (чтобы могли работать и другие пользователи);
- **parallel_automatic_tuning** – выставление значении true приводит к тому, что Oracle самостоятельно устанавливает значения параметров, относящихся к параллельным операциям.

Кроме того, важными являются такие параметры, как:

- **parallel_degree_limit** – который может принимать значения CPU или IO (включает параллелизм процессоров или ввода-вывода),
- **parallel_degree_policy** – который определяет, будет ли автоматическая степень параллелизма (пример его применения показан ниже), и др.

Документированные параметры инициализации являются общесистемными и устанавливаются администратором баз данных, однако на уровне сессии некоторые из них могут быть изменены через хинт (подсказка оптимизатору) opt_param, например:

```
/*+ opt_param('parallel_degree_limit' 'CPU') */
```

Вместе с тем для пользователей Oracle не меньший интерес представляют параметры, управляющие процессами распараллеливания на уровне сессии (назовем их сессионными параметрами).

Параметры параллелизма на уровне сессии

Понять возможности Oracle в распараллеливании процесса на уровне сессии помогает Oracle-представление v$ses_optimizer_env через запрос:

```
Select name, isdefault, value from v$ses_optimizer_env  s ↵
        where sid = sys_context('USERENV', 'SID')  ↵
        and  name like 'parallel%'  order by 1;
```

Замечание. В Oracle 12c появились три новых параметра:

- parallel_dblink
- parallel_degree_level
- parallel_execution_message_size

Среди всех сессионных параметров особый интерес представляют два параметра:

- **parallel_ddl_mode** – работающий для DDL-операций (по умолчанию принимает значение enabled),
- **parallel_dml_mode** – определяющий параллелизм DML-операций (по умолчанию принимает значение disabled).

Первый параметр сообщает, что в сессиях включен режим enabled, разрешающий проводить распараллеливание процессов при DDL-операциях (например, при создании индекса). Однако следует заметить, что это не подразумевает автоматического включения процесса распараллеливания, требуется провести работу по активизации процессов распараллеливания.

Второй параметр имеет значение disable, блокирующее (выключающее) режим распараллеливания процессов DML-операций (назовем параллельные DML-операции PDML), например, таких как Insert, Delete Update, Merge, и, чтобы распараллеливание заработало, необходимо включить (разрешить через значение enable) процесс распараллеливания. Рассмотрим данный параметр подробнее.

Включение распараллеливания DML-операций

Чтобы процесс распараллеливания DML-операций стал доступен в сессии, требуется перевести параметр

parallel_dml_mode в состояние enabled, включающее режим распараллеливания. Это осуществляется командой:

```
ALTER SESSION ENABLE PARALLEL DML
```

В результате выполнения команды параметр parallel_dml_mode принимает значение enabled. Доступным процесс распараллеливания для DML-операций становится также по команде:

```
ALTER SESSION FORCE PARALLEL DML
```

с фразой FORCE. При этом параметр parallel_dml_mode после выполнения команды принимает значение forced.

Отличие первой команды от второй в том, что команда с фразой ENABLE только разрешает режим распараллеливания, однако требует активизации процесса распараллеливания операций. В качестве средства активизации выступает хинт parallel, который вставляется в команду DML с указанием в ней степени параллелизма (degree of parallelism, или сокращенно DOP), указывающей число параллельных процессов. Например, хинт parallel в операции Insert /*+parallel(32) */ into…) имеет степень параллелизма DOP=32.

При использовании в сессии второй команды с фразой FORCE происходит не только разрешение на распараллеливание, но и автоматическая активизация процессов распараллеливания. В этом случае можно обойтись без хинта parallel. Хинт parallel в режиме FORCE применяется для регулирования числа параллельных процессов.

При наличии FORCE по умолчанию включается в выполнение DML-операции некоторое число параллельных процессов (назовем их число как max_parallel_default).

Величина max_parallel_default вычисляется по формуле на основании значений параметров инициализации. Такими параметрами являются:

- **cpu_count** – определяет число процессоров, доступных для использования базой данных Oracle, а в многоядерной архитектуре определяет число доступных ядер процессора),
- **parallel_threads_per_cpu** – параметр описывает число выполняющихся параллельных процессов, которые процессор может обработать во время параллельного выполнения).

Формула имеет вид:

cpu_count * parallel_threads_per_cpu + 1

где добавляется значение 1 (дополнительный процесс для координации работы остальных процессов). Для выяснения значения max_parallel_default можно воспользоваться запросом:

```
Select value*(select value  from v$parameter ↵
         where name='parallel_threads_per_cpu')+1 ↵
         from v$parameter where name='cpu_count'
```

Например, в случае cpu_count = 16 и parallel_threads_per_cpu = 2 значение max_parallel_default будет иметь значение 16*2 + 1 = 33.

Следует учесть, что при наличии индексов в таблице, используемой в DML-операции, число параллельных процессов может увеличиться в два раза (например, при наличии уникального индекса).

Замечание. Хинт parallel работает эффективно для Oracle 11g и 12c без указания имени таблицы, однако при работе с представлением указание имени или «алиаса» таблицы в хинте позволяет проводить параллелизм эффективнее.

Таким образом, для распараллеливания DML-операций с таблицей buh_t в сессии, в которой запускается Insert, Update, Delete и Merge операции, могут быть два варианта команд по разрешению и активизации параллелизма.

Первый вариант – это команда Alter с фразой ENABLE и хинтом parallel (с указанием в нем степени параллелизма):

```
ALTER SESSION ENABLE PARALLEL DML;
Insert  /*+ parallel(32) */  into  buh_t  ↵
         select * from buh_a; commit;
…
Delete /*+ parallel(32) */  …; commit;
```

где в хинте указывается степень параллелизма, например, число, равное значению параметра инициализации cpu_count.

Его значение можно получить из запроса:

```
Select value from v$parameter where name='cpu_count'
```

При этом практика показала, то при степени параллелизма DOP=cpu_count получается наиболее значимый эффект в сокращении времени выполнения операций.

Важным является тот факт, что, если операции идут друг за другом, следует обязательно завершить предыдущую DML-операцию по commit (или rollback), иначе параллелизм не только не сработает, но еще появится ошибка:

```
ORA-12838: cannot read/modify an object after modifying it in
parallel
```

Второй вариант с FORCE (без хинта parallel):

```
ALTER SESSION FORCE PARALLEL DML;
Insert  into  buh_t select * from buh_a; commit;
…
Delete buh_t where …; commit;
```

В этом случае по умолчанию может быть задействовано max_parallel_default процессов (как вычислить max_parallel_default, показано выше).

При выборе варианта следует учесть, что второй вариант без использования хинта parallel может привести к использованию по умолчанию большого числа процессов. Это, в свою очередь, может привести не к повышению производительности работы PDML-операции, а к замедлению ее выполнения в силу чрезмерного потребления ресурсов.

В силу чего рекомендуется, во-первых, использовать в методе FORCE также хинт parallel с указанием степени параллелизма DOP, а, во-вторых, указывать степень параллелизма начиная со значения параметра cpu_count. При этом в некоторых случаях, например при большом значении cpu_count или при интенсивном использовании

параллелизма разными сессиями, значение DOP имеет смысл указывать меньше cpu_count, чтобы не перегружать ресурсы, например, начинать со степени параллелизма в два или даже более раз меньше значения cpu_count.

Выводы:

- Для включения режима параллелизма для DML-операций в сессии в одинаковой степени можно использовать обе команды, как ALTER SESSION ENABLE PARALLEL DML, так и ALTER SESSION FORCE PARALLEL DML.
- Для активизации процесса можно использовать фразу FORCE или хинт parallel.
- Целесообразно независимо от вида команды ALTER (с фразой enable или force) в PDML-операции использовать хинт parallel с указанием в нем степени параллелизма. При этом имеет смысл проверить эффективность распараллеливания при степени параллелизма, равной значению cpu_count или меньшей в два и более раз.
- В цепочке PDML-операций обязательно завершение предыдущей PDML-операции по commit или rollback. Если по технологическим причинам завершение PDML-операции невозможно, то попробовать использовать автономную транзакцию, создавая для PDML процедуру с фразой PRAGMA AUTONOMOUS_TRANSACTION и обязательным commit (rollback) в конце данной процедуры.

В силу вышесказанного базовым для получения PDML-операции в сессии можно считать следующую последовательность команд и операций в сессии.

В начале сессии выполняется команда, разрешающая проведение распараллеливания:

```
ALTER SESSION ENABLE PARALLEL DML;
```

или

```
ALTER SESSION FORCE PARALLEL DML;
```

В PDML-операциях, используемых в сессии, ставится хинт parallel с указанием степени параллелизма

```
Insert  /*+ parallel(32) */   …  ; commit;
Update  /*+ parallel(32) */   …  ; commit;
Delete  /*+ parallel(32) */   …  ; commit;
Merge  /*+ parallel(32) */   …  ; commit;
```

Замечание. Команду ALTER SESSION FORCE PARALLEL DML с фразой FORCE, автоматически включающей параллелизм, имеет смысл использовать, например, в случае, если нет возможности скорректировать текст DML-операции в хранимой процедуре, чтобы вставить в нее хинт parallel.

Если требуется создать число параллельных процессов, которое больше максимально возможного по умолчанию,

т.е. больше max_pocess_default, например 200, то следует указать это число в хинте /*+ parallel(200) */. Однако в этом случае допустимое число параллельных процессов не превысит удвоенного числа max_pocess_default. Обойти это ограничение позволяет системный параметр PARALLEL_DEGREE_POLICY, который следует перевести из режима MANUAL (установленного, как правило, по умолчанию) в режим AUTO, т.е. выполнить команду:

```
ALTER SESSION SET PARALLEL_DEGREE_POLICY=AUTO;
```

Но даже в этом случае реально число параллельных процессов Oracle автоматически ограничит.

Следует отметить особенности при работе команды ALTER SESSION SET PARALLEL_DEGREE_POLICY в режиме AUTO. Так, при наличии команды:

```
ALTER SESSION ENABLE PARALLEL DML;
```

разрешающей параллелизм, режим AUTO работает при наличии хинта parallel в DML-операциях или указании параллельности в таблице при ее создании фразой parallel. В то же время при наличии команды:

```
ALTER SESSION FORCE PARALLEL DML
```

хинта parallel или параллелизма в таблице для включения в работу режима AUTO не требуется.

Таким образом, для увеличения числа параллельных процессов при вводе данных в таблицу следует выполнить команды:

```
ALTER SESSION SET PARALLEL_DEGREE_POLICY=AUTO;
ALTER SESSION ENABLE PARALLEL DML;
Insert  /*+ parallel(200) */  …;
```

Учет временного пространства на распараллеливание операций

Следует учесть расходы временного пространства на параллельные процессы. Для контроля можно использовать запрос:

```
Select tablespace_name, round(free_space/1024/1024) free_mb  ↵
         from dba_temp_free_space;
```

Отключить параллелизм DML-операций

Отключить параллелизм в сессии можно командой:

```
ALTER SESSION DISABLE PARALLEL DML;
```

или через хинт no_parallel, а также через хинт opt_param, например:

```
insert  /*+ opt_param ('parallel_execution_enabled' ↵
         'false') */ into…
```

Необходимость отключения параллелизма может возникнуть, например, для отключения большого числа параллельных процессов, создающих проблемы в работе базы данных Oracle (к примеру, в режиме FORCE).

Представления Oracle для мониторинга процессов распараллеливания

Увидеть число выполняемых параллельных процессов можно по запросу:

```
Select count(*) from v$px_session;
```

Кроме того, полезными являются запросы, построенные на других представлениях Oracle, таких как v$px_process_sysstat, v$sql_optimizer_env и др., например, по запросам:

```
Select value+1 from v$px_process_sysstat ↵
         where statistic like 'Servers In Use%';
Select sql_id, (select v.sql_text from v$sql v ↵
         where v.sql_id=s.sql_id and rownum=1) sql_text,  ↵
         p.* ,s.* from v$process p,v$session s ↵
         where s.paddr=p.addr  and pname like 'P%'
order by s.sid;
```

Если известно sql_id выполняемой DML-операции, например значение sql_id= '66n9v45wqrd9c' , полученное из предыдущего запроса, то увидеть параллелизм позволяет запрос:

```
Select e.* , (select sql_text from v$sql v ↵
         where v.sql_id=e.sql_id and rownum=1) ↵
         sql_text from v$sql_optimizer_env e  ↵
         where e.sql_id='66n9v45wqrd9c' ↵
         and name='parallel_dml_mode' ↵
         and value<>'disabled';
```

Ограничения, накладываемые на распараллеливание операций

- Нельзя получить доступ к таблице, которая была изменена с помощью распараллеливания PDML-операции, пока не завершится или откатится транзакция.
- Во время выполнения операции триггеры не поддерживаются.
- Распараллеливание DML не производится для колонок таблицы, содержащих объекты LOB, но поддерживается для других колонок этой таблицы.

Общие выводы

- Настройка режима распараллеливания на общесистемном уровне осуществляется настройкой параметров инициализации.
- По умолчанию PDML-операции в сессии заблокированы. Для включения режима распараллеливания в сессии требуется выполнить в сессии команды ALTER SESSION ENABLE PARALLEL DML или ALTER SESSION FORCE PARALLEL DML.
- После включения распараллеливания требуется активизировать процесс распараллеливания, например, хинтом parallel, а отключить хинтами no_parallel или opt_param.
- Требуется контролировать расход временного табличного пространства и учесть ограничения, накладываемые на параллелизм. В том числе учесть необходимость завершения транзакций при PDML-операциях. EOF

Ключевые слова: Oracle, СУБД, распараллеливание операций, DML-операции.

Визитка

ОЛЕГ ФИЛИППОВ,
АНТ-Информ, заместитель начальника отдела разработки, comol@mail.ru

Красивые отчеты из 1С

Внешние системы отчетности

Предпочитаете использовать красивый Dashboard? Руководитель не хочет заходить в 1С, чтобы просматривать отчеты? Отчеты должны выглядеть «живыми»? Тогда эта статья может оказаться вам полезной

Чем же плохи отчеты в 1С?

В последних версиях платформы 1С достаточно удобная и развитая система отчетности. Так называемая Система Компоновки Данных, или СКД, является удобным и мощным инструментом быстрой разработки отчетных форм. Но остаются вопросы итогового представления результата работы СКД.

Как правило, это обычный «Табличный документ» 1С, возможности которого не претерпевали существенных изменений еще со времен версии 1С 7.7. Универсальность данного элемента, конечно, вызывает уважение, но, так или иначе, графические возможности современных пакетов построения отчетности существенно превзошли то, что «можно выжать» из табличного документа.

Вторая проблема – это тот факт, что веб-технологии плотно укрепились в нашей жизни. Если для повседневной работы и ввода большого количества информации данные технологии не всегда удобны, то их графические возможности для оформления отчетов поражают воображение.

В 1С есть веб-клиент, но его принципиальные особенности в том, что все оформительские функции и возможности интерфейса инкапсулированы от прикладного разработчика. Кроме того, веб-приложение 1С все-таки построено по принципу полноценного клиента, а не набора отдельных модулей.

Это принципиально отличает его от большинства современных систем отчетности. Если вы, к примеру, хотите посмотреть отчет по продажам, то вы запустите полноценный клиент 1С, затем нажмете кнопку «сформировать» и только потом увидите результат отчета в том же окне клиента 1С. Представителей бизнеса, конечно, это не всегда устраивает.

Итак, получается, для того, чтобы видеть красивые, быстрые и независимые от 1С отчеты, необходимо использовать сторонний инструментарий. Но любому инструментарию построения отчетности, так или иначе, нужны данные, которые с его помощью будут представлены в удобном для пользователя виде. Поэтому нам с вами прежде всего необходимо разобраться, каким образом эти данные из 1С получить.

Способы получения данных из систем на базе 1С

На эту тему написаны тома литературы. Вариантов бывает великое множество. Но все они, по сути, сводятся к пяти базовым технологиям:

- 1С выгружает данные в XML/CSV/JSON/DBF/XLS и прочее, система отчетности получает данные из файлов и загружает их в память/свою БД.
- Создается DWH. 1С наполняет его данными, система отчетности использует для построения кубов и из них уже отчетов.
- Система отчетности создает COM объект клиента 1С и получает из 1С данные «нативными» средствами.
- Система отчетности получает данные из 1С посредством обращения к веб-сервисам, экспортируемым 1С.
- Система отчетности получает данные из 1С посредством прямого обращения к БД 1С на уровне СУБД.

Рисунок 1. Отчет, созданный с помощью MS Reporting Services

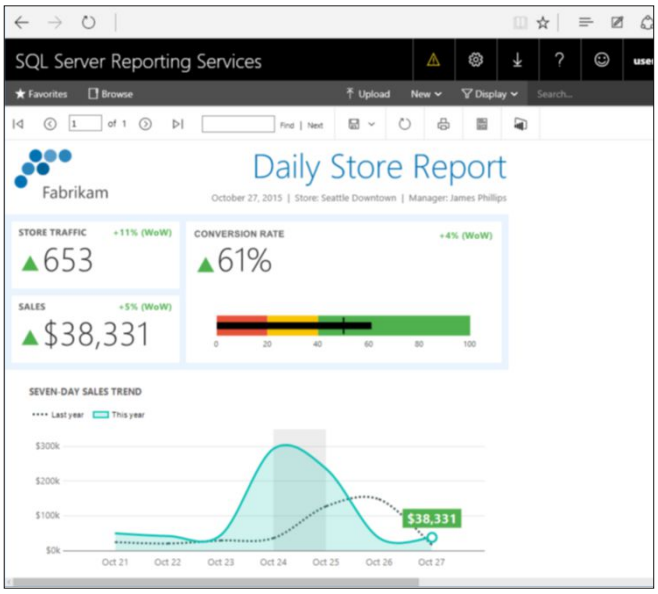


Рисунок 2. Отчет, созданный с помощью SAP Crystal Reports

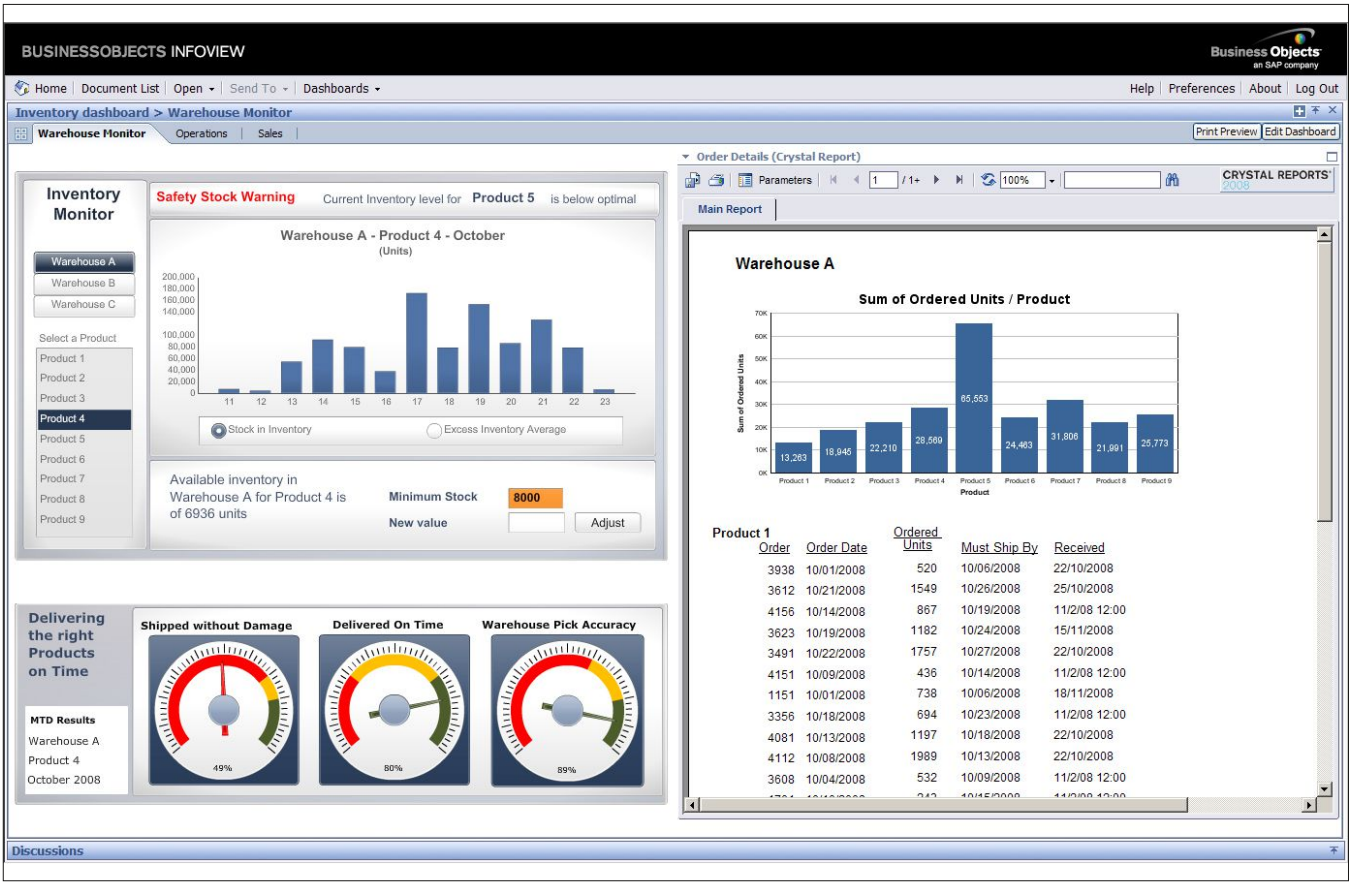
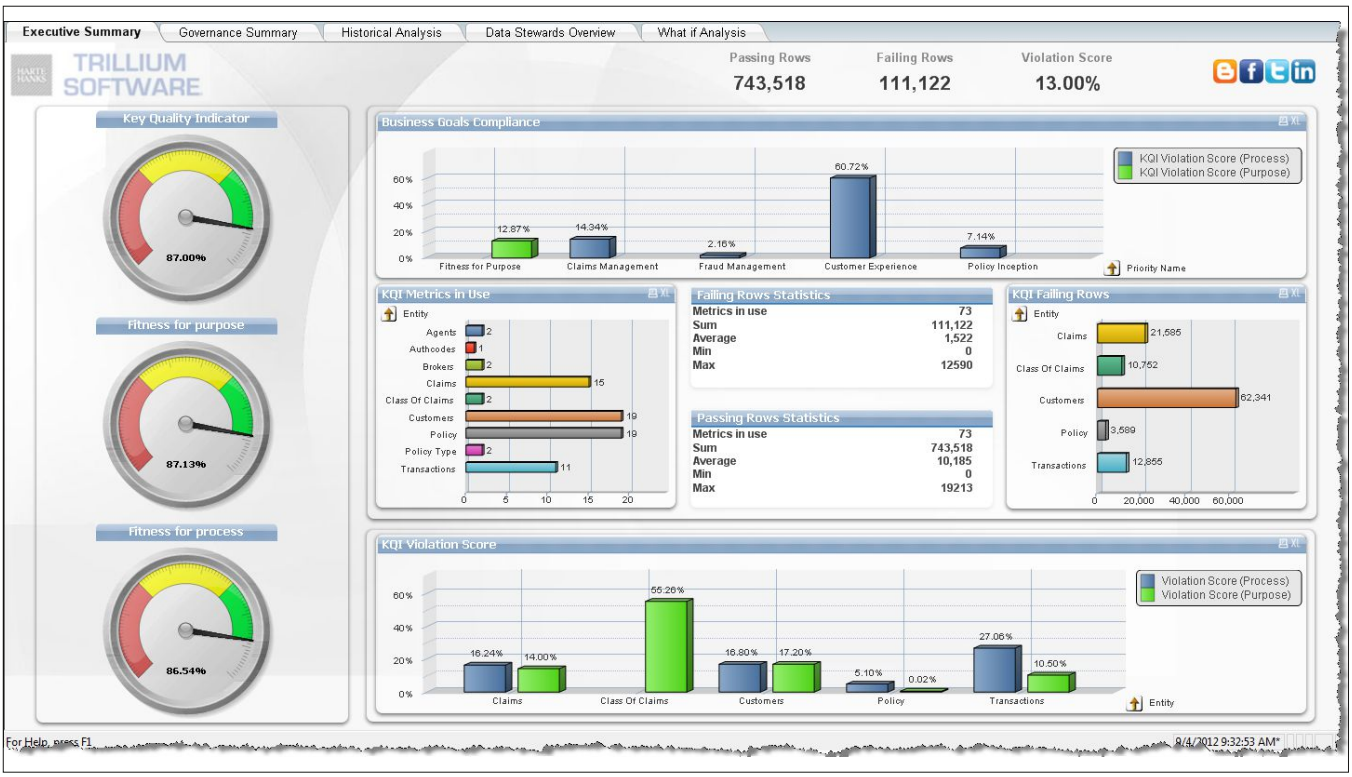
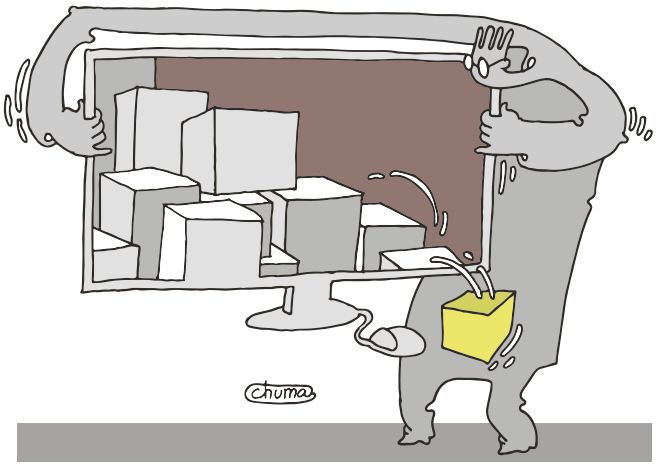


Рисунок 3. Внешний вид отчетов QlikView





Чтобы видеть красивые, быстрые и независимые от 1С отчеты, нужно использовать сторонний инструментарий

Давате постараемся сравнить их основные достоинства и недостатки. Результаты моего сравнения приведены в таблице 1.

Итак, выбор технологии неочевиден и зачастую зависит от многих факторов, будь то необходимость получения данных online и анализа больших объемов данных, требования к ОС и лицензиям.

На мой взгляд, наиболее универсальным является последний вариант, хотя и расходится с основной методологией 1С. Конечно, в случае небольших объемов данных гораздо практичнее использовать веб-сервисы, в случае очень больших – создавать DWH, а если нет компетенции по SQL в компании, то прямой доступ можно заменить на COM.

После того, как мы с вами определились с получением данных, самое время решить, какой «движок» отчетности использовать.

Наиболее популярные системы отчетности

Итак, какие системы отчетности мы с вами сможем использовать? Вариантов не так много, как кажется на первый взгляд:

- MS Reporting Services
- SAP Crystal Reports
- Qlik View, MS Power BI (Desktop OLAP)
- JavaScript-библиотеки:
  - YARG – для отчетов,
  - Amcharts и Hightcharts – для диаграмм.

Таблица 1. Результаты сравнения технологий доступа к данным 1С

Вариант	Достоинства	Недостатки
Выгрузка в файл	> Простота разработки на стороне 1С	> Данные получаются не online > На стороне системы отчетности нужна отдельная БД > Необходима отдельная процедура загрузки данных в систему отчетности > Сложность модификации подобной системы
DWH	> Сложность и дороговизна разработки > Данные получаются не online	> Высокая скорость работы с большими объемами данных > Независимость от архитектуры корпоративной информационной системы
COM доступ	> Со стороны 1С нет необходимости разработки > Использование для получения данных средств самой 1С > Достаточно высокая скорость доступа к данным > Получение данных online	> Необходимость создавать COM объект, поддерживать соединение > Использует лицензию 1С > Только для ОС Windows
Веб-сервисы	> Универсальный протокол, поддерживаемый большинством систем отчетности > Возможность использования протокола ODATA для доступа к данным	> Внутри используются протоколы HTTP и XML при больших объемах передаваемых данных могут быть проблемы > Большие объемы передаваемых вспомогательных данных > Расходуются лицензии 1С > При каждом вызове происходит установка соединения с 1С, что занимает время и ресурсы
Прямой доступ SQL	> Высокая скорость получения данных > Данные получаются online > Разработка только на уровне системы отчетности > Нет необходимости в дополнительных установках соединений и лицензиях	> При изменении структуры метаданных системы необходимо модифицировать систему отчетности > Достаточно существенно привязана к существующей корпоративной информационной системе

Полноценные «большие» OLAP-системы вида SAP BO, IBM Cognos и MS Analysis Services в данном случае мы рассматривать не будем, т.к. едва ли их можно называть «системами отчетности». Все-таки задачи внедрения BI-системы и построения красивых отчетов для руководства нужно разделять.

Рассмотрим их более детально по порядку.

Если есть время, ресурсы и соответствующие компетенции, то можно использовать бесплатные библиотеки и рисовать отчеты, до которых «готовым» системам еще далеко

MS Reporting Services

Поставляется как один из компонентов MS SQL Server. Данная система отчетности используется практически во всех современных продуктах Microsoft, включая MS Dynamics Ax и MS Dynamics CRM. Внешний вид отчета, созданного с помощью MS Reporting Services, представлен на рис 1.

В состав MS Reporting Services входит достаточно удобный и простой инструмент MS Report Builder, который позволяет создавать основные виды отчетов. Отчеты также можно создавать с помощью MS Visual Studio, где уже доступны расширенные средства.

Официальный сайт – [1]. Последняя версия использует все возможности HTML 5, в целом отчеты выглядят очень даже симпатично и в общем стиле Microsoft. Естественно, интегрируется с MS Power BI и MS SharePoint Server. Есть мобильная версия.

SAP Crystal Reports

Давно всем известная система отчетности. Пожалуй, одна из первых, что появились на свет. В 2003 году была куплена компанией Business Objects, которая, в свою очередь, в 2011-м была куплена компанией SAP.

Выглядят отчеты тоже достаточно симпатично (см. рис. 2).

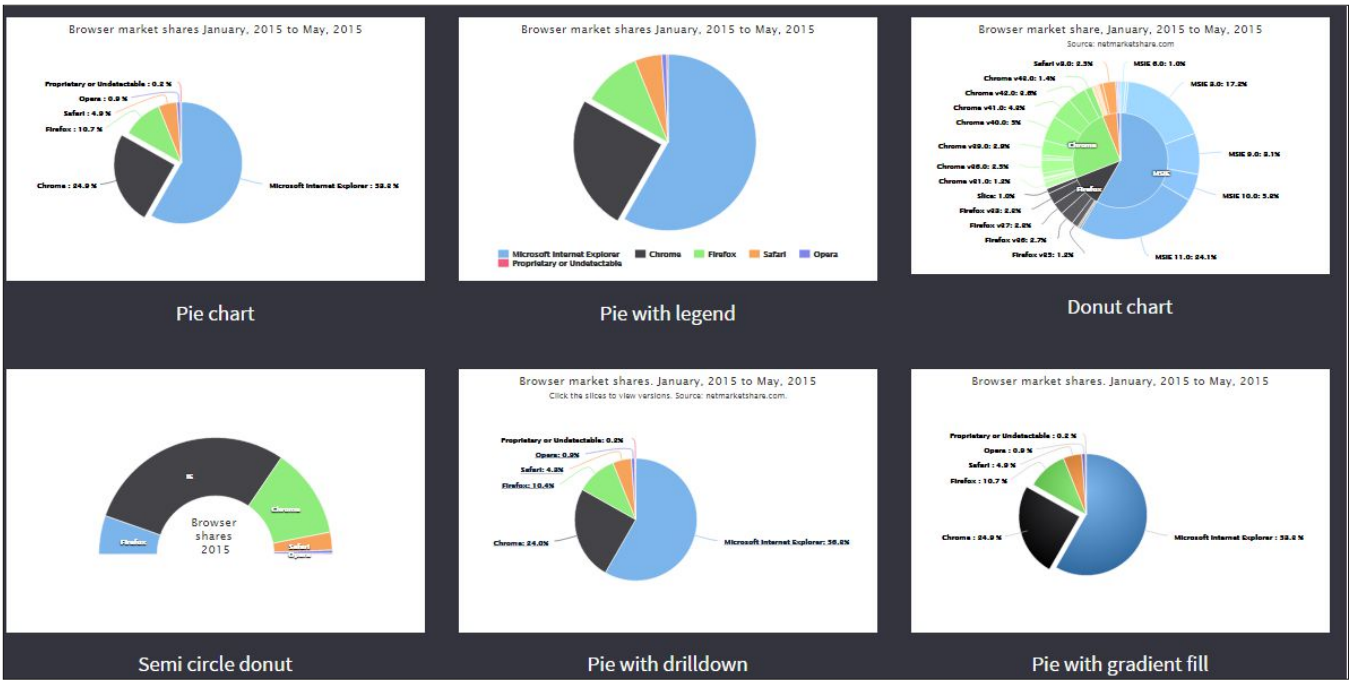
Так же, как и для Microsoft Reporting Services, есть дизайнер отчетов и интеграционные дополнения для MS Visual Studio. В последней версии поддерживается HTML 5, есть мобильная версия. В общем, все, что нужно для полноценной системы отчетности. Разница лишь в цене. Если для пользователей MS SQL Server – Reporting Services, по сути, бесплатное дополнение, то Crystal Reports очень даже платная. Официальный сайт [2] сразу перенаправляет нас на страницу «SAP магазина».

Системы Desktop OLAP

MS Power BI и QlikView представляют собой один класс систем. Суть их работы заключается в создании куба «на лету» в оперативной памяти, что позволяет использовать как преимущества OLAP для анализа данных, так и достоинства получения данных непосредственно из источников, без создания DWH. Внешний вид отчетов Power BI соответствует отчетам MS Reporting Services. Внешний вид отчетов QlikView представлен на рис. 3.

Главный минус данных систем отчетности заключается в отсутствии развитых средств кастомизации итоговых отчетов. Выражаясь простым языком, эти системы – инструмент аналитика для анализа данных и предоставления руководству уже итогового результата. Официальный сайт

Рисунок 4. Внешний вид диаграмм HightCharts



Qlik – [3], Power BI – [4]. Обе системы достаточно недороги, кроме того, существуют бесплатные версии на одного пользователя.

Библиотеки JavaScript

Сразу стоит оговориться, что библиотеки JavaScript – это не готовые системы построения отчетности, в отличие от рассмотренных выше. Но в качестве отчетов для 1С они используются очень часто. Программный интерфейс данных библиотек достаточно удобен, что делает разработку отчетов с их помощью не таким трудоемким занятием, как может показаться на первый взгляд. Основные библиотеки можно назвать три:

- AmCharts [5]
- HightCharts [6]
- YARG [7]

На рис. 4 представлен внешний вид части диаграмм HightCharts, а на рис. 5 соответственно AmCharts. Как по мне, так внешний вид этих диаграмм даже симпатичнее, чем у SAP или Microsoft. И, что самое интересное, все это абсолютно открыто и бесплатно.

Удобного конструктора отчетности в этом случае, конечно же, нет, никакой штатной интеграции, естественно, тоже. Но возможности ничуть не уступают дорогим системам отчетности. Данные библиотеки уже нередко используются внутри решений 1С для придания отчетам 1С соответствующего внешнего вида [8].

Итак, выводы по системам отчетности тоже могут быть весьма разнообразными. Однозначного решения нет. Если есть время, ресурсы и соответствующие компетенции, то, конечно, можно использовать бесплатные библиотеки и рисовать отчеты, до которых «готовым» системам еще далеко. Но если есть ограничение во времени/ресурсах/финансах, а в компании повсеместно используются продукты Microsoft, то выбор тоже достаточно очевиден. Если же есть «специально обученный аналитик» и вся отчетность замкнута на нем, то ему в руки нужно всего лишь дать полноценный инструмент. EOF

[1] Официальная информация об MS Reporting Services – https://msdn.microsoft.com/ru-ru/library/ms159106(v=sql.120).aspx.
[2] Официальный сайт SAP Crystal Reports http://www.crystalreports.com.
[3] Официальный сайт Qlik – http://global.qlik.com/ru.
[4] Официальный сайт MS Power BI – https://powerbi.microsoft.com/ru-ru.
[5] JavaScript-библиотека Amcharts – https://www.amcharts.com.
[6] JavaScript-библиотека HightCharts – http://www.highcharts.com.
[7] JavaScript-библиотека генерации отчетов YARG – https://github.com/Haulmont/yarg.
[8] Пример использования библиотеки AmCharts для построения красивых диаграмм в отчетах из 1С – http://infostart.ru/public/225601.

Ключевые слова: 1С, система отчетности, библиотеки.

Рисунок 5. Внешний вид диаграмм AmCharts

Визитка

ТИМУР ШАМИЛАДЗЕ,
руководитель, ООО «БухКонсалтинг», timurshamiladze@yandex.ru

Настройка SQL Server
для производительной работы 1С

MS SQL Server – самая распространенная СУБД для клиент-серверного использования 1С. Какие настройки необходимо произвести в ней для обеспечения максимально производительной работы 1С?

Самый лучший вариант – если сервер СУБД расположен на отдельном компьютере, на котором другие серверные роли не установлены (контроллер домена, терминальный сервер и пр.). Но ничто не мешает установить сервер 1С и СУБД на один компьютер, если нагрузка на систему будет небольшой.

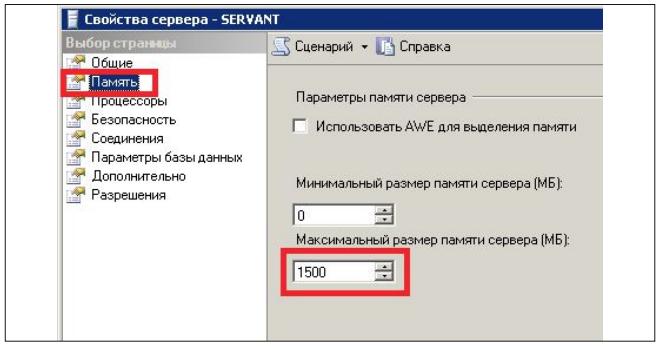
При выборе версии MS SQL Server стоит использовать последние, т.к. в новых версиях исправляют ошибки и оптимизируют механизмы, что приводит к улучшению производительности. Рассмотрим основные настройки.

Использование памяти

Следует ограничивать максимальный объем оперативной памяти, используемой MS SQL Server, если на сервере, помимо СУБД, работают другие ресурсоемкие приложения, например сервер 1С. Если этого не сделать, то SQL Server может занять столько памяти, что это помешает другим приложениям полноценно работать. Для того чтобы вычислить, какой объем памяти необходим, нужно определить, сколько оставить ОС и серверу 1С.

Обычно для нормальной работы операционной системе достаточно 4 Гб. Для систем с большим объемом памяти рекомендуется для ОС оставлять еще 1 Гб памяти на каждые 16 Гб, установленной RAM.

Рисунок 1. Настройка Max server memory



Чтобы вычислить, какой объем памяти оставить для сервера 1С, необходимо посмотреть, сколько памяти занимают процессы кластера серверов 1С в пиковые нагрузки, и добавить к этому числу некоторый резерв. К процессам кластера серверов 1С относят rphost, ragent, rmngr.

Получаем следующую формулу:

Память для SQL Server = Всего память – Память для ОС – Память для сервера 1С

Максимальный объем памяти, выделяемый SQL Server, устанавливается параметром Max server memory (Максимальный размер памяти сервера). Для этого необходимо в Management Studio через контекстное меню открыть свойства сервера и на странице «Память» указать параметр «Максимальный размер памяти сервера» (см. рис. 1).

Расположение файлов базы данных и журнала транзакции

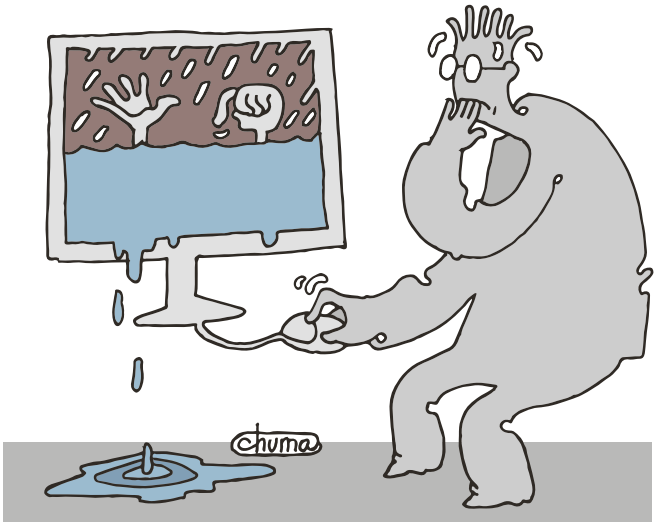
Каждая база данных состоит из файла базы данных и журнала транзакций.

Файл данных имеет расширение *.mdf или *.ndf и содержит непосредственно данные и хранит содержимое таблиц базы данных. Журнал транзакций имеет расширение *.ldf, содержит информацию, необходимую для отката или фиксации транзакции. У одной базы данных файлов обеих типов может быть несколько.

Исходя из назначения этих файлов для них существуют различные требования:

- Для журнала транзакций необходима высокая скорость последовательной записи.
- Для файла данных необходима высокая скорость случайного доступа на чтение и запись.

Например: один пользователь формирует «сложный» аналитический отчет, и в это же время другие пользователи заносят и проводят документы. Если файл данных и журнал транзакций расположены на одном диске, то жесткий диск вынужден постоянно переключаться между массовыми



Задав правильно параметры СУБД для работы 1С, мы можем избавиться от целого ряда проблем

операциями чтения данных при формировании отчета и операциями записи в журнал транзакций при проведении документов. В результате таких действий работа дисковой подсистемы будет неэффективной.

Рекомендация: файлы базы данных и журнала транзакции следует разнести по разным дисковым подсистемам.

Чтобы поменять местоположение файла базы данных или журнала транзакции необходимо:

- Создать бэкап.
- В Management Studio вызвать правой кнопкой мыши контекстное меню на имени нужной базы, выбрать «Задачи» (Tasks) – «Отсоединить» (Detach) и нажать «OK».
- Переместить файлы базы и журнала транзакций в нужный каталог.
- В Management Studio вызвать правой кнопкой мыши контекстное меню на узле базы данных (Databases), выбрать «Присоединить» (Attach).
  - В поле «Базы данных для присоединения» (Databases to attach) добавить новый путь.
  - В поле «Сведения о базе данных» (Database details) изменить путь к файлу журнала транзакций на новый.
- Нажать «OK».

Расположение базы TempDB

TempDB – служебная база данных, используемая экземпляром MS SQL Server для хранения временных таблиц, табличных переменных, версий данных при использовании RCSI, промежуточных наборов данных при выполнении запросов. Так же как и любая другая база в СУБД, она имеет файл данных и журнал транзакций.

Эта база является общей для всех баз данных, работающих на данной СУБД, и нагрузка на нее может быть очень высокой, поэтому часто она становится узким местом.

Также для минимизации транзакционных блокировок в 8.3 используется режим версионности – а он увеличивает нагрузку на TempDB, т.к. версии строк данных хранятся в этой базе данных.

Рекомендация: вынести базу TempDB на отдельный быстрый диск (желательно RAM или SSD).

Листинг 1. Запрос, перемещающий базу TempDB на другой диск.

```
use master
go
alter database tempdb modify file (NAME = tempdev, ↵
        FileName = 'Новый диск:\Новый каталог\tempdb.mdf')
go
alter database tempdb modify file (NAME = templog, ↵
        FileName = 'Новый диск:\Новый каталог\templog.ldf')
go
```

Настройка авторасширения базы

С увеличением количества данных в базе растет и ее размер, но он растет не постепенно, а дискретно. Объем базы данных при достижении предела увеличивается на конкретно задаваемый размер. По умолчанию это 1 Мб, а это очень незначительный объем, и поэтому на постоянное увеличение размера базы будут тратиться ресурсы SQL Server.

Рекомендация: задать авторасширение базы 5 Гб (или не менее 500 Мб).

Параметр авторасширения задается в свойствах базы данных, на странице «Файлы» в колонке «Авторасширение» (см. рис. 2).

Рисунок 2. Установка значения авторасширения базы

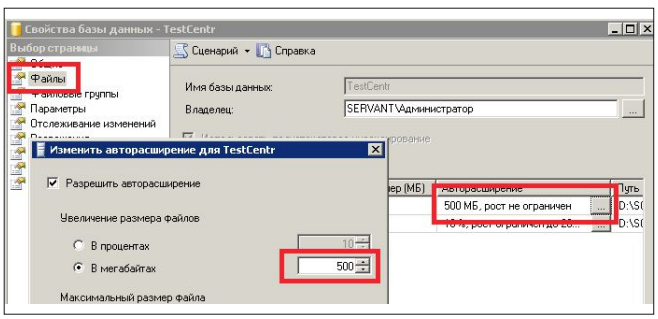


Рисунок 3. Установка модели восстановления

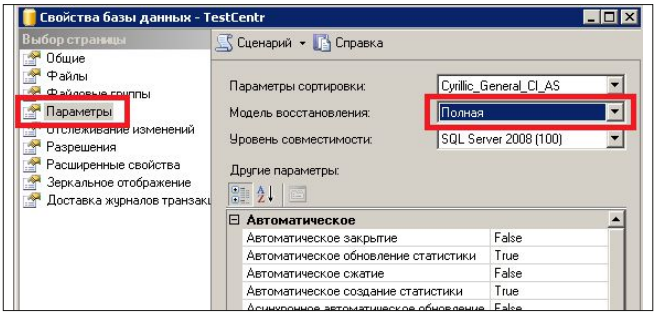


Рисунок 4. Настройка резервного копирования

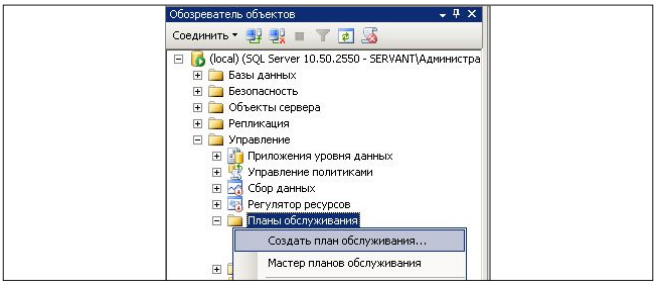


Рисунок 5. Установка максимальной степени параллелизма

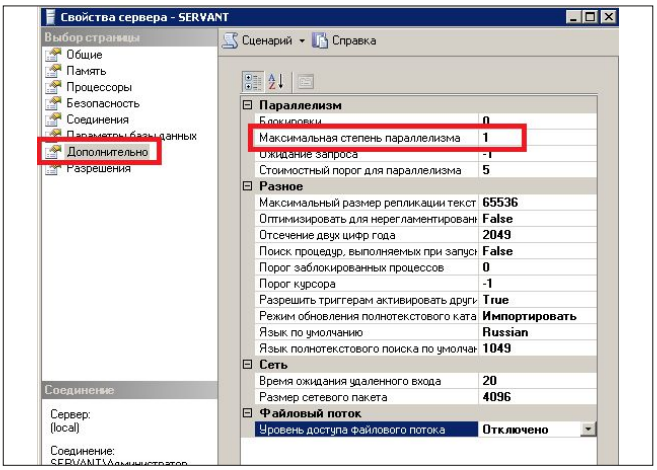
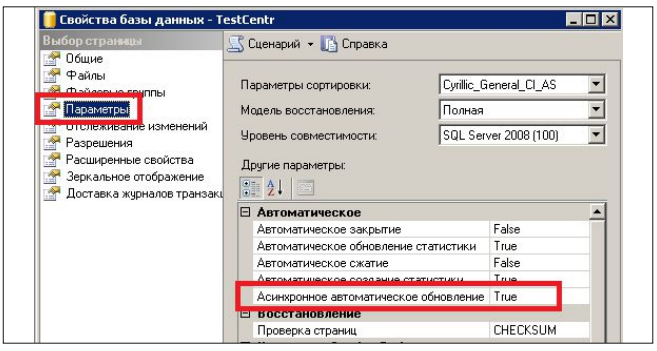


Рисунок 6. Установка асинхронного обновления статистики



Установка полной модели восстановления базы данных

Если мы хотим иметь возможность восстановить состояние базы данных на точно определенный момент времени, то следует использовать FULL (полную) модель восстановления.

Модель восстановления задается в Management Studio. В контекстном меню на имени нужной базы выбираем «Свойства» (Properties) и переходим на страницу «Параметры» (Options). Нужное значение задается в поле «Модель восстановления» (Recovery model) (см. рис. 3).

Настроить резервное копирование и проверить его работу

Старая истина гласит: системные администраторы делятся на тех, кто «делает архивные копии», и тех, кто «будет делать архивные копии». Архивные данные для баз 1С особенно важны, т.к. обычно там хранится вся учетная и финансовая информация по организации.

Рекомендация: создавать архивные копии раз в день.

Архивировать данные можно с использованием планов обслуживания:

- Создаем новый план обслуживания в Management Studio: «Управление» (Management) → «Планы обслуживания» (Maintenance Plans) → «Создать план обслуживания» (New Maintenance Plan) (см. рис. 4).
- Указываем имя нашего плана.
- Добавляем задачу «Резервное копирование базы данных» из панели задач раздела «Общие».
- Указываем расписание для данной операции, а также для каких баз данных делать копию.

Установка параметра Max dergee of Parallelism

Параметр Max dergee of Parallelism (Максимальная степень параллелизма) указывает, сколько процессоров может быть использовано при выполнении одного запроса. По умолчанию параметр равен 0, т.е. для выполнения запроса могут использоваться все процессоры. Для OLTP-систем рекомендуется устанавливать этот параметр в 1. Для его установки в Management Studio открываем свойства сервера и на странице «Дополнительно» указываем параметр «Максимальная степень параллелизма» (см. рис. 5).

Обновление статистики

Статистика – служебный объект, который хранит информацию о количестве и распределении значений в таблице и индексе, используется при построении планов запроса.

Для оптимальной работы запросов в СУБД необходимо, чтобы статистика находилась в актуальном состоянии, т.к. исходя из статистики оптимизатор СУБД строит планы запроса, в соответствии с которыми они выполняются. Если статистика окажется неактуальной, то СУБД может построить плохой план запроса, что замедлит работу пользователей.

Статистика обновляется автоматически, если в свойствах базы данных установлено значение параметра Auto update statistics (Автоматическое обновление статистики) = True.

Порог автоматического обновления статистики зависит от количества измененных данных, и СУБД может не обновить статистику, когда ее действительно нужно обновить, поэтому с помощью регламентного задания необходимо принудительно обновлять статистику.

Рекомендация: обновлять статистику раз в день для часто меняющихся данных.

Обновить статистику можно с использованием планов обслуживания:

- Создаем новый план обслуживания в Management Studio: «Управление» (Management) → «Планы обслуживания» (Maintenance Plans) → «Создать план обслуживания» (New Maintenance Plan).
- Указываем имя нашего плана.
- Добавляем задачу «Обновление статистик» (Update Statistics Task) из панели задач.
- Указываем расписание для данной операции, а также для каких баз и таблиц обновлять статистику.

Если будет включено асинхронное обновление статистики, то СУБД, обнаружив при выполнении запроса устаревшую статистику, продолжит выполнение запроса, но параллельно запустит процесс обновления статистики, и уже следующий запрос к этой таблице будет выполняться с актуальной статистикой (см. рис. 6).

Очистка процедурного кэша

СУБД, чтобы постоянно не строить планы запроса, их помещает в процедурный кэш, тем самым облегчая себе работу. И когда в СУБД придет такой же запрос, то уже не придется его заново строить, а просто берется уже готовый. Но может случиться так, что из-за неактуальной статистики оптимизатор СУБД построил неоптимальный план запроса и его закэшировал. Поэтому, чтобы мы всегда имели актуальный план запроса, после обновления статистики рекомендуется сразу сделать очистку процедурного кэша.

Очистку можно произвести с помощью команды:

```
DBCC FREEPROCCACHE
```

Для автоматизации выполнения очистки процедурного кэша следует также использовать планы обслуживания.

Дефрагментация индексов

Из-за интенсивной работы с таблицами базы данных возникает эффект фрагментации индексов, который приводит к снижению эффективности работы запросов. Дефрагментация (реорганизации) индексов – процесс устранения фрагментации. Дефрагментацию рекомендуется делать, если фрагментация не более 30%. Выполнение дефрагментации не блокирует работу пользователей, но создает некоторую нагрузку на оборудование, поэтому не рекомендуется выполнять дефрагментацию в рабочее время.

Рекомендация: выполнять дефрагментацию минимум один раз в неделю.

Дефрагментация выполняется командой:

```
ALTER INDEX ALL ON «ИмяТаблицы» REORGANIZE
```

Реиндексация индексов

Реиндексация – операция пересоздания индексов. Эту операцию рекомендуется выполнять, когда в индексе большая степень фрагментации (> 30%).

Реиндексация выполняется командой:

```
ALTER INDEX ALL ON «ИмяТаблицы» REBUILD
```

• • •

Производительность 1С очень сильно зависит от MS SQL Server. Задав правильно параметры СУБД для работы 1С, мы можем избавиться от целого ряда проблем. Хотя большинство советов касается и других приложений, работающих в связке с этой СУБД. EOF

Ключевые слова: MS SQL Server, 1С, оптимизация, СУБД.

Визитка

АНДРЕЙ ПАХОМОВ,
Android-разработчик, mailforpahomov@gmail.com

Механизмы уведомлений в ОС Android

Изучаем способы, как неназойливо привлечь внимание пользователя

Напомнить пользователю

Мобильные устройства с полноценными операционными системами появились относительно недавно, но мы к ним уже настолько привыкли, что без них жизнь невозможна. Некоторые приложений для смартфонов и планшетов стали настоящими помощниками, существенно облегчающими как бизнес-процессы или учебу, так и повседневную жизнь.

Вполне привычно, когда распорядок дня или целой недели может серьезно поменяться после прочтения письма по электронной почте, сообщения из мессенджеров или уведомления о том, что таксист опаздывает из-за вечерних пробок. Современное мобильное приложение должно вовремя оповещать о пропущенном звонке или предстоящей встрече. Давайте проверим, позволяет ли техническое оснащение современных носимых устройств на базе Android свести такие казусы к минимуму.

Для оповещения владельца мобильного устройства о новых событиях реализован механизм уведомлений. Уведомление (официальное название notification) – это сообщение от приложения, которое ОС Android выводит пользователю вне интерфейса приложения. Для показа уведомлений в Android в верхней части дисплея расположена панель уведомлений, куда система помещает сообщения от всех приложений на устройстве.

Чтобы лучше понять, как все работает, попрактикуемся сегодня в создании собственных уведомлений. Хотя задача и выглядит достаточно простой, существуют нюансы, которые следует держать в уме при создании информационного сообщения для пользователя.

Создаем уведомление

Итак, приступим. Первым делом подключимся к системному сервису NOTIFICATION_SERVICE, который обслуживает все уведомления в системе. Для этого воспользуемся классом NotificationManager, он позволит передать в ОС Android сформированное сегодня уведомление.

Подключение к сервису осуществляется вызовом метода getSystemService.

```
mNotificationManager= (NotificationManager) ↵
          getSystemService(NOTIFICATION_SERVICE);
```

Удивительно, но все приготовления на этом уже закончены. Теперь пора сконструировать наше первое уведомление, основная работа будет возложена на класс NotificationCompat.Builder.

Мы создадим объект mBuilder, а данные в уведомление будем помещать с помощью уже существующих в объекте методов. В минимальном наборе полноценное уведомление должно содержать иконку, отображаемую в панели уведомления, и текст сообщения для пользователя. Как это принято в Android, текст будет показан пользователю на заблокированном экране или при развертывании вниз панели уведомлений.

```
NotificationCompat.Builder mBuilder = ↵
         new NotificationCompat.Builder(this)
.setSmallIcon(R.mipmap.ic_launcher)
.setContentTitle("Notification message")
.setContentText("Text for user")
```

Хорошее уведомление должно иметь практический характер и давать возможность как-то отреагировать. Вероятнее всего, пользователю потребуется запустить наше приложение и внести какие-то изменения. С этой целью предусмотрена возможность запуска нового Activity, принадлежащего нашему приложению, путем простого касания уведомления.

Как всегда, для запуска нового Activity требуется сформировать Intent, содержащий в себе нужный класс нашего приложения.

```
Intent resultIntent = new Intent(this, ↵
         NotificationReceiverActivity.class);
```

Вполне вероятна ситуация, когда новое событие происходит в тот момент, когда приложение не используется пользователем, и в системе остается активным только сервис, выполняющий в фоне действия по обновлению контента.

Мобильное устройство должно оперативно оповещать об изменениях в окружающем мире

В таком случае для работы с приложением требуется создать новый стек.

Стек в Android – это набор следующих друг за другом «окон» приложения, именуемых Activity. При выходе из последнего Activity на экране появится предыдущий. Самый первый же (тот, который использовался раньше всех) может быть окончательно выгружен из системы в целях экономии ресурсов. Стек в Android создается автоматически самой системой, также его возможно задать вручную, используя класс TaskStackBuilder.

```
TaskStackBuilder stackBuilder = ↵
         TaskStackBuilder.create(this);
stackBuilder.addParentStack(NotificationReceiverActivity.class);
stackBuilder.addNextIntent(resultIntent);
```

При касании уведомления на устройстве будет запущен объект NotificationReceiverActivity. По логике работы с уведомлениями он должен дать точный ответ, что же изменилось в приложении.

```
public class NotificationReceiverActivity extends Activity {
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.result);…}…}
```

Теперь нужно продумать логику дальнейших действий пользователя. Скорее всего после изучения повода для вывода уведомления он захочет попасть в главное меню приложения. Чтобы это позволить, внесем небольшие изменения в манифест приложения.

```
<activity
android:name=".NotificationReceiverActivity"
android:parentActivityName=".MainActivity".../>
```

Мы создали новый параметр parentActivityName, который говорит системе, что у класса, выдающего информацию об уведомлении, есть родительский класс, в который будет выполнен переход после нажатия кнопки «назад».

Приготовления закончены, теперь созданный Activity и сформированный стек требуется подключить к созданному уведомлению, объединив все в объект класса PendingIntent. Этот объект возможно передать системе или другому приложению с тем, чтобы они смогли выполнить часть кода нашего приложения, помещенного в PendingIntent. Поскольку запущенный системой или сторонним приложением PendingIntent будет обладать всеми полномочиями нашего приложения, нужно быть внимательным и не помещать в стек больше, чем нужно.

```
PendingIntent resultPendingIntent = ↵
         stackBuilder.getPendingIntent(0, ↵
         endingIntent.FLAG_UPDATE_CURRENT);
mBuilder.setContentIntent(resultPendingIntent);
mNotificationManager.notify(messageId, mBuilder.build());
```

Итак, наше уведомление сформировано! Как только будет вызван метод notify, ОС Android оповестит пользователя о новых событиях в нашем приложении. Хорошее уведомление должно обладать большей функциональностью. Новые параметры возможно добавить, вызывая методы в созданном объекте mBuilder.

Например, после касания пользователем уведомления оно должно исчезнуть с экрана устройства.

```
.setAutoCancel(true)
```

Привлекаем внимание

Визуальную нотификацию будет логичным продублировать звуком и вибрацией устройства. Для доступа к виброзвонку нужно запросить разрешение у системы, звук же доступен по умолчанию.

```
<uses-permission android:name="android.permission.VIBRATE" />
```

Чтобы телефон вибрировал, нужно указать системе, в какой момент времени необходимо включить моторчик и через сколько его выключить. Для этого нужно сформировать массив переменных типа long. В нашем случае вибрация

будет недолгой: мы включим моторчик и выключим его через 100 миллисекунд. Сформированный массив подадим в качестве аргумента соответствующему методу.

```
long [] pattern=new long[]{0, 100};
mBuilder.setVibrate(pattern);
```

Теперь поработаем со звуком. Мы не будем использовать какие-то экзотические для пользователя звуки и для нашего уведомления, лучше возьмем мелодию, которую пользователь выбрал сам для такого рода событий. Доступ ко всем стандартным звукам осуществляется через класс RingtoneManager.

```
Uri soundUri = RingtoneManager.getDefaultUri( ↵
        RingtoneManager.TYPE_NOTIFICATION);
mBuilder.setSound(soundUri);
```

Во многих устройствах на базе Android есть LED-индикатор, который тоже возможно использовать в своих приложениях. Достаточно в созданном объекте-уведомлении задать параметры цвета лампочки и характер сигнала. Сегодня мы с небольшой задержкой включим индикатор, окрасим его в зеленый цвет и выключим через 100 миллисекунд.

```
Notification fullNotif = mBuilder.build();
fullNotif.ledARGB = Color.argb(255, 0, 255, 0);
fullNotif.flags |= Notification.FLAG_SHOW_LIGHTS;
fullNotif.ledOnMS = 200;
fullNotif.ledOffMS = 300;
```

Уведомления с выбором

Реакция на наше уведомление может быть разной. К двум базовым – зайти в приложение или не обращать внимание вообще – можно добавить еще любых три.

```
mBuilder.addAction(R.mipmap.ic_launcher,"ActionButton",↵
        buttonIntent);
```

Теперь внизу уведомления появится небольшая кнопка с заданными иконкой и текстом. При ее касании будет запущен объект уже знакомого нам класса PendingIntent. Это может быть запуск Activity как из нашего приложения, так и стороннего. К примеру, можно предложить пользователю написать текстовое сообщение.

```
Intent smsIntent = new Intent(Intent.ACTION_VIEW);
smsIntent.setData(Uri.parse("sms:"));
TaskStackBuilder smsStack = TaskStackBuilder.create(this);
smsStack.addNextIntent(smsIntent);
PendingIntent smsPIntent = smsStack.getPendingIntent(0, ↵
        PendingIntent.FLAG_CANCEL_CURRENT);
```

Для написания сообщений мы запросили у системы показать новое окно с данными (ACTION_VIEW). Метод setData позволяет указать системе, какие именно данные мы хотим показать пользователю. Как обычно, для открытия нового Activity нам необходим стек. Поскольку отправка сообщения будет происходить через стороннее приложение, в стеке будет только сформированный интент.

Приватность

Современный житель мегаполиса большую часть дня проводит в окружении случайных попутчиков или коллег. Легко представить себе ситуацию: в общественном месте на его мобильное устройство поступает информация, которой он не готов делиться с окружающими. В некоторых ситуациях даже уведомления могут быть излишними, на этот случай в Android введено дополнительное разделение.

```
mBuilder.setVisibility(NotificationCompat.VISIBILITY_SECRET);
```

Существует три варианта видимости уведомлений: показывать все без ограничений (VISIBILITY_PUBLIC), отображать

Рисунок 1. Уведомление на заблокированном экране

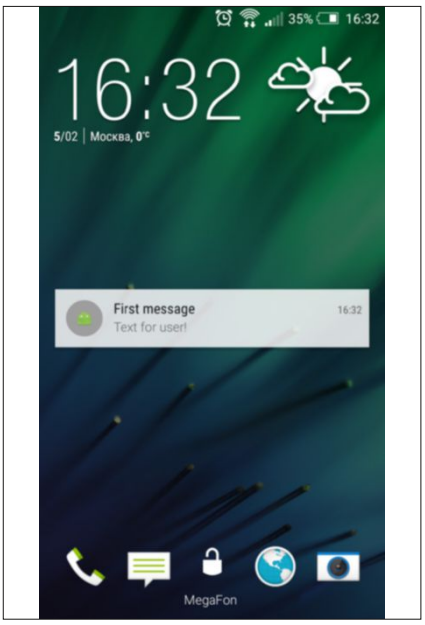


Рисунок 2. Полная версия уведомления

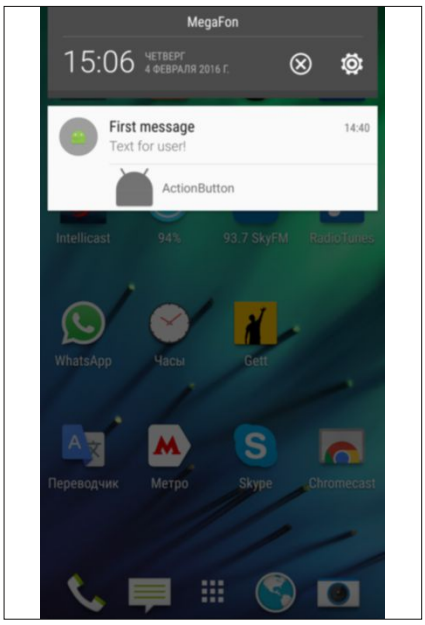
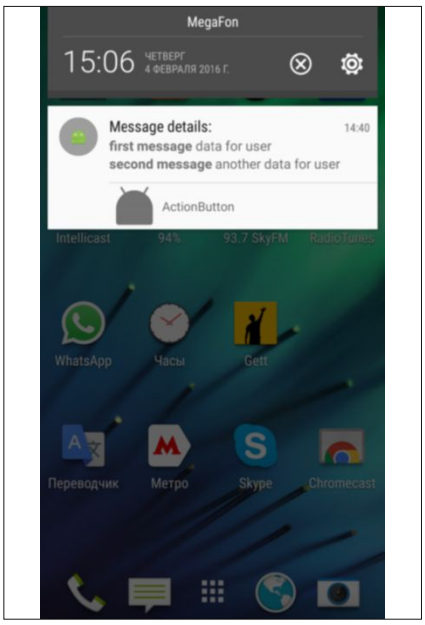


Рисунок 3. Сгруппированные уведомления



на заблокированном экране только иконку и заголовок уведомления (VISIBILITY_PRIVATE) и не отображать уведомление на заблокированном экране вообще (VISIBILITY_SECRET).

Любопытно, что данный параметр учитывается системой только в случае, если пользователь использует пароль или паттерн для блокировки экрана, а также в настройках системы выставлен пункт «показывать только заголовки, скрывать содержимое» (название может немного отличаться в зависимости от устройства).

Группировка уведомлений

Очень важно соблюдать баланс при использовании уведомлений. Согласитесь, что, если на заблокированном экране появится сразу с десяток одинаковых сообщений, это будет выглядеть странно. Подобный «спам» у неопытных пользователей может вообще отбить желание пользоваться современными средствами связи. Во избежание подобных инцидентов уведомления о схожих событиях следует группировать.

Для группирования текстовых сообщений воспользуемся классом NotificationCompat.InboxStyle. Это уже готовый инструмент для формирования уведомления-списка о нескольких событиях, несущих прежде всего смысловую (текстовую) нагрузку.

```
NotificationCompat.InboxStyle inboxStyle = ↵
        new NotificationCompat.InboxStyle();
inboxStyle.setBigContentTitle("Message details:");
```

В созданном списке есть возможность выделить определенные сообщения или их часть. Применим возможности класса Spannable, который позволяет в тексте менять кегль и другие параметры.

```
Spannable[] events = new SpannableString[2];
events[0]=new SpannableString("first message data for user");
events[0].setSpan(new StyleSpan(android.graphics. ↵
        Typeface.BOLD), 0, "first message".length(), ↵
        Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
```

В данном случае начальная фраза сообщения будет выделена жирным. Это распространенный прием, к примеру, таким способом выделяется имя абонента, приславшего сообщение через один из популярных мессенджеров. Чтобы заполнить список InboxStyle, достаточно организовать простой цикл.

```
for (int i=0; i < events.length; i++) {inboxStyle.↵
        addLine(events[i]);}
```

Созданный стиль мы передадим в объект mBuilder, а затем уведомим систему об изменении текста уведомления для пользователя. К слову, приложение может выдавать несколько сообщений одновременно, задав для каждого уникальный идентификатор messageId.

```
mBuilder.setStyle(inboxStyle);
mBuilder.setContentTitle("extra info");
mBuilder.setContentText("more messages received");
Notification fullNotif = mBuilder.build();
```

Расширенное уведомление готово! Если пользователь пропустил первое уведомление, то теперь оно будет заменено системой на новое. Параметры ContentTitle и ContentText тоже следовало обновить, т.к. они будут выведены на заблокированном экране.

Кроме списка IndoxStyle, нам доступны и другие форматы расширенных уведомлений. К примеру, можно поместить крупное изображение с помощью BigPictureStyle или прислать уведомление от часов Android Wear, используя WearableExtender.

Иерархия уведомлений

Бывают ситуации, когда о произошедшем событии пользователю следует знать не только безотлагательно, но и раньше, чем обо всех других событиях в системе. Для этого у нас есть возможность выставить приоритет для уведомления.

```
mBuilder.setPriority(NotificationCompat.PRIORITY_MAX);
```

Всего существует пять градаций важности сообщений, с высшим приоритетом приходят уведомления о телефонных звонках или критическом разряде батареи. Сообщения с минимальным приоритетом (например, обновление данных о прогнозе погоды) не отображаются на панели уведомлений.

Если вы точно не уверены, какой приоритет выставлять для уведомлений в вашем приложении, в классе Notification есть более подробная градация. В этом случае нужно воспользоваться методом setCategory. Так, уведомления о сообщениях в социальных сетях должны приходить с приоритетом CATEGORY_SOCIAL.

```
mBuilder.setCategory(NotificationCompat.CATEGORY_SOCIAL);
```

• • •

В современном обществе время – один из самых ценных ресурсов, и логично, что мобильное устройство должно оперативно оповещать об изменениях в окружающем мире. В то же время сервис уведомлений следует использовать только в тех приложениях, результат работы которых важен для пользователя именно здесь и сейчас.

Компания Google просит очень ответственно относиться к созданию уведомлений. Несвоевременные или пустяковые сообщения могут серьезно испортить впечатления обо всем приложении. Так, некоторые разработчики пытаются удержать пользователя уведомлениями вида «Вы давно не открывали наше приложение, возвращайтесь скорее», что скорее отпугивает. Используйте уведомления исключительно по делу, и популярность ваших электронных продуктов будет только расти.

Сегодня нам удалось достаточно подробно разобрать, как, когда и каким образом стоит привлекать внимание пользователя. Исходный код приложения доступен на сайте журнала, если же будут какие-то вопросы, рекомендую заглянуть в официальное руководство от Google или написать мне по электронной почте. Удачи! EOF

[1] Официальное руководство от Google по созданию уведомлений – http://developer.android.com/guide/topics/ui/notifiers/notifications.html.

Ключевые слова: Android, уведомления, программирование.

Визитка

АЛЕКСАНДР КАЛЕНДАРЕВ,
РБК Медиа, программист, akalend@mail.ru

Hack. Асинхронность

Рассмотрим возможность асинхронного выполнения кода приложения, написанного на языке Hack [1]

Происходит слово «асинхронность» от греческого α – отрицание, συν – вместе, χρονος – время, т.е. несовпадение с чем-либо во времени. Асинхронность характеризует процессы, не совпадающие во времени, т.е. такие, которые запускаются не синхронно, не строго, как только один закончился, так сразу начал исполняться следующий.

Рассмотрим работу некоторого абстрактного скрипта. Как правило, скрипт получает необходимые данные, обрабатывает их, записывает куда-то промежуточные результаты и демонстрирует результаты работы. В то время, когда выполняются операции чтения/записи, процессор простаивает. Как мы можем изменить логику работы на более низком уровне.

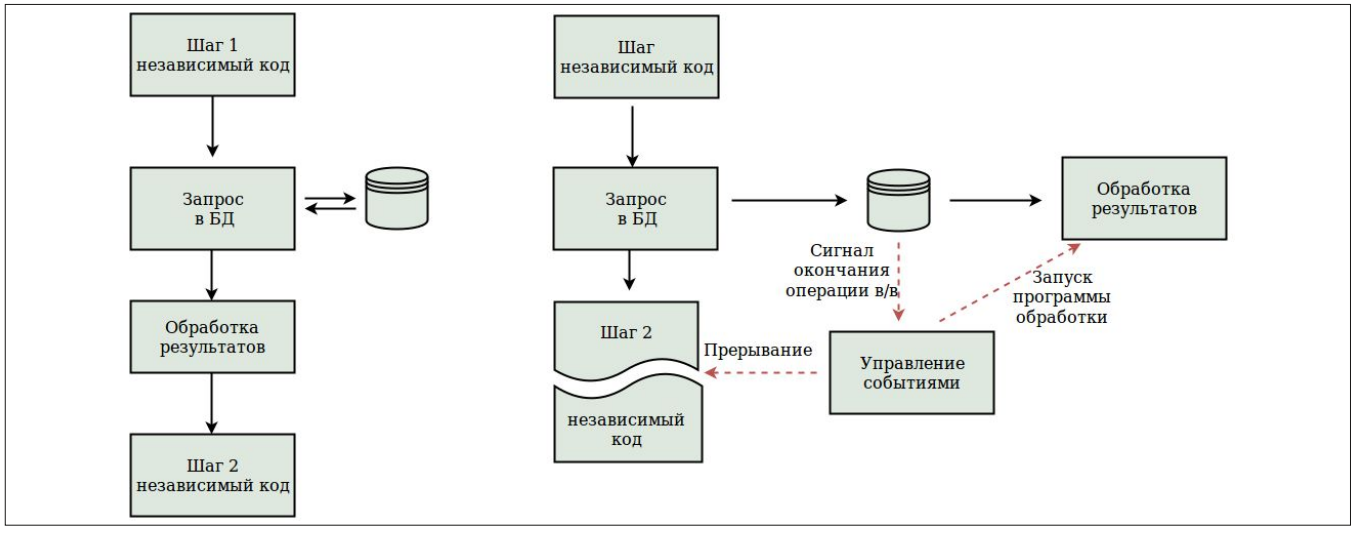
Можно в точке выполнения операций, не требующих процессорного времени, сделать ветвление и одной ветвью продолжить выполнение этих операций, а другой осуществлять выполнение остальной логики скрипта. А когда окончится выполнение операций, не требующих процессорного времени, то возникнет прерывание, по которому произойдет передача управления на функцию обратного вызова обработки результатов, если это предусмотрено логикой программы.

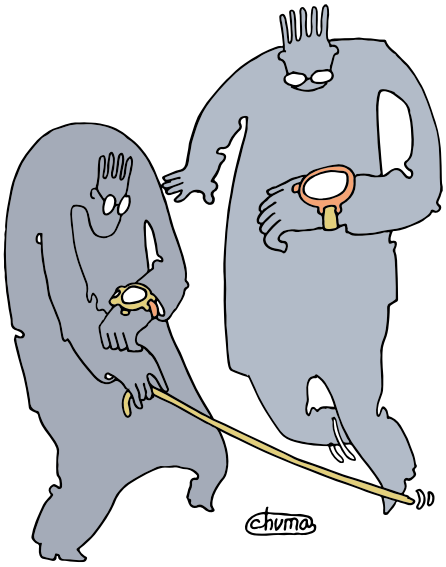
Слева на рис. 1 показана схема синхронного выполнения кода: сначала выполняется блок кода шаг 1, далее идет запись в БД и далее выполняется блок кода шаг 2. С асинхронным выполнением все немного иначе: после выполнения блока шаг 1 начинает выполняться запрос к БД, и далее исполняется блок кода шаг 2. В какой то момент от операционной системы приходит сигнал о завершении операций ввода/вывода, и происходит передача управления функции обратного вызова обработки результатов выполнения. В результате освободилась та часть времени, в которой ранее простаивал процессор.

Когда же рационально использовать асинхронность:

- когда есть дисковые операции ввода/вывода;
- когда есть сетевые операции ввода/вывода;

Рисунок 1. Синхронное и асинхронное выполнение кода





Асинхронность дает нам преимущества — выигрыш в производительности в некоторых ситуациях

- запросы к БД (повторяемся, так как это можно отнести к сетевым операциям ввода/вывода);
- отправление электронной почты, опять же коммуникация с почтовым сервером через сетевые операции ввода/вывода.

Файлы всех примеров можно скачать на сайте журнала http://samag.ru.

Асинхронность в MySQL

Наиболее частые сетевые операции в веб – это запросы к БД. Если у нас имеется несколько независимых запросов, т.е. результаты выполнения одного запроса не влияют на другой запрос, то для ускорения можно использовать асинхронность.

Пусть один запрос исполняется 3 секунды, а второй всего одну (см. рис. 2). Справа представлено синхронное выполнение двух запросов, слева – асинхронное. Как видно, за счет асинхронности мы можем выиграть время, равное разнице суммы времени выполнения запросов и самого длинного запроса. В нашем примере это будет 1 секунда.

Асинхронное выполнение запросов к MySQL появилось в PHP 5.3 в расширении mysqlnd.

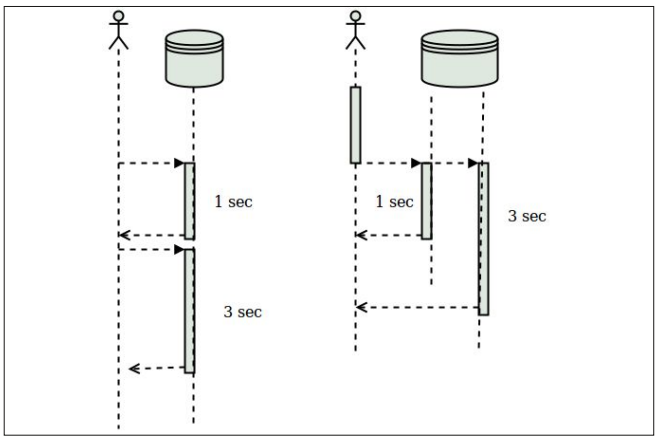
Проверим на практике код синхронного выполнения запросов (файл ex1.php).

```
<?php
$host = '127.0.0.1';
$port = 3306;
$user = '...';
$password = '...';
$mysqli = new mysqli($host, $user, $password, "temp");
/* check connection */
if ($mysqli->connect_errno) {
    printf("Connect failed: %s\n", $mysqli->connect_error);
    exit();
}
$start = time();
$res = $mysqli->query('SELECT sleep(3) as sleep');
if ($res == null) {
    echo $mysqli->error, PHP_EOL;
}
$row = $res->fetch_assoc();
$res = $mysqli->query('SELECT sleep(1) as sleep');
if ($res == null) {
    echo $mysqli->error, PHP_EOL;
}
$row = $res->fetch_assoc();
$mysqli->close();
echo 'time ', ( time() - $start ), ' sec'  , PHP_EOL;
```

Код выполнения асинхронных запросов (файл ex2.php).

```
<?php
$hosts = array_fill( 0, 2, '127.0.0.1');
$port = 3306;
$user = '...';
$password = '...';
$mysqli = [];
$queryes = [];
$sleep = [3, 1];
$sleep_value = current($sleep);
foreach( $hosts as $host) {
    $mysqli[] = $mysql_i = new mysqli($host, $user, ↵
        $password, "temp");
    /* check connection */
    if ($mysql_i->connect_errno) {
        printf("Connect failed: %s\n", ↵
                $mysql_i->connect_error);
```

Рисунок 2. Время синхронного и асинхронного выполнения кода



```
        exit();
    }
    // формируем запрос
    $queryes[] = "SELECT sleep($sleep_value) as sleep";
    $sleep_value =  next($sleep);
}
$start = time();
reset($queryes);
$query = current($queryes);
foreach ($mysqli as  $link) {
    // выполняем запрос
    $link->query($query, MYSQLI_ASYNC);
    $query = next($queryes);
}
// событийный цикл,
$processed = 0;
do {
    $links = $errors = $reject = [];
    foreach ($mysqli as $mysql) {
        $links[] = $errors[] = $reject[] = $mysql;
    }
    # опрашиваем все коннекции на наличие ответа
    if (!mysqli_poll($links, $errors, $reject, 60)) {
        continue;
    }
    foreach ($links as $k=>$link) {
        # получаем ответ из асинхронного запроса
        if ($result = $link->reap_async_query()) {
            $res = $result->fetch_row();
            # Handle returned result
            //echo 'get result link #',$k,PHP_EOL;
            //var_dump( $res);
            mysqli_free_result($result);
        } else die(sprintf("MySQLi Error: %s", ↵
                mysqli_error($link)));
        $processed++;
    }
} while ($processed < count($mysqli));
echo 'time ', ( time() - $start ), ' sec'  , PHP_EOL;
```

Результаты выполнения примеров ex1.php и ex2.php:

```
$ php ex1.php

time 4 sec

$ php ex2.php

time 3 sec
```

Когда выгодно использовать асинхронные запросы? Например, когда обращаемся к нескольким базам данных, находящимся на разных серверах. Обычно это используется при анализе данных, расположенных [2] на нескольких серверах. Также асинхронное выполнение запросов выгодно использовать при вставке или изменении данных: отправил запрос и забыл о нем... Однако предупреждаю, что нельзя использовать асинхронную вставку в одну таблицу в одной и той же базе данных, что неизбежно вызовет блокировку таблицы. Но параллельно можно вставлять разные данные в свои таблицы, например, заполнять таблицу users и users_info, или один и тот же тип данных разливать на разные серверы (шардинг). Очень удобно использовать при миграции данных с монолитного сервера на шардированные серверы.

Теперь рассмотрим на примере использования MySQl в Hack. Тут асинхронность в Hack встроена в сам язык. Использовать асинхронное расширение MySQL проще, чем стандартное mysqli. Основной класс для создания соединений: AsyncMysqlConnectionPool, который содержит пул соединений с БД и имеет основной асинхронный метод connect():

```
class AsyncMysqlClient {
        public static async function connect(
                string $host,
                int $port,
                string $dbname,
                string $user,
                string $password,
                int $timeout_micros = -1
        ): Awaitable<?AsyncMysqlConnection>;
}
```

Как мы видим, при объявлении метода connect() использовалось ключевое слово «async», которое сообщает HHVM, что метод асинхронный. Данный метод возвращает объект AsyncMysqlConnection, который будет ожидать завершения всех запущенных асинхронных операций. Знак вопроса в объявлении возвращаемого типа указывает, что может вернуться как объект типа AsyncMysqlConnection, так и null.

Асинхронный класс AsyncMysqlConnection реализует запросы к БД, используя два основных метода:

- **query()** – выполнение запроса;
- **queryf()** – форматирование запроса наподобие функции sprintf() и его исполнение.

При необходимости используется метод escapeString(), название которого говорит само за себя.

Ниже приведен пример выполнения асинхронного запроса:

```
<?hh
// объявляем асинхронную функцию
async function async_mysql() : Awaitable<void> {
    $pool = new \AsyncMysqlConnectionPool([]);
    $host = '127.0.0.1';
    $port = 3306;
    $db = 'temp';
    $user = 'akalend';
    $password = '12345';
    //осуществляем соединение
    $conn1 = await $pool->connect($host, $port, $db, $user, ↵
         $password);
    $conn2 = await $pool->connect($host, $port, $db, $user, ↵
         $password);

    // синхронизируем ответ от выполнения двух асинхронных
    // запросов
    $res_pool = await \HH\Asio\v([
        $conn1->query('SELECT sleep(1) as sleep1'),
        $conn2->query('SELECT sleep(3) as sleep2'),
    ]);

    // получение результата, данные выполнения в массиве
    // $res_pool
    var_dump($res_pool[0]->mapRows(), ↵
         $res_pool[1]->mapRows());
    $conn1->close();
    $conn2->close();
}

// запуск асинхронной функции и ожидание результата
$time = time();
\HH\Asio\join(async_mysql());
echo time()-$time, ' sec',PHP_EOL;
```

Результат исполнения, как и ожидалось, при асинхронном выполнении должен составлять 3 секунды (максимальное

время выполнения самого тяжелого запроса), а не 5 секунд (сумма времени исполнения каждого запроса):

```
object(HH\Vector)#16 (1) {
  [0]=>
  object(HH\Map)#17 (1) {
    ["sleep1"]=>
    string(1) "0"
  }
}

object(HH\Vector)#18 (1) {
  [0]=>
  object(HH\Map)#19 (1) {
    ["sleep2"]=>
    string(1) "0"
  }
}

3 sec
```

Если сравнить, то написание асинхронного кода на Hack намного проще, чем PHP.

Какие основные ключевые моменты в представленном выше коде? Каждая функция, которая должна выполняться асинхронно, должна быть объявлена с ключевым словом «async» и иметь возвращаемый тип Awaitable<тип>. Для получения асинхронных данных как результат исполнения асинхронной функции используется ключевое слово «await». Если мы хотим получить несколько результатов асинхронно запущенных функций, то необходимо использовать функцию синхронизации await HH\Asio\v (возвращает Vector) или await HH\Asio\m (возвращает Map):

```
// объявляем асинхронную функцию
async function f1(): Awaitable<int> {
        // ...тело функции
}
async function f2(): Awaitable<int> {
        // ...тело функции
}

// составляем массив ссылок на функции
$handles = [f1(), f2()];

// асинхронно выполняем обе функции и синхронизируем
// результат в массиве $result
$result = await HH\Asio\m($handles);
// используем полученный результат
var_dump($result['f1'], $result['f2']);
```

Какие могут быть еще интересные моменты при написании асинхронного кода?

```
// объявляем асинхронную функцию
async function f(): Awaitable<int> {
        // ...тело функции
}

async function main(): Awaitable<void> {
        $wait_handle = f();
        // далее может идти выполнение некоторого
        // синхронного кода

        // …
        // … тут происходит ожидание выполнения асинхронной
        // функции
        // … и получение результата целого типа: int
        $result = await $wait_handle;
}
```

Если мы в синхронном коде вызываем асинхронную функцию, то необходимо использовать функцию HH\Asio\join():

```
async function f(): Awaitable<mixed> {
    // … код асинхронной функции
}

// синхронный код
function main(): void {
     // выполнение асинхронной функции
     // и ожидание окончания выполнения
     $result = HH\Asio\join(f());
}
```

Что нужно помнить при написании асинхронного кода?

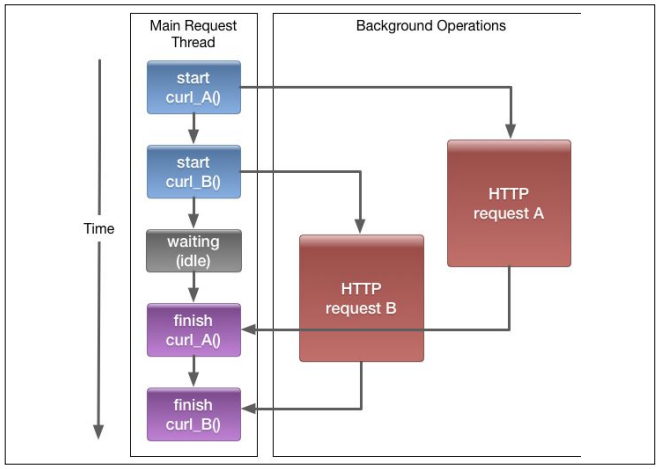
Выполнение всегда происходит в пределах одной нити (треда, thread), и надо понимать, что нет параллельности исполнения кода, а используется простой процессор в асинхронных операциях ввода/вывода. Поэтому использование блокирующих операций или sleep() всегда будет «тормозить» весь код и сведет на нет всю асинхронность. В асинхронном коде необходимо использовать асинхронный sleep \HH\Asio\sleep().

При написании асинхронного кода нельзя использовать никакие синхронные функции, которые имеют блокирующие операции. Все функции, вызываемые в асинхронном коде, должны быть объявлены как async. Если используются анонимные функции, например, в качестве функции обратного вызова, то они также должны быть объявлены как асинхронные:

```
function f((function(int): Awaitable<string>) $callback): ↵
         void {
         // ...
}
```

Все то же самое относится и к классам. Все синхронные методы должны быть объявлены с ключевым словом «async». Но тут есть немаловажный момент: если объявляется абстрактный метод, или метод интерфейса, то он должен быть объявлен как синхронный:

Рисунок 3. Схема выполнения асинхронных HTTP-запросов



```
interface I {
    // ошибка
    public async function bad(): Awaitable<void>;
    // объявляем как синхронный
    public function good(): Awaitable<void>;
}
abstract class C {
    // ошибка
    abstract public async function bad(): Awaitable<void>;
    // объявляем как синхронный
    abstract public function good(): Awaitable<void>;
}
```

Асинхронность в cUurl

Второй долгоиграющей операцией по частоте использования в веб является выполнение HTTP-запросов. Это может быть парсинг RSS, JSON, загрузка страниц XML-справочников товаров или просто запросы к стороннему API. Для этих целей используется PHP-расширение cUrl.

На рис. 3 представлена схема, взятая из официальной документации, выполнения асинхронных HTTP-запросов.

Рассмотрим пример. Пусть мы должны сделать запросы на разные долгоиграющие скрипты, которые эмулируем скриптом sleep.php, в строке запроса скрипта sleep.php передаем количество секунд задержки и это же число получаем в ответе:

```
<?php
$sleep = $_SERVER['QUERY_STRING'];
sleep($sleep);
echo $sleep, PHP_EOL;
```

Наш скрипт обработки асинхронных HTTP-запросов на PHP:

```
<?php
$ch1 = curl_init();
$ch2 = curl_init();
    // установка url на запрос со временем исполнения 2 сек
curl_setopt($ch1, CURLOPT_URL, ↵
        "http://127.0.0.1/sleep.php?2");

    // установка url на запрос со временем исполнения 3 сек
curl_setopt($ch2, CURLOPT_URL, ↵
        "http://127.0.0.1/sleep.php?3");

    // инициализация curl
$mh = curl_multi_init();
curl_multi_add_handle($mh,$ch1);
curl_multi_add_handle($mh,$ch2);
$time = time();
$active = null;
do {
    do {
    // выполнение запросов
        $mrc = curl_multi_exec($mh, $active);
    } while ($mrc == CURLM_CALL_MULTI_PERFORM);
    usleep(1000);
    // выполнение цикла по всем дескрипторам
} while (curl_multi_select($mh) === -1);

    // выполнение цикла пока не  придут данные
while ($active && $mrc == CURLM_OK) {
    if (curl_multi_select($mh) != -1) {
            // проверка дескриптора на наличие  данных
        do {
            $mrc = curl_multi_exec($mh, $active);
            //echo $mrc, PHP_EOL;
        } while ($mrc == CURLM_CALL_MULTI_PERFORM);
    }
}
```

```
echo 'time=', (time() - $time), "sec\n";
curl_multi_remove_handle($mh, $ch1);
curl_multi_remove_handle($mh, $ch2);
curl_multi_close($mh);
```

Как и ожидалось, для асинхронного выполнения результат этого примера – 3 секунды:

```
$ php ex3.php

2
3
time=3sec
```

Теперь аналогичный пример для Hack, где API cUrl представлено двумя основными функциями:

```
async function curl_multi_await(resource $mh, ↵
        float $timeout = 1.0) : Awaitable<int>;
namespace HH\Asio {
        async function curl_exec(mixed $urlOrHandle): ↵
                Awaitable<string>;
}
```

Если сравнить ниже на примере для Hack, то увидим, что код лаконичнее, красивее и проще:

```
<?hh
use HH\Asio;
    // асинхронная функция - обертка над curl
async function async_curl(int $sec) : Awaitable<string> {
    return await  HH\Asio\curl_exec("http://127.0.0.1/ ↵
        sleep.php?" . $sec );
}
async function example_async_curl() : Awaitable<void> {
    // асинхронный запуск необходимого кол-ва HTTP запросов
    $res1 = async_curl(1);
    $res2 = async_curl(3);

    // синхронизация ответов
    list($str1,$str2) = await  HH\Asio\v([$res1, $res2]);
    echo $str1,$str2, PHP_EOL;
}
$time = time();
\HH\Asio\join(example_async_curl());
echo time()-$time, ' sec',PHP_EOL;
```

McRouter – асинхронный memcached-клиент

В Hack есть асинхронное расширение для работы с memcached. Принципиально оно не отличается от PHP memcache. Для создания соединений используется либо конструктор, либо статический метод createSimple(). Для доступа к memcached используются асинхронные методы get, set, add, delete...

```
class MCRouter {
    public function __construct(array<string, mixed> ↵
        $options, string $pid = '');
    public static function createSimple(ConstVector<string> ↵
        $servers): MCRouter;
    public async function add(string $key, string $value, ↵
        int $flags = 0, int $expiration = 0): ↵
        Awaitable<void>;
    public async function get(string $key): ↵
        Awaitable<string>;
    public async function del(string $key): Awaitable<void>;
    public async function incr(string $key): Awaitable<int>;
    public async function decr(string $key): Awaitable<int>;
    ...
}
```

Конструктор имеет разное поведение, которое зависит от значения строки $pid (Persistent Identificator), если переменная $pid пустая, то конструктор возвращает трансистный (transient) объект, иначе осуществляется поиск среди уже созданных объектов по значению $pid.

Переменная $options содержит параметры, которые конфигурируют объект или ключ с именем 'config_str', значение которого указывает на имя конфигурационного файла.

Более простой путь получения memcached-клиента – это вызов статического метода MCRouter::createSimple() с единственным параметром – строка соединения, представленная как 'host:port', например '127.0.0.1:11211'.

Методы get, set, del, incr, decr, cas, add, append, prepend, replace и пр. соответствуют методам текстового протокола memcached [4]. Эти методы вызывают исключение, если ключ не существует и их можно вызывать через оберточную функцию HH\Asio\wrap().

Ниже приведен пример использования MCRouter:

```
<?hh
use HH\Asio;

// создание объекта MCRouter
function get_mcrouter_object(): \MCRouter {
    $servers = Vector { '127.0.0.1:11211' };
  $mc = MCRouter::createSimple($servers);
  return $mc;
}

// Добавление данных в memcached
async function add_user_name( \MCRouter $mcr, int $id, ↵
         string $value): Awaitable<void> {
    $key = 'name:' . $id;
    await $mcr->set($key, $value);
}

// Получение данных из memcached
async function get_user_name(\MCRouter $mcr, int $user_id): ↵
         Awaitable<string> {
    $key = 'name:' . $user_id;
    try {
        $res = await \HH\Asio\wrap($mcr->get($key));
        if ($res->isSucceeded()) {
            return $res->getResult();
        }
        return "";
    } catch (\MCRouterException $ex) {
        echo $ex->getKey() . PHP_EOL . $ex->getOp();
        return "";
    }
}

async function run(): Awaitable<void> {
    $mcr = get_mcrouter_object();
    await add_user_name($mcr, 1, "Вася Пупкин");
    $name = await get_user_name($mcr, 1);
  var_dump($name); // выведет "Вася Пупкин"
}

\HH\Asio\join(run());
```

Урок повторения, или Еще раз о синхронизации

Для большего понимания материала, хочу повторить некоторые основные моменты. Код, выполняемый асинхронно, должен содержаться в функциях, объявленных асинхронными async function, и возвращаемый результат должен быть Awitable<возвращаемый тип>. Основной код, который вызвал асинхронную функцию, может закончиться раньше, чем сама функция, и мы не увидим результата выполнения, поэтому необходимо дождаться выполнения асинхронной функции, присоединив ее результат к основному коду функцией HH\Asio\join();

Асинхронную функцию можно обернуть в HH\Asio\wrap(), результат которой реализует интерфейс HH\Asio\ResultOrExceptionWrapper:

- **public function isSucceeded(): bool;** – проверяет, нормально ли завершилась функция;
- **public function isFailed(): bool;** – проверяет, завершилась ли функция аварийно;
- **public function getResult(): T;** – возвращает результат или вызывает исключение;
- **public function getException():** – возвращает вызванное исключение.

Если необходимо засинхронизировать ответ нескольких асинхронных функций, то используются функции HH\Asio\v() – получение результата выполнения в виде вектора или HH\Asio\m() – результат в объекте Map. То же, но с оберткой Exceptions во wrap – аналог использования HH\Asio\wrap(), – для вектора HH\Asio\vw() и HH\Asio\mw() для Map. А также с применением маппинга (см. использование функции map() [1, 3]) к результату HH\Asio\vm() и HH\Asio\mm() и применением фильтров HH\Asio\vf() и HH\Asio\mf(). В официальной документации можно найти более подробный перечень функций синхронизации.

Для эмуляции задержек функций sleep/usleep необходимо использовать их асинхронные аналоги HH\Asio\sleep/usleep. Получить результат уже завершившейся асинхронной функции можно, используя функцию HH\Asio\result().

Если необходимо устроить циклы ожидания, то получить запущенные хандлеры можно функцией HH\asio_get_running() и проверить оконченные HH\Asio\has_finished().

Также возможно использовать асинхронные генераторы, но это выходит за рамки данной статьи, как и описание использования асинхронных потоков.

• • •

Как мы видим, использование асинхронности дает нам некоторые преимущества и в определенных ситуациях даже выигрыш в производительности. Использование асинхронных расширений MySQL, cUrl, MCRouter за счет встроенной асинхронности оказывается еще проще. Думаю, что с переходом новых проектов на Hack вы оцените все его фишки. EOF

[1] Календарев А. Введение в Hack. // «Системный администратор», №3, 2016 г. – С. 46-50 (http://samag.ru/archive/article/3150).

[2] Календарев А. Горизонтальное масштабирование. Проблемы и пути решения. // «Системный администратор», №10, 2014 г. – С. 54-62 (http://samag.ru/archive/article/2797).

[3] Описание асинхронности в официальной документации Hack – http://docs.hacklang.org/hack/async/introduction.

[4] Описание текстового протокола memcached – https://github.com/memcached/memcached/blob/master/doc/protocol.txt.

Ключевые слова: Hack, асинхронность, разработка.

Визитка

АЛЕКСАНДР МАЙОРОВ,
Tutu.ru, руководитель отдела Frontend-разработки, alexander@majorov.su

Шаблонные строки в ES6
Больше, чем строки

В новом стандарте ES2015 (ES6) добавлено много разнообразных улучшений. При этом некоторые фичи разработчики не используют на полную либо по причине дефицита фантазии, либо по незнанию всех аспектов и нюансов. Рассмотрим шаблонные строки и их расширение — тегированные шаблонные строки

Шаблонные строки ES6

Итак, что же такое шаблонные строки? Из названия уже может быть понятно, что это возможность создавать некие шаблоны в формате строки. Вспомним, как мы конкатенировали данные со строками раньше (и можем продолжать писать сейчас и в будущем, при желании):

```
var myTooLongString = "A long time ago, in a galaxy far," +
   "far away...." + someVariable +
   "It is a period of civil war";
```

Это было всегда не очень удобно, но терпимо. Разработчики создавали различные вспомогательные функции (хелперы) для работы со строками. Но потребность в создании некоторого шаблона, в который можно вставлять переменные и логику (вызов функции, условия), привела к появлению различных, более сложных библиотек для работы со строками и данными – шаблонизаторов [1-3].

Самый простой вариант шаблонизатора, назовем его Nano, может выглядеть следующим образом:

```
/**
 * @param tpl - шаблон
 * @param data - данные (не обязательно)
 * @param tags
 * @returns {XML|string|void}
 */
function template(tpl, data, tags){

   tags = tags || { open: '${', close: '}' };

   return tpl.replace(
      new RegExp(tags.open + "\\s*([\\w\\.]*)\\s*" + ↵
         tags.close, "g"),
      function(s, k){
         var k = k.split('.'),
            v = data[k.shift()],
            prop;

         for (prop in k) {
            if(k.hasOwnProperty(prop)) {
               v = v[k[i]];
            }
         }
         return (typeof v !== void 0 && v !== null) ? v : '';
      }
   );
}
```

Пользоваться таким шаблонизатором просто:

```
var data = {
   person: {
      name: 'Alexander',
      surname: 'Majorov'
   }
};

console.log(
   template('hello ${ person.name } ${ person.surname }', data)
);
```

Это простейший вариант. Здесь нельзя вызывать функции и добавлять логику в шаблон. Для этого нужно усложнять реализацию.

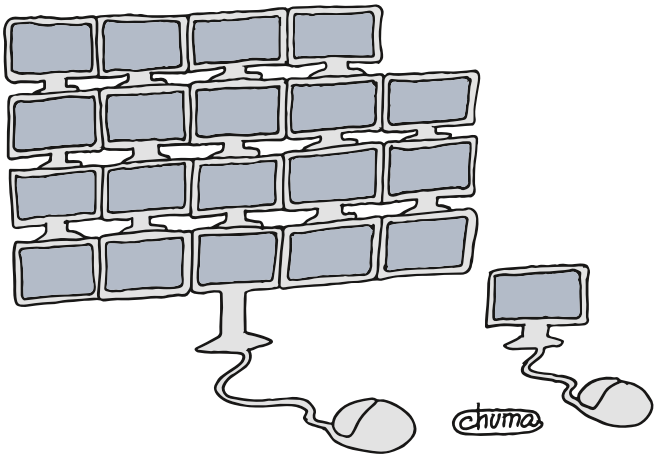
Также проблема существующих строк в JavaScript – невозможность их переносить и делать мультистроковые без оператора конкатенирования (как было показано выше). Точнее, можно и без оператора, но с использованием специальных эскейп-последовательностей (символов), таких как \n. Все эти недостатки были реализованы в новом синтаксисе шаблонных строк.

Шаблонные строки (шаблоны) являются строковыми литералами, допускающими использование выражений. Вы можете использовать многострочные литералы и возможности интерполяции.

А если говорить простым языком, то, используя литералы «`», можем теперь записывать строки в таком формате:

```
var data = {
   person: {
      name: 'Alexander',
      surname: 'Majorov'
   }
};

console.log(`hello ${ person.name } ${ person.surname }`);
```



Шаблонные строки (шаблоны) являются строковыми литералами, допускающими использование выражений

В многострочных блоках уже не нужны эскейп-символы (\n) и операторы конкатенирования:

```
var myTooLongString = `A long time ago, in a galaxy far
far away....
It is a period of civil war.
Rebel spaceships, striking
from a hidden base, have won
their first victory against
the evil Galactic Empire`;
```

Вроде бы все просто. Такой механизм уже используется в других языках, например в PHP. Но в ES2015(ES6) есть возможность создавать тегированные шаблонные строки. Что это такое?

Тегированные шаблонные строки

Это совершенно новый синтаксис, позволяющий описывать функции-обработчики для шаблонных строк. Как это выглядит:

```
function parse(strings, ...values) {
   return strings.raw[0];
}

let myStr = parse`Any string with any var = ${ myVar }`;
```

Как видите, это похоже на вызов функции, но нам не нужно обрамлять строку в скобки (как при вызове функции), и в функцию передаются различные аргументы автоматически.

Первый аргумент – это коллекция типа массив, в которой содержатся части строки. Строка разбивается разделителем ${}. Также в коллекции есть массив raw (в нашем случае доступ можно получить через вызов str.raw), который содержит в себе необработанную строку. Остальные аргументы – это значения, которые были вычислены внутри операторов ${}.

Кстати, чтобы извлечь необработанную (сырую) строку, не обязательно писать свою функцию. В объекте String уже добавлен метод raw(), который используется для получения необработанной шаблонной строки. Вот как это можно сделать:

```
String.raw`Hi\n${2+3}!`; //Выведет "Hi\\n5!"
```

Вроде бы интересная возможность, но, когда программисты сталкиваются с новым синтаксисом в языках программирования, первым делом происходит дефицит фантазии. А для чего вообще можно использовать этот синтаксис?

Можно придумать интересные, необычные примеры использования ради шутки или академического интереса. Например, мы можем реализовать что-то наподобие языка ассемблера прямо в JavaScript. Точнее, лишь пару инструкций, которые похожи. Следующий код совершенно валидный и рабочий, это код на ES6:

```
mov   `ax` `2`
mov   `dx` `3`
add   `ax` `dx`
print `ax`
```

Почему этот код работает? Мы используем синтаксис тегированных шаблонных строк, при этом вместо строки возвращаем функцию для обработки шаблонных строк. Выходит, что мы можем организовывать целые цепочки таких вызовов.

```
var mem = {};
function mov(s1) { return s2 => mem[s1[0]] = s2[0] }
function add(s1) { return s2 => mem[s1[0]] + mem[s2[0]] }
function print(s1) { console.log(mem[s1[0]]) }
```

Конечно, в жизни такое писать скорее всего не придется. Хотя, согласитесь, выглядит красиво. Но, если серьезно, что можно делать с тегированными шаблонными строками?

Примеры использования тегированных шаблонных строк

Вы можете описать свой вариант шаблонизатора, решающий ваши конкретные задачи. Вот пример универсальной функции для шаблонизации строк с передачей аргументов:

```
function template(strings, ...keys) {
   return (function(...values) {
      var dict = values[values.length - 1] || {};
      var result = [strings[0]];
      keys.forEach(function(key, i) {
         var value = Number.isInteger(key) ? ↵
                  values[key] : dict[key];
         result.push(value, strings[i + 1]);
      });
      return result.join('');
   });
}

template `${0}${1}${0}!` ('Y', 'A');  // return "YAY!"
// return "Hello World!"
template `${0} ${'foo'}!` ('Hello', {foo: 'World'});
```

В данном случае наша функция (тег) template возвращает функцию, которая принимает аргументы. Это аналогично вызову template(…)(…). Этот пример хорошо показывает возможности тегированных шаблонных строк. Но опять же все это мы могли бы реализовать и без этого синтаксиса. А что реально было бы интересно реализовать через такой функционал?

Функция логирования

Когда начнете пользоваться шаблонными строками, то в какой-то момент потребуется при отладке проверять значения переменных, приходящих в строку. Можно по старинке писать console.log() и выводить отдельно строки и отдельно переменные. Но что если мы реализуем такую функцию логирования? Использование будет выглядеть следующим образом:

```
function log(strings, ...values) {

   let args = [], res = [], n = values.length;

   for (let i = 0; i < n; i++) {
      args.push(strings[i], '${ ' + values[i] + ' }');
      res.push(strings[i], values[i]);
   }

   args.push(strings[n]);
   res.push(strings[n]);

   console.log(...args);

   return res.join('');
}

const week  =  7;
const month = 31;

let myStr = log `A week has ${week} days. ↵
         A month has ${month} days.`;

// in console log: A week has  ${ 7 }  days.
// A month has  ${ 31 }  days.
// myStr = A week has 7 days. A month has 31 days.
```

Вместо того чтобы отдельно вызывать console.log() для myStr, где будет уже результирующая строка, мы всего лишь дописали маленькую строчку перед присвоением шаблонной строки со значениями, т.е. тегировали ее.

Развивая тему логирования, мы могли бы сделать улучшенную версию console.log() для форматированного вывода значений. Например, так:

```
function vars(templateStrings, ...substitutions) {
   let result = templateStrings[0];
   for (let [i, obj] of substitutions.entries()) {
      let propKeys = Object.keys(obj);
      for (let [j, propKey] of propKeys.entries()) {
         if (j > 0) {
            result += ', ';
         }
         result += propKey+'='+obj[propKey];
      }
      result += templateStrings[i+1];
   }
   return result;
}

let foo = 123;
let bar = 'abc';
let baz = true;

console.log(vars`Variables: ${{foo, bar, baz}}`);
// Output:
// Variables: foo=123, bar=abc, baz=true
```

Тег vars для шаблонной строки распечатывает нам не просто значения переменных, но и пишет имена этих переменных.

Для чего еще можно использовать этот синтаксис, кроме логирования, может спросить читатель. На самом деле фантазия может увести далеко. К примеру, можно реализовать функции для локализации. Представьте, что вы могли бы перевести строку на другой язык таким образом:

```
let myString = ru `Hello world!`;
// Привет мир!
```

Или вы можете реализовать функцию i18n. Кстати, такое уже есть, подробности можно почитать по [4].

Функции-хелперы для работы с Url – вполне себе интересный юзкейс. Можно описывать URL в вашем фреймворке в формате:

```
let myString = url `/hello/world`;
// http://majorov.su/hello/world.html
```

При этом функция-тег сама добавит доменное имя и расширение, сделает проверки или какие-то дополнительные действия над URL. К примеру, обработает его через encodeURIComponent.

Если пофантазировать, то можно придумать с десяток интересных применений. Главное – знать, что такой механизм есть и что он уже работает нативно в некоторых браузерах. Но если вы пользуетесь TypeScript или Babeljs, то ваш код будет работать в любых браузерах. И все это вы можете уже использовать сейчас в вашем любимом JavaScript.

[1] Getting Literal With ES6 Template Strings – https://developers.google.com/web/updates/2015/01/ES6-Template-Strings.
[2] Template literals – https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Template_literals.
[3] Строки шаблона (JavaScript) – https://msdn.microsoft.com/ru-ru/library/dn858580(v=vs.94).aspx.
[4] i18n with tagged template strings in ECMAScript 6 – http://jaysoo.ca/2014/03/20/i18n-with-es6-template-strings.

Ключевые слова: TypeScript, веб-разработка, JavaScript, программирование, template strings.

Визитка

ВЛАДИМИР ЗАКЛЯКОВ,
советник налоговой службы 2-го ранга

Лабораторная работа
Исследуем сокеты. Часть 1

В работе исследуется взаимодействие процессов в ОС Linux через интернет-сокеты, созданные на базе протоколов TCP и UDP и работающие поверх протокола IP версии 4

Крик души (введение)

Первая публикация о сокетах в «СА» была в первом номере в 2002 году, где Всеволод Стахов описал программирование сокетов на Си [1]. С того момента и до сегодняшнего номера публикаций по этой теме не было. Почему не было публикаций? Ничего не изменилось? Нет, всё же небольшие изменения по коду есть. Сокеты не востребованы? Востребованы! Все умеют программировать сокеты, и это никому не нужно? Тоже вряд ли. Тема в той публикации была раскрыта полностью, писать больше нечего? Скорее да, чем нет. Но, может, что-то осталось за кадром? Несомненно, да, за кадром остались вопросы практического применения и получения навыков работы с сокетами. Одно дело прочитать статью и «знать» о сокетах, другое – «уметь» выполнить предложенные по теме сокетов задания и совсем третье – «владеть» технологией использования сокетов так, чтобы использовать их на практике, самостоятельно исправлять возникающие ошибки. Цель данной работы – сократить пропасть между программистами и сетевыми администраторами, поскольку первые могут ничего не знать о тонкостях сетевых технологий и настройки операционных систем, обращаясь с сокетами, как с некоторой программной абстракцией, а вторые могут использовать снифферы, тонко настраивать всевозможные программы, пакетные фильтры, сетевые интерфейсы и аппаратные компоненты, но совсем не дружить с компилированием программ, структурами, шаблонами и т.п.

Терминология

В английском языке слово «socket» может употребляться как в обычной жизни, так и во многих отраслях науки: биологии, медицине, механике, электронике и компьютерных науках.

В русском языке термин «сокет» имеет более узкое смысловое наполнение, поскольку для большинства случаев у нас есть свои аналоги. Однако аналоги не успели появиться либо были вытеснены в таких быстро развившихся областях науки и техники, как электроника, вычислительная техника (компьютеры), компьютерные сети и интернет, где у сокета остаётся несколько смысловых значений, поэтому его употребление происходит либо в контексте, откуда становится ясно его смысловое наполнение, либо этот термин является составной частью другого многословного термина. Так, в литературе можно встретить без перевода: Berkeley sockets, Internet sockets, Network socket, Unix sockets, Unix domain sockets, raw socket, IC socket, CPU socket, ZIF Socket и др., а также их различные переводные и транслитерированные аналоги.

Таким образом, в отечественной литературе преимущественно сокет – это:

- либо соединительный разъём или панелька (панель) для установки микросхемы, в том числе разнообразные детальки исключительно для конкретной микросхемы процессора;
- либо название программного интерфейса для обеспечения обмена данными между процессами. Процессы при таком обмене могут исполняться как на одной ЭВМ, так и на различных, связанных между собой сетью.

В рамках данной работы будем использовать термин «сокет» для обозначения абстрактного объекта, представляющего конечную точку, используемую программой на языке Си, для соединения двух различных процессов поверх протокола IP версии 4. Или, по сути, сокет – это то, через что две различные программы могут взаимодействовать друг с другом, обмениваться информацией.

Цели работы

- Показать, что, используя метод декомпозиции (т.е. решения одной большой задачи путем её разделения и решения серии меньших задач), возможно организовать силами начинающих программистов и/или системных администраторов относительно сложные процессы информационного обмена между двумя процессами в ОС Linux.
- Показать, что при использовании абстракции сокет возможно понизить требования к знаниям программиста для решения задачи информационного обмена между процессами.

- Закрепить теоретические знания и дать (выполняющим работу) практические навыки использования сокетов.
- Усовершенствовать навыки работы через интерфейс командной строки (CLI).
- Немного попрактиковаться в компилировании небольших прикладных программ.

Описание лабораторного стенда

Для выполнения работы используется ОС Linux CentOS 6.7. Данная версия является последней на момент подготовки материалов в печать, для неё имеются 32- и 64-битные сборки. Поскольку понятие «сокета» изначально абстрактное, то тексты программ и задания разработаны таким образом, чтобы была возможность расширить (с минимальными изменениями или совсем без них) как перечень исследуемых вопросов, так и набор используемых для этого программных и аппаратных компонентов.

Для выполнения работы возможны различные сценарии (в зависимости от варианта установки), в каждом из которых указаны минимальные требования к организации лабораторного стенда, определены задания для последовательного самостоятельного выполнения.

Поскольку для выполнения некоторых заданий необходимо наличие прав root, самое простое решение – использовать виртуализацию, например, на основеVirtualBox (поскольку он кроссплатформенный, свободный и имеет интуитивно-понятный графический интерфейс), что позволит иметь полный доступ к возможностям ОС. С другой стороны, существует более сложный, но имеющий право на жизнь вариант, – пользователи остаются с обычными правами, но запуск нужных им команд разрешён через прописывание в /etc/sudoers всех необходимых комбинаций запуска программ через sudo.

В этом случае следует помнить, что:

- для iptables придётся прописать разрешения для просмотра статистики с различными комбинациями ключей, а также для добавления и удаления правил без действия. Потенциальная опасность: пользователи увидят текущие настройки системы фильтрации и значения счётчиков не у своих правил;
- для nc придётся прописать различные разрешающие шаблоны для создания серверных сокетов TCP и UDP для диапазона привилегированных портов (0-1023). Потенциальная опасность: пользователи могут «занять» все свободные привилегированные порты и тем самым осложнить возможный запуск других программ на этих портах;
- проблема запуска веб-сервера не актуальна, если он всегда запущен;
- для запуска снифферов придётся прописать различные комбинации в /etc/sudoers. Потенциальная опасность – получение пользователями доступа не к своему трафику, а также в случае использования ключа -w проблемы переполнения диска.

Примеры таких записей в файле /etc/sudoers:

```
user  ALL=(ALL)  NOPASSWD: /usr/sbin/tcpdump -i lo -n -nn -X
user  ALL=(ALL)  NOPASSWD: /usr/sbin/tshark -i lo -x -V
user  ALL=(ALL)  NOPASSWD: /sbin/iptables -L
user  ALL=(ALL)  NOPASSWD: /sbin/iptables -L -v -x -n
user  ALL=(ALL)  NOPASSWD: /sbin/iptables -L -v -x -n ↵
        --line-numbers
user  ALL=(ALL)  NOPASSWD: /sbin/iptables -L INPUT ↵
         -v -x -n --line-numbers
…
user  ALL=(ALL)  NOPASSWD: /usr/bin/nc -l ?
user  ALL=(ALL)  NOPASSWD: /usr/bin/nc -l ??
user  ALL=(ALL)  NOPASSWD: /usr/bin/nc -l ???
user  ALL=(ALL)  NOPASSWD: /usr/bin/nc -l ????
user  ALL=(ALL)  NOPASSWD: /usr/bin/nc -u -l ????
```

Какой путь выбрать – решать вам.

Варианты установок:

Вариант 1. «Минимум». Минимальная установка ОС (в неё уже включены программы netstat, ss, ps, iptables) + дополнительная установка программ: Apache + GCC + lsof + nc + tcpdump + telnet + tshark + wget.

Рисунок 1. Функции сокетов для элементарного клиент-серверного TCP-соединения

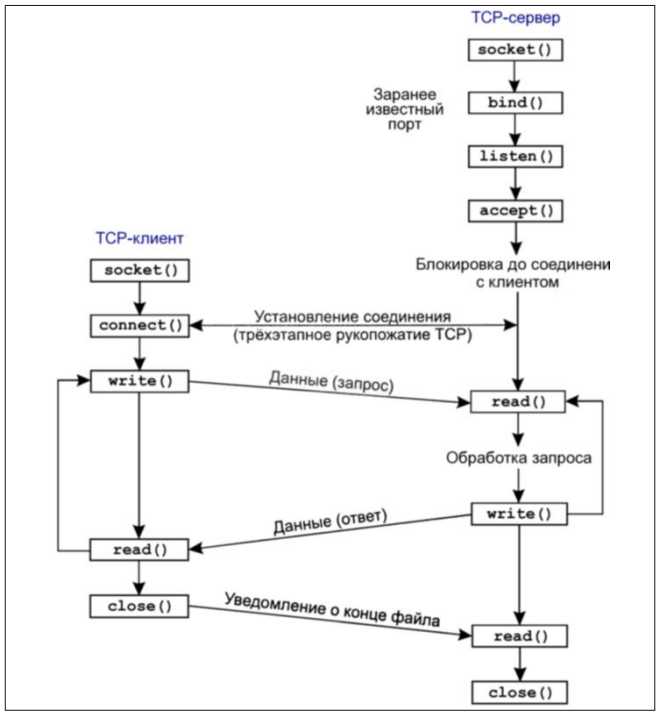
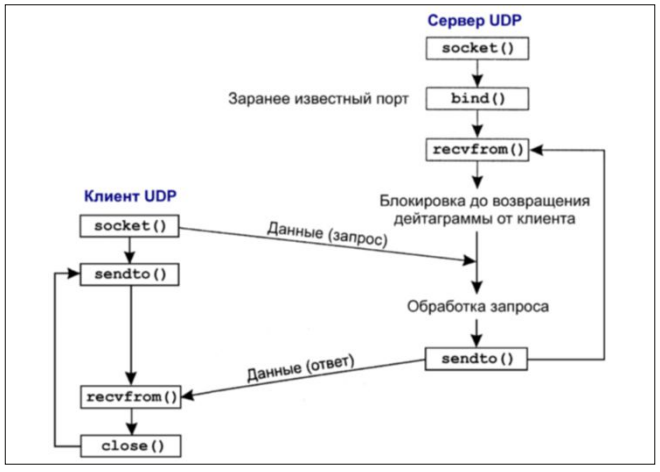
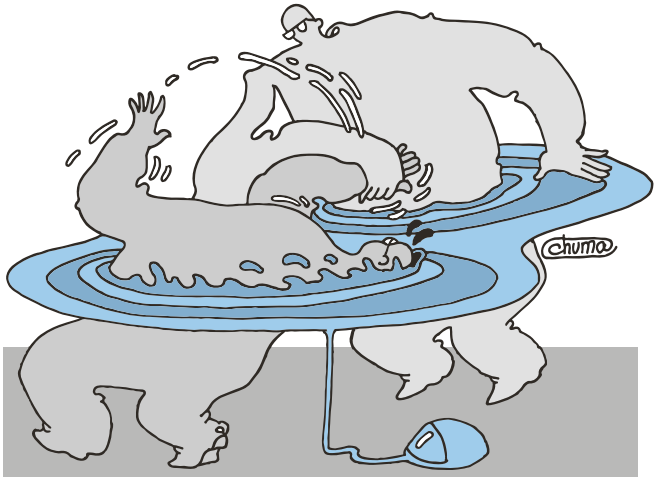


Рисунок 2. Функции сокетов для элементарного клиент-серверного UDP-соединения





Цель данной работы — сократить пропасть между программистами и сетевыми администраторами

Консольный интерфейс. Исследование взаимодействия двух процессов производится через интерфейс lo. Подключение во внешнюю сеть отсутствует.

Вариант 2. «Графический интерфейс». Установка системы с графической средой в варианте «Desktop» + дополнительные программы как в Варианте 1 + дополнительно сниффер Wireshark (пакет wireshark-gnome).

Сценарий проведения работы аналогичен Варианту 1, за тем лишь исключением, что дополнительно появляется возможность использования двух графических программ: браузер Firefox и Wireshark.

Вариант 3. «Сетевой интерактив». Установка программ производится как в Варианте 2 (или 1), но с поддержкой сети. То есть работа проводится не одним человеком в одной операционной системе, а несколькими, например, в компьютерном классе. Появляется возможность создания сокетных соединений между физически разными компьютерами.

При наличии в локальной сети веб-серверов или прямого доступа к сети Интернет возможно расширение экспериментов заданиями по подключению к реальным веб-серверам.

Несомненно то, что можно смоделировать работу любой сети и все задания выполнять с одного рабочего места на двух виртуальных машинах в пределах одного физического компьютера, однако остаётся больше впечатлений, если компьютер с сокет-клиентом и компьютер с сокет-сервером оказываются физически разными на небольшом удалении друг от друга, а сеть между ними – реальной.

Таким образом, оптимальным вариантом для проведения лабораторной работы можно считать использование нескольких физических компьютеров, объединённых в сеть, на каждом из которых в независимости от используемой ОС установлен VirtualBox, внутри которого произведена установки одной виртуальная машины с Linux согласно Варианту 3. Для обеспечения сетевого обмена между всеми виртуальными машинами администратором оговорены правила использования локального IP-адресного пространства, а виртуальные машины подключены к реальным сетевым интерфейсам хостовых машин в режиме «Сетевой мост».

Подготовка лабораторного стенда

Вариант 1. По завершении минимальной установки запустите систему и добавьте учетную запись пользователя user с паролем user (там, где это возможно, старайтесь приучать себя работать с минимальными привилегиями, то есть с правами обычного пользователя):

```
# adduser user
# passwd user
```

Замечание. Подробнее, как определить источник получения пакетов и настроить сеть (присвоить IP-адрес сетевому интерфейсу, прописать маршруты и т.п.), см. [1].

Установите пакеты согласно выбранному варианту, добавьте компилятор g++ (пакет gcc-c++) и файловый менеджер mc. В работе они не используются, но пригодятся в дальнейшем.

```
# yum install mc httpd gcc gcc-c++ lsof lynx nc ↵
        tcpdump telnet wireshark wget
```

Вариант 2. Доустановите пакеты как для Варианта 1 + wireshark-gnome.

Вариант 3. Создайте в VirtualBox новую виртуальную машину и выполните действия для Варианта 2. Выключите машину. Клонируйте виртуальную машину. Для этого используйте копирование внутри менеджера виртуальных машин, поскольку в этом случае будет произведена замена идентификаторов дисков и MAC-адресов сетевых интерфейсов, что позволит избежать проблем взаимодействия двух виртуальных машин между собой в будущем.

Перенесите клонированную виртуальную машину на другой компьютер. Убедитесь, что в настройках сетевых интерфейсов у виртуальных машин тип подключения указан как «Сетевой мост», а MAC-адреса у всех используемых копий виртуальных машин разные.

Определитесь с используемыми в сети сетевыми адресами (либо возьмите сеть 192.168.0.0/24, либо уточните у администратора). Далее полагаем, что у первой виртуальной машины адрес 192.168.0.2, у соседней – 192.168.0.3 и т.д.

Запустите нужное число виртуальных машин, настройте их сетевые интерфейсы и маршрутизацию (см. [1]). Отключите средства фильтрации установлением разрешительной политики и удалением всех имеющихся правил. (После установки ОС в ней по умолчанию запрещены входящие соединения, что делает невозможным подключение к прослушивающим сокетам (которые будут создаваться в процессе работы) извне системы.)

```
# iptables -P INPUT ACCEPT
# iptables -P OUTPUT ACCEPT
# iptables -F
```

Запустите веб-сервер.

```
# service httpd start
```

Подготовка лабораторного стенда завершена.

Краткие теоретические сведения

Предполагается, что, приступая к выполнению лабораторной работы, лица, планирующие её выполнить, уже имеют достаточный багаж теоретических знаний о сокетах. Если вы таковыми не являетесь либо хотите освежить в памяти теорию, то см. [1-3]. Общая схема, по которой производится сокетное соединение для протоколов TCP (тип сокета SOCK_STREAM) и UDP (тип сокета SOCK_DGRAM), представлена на рис. 1 и 2.

Описание хода проведения работы и отдельных компонентов

Для исследования межпроцессного информационного обмена (обмена данными между двумя процессами) в работе предлагается произвести связь процесса клиента и процесса сервера, используя сокеты в режиме клиента и сервера соответственно. Для этого в распоряжении обучаемых имеется (см. таблицу 1) шесть вариантов программ, использующих сокет в режиме клиента, три варианта программ, открывающих сокет на прослушивание на стороне сервера, и семь различных инструментов для исследования тех и иных аспектов, связанных с работой указанных программ и передачей данных. Хотя общее число комбинаций записей в таблице 6×3×8 = 144 и кажется большим, число принципиально исследуемых моментов на порядок, а то и два меньше. То, что в ряде комбинаций наблюдаемые результаты будут не так сильно отличаться друг от друга, должно быть очевидным после ознакомления с теорией, а на проверку каждого такого случая потребуется не более 5-10 секунд.

Таблица 1. Схема исследований

Клиент	Инструмент исследования	Сервер
telnet nc* программа на Си* браузер Firefox браузер lynx wget	tcpdump tshark Wireshark netstat ss lsof iptables ps	nc* программа на Си* веб-сервер Apache

* Все приведённые в таблице клиент-серверные средства поддерживают TCP-сокеты, а поддерживать протокол UDP умеют лишь nc и программа на Си

При наличии технической возможности и достаточного опыта у обучаемых число используемых программ и инструментов, как и число вариантов использования их друг по отношению к другу, в процессе исследования можно самостоятельно увеличить, а интересные наблюдения отразить в отчёте. Ниже даётся краткое описание всех компонентов из таблицы 1, типичные способы их использования, примеры запуска и ожидаемые результаты.

telnet

Изначально протокол telnet (уровень приложения (прикладной), RFC 854) использовался для предоставления двухстороннего доступа терминальным устройствам к серверу. Таким образом, пользователь мог через терминал взаимодействовать с различными процессами на сервере, в том числе и с командным интерпретатором. На транспортном уровне он использует протокол TCP. В протоколе telnet все данные передаются по сети в открытом виде и могут быть легко перехвачены. По этой причине сегодня для удалённого управления протокол telnet практически не используется (его заменил SSH), но для данной работы он прекрасно подходит. Программная реализация консольного клиента называется также telnet. Этой программой нам и предстоит воспользоваться в процессе работы.

Например, подключение к серверу 127.0.0.1 на порт 1234 протокола TCP:

```
$ telnet 127.0.0.1 1234
```

то же самое, но на порт 80 протокола TCP:

```
$ telnet 127.0.0.1 80
```

Замечание 1. Аналогичная программа в ОС Windows называется «telnet.exe». Если в Windows XP она была установлена по умолчанию, то начиная с Windows Vista (в том числе и на серверных линейках) её убрали. Установить её можно следующим образом: «Панель управления → Программы и компоненты → Включение или отключение компонентов Windows → Клиент Telnet» – поставить галочку. Либо можно произвести установку через консоль командой:

```
pkgmgr /iu:"TelnetClient"
```

Замечание 2. Вместо клиента telnet сегодня довольно часто используется бесплатная графическая утилита PuTTY, имеющая не только ряд дополнительных полезных свойств (например, поддержка протокола SSH, работа с различными кодировками), но и реализацию как под Windows, так и под UNIX и Linux [4].

Замечание 3. Программа telnet в ОС Linux не использует пользовательский сигнал SIGINT, возникающий при нажатии

клавиш <CTRL> + <C>, для выхода из программы. Чтобы выйти из сеанса работы telnet-клиента в случае зависания сервера, нажмите <ctrl> + <]>, после чего у вас появится приглашение «telnet>», где наберите quit для выхода из программы.

nc

nc – программа для создания соединений с использованием сокетов протоколов TCP и UDP и последующей передачей данных по ним [5]. Она позволяет как осуществлять клиентские подключения с использованием указанных протоколов, так и создавать на серверной стороне сокеты, находящиеся в режиме ожидания входящих соединений от клиентов.

Примеры использования nc в режиме «сервера»:

Создание прослушивающего сокета протокола TCP на порту 1234 для всех имеющихся у хоста IP-адресов.

```
$ nc -l 1234
```

Заметим, что для обычного пользователя доступны для прослушивания (для создания входящих соединений) лишь непривилегированные номера портов, то есть с номерами из диапазона 1024-65535. Для портов с номерами 1023 и меньше потребуются права суперпользователя.

```
# nc -l 1023
```

По умолчанию программа nc использует протокол TCP. Для использования протокола UDP необходимо дополнительно указывать ключ -u. Например, открытие на прослушивание порта 1234 протокола UDP:

```
$ nc -l 1234 -u
```

Выдавать содержимое файла 1.txt всем присоединившимся к сокету на 1080-м порту протокола TCP:

```
$ nc -l 1080<1.txt
```

Совместно с циклом на bash возможно создание временной заглушки для замены веб-сервера на 80-м порту:

```
# while true; do nc -l 80<index.html; done
```

Работает это следующим образом. При обращении к сокету HTTP-запросы клиента игнорируются, и сразу начинается передача данных из файла index.html клиенту. Большинство браузеров при получении такого потока данных игнорируют отсутствие заголовков от сервера и выводят полученное содержимое пользователю. Имя файла (index.html в нашем случае) может быть любым, поскольку клиент его не видит.

Примеры использования nc в режиме «клиента»:

Подключение к хосту с именем samag.ru на порт 80 по протоколу TCP:

```
$ nc samag.ru 80
```

Подключение к хосту с IP-адресом 192.168.3.1 на порт 53 по протоколу UDP:

```
$ nc -u 192.168.3.1 53
```

В man к nc можно найти много других полезных примеров по использованию.

Обратите внимание, что программа nc в режиме клиента в случае отсутствия возможности установления соединения по-разному себя ведёт для протоколов TCP и UDP.

Поскольку почтовые и многие другие сетевые службы используют текстовый режим работы с клиентом (либо всегда, либо только вначале соединения), то nc и telnet могут использоваться администраторами в целях «ручной» диагностики работы сетевых сервисов.

Замечание. Обратите внимание, что существуют и другие, аналогичные nc (netcat), программы: ncat, socat (Multipurpose relay (SOcket CAT) . Первая была специально написана для проекта Nmap. Она имеет несколько дополнительных возможностей, которые не потребуются в данной работе, но их, возможно, стоит знать, например поддержка прокси, работа с SSL:

```
$ ncat -l -p 6500 --ssl --ssl-cert /etc/ssl/host.crt ↵
        --ssl-key /etc/ssl/host.key > out.tgz
$ tar -zc ~ | ncat --ssl machineb 6500
```

Вторая программа умеет делать множественные перенаправления сокетов.

Браузер Firefox

Популярный кроссплатформенный браузер с GUI-интерфейсом. Удобен тем, что через обращение к странице about:config позволяет редактировать многие параметры своей работы. Также имеется возможность подключения большого числа расширений (файлы с расширением .xpi), добавляющих различную полезную функциональность. При необходимости возможна его замена на любой другой браузер. Для обращения к какому-либо серверу в адресной строке браузера следует указать его адрес и нажать «Enter». По умолчанию используется протокол TCP, и обращения направляются на порт 80. В случае необходимости использования отличного порта, например 1180, его следует указать через знак «:» в адресной строке. Например, http://192.168.1.1:1180.

Продолжение – в следующем номере.

[1] Закляков В. Looking Glass своими руками, или Сервер диагностики сетевой доступности. // «Системный администратор», №1-2, 2016 г. – С. 10-15 (http://samag.ru/archive/article/3110).

[2] Стахов В. Программирование сокетов. // «Системный администратор», №1, 2002 г. – С. 78-82 (http://samag.ru/archive/article/33).

[3] UNIX: разработка сетевых приложений /У. Стивенс. – СПб.: «Питер», 2003. – 1088 с. ISBN 5-318-00535-7.

[4] PuTTY: a free SSH and Telnet client – http://www.chiark.greenend.org.uk/~sgtatham/putty.

[5] man для программы nc (русский язык) – http://bsdadmin.ru/index.php/mans-freebsd/295-nc-netcat.

[6] Сайт Mozilla – http://www.mozilla.org.

Ключевые слова: сокет, сетевые утилиты, CentOS.

Ирина Попова:
«ИТ-инфраструктура ИТМО – один из инструментов развития университета»

В гостях у «Системного администратора» — начальник Департамента информационных технологий Санкт-Петербургского национального исследовательского университета информационных технологий, механики и оптики (ИТМО) Ирина Попова

Подготовила Ирина Ложкина

Ирина Попова, начальник Департамента информационных технологий Санкт-Петербургского национального исследовательского университета информационных технологий, механики и оптики (ИТМО)

О Департаменте информационных технологий

Департамент информационных технологий Университета ИТМО (ДИТ) занимается проектированием и внедрением передовых информационных технологий в систему управления и администрирования, совершенствование сетевых ресурсов и разработку наиболее эргономичных и эффективных моделей организации для позиционирования ИТМО и взаимодействия в сети Интернет.

Университет ИТМО: информационные технологии – ключ к успеху

Расхожий афоризм, приписываемый то Натану Ротшильду, то Уинстону Черчиллю – «кто владеет информацией, тот владеет миром» – актуален был всегда, а со стремительным развитием ИТ- и менеджмент-технологий приобрел глобальное значение.

Информационная инфраструктура, информационная система управления, представленность в интернет-пространстве становятся сегодня инструментами стратегического развития организации. Они выполняют важную роль в процессе интеграции в мировое информационное поле, способствуют созданию корпоративной (проектной, персональной) информационной среды, обеспечивая тем самым условия для стабильного функционирования и динамического развития любой структуры.

Мы не раз задавали себе вопрос, почему Университет ИТМО в своей работе реализует именно такой подход к пониманию ИТ-инфраструктуры не только как средства решения текущих операционных задач (хотя это, несомненно, очень важно, поскольку обеспечивает «устойчивое сегодня»), но прежде всего как одного из инструментов развития университета. Ответ заключается в специфике нашей деятельности.

Университет ИТМО – ИТ-вуз, и это проявляется на всех уровнях: от направленности образовательных программ до организации системы администрирования. Своим студентам мы предлагаем:

- емкий перечень специальностей для получения ИТ-компетенций,
- высокий уровень академической мобильности (возможность получить дополнительные знания и навыки в зарубежных вузах-партнерах),
- активное взаимодействие с ИТ-компаниями (получение практических компетенций в конкретных областях и трудоустройство в профильных организациях уже с младших курсов),

> поддержку научно-исследовательской деятельности учащихся (участие в проектах международных научных лабораторий).

В университете реализована эффективно организованная и постоянно развивающаяся информационная инфраструктура, обеспеченная в первую очередь собственными ИТ-разработками.

Мы гордимся нами созданной современной системой управления, построенной на сетевых принципах и позволяющей студентам и сотрудникам быть активными участниками жизни вуза.

Это стало возможным благодаря широкому использованию современных корпоративных и коммуникационных ИТ-решений, а также команде высококлассных специалистов, многие из которых – студенты университета.

Университет ИТМО имеет долгую историю, связанную с разработкой информационных систем управления для образовательных учреждений.

В 1999 году в Центрально-Европейском университете Будапешта состоялся семинар «Информационные системы в управлении вузом», где обсуждались идеи интегрированных решений и специфика их применения в высших учебных заведениях. Участники семинара признали необходимость разработки Интегрированной информационно-аналитической системы управления вузом (ИИАС).

И уже в ноябре в Санкт-Петербурге был создан консорциум из одиннадцати университетов, руководителем которого стал ректор Университета ИТМО В.Н. Васильев. Главной задачей организации стала разработка инновационного продукта – ИИАС. В результате данного проекта появились идеи, нашедшие отражение в вузовских системах управления, в том числе и у нас.

В 2010 году перед Департаментом информационных технологий были поставлены задачи кардинального реформирования подхода к организации информационной инфраструктуры университета. И связано это было прежде всего с серьезными изменениями в стратегии развития вуза и меняющейся практикой управления.

На сегодняшний день информационная инфраструктура Университета ИТМО является комплексом деловых и ИТ-решений, реализованных с использованием современных технологий.

Что у нас есть:

> **Информационная система управления (ИСУ)** – интегрированное решение для поддержки основных направлений деятельности вуза. С приложениями работают все службы университета, инструменты обеспечивают непрерывность бизнес-процессов планирования, учета, анализа и оценки деятельности университета. Интранет-приложение «Монитор руководителя» предоставляет руководству вуза интегрированную информацию для поддержки принятия решений.

Справка

В 2011 году Университет ИТМО и «Сколково» создали Ассоциацию предпринимательских университетов России, а в 2012-м совместно со Сколтехом инициировали создание Международной ассоциации центров внедрения технологий (IPOCA). Университет – венчурный партнер ОАО «РВК» с 2013 года. С 2011-го по настоящее время Университет ИТМО выступает хабом (организационно-методическим центром) по развитию инноваций и предпринимательства в регионах России в рамках американо-российской программы повышения исследовательского и предпринимательского потенциала вузов РФ «ЭВРИКА».

Справка

Университет ИТМО (Санкт-Петербург) – национальный исследовательский университет, ведущий вуз России в области информационных и фотонных технологий. Alma-mater победителей международных соревнований по программированию: ACM ICPC (единственный в мире шестикратный чемпион), Google Code Jam, Facebook Hacker Cup, Яндекс.Алгоритм, Russian Code Cup, Topcoder Open и др. Приоритетные направления: ИТ, фотонные технологии, робототехника, квантовые коммуникации, трансляционная медицина, урбанистика, Art&Science, Science Communication. С 2013 года участник проекта «5 в 100».

В составе Университета ИТМО более 20 институтов и факультетов, где обучаются около 12 тысяч студентов и аспирантов, работают 1200 преподавателей и научных сотрудников (из них около 700 — доктора и кандидаты наук)

> **Система интранет-решений** – инструмент развития корпоративной культуры университета. Количество пользователей интранет-портала достигло 20 тысяч, ежедневно несколько тысяч сотрудников и учащихся активно используют его сервисы и ресурсы: коммуникационные инструменты, систему электронных заявок, инструменты для размещения новостей, деловых регламентов и другие.

> **Система интернет-решений** – комплекс ресурсов и ИТ-решений (более 150), обеспечивающих активное представительство и продвижение университета в глобальном информационном пространстве. В международном рейтинге Webometrics ИТМО занимает 996-ю позицию, среди университетов Российской Федерации – 7-ю; в международном рейтинге репозиториев – 746-ю позицию, в РФ – 4-ю.

> **Мобильная инфраструктура** – комплекс приложений для мобильных платформ iOS и Android, включая приложения «Университет ИТМО», «Ученый совет» и др.

И информационная инфраструктура продолжает развиваться, это является общим делом всего вуза, а не только ИТ-службы. В этом плане Университету ИТМО, пожалуй, повезло больше других, ведь специалистов, способных в будущем совершить революционные открытия и реализовать масштабные проекты, мы воспитываем сами. EOF

София Савинова: «ИТМО в интернете – понятно, удобно, привлекательно!»

София Савинова, начальник отдела продвижения интернет-ресурсов Департамента информационных технологий Университета ИТМО

Сегодня все мы видим глобальные перемены в области высшего образования. Конкуренция в высшем образовании растет, и в нынешних быстро меняющихся условиях показателем эффективности деятельности вуза становятся рейтинги.

Позиционирование университета в сети Интернет и его продвижение в различных рейтинговых системах является неотъемлемой частью стратегии развития современной высшей школы. И Университет ИТМО следует мировым тенденциям. В рамках правительственной программы развития «5 в 100» мы стремимся выйти в лидеры среди высших учебных заведений и, надо сказать, демонстрируем стабильный положительный результат!

За последние два года Университет ИТМО значительно укрепил свои позиции, войдя в тысячу лучших вузов мира и в первую сотню вузов стран БРИКС. Наша команда ежедневно работает над повышением качества ресурсов, их эффективности и привлекательности для пользователей. Мы открываем интернет-площадки для рекрутинга и взаимодействия с иностранными коллегами, обеспечиваем поддержку наших ресурсов, применяя новейшие информационные технологии. Мы – ведущий ИТ-вуз страны, и рейтинги только подтверждают это! EOF

Справка

7 мая 2012 года Президент РФ Владимир Путин подписал Указ №599 «О мерах по реализации государственной политики в области образования и науки». Одной из задач, поставленных главой российского государства перед Правительством РФ, было «вхождение к 2020 году не менее пяти российских университетов в первую сотню ведущих мировых университетов согласно мировому рейтингу университетов». Так начался проект по повышению конкурентоспособности ведущих университетов Российской Федерации среди мировых научно-образовательных центров (проект «5 в 100»).

Путь длиной в 116 лет

В 1900 году в Ремесленном училище цесаревича Николая открылось механико-оптическое и часовое отделение. На тот момент это было единственное в России учебное заведение, где готовили мастеров в области точной механики и оптики.

Из отделения выросло уже самостоятельное среднее техническое училище, а в 1920-м, когда основные классы училища были преобразованы в техникум точной механики и оптики, учебное заведение получило право подготовки инженеров узкой специализации.

В 1931 году техникум подготовил первый в России выпуск инженеров-приборостроителей.

В 1930-м техникум был преобразован в Учебный комбинат точной механики и оптики, а спустя три года от него отделился знаменитый Ленинградский институт точной механики и оптики – ЛИТМО.

В 1937 году в ЛИТМО открылась одна из первых в СССР лабораторий счетно-решающих приборов, преобразованная впоследствии в кафедру математических и счетно-решающих приборов и устройств.

С первых дней Великой Отечественной войны в ЛИТМО действовала военно-ремонтная база, где изготавливались и ремонтировались контрольно-измерительные приборы для армейских и флотских подразделений.

В послевоенные годы институт активно развивался, многие сотрудники были удостоены Государственных и Ленинских премий. А в 1958-м была создана первая ЭВМ для инженерных расчетов «ЛИТМО-1».

В семидесятые годы в ЛИТМО открылась лаборатория лазерной технологии, а также был построен учебный корпус на Саблинской улице, являющийся сейчас основным зданием Университета ИТМО.

В восьмидесятые специалистами ИТМО были начаты работы в области микропроцессорной техники, был создан межотраслевой институт повышения квалификации специалистов промышленности по новым направлениям развития техники и технологии.

В девяностые годы Университет ИТМО выступил инициатором и основным разработчиком сети RUNNet, объединившей все крупные научно-образовательные центры России; был создан учебно-научный центр «Компьютерная оптика», а также открыт факультет компьютерных технологий и управления. Так началось развитие нового, «компьютерного», направления в ИТМО.

Подготовка новых специалистов быстро набрала обороты, и сборная команда Университета ИТМО по программированию вышла на лидирующие позиции в мировых студенческих чемпионатах.

В 1992 году ЛИТМО был переименован в Санкт-Петербургский институт точной механики и оптики, а в 1994-м получил статус университета.

Развитие информационных технологий стало одной из предпосылок больших изменений в университете. Он значительно обновился, расширились сферы деятельности, открылись новые возможности.

Все это нашло отражение в изменении названия – с 2003 года учебное заведение именуется Санкт-Петербургский государственный университет информационных технологий, механики и оптики.

В настоящее время Университет ИТМО – ведущий университет России в области информационных и фотонных технологий, один из немногих вузов нашей страны, получивших в 2009 году статус национального исследовательского университета.

С 2013 года университет является участником программы повышения конкурентоспособности российских университетов среди ведущих мировых научно-образовательных центров (известной как «5 в 100»).

Антон Клочков: «Успех поступления в вуз зависит не от пролитых слез, а только от усердия школьника»

Антон Клочков, абитуриент из Хабаровского края, номинант губернаторской стипендии и призер региональной олимпиады по информатике

Наш собеседник рассказал, почему только в Университете ИТМО готовят профессионалов, каково иногородним студентам в Петербурге, и о том, что еще, кроме образования, школьники ценят в вузах

– Что повлияло на выбор заняться информационными технологиями?

– В детстве, как и многие, я хотел стать космонавтом или хотя бы летчиком. А в четвертом классе учительница сказала, что у меня хорошо получается работать с компьютерными системами. Окончательное решение связать свою жизнь с информационными технологиями сформировалось в седьмом классе. Я понял, что мне нравится работать с компьютером. Знаете, есть дела, которые нужно делать, а есть те, которые хочешь делать, а здесь для меня объединяется и то, и другое.

– Почему был выбран университет на Северо-Западе, а не где-нибудь на Дальнем Востоке, ближе к дому?

– Если рассматривать высшее образование в сфере информационных технологий, то, на мой взгляд, университеты западной части России существенно выигрывают у вузов на том же Дальнем Востоке. Красноречивый показатель – средний проходной балл. В Хабаровском крае для поступления на специальность «Информационные технологии» требуется примерно 180 баллов, а в Петербурге и Ленинградской области – около 270. Это о многом говорит, в том числе о престиже университета.

– А почему именно Университет ИТМО?

– Я начал искать вузовские образовательные программы еще в девятом классе и заметил, что много людей, востребованных в IT-профессиях, закончили именно Университет ИТМО. Я планирую поступить либо на кафедру компьютерных технологий, либо на кафедру безопасных информационных технологий. Многое зависит от результатов ЕГЭ. На КТ пройти непросто: туда берут, как правило, призеров олимпиад. Скоро будет проводиться многопрофильная инженерная олимпиада «Звезда», в которой я приму участие. Займу призовое место – буду пробовать поступать на кафедру компьютерных технологий. Вполне вероятно, что также подам документы в СПбГУ, но этот вуз готовит по всем специальностям, а Университет ИТМО все-таки более профильный, поэтому приоритет отдаю ему.

– Какие еще «бонусы» должны быть у университета, чтобы абитуриенты хотели туда поступать?

– Часто к нам в школу приезжают выпускники различных университетов и отвечают на вопросы школьников. Абитуриентов интересует не только сама учеба, а и «университетская жизнь» – возможность заниматься различными активностями в стенах вуза: участвовать в конференциях, заниматься спортом, играть в КВН и тому подобное. Кроме того, здорово, когда в вузе работают иностранные преподаватели – они могут передать свой уникальный опыт работы за рубежом.

– От чего зависит успех абитуриента при поступлении?

– Все зависит от самого ученика. У меня в городе не у кого было обучаться компьютерным технологиям на должном уровне: все, что получалось узнать, я находил сам в книгах, в интернете. Если у ученика есть цель, он просто идет к ней: сидит и учит, ищет дополнительную информацию. А если ничего не делать, оставить все на последний момент, конечно, начнутся переживания о том, куда поступать, как успешно сдать экзамены. У меня в классе, например, есть ребята, которые еще не определились со своей будущей профессией, хотя большинство – достаточно целеустремленные люди.

– Сложно ли иногороднему студенту адаптироваться в Санкт-Петербурге?

– Я много думал об этом. Все-таки начнется самостоятельная жизнь, вдали от родителей. Сейчас чувствую себя более уверенно: я уже несколько раз был в Санкт-Петербурге, жил там пару месяцев, поэтому более-менее ориентируюсь в городе. В любом случае, с какими-то бытовыми проблемами всегда могут помочь одногруппники, а в учебе – преподаватели. Если человек хочет учиться, они это видят.

Мне нравится представлять жизнь в большом городе. Санкт-Петербург, мне кажется, по праву называется «культурной столицей». Как-то во время перелета из Хабаровска в Петербург я сравнил поведение людей в аэропортах обоих городов. Если в Хабаровске они просто разговаривали, ожидали, то в Северной столице много пассажиров коротали время с книгой.

– В будущем займешься наукой или будешь работать в какой-нибудь компании?

– Скорее первое. Хотелось бы поступить в аспирантуру. Возможно, я мог бы заняться алгоритмами для распознавания образов или моделирования физических процессов, например, в робототехнике. Пока я хочу работать только в России. Сейчас стране нужны новые технологии, а для того, чтобы их создавать, необходимо развивать науку. Сейчас ученым стали уделять больше внимания, поскольку они очень нужны стране. Ставку делают и на нас – новое поколение. Нужно не подвести. EOF

Вакансия: программист Perl

Perl был ранее достаточно популярным языком программирования. За долгий период развития создано множество решений, работающих и сейчас. А значит, Perl до сих пор остается актуальным

1. Какими знаниями и навыками должен обладать программист Perl?
2. Каков инструментарий программиста Perl?
3. Каковы требования компании к уровню образования потенциальных сотрудников?
4. Какой необходим опыт работы?
5. Есть ли особые требования, которые обусловлены спецификой деятельности компании?

Александр Горный, директор по информационным технологиям Mail.Ru Group

1 Главное, чтобы программист Perl просто был. Язык давно потерял популярность, разработчики в большинстве своем перешли на Python или Go или, как я, стали руководителями.

«Перловиков» в России осталось очень мало, новые люди в этот язык почти не приходят, и любой проект, в котором остался Perl, всегда страдает от того, что ему не хватает программистов на рынке. У нас в Технопарке (это один из образовательных проектов Mail.Ru Group, созданный на базе МГТУ имени Баумана) даже запущен специальный полугодовой курс для подготовки сильных «перловиков». Мы набираем примерно 40-50 человек каждые полгода, до конца доходят 10-12. Лучших трех-четырех мы берем к себе в компанию.

Над курсом работают три разработчика из трех разных бизнес-подразделений: Почты, Поиска и Моего Мира.

2 В этом нет серьезных технических причин, но исторически сложилось, что разработчики на Perl – самые хардкорные из всех программистов на скриптовых языках. «Перловик» может не знать JS или даже основ CSS, но зато он обязан отлично разбираться в SQL, оптимизации запросов и архитектуре Unix. Многие из них знают C на уровне, достаточном для написания собственных модулей на технологии XS.

3 Программисты Perl обычно в том возрасте, когда вопрос об образовании уже не слишком актуален. Какая по большому счету разница, что за вуз был закончен программистом 15 или 20 лет назад, нынешний специалист совсем не похож на того выпускника, хоть и носит ту же фамилию. Да и языку Perl его там точно не учили. Мне самому, например, до сих пор проще всего писать именно на Perl, и, думаю, я мог бы снова работать программистом на нем, но диплом МГУ 2000 года работодателя в этом никак не убедит.

Если же мы говорим о выпускниках нашего Технопарка, то в ходе обучения они проходят огонь, воду и медные трубы, так что сомневаться в их квалификации не приходится. Кроме того, как я уже сказал, на работу в Mail.Ru Group приглашаются лучшие из них.

4 Если это Perl-разработчик и у него нет за спиной хотя бы пяти – семи лет опыта, то, наверное, это американский шпион. Если серьезно, то требования к опыту от языка почти не зависят. Обычно пары лет работы достаточно, чтобы полностью освоиться с любой технологией, Perl тут не исключение.

5 На Perl написана значительная части Почты Mail.Ru, это сердце нашей компании. Естественно, мы ждем от программистов-«перловиков» максимальной ответственности и умения работать со сложными высоконагруженными проектами.

Руслан Закиров, Team-lead команды Perl-разработки в ООО «Спортс.ру»

1 Знать Perl, уметь пользовать perldoc. Поскольку Perl придерживается принципа TIMTOADY (There's more than one way to do it), подразумевающего, что существует больше одного способа сделать что-либо, то программист должен иметь хорошие навыки проектирования поддерживаемого и читаемого кода для работы над долгосрочными проектами.

2 Perl, Vim/Emacs, SSH, Git, man, perldoc, CPAN, DBI, Dancer, Mojo.

3 У нас нет жестких требований в плане наличия у человека высшего технического образования, он вполне может быть и самоучкой. Но наличие хорошей теоретической базы поможет ему в работе и в развитии внутри компании.

4 Опыт от трех лет – прям хорошо, но больше не значит лучше. Важен опыт работы в реальных проектах, в реальных боевых условиях и в команде.

5 Желание разбираться не только в бэкенд-разработке, хорошее чувство юмора.

Алексей Мележик, senior software developer, EPAM Systems

1 Программировать в принципе, понимание базовых понятий в разработке ПО, рефакторинга кода, тестирования, знание общих структур данных (массивы, хеши), ООП.

Также важно умение обосновывать свою точку зрения при выборе того или иного решения, анализировать, находить больше, чем один подход, при решении задачи, умение сравнивать выбранные методы.

2 Юнит-тесты Test::More, веб-фреймворки – хотя бы знать, что такие есть, Mojolicious, Dancer2, Catalyst, Kelp и т.д. Средства для работы с реляционными базами данных, DBI must have, ORM (DBIx::Class) – желательно, но не обязательно – опять-таки, чтобы показать свой кругозор. Ну и, конечно, умение написать какой-нибудь простой SQL join-запрос.

3 Опять же знание теории как таковой неплохо, и какой-то минимум важен, но гораздо существеннее умение думать и находить решения.

4 Смотря кого ищем. Если синьора, то, несомненно, соответствующий стаж, если юниора, можно взять с минимальным опытом, даже не обязательно на Perl. В любом случае главное, чтобы человек показал, что он умеет думать.

5 Все зависит от конкретной позиции и ситуации внутри компании, какой человек ищется, а это всегда по-разному.

Илья Чесноков, проект PerlJobs.Ru

1 Знание языка Perl – основы, структуры данных, ООП, основы регулярных выражений. Хороший список вопросов для самопроверки опубликован по ссылке http://modernperlbooks.com/mt/2011/01/how-to-identify-a-good-perl-programmer.html (документ этот на английском, и, разумеется, английский язык тоже необходим).

Неплохо знать язык Си, чтобы иметь возможность писать XS-модули или хотя бы понимать их код.

Обязательно понимание реляционных баз данных и способов их использования в Perl – «чистый» DBI и ORM (DBIx::Class, Rose::DB...).

Если речь идет о веб-разработке, то нужны базовые знания HTML/CSS/JavaScript и популярных веб-фреймворков (Catalyst/Mojolicious/Dancer и т.д.) – в каждой компании обычно используется только один, но желательно иметь о каждом общее представление и понимать основные принципы шаблона MVC и протокола PSGI, на которых они основаны.

Нужно понимание объектно-ориентированного программирования и современных способов работы с ним в Perl. Зачастую на интервью спрашивают, как создать объект в Perl: при этом классический пример вида bless $ref, $class; уже не является 100% правильным ответом – более правильно будет предложить использовать один из ООП-фреймворков.

Так как на Perl легко написать неудобочитаемую кашу вместо кода, программист должен придерживаться определенного стиля кодирования – принципы написания наиболее читабельного кода описаны в книге Damian Conway «Perl Best Practices», которая весьма рекомендуется к прочтению. Как правило, компании выбирают предпочтительные для них принципы и оформляют их как локальный style guide, которого необходимо придерживаться.

Также советую прочитать и иметь под рукой «Рефакторинг» Мартина Фаулера.

Еще одна хорошая книга – «Modern Perl», ее особенность в том, что в ней говорится, как писать идиоматический код и использовать возможности Perl по максимуму (чтобы Perl-код выглядел и работал как код на Perl, а не как код на Си).

Мы ждем от программистов максимальной ответственности и умения работать со сложными высоконагруженными проектами

2 Программы на Perl запускаются, как правило, в nix-среде, так что претендующий на роль Perl-программиста должен уметь хорошо ориентироваться в nix-системах. Умение писать и отлаживать код в консоли помогает отлаживать программы на удаленных серверах или виртуальных машинах.

Говоря об отладке, стоит вспомнить встроенный консольный отладчик Perl – он довольно функционален, так что его обычно достаточно для большинства задач по отладке программ, хотя программисты часто предпочитают просто выводить отладочные данные на экран. Также имеются другие отладчики – например, один из лучших – в (платном) Komodo IDE – если не ошибаюсь, он является оберткой над тем же самым консольным отладчиком.

Безусловно, необходимо уметь пользоваться какой-либо системой контроля версий кода, моя любимая – Git.

Помимо этого, Perl сам предоставляет множество инструментов для эффективной работы. Их в первую очередь стоит искать на CPAN. Например, это системы управления инсталляциями Perl и его модулей – perlbrew, local::lib, cpanminus.

Perltidy позволяет отформатировать код в соответствии с принципами из Perl Best Practices (по крайней мере теми из них, которым вы готовы следовать), perlcritic покажет нечитабельные или потенциально опасные места в коде, а Devel::NYTProf позволит выявить узкие места в производительности программы.

Хороший Perl-разработчик должен уметь писать тесты для своего кода (на CPAN большой выбор модулей с именем Test::*), запускать их утилитой prove, а также документировать свои программы с помощью языка разметки POD.

3 Зависит от компании, но я бы сказал, что сейчас требования к образованию не слишком высокие

(особенно за рубежом, где обучение в университете зачастую стоит неоправданно дорого), а на первом месте стоят скорее способность и желание обучаться самостоятельно. Например, я знаю замечательного программиста, который вообще не имеет никакого технического образования, а закончил, если я не ошибаюсь, только факультет психологии.

Хотя в целом высшее техническое образование – несомненный плюс, а еще лучше, если будущий Perl-программист поработал до этого некоторое время системным администратором (плохо, если он работал C, JS или, не приведи господь, PHP-программистом).

> В мире ИТ в первую очередь смотрят на то, что ты умеешь делать и где ты ранее работал, а высшее образование играет меньшую роль

4 Это также зависит от специфики деятельности компании и ее потребности в Perl-программистах. Некоторые компании не предъявляют высоких требований к опыту и готовы сами обучать сотрудников (спасибо им за это!), другие же предпочитают сразу найти опытного сотрудника, который будет учиться сам и делиться своими знаниями с коллегами.

Как правило, приветствуется некоторый опыт работы в той же области, которой занимается компания, наличие написанного программистом кода где-нибудь на GitHub (а еще лучше – на CPAN), на который можно посмотреть и оценить качество программы, образ мыслей разработчика и стиль кодирования, – в общем, по-моему, тут все так же, как и с другими языками.

5 Как Perl-разработчик я работал в основном в хостинг-провайдерах – там нужно понимать общепринятые протоколы обмена данными (как правило, это что-нибудь основанное на XML/JSON, хотя бывают и неприятные исключения), потому что деятельность хостинг-провайдеров по большей части связана с перепродажей сервисов других компаний, заказ которых происходит через API. Кроме того, нужно уметь принимать платежи, используя API платежных систем. Плюс еще зачастую необходимо писать API для своих клиентов – так что да, в хостинг-компаниях API со всех сторон.

Обязательно знание SQL баз данных. Часто NoSQL, очереди задач, конкурентное программирование – если нужно выжать из программы максимум производительности. Разумеется, нужно хорошо разбираться в принципах работы протокола DNS. Как правило, все это рано или поздно приходит в процессе работы.

Виктор Турский, технический директор компании WebbyLab (http://webbylab.com)

1 Perl-разработчиков на рынке не так уж много. И в целом наиболее важное – это знание самого языка Perl, понимание его идиоматики. Также важно следование современным практикам разработки, так называемый Modern Perl. Это использование систем контроля версий (типа Git), разработка через тестирование, правильная работа с исключениями, использование PSGI-абстракции для веб-приложений, современные подходы к структурированию приложений (Dependency Injection) и т.д.

Также важно, чтобы кандидат обладал системным мышлением и системными знаниями. Мы занимаемся интернет-приложениями и соответственно хотим, чтобы кандидат не только умел писать код на Perl, но и понимал, как работает интернет.

2 С того, что мы используем каждый день, и то, что присутствует почти в каждом нашем проекте:

- **plenv** – инструмент для управления версиям Perl;
- **Carton** – для управления зависимостями и фиксации их версий;
- **Perl::Tidy** – для форматирования кода;
- **Test::Most** – для тестирования;
- **Devel::Cover** – для оценки покрытия кода тестам;
- **Devel::NYTProf** – лучший профайлер под Perl;
- **Moo** – ООП-фреймворк;
- **Mojolicious** – веб-фреймворк;
- **Rose::DB::Object** – ORM;
- **Validator::LIVR** – валидатор;
- **Try::Tiny** – для обработки исключений.

Остальные инструменты уже специфичны для проекта. Разработку ведем под Linux.

3 Мы всегда отдаем предпочтение кандидатам с высшим образованием, но образование не обязательно должно быть профильным. Просто университет дает студентам подходы к изучению и структурированию новой информации.

Конечно, мы рассматриваем кандидатов и без высшего образования, но тогда от него требуется релевантный опыт работы. Если кандидат без образования и без опыта, то либо он только закончил школу и ему 17, и, возможно, он слишком мал для ответственной работы, либо он старше (например, 21), тогда возникает вопрос, если он не учился и не имеет опыта работы, то что он делал четыре года и умеет ли он работать вообще?

4 Для нас не так важно количество лет, как наличие грамотного подхода к разработке и готовность разобраться в новых вещах. Конечно, если мы говорим о старших разработчиках, то для нас всегда важно увидеть, что человек до этого строил проекты сопоставимой сложности и в состоянии справиться с такими же задачи в нашей компании.

5 Мы создаем веб-приложения, соответственно желательно разбираться в веб-технологиях. Каких-то других специфических требований нет.

Александр Ружников, Perl-программист в компании Intermedia

1 Ну в первую очередь конечно, знание языка Perl, типы переменных, функции, операторы, ООП. Часто приходится смотреть в чужой код, поэтому необходимо знать что делает тот или иной оператор или конструкция.

Тут нет большого отличия от других языков программирования. Perl позволяет одну и ту же вещь сделать многими способами, поэтому желательно сразу выработать у себя единый подход к написанию кода для того, чтобы у других разработчиков не вытекали глаза от того, как вы реализовали задачу.

Также часто требуется знать один-два веб-фреймворка (например, Catalyst и Mojolicious) + какой-нибудь из шаблонизаторов, например, Template-Toolkit.

2 Тут тоже как у остальных разработчиков: IDE или текстовый редактор, например, Eclipse или Sublime Text. Также в мире разработки сейчас никуда без системы контроля версий. Git тут правит балом, хотя другие системы (svn, mercurial) тоже встречаются порой в различных компаниях.

Как правило, код на Perl запускают на операционной системой GNU Linux. Самые распространенные из них – это Centos и Debian/Ubuntu. На рабочих машинах разработчиков тоже нередко используется Linux, т.к. он прям-таки создан для того, чтобы писать под него на скриптовых языках. Иногда можно встретить людей, пишуших код на Perl на Windows, но мне сложно представить, чтобы я когда-то такое начал делать – сам использую Linux Mint.

Ну и часто (даже правильнее сказать, почти всегда) в проекте используется одна из распространенных СУБД: MySQL, PostgreSQL или Oracle. Иногда также используются No MySQL базы типа MongoDB и различные кэши: Redis, Memcache и пр.

3 Ну лично я ни разу не сталкивался с таким, чтобы человека без высшего образования не взяли на работу программистом из-за отсутствия этого самого ВО. В мире ИТ в первую очередь смотрят на то, что ты умеешь делать и где ты ранее работал, а высшее образование играет меньшую роль. Это, конечно, не касается компаний, где разработка ПО ведется на сложных математических алгоритмах. Здесь да, будут смотреть, откуда пришел человек, что он заканчивал и какой экспириенс в плане алгоритмов он имеет. Но в большинстве же компаний большой математический бэкграунд не требуется, важнее иметь знания в области используемых языков и фреймворков.

4 Если устраиваешься джуниором (начинающим) разработчиком, то опыт работы часто вообще может быть нулевым. Важно желание развиваться и писать код на используемом языке программирования.

Для средних разработчиков (мидлов) обычно просят не менее 1-2 лет опыта работы на данном языке.

По стеку технологий опять же зависит от того, что используется в компании. Как правило, это веб-фреймворк, как писал выше, Catalyst, Mojo или аналогичный, далее шаблонизатор для рендеринга страниц, ну и, конечно же, опыт работы SQL. Чаще с MySQL, реже с PostgreSQL и/или Oracle.

Иногда также требуется опыт работы с верстской, чаще с JavaScript. Но, как правило, это требование является желательным, но не обязательным.

5 Конечно. Каждая компания имеет свою специфику работы и каждый раз требуется некоторое время для того, чтобы вникнуть во внутренние бизнес-процессы. Как правило, у опытного разработчика это занимает не более 1-2 недель.

Мое мнение такое, что чем больше у человека опыта, чем больше он интересуется различными технологиями и языками программирования, тем проще ему бывает вникнуть в особенности работы на новом месте.

Порой в вакансии декларируется, что от человека требуется знание языка Perl, а на месте выясняется, что неплохо бы ему еще знать PHP, JavaScript и в особых случаях C++. Знающему и подготовленному человеку будет проще, неподготовленному придется в спешном порядке получать требуемые знания.

Пронин Олег, технический директор Crazy Panda

1 Программист Perl, как и любой другой программист, в первую очередь должен обладать аналитическим складом ума и умением быстро писать эффективный и понятный код, хорошо владеть отладчиком и уметь быстро искать утечки памяти (а лучше – писать код, не требующий таких операций). Программисты Perl высокого уровня также хорошо разбираются во множественном горизонтальном наследовании, различных фреймворках типа Catalyst, DBIx::Class, а также на отличном уровне владеют языками C/C++ для написания XS-модулей к Perl.

> Программист Perl, как и любой другой программист, прежде всего должен обладать аналитическим складом ума и умением быстро писать эффективный и понятный код

2 Из обязательного, пожалуй, можно отметить отладчик и разнообразные профилировщики. Что касается среды разработки, то она может быть совершенно разнообразной – от консольного VI через терминал или блокнота в Windows до IDE типа Eclipse/IDEA с графическими отладчиками и прочими плюшками. Обычно компании никак не регламентируют и не ограничивают этот инструментарий.

3 По моему опыту, большинство компаний не требуют высшего образования при приеме на работу программистов, и, более того, это обычно и плюсом то почти не является. Если бы, например, к нам пришли программист с красным дипломом и другой программист с двумя классами образования, но более соображалистый/умный, то дипломник однозначно в пролете.

4 Формально у нас опыт работы не требуется. В любом случае, уровень кандидата и его возможности оцениваются на собеседовании. Если начинающий толковый программист решит у нас поработать, мы с удовольствием его возьмем на Junior Perl.

5 В нашей компании специфических требований нет. Плюсами являются технический английский и положительное отношение к играм.

Подготовил Игорь Штомпель

Виктор Иванников:
«Я понял, что программирование – прекрасный мир!»

Как зарождалось системное программирование в СССР? Как создавался первый советский суперкомпьютер? Какую роль сыграла IBM в возникновении индустрии ПО? Как в 70-е годы преодолевали технологическое отставание нашей страны и к чему это привело?

Об этом рассказал «Системному администратору» академик Виктор Петрович Иванников, создатель и бессменный глава Института системного программирования РАН, заведующий кафедрами системного программирования на факультете ВМК МГУ и в Московском физико-техническом институте, главный редактор журнала «Программирование»

Беседовала Анна Новомлинская

– Виктор Петрович, как вы выбрали вуз и почему решили пойти в программисты? Тогда эта профессия была очень редкой.

– Есть теория у академика Дмитрия Павлович Костомарова, которая мне очень нравится. Он говорил о том, почему молодежь наших поколений шла в технические вузы. Потому что в то время это давало свободу, там было меньше лжи, а была реальная работа. Я этого особенно не осознавал, но, наверное, эти мотивы все-таки сказывались на принятии решения. Один мой дальний родственник учился в Физтехе. Когда я был старшеклассником, он мне прислал задачники, где были собраны экзаменационные задачи по физике и математике. В буклете Физтеха было написано, что выпускники идут работать в научные учреждения.

– Вы учились в физико-математической школе?

– Я жил в поселке на Урале, какие уж там физматшколы. Мама работала фельдшером, отец – мастером на заводе, никто мне не помогал. Была обычная школа, но учился я хорошо. В итоге поступил в МФТИ. Модель физтеха достаточно своеобразная и интересная. Отцы-основатели института – Капица, Христианович – еще перед войной предлагали ввести новую модель образования, направленную на то, чтобы готовить специалистов, которые могут разрабатывать новые технологии. Обучение на первых трех курсах происходит в метрополии, в городе Долгопрудном. А потом – как в метрополии, так и в диаспорах – на базовых кафедрах Физтеха в академических институтах или конструкторских бюро. Там тоже читали лекции, но самое главное, что постепенно студенты втягивались и в те проекты, которые велись в этих учреждениях. Одновременно студентам давали стабильные классические курсы и новые, связанные с теми работами, которые велись в этих институтах, обеспечивали постепенность врастания в разработки. Была очень большая свобода.

– Вы о ней уже упоминали. Имеете в виду политическую свободу?

– Свободу перемещения, выбора, специализации… В Физтехе я мог выбирать, занимался разными вещами. Год проработал с цифровыми дифференциальными анализаторами, но мне не понравилось – надо было много паять, я понял, что это не мое. Еще год занимался функциональным анализом, математикой, а потом мой руководитель уехал в академгородок, и пришлось на шестом курсе выбирать что-то другое. Слышал о программировании, что это такое, не известно, я тогда считал – это занятие для пожилых женщин после 35 лет. Но в отделе программирования открывалась возможность получить полставки, ребята говорили, что это здорово. Главное, я столкнулся с тем, что там было не так много рутинной работы, не надо было паять, установки делать, прежде чем дойдешь до эксперимента. Руководитель дал мне задачу по экономии памяти для переменных, она была очень интересной, и результаты мне нравились. С этого все и началось, я понял, что программирование – прекрасный мир!

– Так вы начали работать в Институте точной механики и вычислительной техники?

– Я попал на кафедру Сергея Алексеевича Лебедева, основоположника вычислительной техники в СССР, в Институт точной механики и вычислительной техники. Там работал Лев Николаевич Королев, он стал очень близким мне человеком, учителем и другом. Постепенно я познакомился со многими выдающимися программистами, которые работали в вычислительном центре РАН, академгородке в Новосибирске, в ИПМ. Программистов было не так много, все друг друга знали. Я лично был знаком с академиком Николаем Николаевичем Говоруном из Дубны, Андреем Петровичем Ершовым, одним из основоположников программирования, создателем Сибирской школы информатики,

Эдуардом Зиновьевичем Любимским, известным ученым в области системного программирования.

Можно назвать массу прекрасных имен и многое вспомнить – конференции, зимние школы, открытость обсуждений. Все оказалось крайне интересным, тем более такому молодому человеку, каким я тогда был.

В ИТМиВТ была потрясающая среда, доброжелательная и открытая обстановка. Там работали великие инженеры – Мельников, Соколов, Бабаян – талантливые люди, с которыми было очень интересно работать. Когда я окончил институт, сразу попал на разработку программ, операционных систем для БЭСМ-6. Потом был еще один запомнившийся проект – разнородный многомашинный комплекс операционной системы, которая была предназначена для АС-6, для центра управления полетами. Мы сделали несколько таких комплексов, в конце 1979 года они прошли испытания.

– То есть сразу после института получили доступ к засекреченным проектам?

– Уже после второго курса нам давали вторую форму допуска, с грифом «совершенно секретно», еще в Физтехе, мы же занимались новыми технологиями…

– Вы участвовали в разработке первых советских компьютеров?

– В середине 70-х Королев полностью ушел работать в университет. Из ИТМиВТ выделилась большая группа людей, которая ушла с Владимиром Андреевичем Мельниковым в один из институтов Минэлектронпрома СССР, чтобы делать советский суперкомпьютер. Я там вел два дела – все системное ПО и САПР для разработки этого компьютера. Все работы шли параллельно – разработка аппаратуры, архитектуры, операционной системы, компиляторов, разных системных вещей и САПР для суперкомпьютера. В итоге к концу СССР было сделано несколько таких компьютеров.

Они работали, но государственные испытания не проводились. В серийное производство они не пошли и не использовались. Закончилась эпоха, и все быстро рухнуло. У нас были отдельные макеты, а сама сборка шла на заводе в Калининграде. Так эти компьютеры там и остались.

– Ведь все рухнуло не в одночасье...

– Советский Союз развалился не мгновенно. Накапливались дефекты в экономике, и в технологическом плане петух прокукарекал достаточно давно. Проблем было много. Со второй половины семидесятых годов наметилось технологическое отставание страны, прежде всего в микроэлектронике, в разных компонентах. Конечно, над этим работали, но очень сильно было стремление получить быстрый результат за счет чужих технологических разработок, и Министерство радиопромышленности, Минэлектронпром взяли курс на копирование. Это началось не в 90-х, а намного раньше – в 70-х. Существовало много школ по вычислительной технике, была великая машина БЭСМ-6, однако началось копирование линий IBM-360 и DEC (Digital Equipment Corporation).

– Получается, сегодня в битве за импортозамещение в ИТ мы пожинаем то, что посеяли в 70-х?

– Искали пути преодоления технологического отставания от Запада. И приняли решение по копированию линии

Виктор Петрович Иванников, академик, ведующий кафедрами системного программирования на факультете ВМК МГУ и в Московском физико-техническом институте, главный редактор журнала «Программирование»

IBM-360. У такой тактики были противники, например, Лебедев отказался участвовать в этом. ИТМиВТ занимал среди школ особенную позицию, он сам делал свою архитектуру, сам искал пути. Творчество тоже существовало. В Зеленограде были очень интересные физики. Изначально Зеленоград создавался как законченная экосистема, центр микроэлектроники, но, к сожалению, доминировало технологическое копирование. Это решение было принято на самом высоком государственном уровне. В авиации, ракетной технике, автомобильной и в других отраслях было то же самое. Например, как произошло врастание в ракетной области – просто скопировали баллистическую ракету ФАУ-2. Германия в то время доминировала в технологиях, наиболее распространенным в физике, математике, инженерном деле был немецкий язык. Нам на военной кафедре в Физтехе читали не что-нибудь, а курс по ФАУ-2. Я учился на радиотехническом факультете, меня восхищало, как в этой ракете была сделана система управления. Тем не менее это было только врастание, а не постоянное копирование.

– В СССР должны были понимать, к чему может привести эта стратегия. Ведь и тогда думали об информационной безопасности, о которой мы столько говорим?

– Думали в основном о вопросах, связанных с надежностью, как избежать ошибок, что произойдет в случае ошибки. Защита линий связи, шифрование – все это было, но термин «информационная безопасность» тогда не использовался.

Что касается программирования, у нас при советской власти так и не создали индустрию программного обеспечения. Были какие-то наметки, прежде всего в интересах оборонных отраслей, но полноценной индустрии ПО не было. В основном софт создавался либо как некоторый компонент вычислительной машины, либо как важное приложение в противоракетной обороне, при моделировании ядерных бомб, в управлении ракетной техникой и в подобном.

Участники учебного курса в ИСП РАН

– Как шел процесс индустриализации сферы ИТ в других странах, например, в США?

– В Штатах происходило то же самое, существовал определенный задел, но нужен был толчок к созданию индустрии ПО. Таким толчком стал антитрестовский процесс против IBM. Во второй половине 60-х корпорация захватила почти 90% рынка за счет продаж IBM-360. И ей грозило антитрестовское разъединение, как это уже произошло в телефонии, когда монополию (AT&T) разделили на три компании. Журнал Computer Society в свое время разместил большой материал «Анналы истории компьютеров» о зарождении индустрии ПО, о том, как проходил этот процесс. Юристы IBM придумали интересный ход – полностью раскрыли архитектуру IBM-360. В итоге это дало возможность независимым компаниям создавать софт для системы 360, в том числе системный. Благодаря такому ходу IBM избежал антитрестовского расчленения. А поскольку это был самый популярный компьютер, та история стала мощным толчком для создания индустрии. В США это стало возможным, поскольку компьютеры использовались в большей степени в гражданской сфере – в экономике, банковской области. А у нас в основном в оборонной промышленности.

– Кому пришла идея основания Института системного программирования РАН?

– Мы говорили с Андреем Петровичем Ершовым, Михаилом Романовичем Шура-Бура, Эдуардом Любимским – все сходились во мнении, что в стране нужны академические институты по системному программированию, была такая мечта. В 1994 году удалось создать такой институт (ИСП РАН), и мне повезло стать его директором.

– 1990-е годы, советская хозяйственная система разваливается, инженеры уходят в торговлю, а вы создаете академический институт...

– Было очень тяжелое время, когда финансирования практически не было. Но к моменту создания ИСП у меня появились контакты за границей, можно было искать гранты. В 1984 году меня избрали членом-корреспондентом Академии наук, и я смог выезжать из страны. До этого здесь участвовал в конференциях, но не должен был иметь контактов с зарубежными коллегами. И вот настал сложный период – институт создан, а больше половины сотрудников в 90-х уехали за границу, нет ни денег, ни людей.

А к этому времени я еще стал завкафедрой на факультете вычислительной математики и кибернетики (ВМК) МГУ. Шура-Бура с его соратником и моим другом Любимским пригласили меня заведовать кафедрой системного программирования. Я в МГУ и Физтехе давно преподавал, пару лет читал лекции в МАИ, потом появилась кафедра системного программирования в Физтехе. К нам приходили единичные студенты – 1-2 человека в год. А когда стал завкафедрой, тут уже пошел серьезный поток молодежи, десятки ребят, которых тоже надо было учить. Сейчас в значительной степени сотрудники института – наши бывшие студенты. Процентов 20-30 из каждого потока приходили и оставались здесь.

– Как вам это удалось? Воплощали везде физтеховскую модель?

– Существуют разные успешные модели. Например, есть репертуарные театры, а есть антрепризы, когда собирается команда для того, чтобы поставить мюзикл или снять кинокартину. Есть модели школ, а есть модели стартапов. У нас в стране, может быть, в силу российского менталитета, памяти о крестьянских общинах, доминировал подход школ, вмещающих сразу несколько вещей. В научной школе есть все – академические исследования, практическая разработка, технологии, образование – все перемешано, и создана сложная экосистема. Так мы и работаем.

– Расскажите, как решались задачи финансирования института?

– В 90-х да и в 2000-х наша промышленность не была заинтересована в новых разработках технологий, так как это дело долгое, для достижения результата требуются годы. Проще покупать софт, чем с высокой степенью риска вкладываться в его разработку. В 1992-м создали Российский фонд фундаментальных исследований (РФФИ), но объем финансирования там был крайне маленьким. Поэтому мы были предоставлены сами себе и занялись аутсорсингом. По академической линии использовали всевозможные западные гранты, делали совместные проекты с Францией, Германией, Соединенными Штатами и другими странами. Сотрудничали с компаниями Intel, Hewlett Packard, они давали нам работу не рутинную – мы решали задачи, связанные с разработкой новых технологий. Обидно, что права интеллектуальной собственности на то, что мы тогда делали, в значительной степени принадлежат им.

Таким образом, у нас в ИСП РАН было два встречных потока – контракты с западными компаниями и поток

талантливой молодежи, которую мы учили на реальных проектах по физтеховской модели.

– А что происходит сегодня?

– Появилось достаточно много заказов и от государства, и от крупных отечественных компаний. Так что мы можем выбирать, чем заниматься, и выбор всегда очевиден – делать что-то новое, не рутинное. В общем, можно сказать, бог миловал – удалось избежать стиля индийского аутсорсинга.

Из потрясений последних лет – реорганизация Академии наук: в 2013 году у нее были отобраны все институты, и их передали в ведение Федерального агентства научных организаций (ФАНО). Люди там нормальные, мы с ними наладили контакт и работаем. Но придумывается очень много новых организационных форм, и существует большая степень риска, что в результате реорганизаций могут быть угроблены академические институты, как это происходит с университетами, со средним образованием. Сплошные реорганизации ведут к тому, что мы теряем немногое оставшееся, чем могли гордиться. Реформаторы очень небрежно ведут себя в таких деликатных областях, как наука, образование. Они могут нормально развиваться при достаточном финансировании и с большой долей свободы, творчества. А сейчас идет жесткая регламентация, не знаю, чем это закончится. Если судить по среднему образованию, то печально. Дошло до того, что Путин просит патриарха возглавить Общество русской словесности. Настолько отчаянное положение сложилось в школе не только с математикой, но и с русским языком, раз принято такое решение. Но до этого положения нужно же было сначала дойти!

– Напоминает 1920-е, когда в результате реформ выпускники школ показали настолько низкий уровень знаний, что пришлось в начале 30-х возвращаться к дореволюционной гимназической модели.

– Остается надеяться, что мы до такого возврата тоже доживем. Важно понимать, что образование – очень деликатная область, как я уже говорил. Раньше ученая степень профессора пользовалась огромным уважением. Сегодня общественное мнение настраивается на другие приоритеты, к педагогам начинают относиться как к людям второго сорта. Не в последнюю очередь за счет сокращения финансирования. Но ведь нельзя все мерить только деньгами – это катастрофа. Есть много разных стимулов, и деньги – лишь один из них.

Исправляется все тоже неуклюже. Не видно бережливости к людям, которые должны создавать интеллектуальную, техническую элиту. Даже неплохо, если их предоставят самим себе – история института показывает, что можно не просто выжить, а нормально жить, создавать высокотехнологичные продукты и при этом приносить доход стране. В 90-х ИСП РАН ежегодно платил налогов больше, чем получал денег от государства. Примерно в том же духе мы и продолжаем.

– Какие направления работы сейчас приоритетны в вашем институте?

– Наши текущие проекты связаны с проблемами, которые существуют сегодня. Если классифицировать, это в первую очередь направление информационной безопасности. На моей жизни размер программ вырос на несколько порядков, сложность стала невероятная, усложнился анализ, нужно разрабатывать новые технологии, чтобы программы были надежными, менее уязвимыми. Это связано с задачами операционных систем реального времени, с обработкой больших данных, анализом текстов, со многими направлениями, о которых мы написали в Уставе, по ним институт и продолжает работать. Хотя сейчас время смутное, очень нестабильное. Я уже говорил, но еще раз повторю: когда государство делает попытки регламентировать все, это крайне опасно. В таких областях, как наука, образование, технологии, необходимо больше свободы, если мы хотим, чтобы в стране шло развитие.

– Что, на ваш взгляд, надо менять в первую очередь?

– Импортозамещение принимает иногда чудные формы, как обычно у нас происходит. Ведь это длительный процесс, на него требуются годы. Для развития технологий, для разработки ПО нужно все в комплексе, и образование в том числе. В это нужно вкладывать все ресурсы в первую очередь. Это области небыстрые и требующие прежде всего бережного отношения. А не высасывания из пальца новых организационных структур или придумывания лозунгов. К сожалению, государственный процесс двигается в ту сторону.

Исправлять надо и частности – некоторые законы. Например, ФЗ-44 о закупках стал сущим кошмаром – компьютеры покупать надо, но создаются невероятно сложные проблемы, которые приходится преодолевать. И надо что-то менять в подходе к сфере технологий и образования в ИТ. Повторю – нужен бережный подход и годы на выращивание этих людей.

Конечно, в Академии наук накопились серьезные проблемы, они начались раньше, еще в советское время. Рецептов не имею, все достаточно сложно. Но я не уверен, что резкие движения – то, что надо. Потому что такого рода авралы, как патриарх во главе Общества русской словесности, – это уже ход отчаяния. Другая крайность: поняли, что упустили подготовку инженеров, кадров не хватает, тут же новая идея – давайте перестанем готовить гуманитариев, а будем готовить только инженеров. Такие решения ничего хорошего не дают. Все нужно взвешивать очень сильно. А то реформы где-то разрабатываются, потом ты их видишь и за голову хватаешься – как избежать последствий этого. С нами не советуются, приходят распоряжения, скажем, из Минобрнауки, и мы узнаем обо всем, когда все уже решено.

– У ИСП РАН все еще есть та самая свобода? Вам удалось вырастить уже не одно поколение ИТ-специалистов...

– Академические институты могут многое. Сейчас мы имеем право, например, создавать компании хоть за рубежом, никто не запрещает, раньше это было запрещено. Но критическая масса доведена до минимума. Если мы выделим группы для работы в отдельных компаниях, то кто будет учить следующее поколение, кто будет их готовить? Я чувствую большую тревогу. Вообще научное образование и разработки – все перемешано. Мы по мере сил этим занимаемся в ИСП РАН, на ВМК МГУ, в Физтехе, проводим конференции, выездные школы, семинары. Стараемся передавать наши знания, создавать экосистему для работы и учения. Хочется надеяться, что мы сможем эффективно продолжать нашу работу. EOF

Визитка

ВЛАДИМИР ГАКОВ, писатель-фантаст, лектор. Окончил физфак МГУ. Работал в НИИ. С 1984 г. на творческой работе. В 1990-1991 гг. – Associate Professor, Central Michigan University. С 2003 г. преподает в Академии народного хозяйства. Автор 8 книг и более 1000 публикаций

Семьдесят лет компьютерной эры

Хроники: XX век – начало

Продолжаем публикацию хроник возникновения, становления и развития информационных технологий*

Режим реального времени

Век начался событием принципиальным и удивительно «угадавшим» со временем (если так можно говорить о событии): в 1901 году итальянский радиотехник Гульельмо Маркони впервые успешно передал радиосообщение через Атлантику. Целых два века до того коммуникации между Старым и Новым Светом обеспечивали исключительно парусники. И лишь под самый занавес – пароходы...

А в следующем 1902 году появился первый радиотелефон, а еще спустя два года – первая диодная лампа Джона Флеминга и первый же телефонный автоответчик.

В том же 1902-м по телеграфу передали и перво е фотоизображение. Что произвело настоящую революцию в СМИ: теперь репортеры могли не только оперативно доложить в свои редакции о каком-то экстренном событии, но и быстро выслать фото – хотя бы из другого города и даже чужой страны.

Ученый и писатель Александр Богданов (Малиновский), создатель науки тектологии

Особо удачным для информационных технологий выдался 1906 год. Была проведена первая радиотелефонная передача, Джей Мортон и Чарлз Крам создали прототип телекса, а американский изобретатель Ли де Форест получил патент на новый прибор, которому предстоит стать основой будущей электроники, – триод.

Но самое удивительное изобретение, все перспективы которого в ту пору, к сожалению, оценить не смог никто, продемонстрировали инженеры компании Bell Telephone. Они построили аппарат, который многие историки науки считают первым прототипом «электронного мозга» – если не относиться слишком буквально к обоим составляющим этого термина (одинаково популярного среди писателей-фантастов и публицистов, писавших о перспективах кибернетики). Сами изобретатели окрестили свое устройство скромнее – «регулировщиком», поскольку оно позволяло удерживать в цепи электроимпульсы на то время, пока переключатели не установят устойчивую связь. Годом позже, в 1907-м, русский изобретатель Борис Розинг создал – пока, правда, в чертежах – первую систему телевидения с электронно-лучевой трубкой. Оттуда пойдут все мониторы будущих компьютеров, пока на самом излете века их окончательно не сменят экраны на жидких кристаллах...

Переход в виртуальный режим

В первую декаду века, как легко догадаться, еще продолжала «раскачиваться» и молодая научная фантастика – во всяком случае, та, которая нас сейчас интересует. Однако редкие прогностические попадания фантастов дорогого стоили.

Взять хотя бы появление научно-фантастического кино! В 1902 году на экраны вышла более чем вольная экранизация Жюля Верна – фильм «Путешествие на Луну» его соотечественника, неутомимого фантазера и выдумщика Жоржа Мельеса. Сейчас его «спецэффекты» могут вызвать только снисходительную улыбку, однако не пройдет и полувека с небольшим, как в кино придет компьютер и все перевернет в наших традиционных представлениях о том, каким может – каким должно быть зрелище! А закрыл первое десятилетие фильм, снятый на киностудии Томаса Эдисона и положивший начало целой серии картин, не утративших популярности и в конце века. Это был первый из кинематографических «Франкенштейнов» (1910)…

Что касается литературной научной фантастики (которую, кстати, тогда еще так никто не называл), то в указанное десятилетие для нас представляют интерес два романа. Во-первых, это социальная утопия видного деятеля русской социал-демократии, ученого и писателя Александра Богданова (Малиновского) «Красная звезда» (1908). В ней автор описал не только будущую социалистическую утопию

на Марсе, но и многое другое, что имеет прямое отношение к теме разговора. А именно: кинематограф (звуковой и стереоскопический), пишущие машинки, способные записывать текст под диктовку, а также – внимание! – цифровые машины, обслуживающие производство, и бортовой компьютер на марсианском межпланетном космическом корабле. Описание последнего, к сожалению, дано слишком бегло и обще: «непонятные машины с множеством циферблатов и стрелок». Чего не скажешь о поразительной догадке насчет перфолент, «тянувшихся из самой большой машины» и «содержавших результаты вычислений»!

Что же касается «компьютерного производства»... В романе-продолжении «Инженер Мэнни» (1913) Богданов описал грандиозные «АСУ», управляющие марсианскими заводами и фабриками, «компьютеризированную» систему учета, маркетинга, планирования (поневоле приходится переходить на сегодняшние термины!) и многое другое.

Впрочем, удивляться нечему: это писал будущий создатель непонятой современниками науки тектологии. В своем двухтомном труде «Всеобщая организационная наука» (вышел в том же году – 1913-м) Богданов додумался и до принципа обратной связи, и до идеи математического моделирования (в том числе и экономических процессов), и до системного подхода и прочих азов современной кибернетики...

Предсказания Хьюго Гернсбека

И, наконец, в романе «Машина останавливается» (1909) английский писатель (на сей раз никакой не «фантаст», а самый что ни на есть кондовый реалист) Эдуард Морган Форстер описывает классический образ «машинной» утопии, которая, как и следовало ожидать, в результате обернулась самой настоящей антиутопией. Человечество, полностью доверившись гигантской Машине (в романе она непременно пишется с заглавной буквы), переложив на ее механические плечи все проблемы экономики, медицины, комфорта и благосостояния, превратились в придаток своего механического детища. И не только в переносном смысле, но буквально! Люди могут виртуально посещать любое место на планете, не покидая своих квартир-клетушек, и обмениваться сообщениями с любым собеседником. Но беда в том, что опутанные проводами вездесущей Машины обитатели, по сути, комфортабельной «добровольной тюрьмы» уже не испытывают желания (а впоследствии им это и категорически запрещается – все той же Машиной!) выбраться на свежий воздух и непосредственно насладиться видами и ароматами дальних стран. Написано более века назад...

Режим реального времени

В 1912 году мир потрясла трагедия «Титаника». Корабль представлял собой последнее слово техники и официально считался непотопляемым. Таким же гордым до беспечности, опьяненным собственным техническим всемогуществом входило в двадцатое столетие человечество. Ему «айсбергов» ждать оставалось недолго... Между прочим, жертв катастрофы могло бы стать еще больше, если бы не посланный по радио сигнал о помощи. По одной из версий сигнал с тонущего «Титаника» первым поймал в США выходец из России, страстный энтузиаст радио Дэвид Сарнов. По крайней мере создатель (в 1920 году) и впоследствии президент Американской радиовещательной корпорации (RCA), сам активно распространявший эту легенду, видел в ней символ наступавшего столетия: если что и спасет человечество, то новейшие средства коммуникации!

Второе десятилетие прошлого века еще не представило решающих доказательств того, что сегодня является неоспоримым фактом: XX век – это прежде всего век массовых коммуникаций, век информатики. Открытия в этой сфере пока не повалили лавиной – ждать этого, впрочем, оставалось недолго; и никто еще не смог бы рассмотреть на горизонте явления науки XX века – кибернетики.

Но зато многие технические новшества успели покинуть недоступные простым смертным лаборатории ученых и уверенно прописались в жилых домах, превратившись в предметы быта. Среди таковых – телефон, переставший будоражить воображение обывателей. В 1915 году была установлена устойчивая радиотелефонная связь через Атлантику, и состоялся первый трансконтинентальный телефонный разговор с США. Двумя годами раньше американец Эдвин Хауард Армстронг получил патент на метод усиления радиосигналов с помощью открытого эффекта «регенерации», а под конец декады (1918) он же создал схему супергетеродинных радиоприемников. В 1919 году были начаты эксперименты с коротковолновым радио. Тогда же состоялся дебют «беспроволочного телефона», первыми абонентами которого стали авиапилоты. Годом позже в Англии установили первый публичный телефон-автомат, и началось коммерческое радиовещание в Штатах.

А на подходе были иные диковины молодой науки электроники. Уже упоминавшийся Борис Розинг впервые осуществил передачу сигналов по телевизионной системе с электронно-лучевой трубкой (1911). А в 1919 году эмигрировал в США талантливый ученик Розинга, чье имя будет также связано с рождавшимся на глазах телевидением, – Владимир Зворыкин... Годом позже неутомимый американец Ли де Форест создал новую систему звукозаписи («фонофильм»), а еще через год была записана первая джазовая пластинка.

В год гибели «Титаника» открылась еще одна «кузница кадров» будущей американской электроники: Институт радиоинженерии (Institute of Radio Engineers - IRE). Наконец, под самый занавес десятилетия, в 1919 году, В.Х. Эккльз

и Ф.У. Йордан создали электросхему типа flip-flop (в переводе с английского это значит «качели», «сальто-мортале»): два усилителя, вход первого соединен с выходом второго и наоборот... Подобными простейшими элементами электронной памяти сегодня снабжены не только обыкновенные настольные лампы, включаемые и выключаемые нажатием одной и той же кнопки, но и блоки памяти (RAM) всех современных компьютеров.

Переход в виртуальный режим

Второе десятилетие века ознаменовано появлением всего двух произведений научной фантастики – из интересующей нас «информационной» части спектра. Но каких!

Роман под невообразимо тоскливым названием «Ральф 124 С 41+» (1912) американского издателя и редактора Хьюго Гернсбека сегодня не назовет шедевром даже самый фанатичный поклонник научной фантастики. Но зато какая россыпь технических прогнозов – один удачнее другого! Ограничусь перечислением только тех, что нас интересуют. Флюоресцентные светильники, световая реклама в небе, музыкальные автоматы, звукозапись на магнитных лентах, громкоговорители, микрофильмы, телевидение, радиосети. И впечатляющее описание – с приведенными чертежами! – «радиодальномера». Иначе говоря, радара.

Гернсбек был профессиональным изобретателем и знал, о чем писал. Среди его достижений – популяризация телевидения (он впервые упомянул это слово на английском!), патент на первый американский домашний радиоприемник и предвиденный, но, увы, не запатентованный радар... Хотя главным изобретением инженера и популяризатора науки стал, конечно, первый в мире журнал научной фантастики – Amazing Stories, успешно стартовавший в 1926 году. Не говоря о самом словосочетании science fiction, которое ввел в оборот также Гернсбек!

А вторым произведением стала пьеса «R.U.R.» (1920) чешского писателя Карела Чапека, давшая миру слово «робот». Вкупе со всеми проблемами, которые будут волновать эту литературу не одно десятилетие... Любопытно, что в год ее выхода в еврейском местечке на далекой Смоленщине родился мальчик, который в трехлетнем возрасте будет увезен родителями в Америку, станет там знаменитым фантастом и, в частности, подхватит эстафету у Чапека, изобретя науку о роботах – «роботехнику». Айзек Азимов (1920-1988)...

Режим реального времени

И все это время люди в разных странах занимались политикой, увлекались джазом (откуда и пошло название «свингующие двадцатые»), совершали революции в искусстве и науке, улучшали свой быт и восторгались техническим прогрессом. И не замечали, как назревает зловещий нарыв. Лопнул он в предпоследний год декады – в «черные» четверг, пятницу и понедельник (24-29 октября) 1929 года, когда грохнулась Нью-Йоркская фондовая биржа.

Мир потряс невиданный экономический кризис, прокатившийся по планете подобно сейсмической волне. Биржи лопались и раньше, но тогда не было радио и телефона, благодаря которым за считанные дни через руки брокеров прошло столько акций, сколько их, наверное, не выпустили за все предшествующие времена... Так новые средства коммуникаций впечатляюще доказали, что мир стремительно глобализуется – нравится это кому-то или нет!

Третье десятилетие века ознаменовано быстрым и неудержимым развитием телевидения, у истоков которого, как уже известно читателям «СА», стояли пять главных фигур: шотландец Джон Лоджи Бэйрд, немец Пауль Нипков, перебравшиеся в Америку русский ученый Владимир Зворыкин, англичанин Чарлз Дженкинс и американец Фило Фарнсуорт.

Не отставали и другие информационные технологии. В 20-е годы особенной популярностью пользовались так называемые кристаллические радиоприемники, основу которых составлял полупроводниковый кристалл. Настройка осуществлялась с помощью тончайшего провода («кошачий ус»), которым на кристалле нащупывалась точка, обеспечивавшая проходимость радиоволн. В США началось радиовещание на средних частотах (1921); в том же году на радио появилась первая платная реклама. В 1924-м в Колумбийском университете была организована первая радиотрансляция образовательных программ. К тому времени в пользовании американцев находились два миллиона радиоприемников. А в Англии к середине десятилетия было установлено полмиллиона телефонных аппаратов.

Уже упоминавшийся Армстронг начал эксперименты по созданию передающих систем с частотной модуляцией (1925). В том же году был образован Исследовательский центр Bell Laboratories, а двумя годами позже – радиовещательная компания Columbia Broadcasting Company. И тогда же BBC начинает регулярно транслировать музыкальную программу, ведущий которой – Кристофер Стоун – стал первым «диск-жокеем» в истории! Спустя два года американец Гарольд Блэк с помощью принципа отрицательной обратной связи решил проблему снятия искажений при усилении сигналов. И состоялся первый трансатлантический телефонный разговор.

Десятилетие ознаменовано и первыми записями в «компьютерную летопись». В 1924 году уже упоминавшаяся компания Германа Холлерита окончательно поменяла название на International Business Machines (IBM). А годом позже американец Ванневар Буш построил в Массачусетском технологическом институте (MIT) первый аналоговый компьютер и под самый занавес декады начал работу по созданию дифференциального анализатора.

Переход в виртуальный режим

В то десятилетие мысль фантастов пока еще в значительной мере двигалась параллельно мысли изобретателей, хотя и были зафиксированы первые примеры решительного «обгона»!

Если в утопии классика Герберта Уэллса «Люди как боги» (1923) описано «просто» телевидение – хотя и с незначительным опережением, то в рассказе некоего Д. Шлосселя «На Луну с Прокси» (1928) описано уже нечто исключительное для третьего десятилетия века: изобретатель-инвалид посылает на Луну своего «радиотелемеханического двойника»! Обратите внимание на его имя – Прокси…

Джек Уильямсон в рассказе «Космический экспресс» (1930) рассуждает о том, что вестерны и «дурацкие драмы» займут львиную долю будущего телевидения. В рассказе Эдмонда Гамильтона «Металлические гиганты» (1926) электронный мозг создает подручных «атомных роботов»

для выполнения конкретных технологических операций. Героиня рассказа Дэвида Келлера «Психофонная няня» (1928) – бизнесвумен недалекого будущего не боится оставить дитя наедине с «железной» нянькой.

Первым же научно-фантастическим произведением о компьютерах – в том смысле, какой мы вкладываем в это слово сегодня, – следует, видимо, считать вышедший в 1928 году рассказ американца Аарона Надела (скрывшегося под вычурным псевдонимом Аммиан Марселлин) «Думающая машина». Автор описывает свое изобретение – «психомашину» (psychomach) – как «устройство, состоящее из сотен тысяч элементов, которые – в различных сочетаниях – могли осуществить все простейшие операции, выполняемые человеческим мозгом».

А в рассказе американцев Лоуренса Мэннинга и Флетчера Прэтта «Город живых мертвецов» (1930) можно с долей фантазии усмотреть то, что сегодня назвали бы «виртуальной реальностью»! Человечество в будущем в буквальном смысле опутано проводами, через которые до людей доходят звуки, картинки, даже запахи окружающего мира. Реального или искусственно созданного самой же Машиной?

Ванневар Буш и первый аналоговый компьютер

Но подобных примеров было кот наплакал. Воображение фантастов занимали иные видения – романтически-инфернальные, «франкенштейновы»! Не случайно самым заметным киберобразом декады стали урбанистический Мир-Машина и дьявольская искусительница-андроид из фильма гения немого кино Фрица Ланга «Метрополис» (1926)... Художников волновали страсти «человекоподобные», а какие эмоции могли испытывать железные ящики с мерцающими лампочками и крутящимися бобинами перфолент! Так что, можно сказать, к научной фантастике компьютер подкрался незаметно...

Режим реального времени

Четвертая декада века открылась событием вполне мирным: в 1931 году британская компания BBC провела первую телетрансляцию финальных заездов со скачек в Дерби. В 1936-м транслировалось уже открытие Олимпийских игр в Берлине; трагическая ирония состояла в том, что одной из первых телевизионных «картинок», посланных человечеством в эфир, была речь, произнесенная на открытии тогдашним главой государства – Гитлером...

А на исходе десятилетия, 1 сентября 1939 года, важное правительственное сообщение – война! – первым прозвучало в Великобритании также с телеэкрана. И в данном случае не обошлось без горькой ухмылки Истории: детский «мультик» про Микки-Мауса, шедший по каналам той же BBC, прервался на фразе кумира детворы: «Кажется, нам пора сваливать...»

Переход в виртуальный режим

Литературная фантастика перестала слепо плестись в кильватере технического прогресса, а смело забегала вперед, высматривая в грядущем «мины», незамеченные учеными и журналистами, восторженно певшими гимн этому самому прогрессу. Чего не скажешь о научно-фантастическом кино: даже в предельно футуристической картине Александра Корды «Облик грядущего» (1935), снятой по роману Уэллса, аэропланов – навалом, есть даже огромные телеэкраны. Зато о компьютерах – ни намека...

Зато телевидение в последнем предвоенном десятилетии стало для фантастов расхожей темой. О том свидетельствуют, например, сразу три рассказа Генри Каттнера – «Лунный Голливуд», «Приговоренный мир» и «Парад звезд» (все вышли в 1938 году), в которых фигурирует телевидение! В рассказе «Дитя эфира» (1939) Теодора Старджона описана особая «электромагнитная жизнь», зародившаяся в... телесети! А в «Свидетеле затмения» (1940) забытого ныне Джона Расселла Фирна описан эксперимент по телетрансляции солнечного затмения с помощью высоколетящего ракетоплана...

Впрочем, фантазия писателей заводила их и дальше. Далекие потомки из рассказа Дэвида Келлера «Церебральная библиотека» (1931) в целях имитации человеческого мозга строят машину, объединяющую пять тысяч банков памяти отдельных «компьютеров». И во всех подробностях описывает гигантскую ЭВМ Джон Кэмпбелл в рассказе «Машина» (1935), не забыв предсказать и неприятное, хотя и очевидное последствие создания искусственного интеллекта: машина в этом произведении развивается настолько, что ей перестают быть нужны... ее создатели! Люди. Они брошены бывшей «служанкой» на произвол судьбы и медленно деградируют, скатываясь к варварству... О том же повествуют и два других рассказа Кэмпбелла, названия которых также символичны, – «Закат» (1934) и «Ночь» (1935).

И в трех произведениях – рассказе американца Лоуренса Мэннинга «Властелин мозга» (1933), знаменитом романе-антиутопии англичанина Олдоса Хаксли «О дивный новый мир» (1932) и романе аргентинца Адольфо Бьой Касареса «Изобретение Мореля» (1940) – нащупана, хотя и робко, тема тем фантастики сегодняшней. Достигшая технического совершенства «виртуальная реальность», в которой человек бьется как рыба в сети, не в состоянии выбраться... Как же давно они все это предвидели, писатели-фантазеры! EOF

** Продолжение. Начало – см. «СА», №1-2, №3, 2016.*

Продолжение следует

В начале было слово

Порой запоминается лишь оно, но изреченное впервые. Некоторые авторы оставили о себе память в виде многотомных собраний сочинений, другие прославились всего-то одной пьесой или парой поэтических строчек. Но встречаются в истории литературы и такие, кто обессмертил себя... одним-единственным словом! Таким стал чешский писатель Карел Чапек. А изобретенным им словом, вошедшим во все языки мира, стало слово «робот».

Впервые оно появилось в тонкой книжке, выброшенной на прилавки пражских магазинов в январе 1920 года. Тираж по тем временам был солидный – 2000 экземпляров. Особенно для пьесы со странным названием, да еще написанным почему-то по-английски: «R.U.R. Rossum's Universal Robots». Какие-то «россумовские универсальные (или вселенские?) роботы»… Что это за «роботы» – ни в одном языке тогда такого слова не было. Тем не менее пьесу спустя год поставили в Пражском национальном театре. И уже к концу жизни – а умер он в 1938-м – автор был знаменит на весь мир.

И все благодаря одному слову. Ну и клубком проблем, которые поднимались в пьесе и с которыми по сей день связана львиная доля научной фантастики. Да уже и наша – реальная – жизнь. Потому что за словом стоял образ, который стал одним из главных символов XX века. Не устаревает он и в веке XXI.

Само слово между тем было придумано как бы походя. И не Карелом Чапеком, а его братом Йозефом, известным театральным художником и книжным иллюстратором. Когда Карелу пришла в голову идея фантастической драмы об искусственных существах, вышедших из-под контроля и восставших против своих создателей, то писатель долго думал, как назвать этих новых «гомункулюсов» технического прогресса. На помощь пришел брат, взяв «с потолка» слово «робот» – от чешского robota. Что переводится не как «труд, работа», а правильнее как «тяжелый, принудительный труд». И даже «барщина»!

Так что Карел Чапек с самого начала создавал драму об искусственных рабах, призванных облегчить человеку жизнь. Рабы же, учит история, остаются безропотными и покорными лишь до поры до времени! Что бесчисленные научно-фантастические роботы не устают доказывать вот уже без малого век. Как и их литературные и мифологические предшественники – те же средневековые гомункулюсы, в пражском же гетто рожденный глиняный истукан Голем, монстр Виктора Франкенштейна, все эти искусственные «куклы» из «ужастиков» француза Вилье де Лиль-Адана, немца Гоффмана, американцев Бирса, По и Мелвилла... Их повсюду разыщешь.

С одной оговоркой. Никаких идей кибернетики все эти литераторы прошлого, разумеется, не предвосхитили: их творения не имели почти ничего общего с реальными роботами современной науки. И, вероятно, науки ближайшего будущего: какими бы совершенными они ни стали, навряд ли мы обнаружим в них хоть какое-то подобие человека. Просто это будет не нужно. Но вот проблему проблем – Человек в роли Создателя – предвидели. Ясно и остро. Заслуга Чапека в том, что к акту создания искусственного существа писатель «привлек» научно-технический прогресс. Между прочим, и фамилия отца и сына Россумов – создателей роботов – с чешского переводится как «разум».

Помнится, и у библейского «конструктора» проблемы тоже начались с создания разумного существа. Раз разум, то уже не раб, не механизм, который слепо подчиняется программе. Поэтому все конструкторы искусственных рабов, которые должны были обеспечить человечеству Золотой век праздности и ничегонеделания, в корне губили свои замыслы, как только снабжали своих «роботов» сознанием, разумом.

Польский писатель-фантаст Станислав Лем – тоже, между прочим, заслуженный «роботехник» литературы! – точно подметил самую суть всей научной фантастики о роботах. Он задал на первый взгляд риторический вопрос: действительно ли они программируемы, эти роботы? Если да, то они неразумны и являются не более чем полезными механизмами, как все ныне существующие промышленные роботы. После чего всякий разговор о «бунте роботов» теряет смысл. Если нет – то есть способны сами себе задать программу, – тогда это уже не роботы, не механические батраки человека, а в полном смысле слова разумные существа. С коими придется считаться и Конструктору.

Пока же с легкой руки Чапека роботы научной фантастики отправились в свой далекий, путаный и многотрудный путь по страницам сотен, если не тысяч произведений, получив благословение «крестника». И кто из любителей научной фантастики хоть на мгновение усомнился, прочитав в одном из таких произведений, написанных много лет спустя, как далекие потомки «россумовских роботов» торжественно клянутся... священным именем Карела! EOF

Владимир Гаков

Системный администратор

Издается с 2002 года

«Системный администратор» включен в перечень ведущих рецензируемых журналов ВАК Минобрнауки РФ

Включен в Российский индекс научного цитирования www.elibrary.ru

Издатель
ООО «Издательский дом Положевец и партнеры»
Адрес редакции
129075, г. Москва, Шереметьевская ул., д. 85, стр. 2, офис 405, тел.: (499) 277-12-41, факс: (499) 277-12-45
Сайт журнала: www.samag.ru

Отпечатано в типографии
ООО «Периодика»Тираж 17000 экз.

Системный администратор

ежемесячный журнал www samag ru

Подписывайтесь и читайте «Системный администратор»!
Подписывайтесь и получайте советы от практиков ИТ!
Подписывайтесь и применяйте знания, которых нет в учебниках! «Системный администратор» дает их вам!

Только полезная
информация

Бумажная + электронная версии!

5440 руб.

Стоимость печатной версии – 4680 руб.
Стоимость электронной версии – 1800 руб.

samag.ru/subscribe

Подпишитесь!

KASPERSKY lab

МАЛИНА ДЛЯ АДМИНА

Специальная акция для системных администраторов «Малина для админа»

Регистрируйте новые базовые лицензии **Kaspersky Internet Security** для всех устройств, **Kaspersky Endpoint Security для бизнеса** (Стартовый, Стандартный, Расширенный) и **Kaspersky Total Security для бизнеса** с защитой до 249 узлов, получайте баллы каждую пятницу и выбирайте подарки! И не забывайте рассказывать об акции своим друзьям-сисадминам! Ведь новые участники, указавшие при регистрации в акции промо-код **OF823**, получают на свой счет 100 приветственных баллов. Эти баллы активируются после одобрения первой лицензии, отвечающей условиям акции.

По просьбам более 3 тысяч постоянных участников акции «Малина для админа» мы продлили акцию до **31 декабря 2016 года**.

MALINA.KASPERSKY.RU