1 Вводная информация

PowerShell используется в IT-индустрии уже более 15 лет, и за этот срок он прошел невероятный путь развития. Если вы пропустили вступительную часть, то повторю: на сегодня PowerShell является кросс-платформенным инструментом, то есть доступен не только в Microsoft Windows. До сих пор не верится, что Microsoft решила открыть его код. Изначально он создавался для решения конкретной задачи по автоматизации административных задач Windows, но для этого хватило бы более простого языка обработки «пакетных файлов». Создатель PowerShell, Джеффри Сновер (Jeffrey Snover), вместе со своей командой разработки смотрел на ситуацию гораздо масштабнее. Они хотели получить инструмент, который будет интересен разнообразной аудитории. По их мнению, администраторы могут начинать с простого использования команд для быстрого выполнения административных задач — как раз об этом наша предыдущая книга, «Изучаем PowerShell за месяц, занимаясь один час в день». Кроме того, разработчики PowerShell представили возможность автоматизации более сложных задач и процессов с помощью различных скриптов, чему посвящена уже эта книга.

Команда PowerShell также предполагала, что разработчики будут использовать их инструмент для создания новой функциональности, о чем мы тоже поговорим. Аналогично тому, как у вашей микроволновки наверняка есть кнопки, которые вы никогда не нажимали, в PowerShell есть инструменты, которые вы никогда не использовали, поскольку они вам не нужны. Но, прочитав эту книгу, вы сможете освоить самую сложную функциональность этой оболочки: скриптинг или — если вы согласны с нашим видением — *создание инструментов*.

1.1. ЧТО ТАКОЕ СОЗДАНИЕ ИНСТРУМЕНТОВ

По нашему опыту, многие подходят к скриптингу PowerShell так же, как подошли бы к работе с пакетными файлами, VBScript, Python и т. д. — и в этом нет ничего плохого. PowerShell позволяет использовать множество различных стилей

и подходов. Но в итоге вы просто усложняете себе жизнь до тех пор, пока не разберетесь в том, как эта оболочка *хочет* работать. Мы считаем, что PowerShell предназначен именно для создания инструментов.

PowerShell можно использовать для разработки универсальных, не зависящих от контекста *инструментов*, которые называются *командами*. Последние обычно выполняют какую-то небольшую задачу и делают это очень хорошо. В отдельных случаях одной команды может быть недостаточно, но PowerShell разработан так, что позволяет упростить «связывание» нескольких команд. Играть с одним кирпичиком «Лего» вряд ли будет весело, но, когда у вас есть целая коробка деталей, собирать конструктор может быть невероятно увлекательно. Именно этот подход мы используем в данной книге и именно поэтому описываем его как *создание инструментов*. Лучше всего вкладывать свои силы и время в разработку небольших автономных инструментов, которые можно будет соединять с другими. Такой подход делает код пригодным к использованию во множестве ситуаций, не только экономя ваше время, но и сокращая издержки, связанные с отладкой и сопровождением.

Скриптинг с помощью PowerShell подразумевает написание последовательностей команд и инструкций в текстовом файле, обычно с расширением `.ps1`. Эти скрипты, по сути, являются программами, написанными на языке PowerShell, предназначенном для автоматизации задач и управления конфигурацией в системах Windows. Скрипты PowerShell можно использовать для решения широкого спектра задач, начиная с выполнения административных заданий и заканчивая автоматизацией сложных рабочих потоков.

Давайте рассмотрим, чем различаются скриптинг PowerShell и работа с командами в консоли PowerShell.

- *Возможность повторного использования*. В скрипте PowerShell можно определять набор инструкций или функций, доступных для повторного использования в разных задачах. Это позволяет составлять модульный и удобный в сопровождении код. Если же вы работаете в командной строке через консоль PowerShell, то зачастую вводите команды интерактивно и возможность их повторного использования ограничена историей ввода или ручным копированием и вставкой.
- *Структура скрипта*. Скрипты PowerShell имеют структурированный формат и содержат такие элементы, как переменные, циклы, условия и функции, поэтому позволяют решать более сложные задачи. Использование же командной строки в консоли PowerShell обычно подразумевает ввод единичных команд, что затрудняет управление в случае сложных операций.
- *Автоматизация*. Скрипты PowerShell прекрасно подходят для автоматизации. Прописывая последовательности команд, вы можете автоматизировать повторяющиеся задачи, выполнять масштабные операции и планировать запуск скриптов в назначенное время. Такого уровня автоматизации сложно добиться интерактивным использованием команд в консоли.

- *Интерактивность.* Работая в консоли PowerShell, вы можете интерактивно вводить команды и видеть непосредственные результаты. Скрипты же, напротив, обычно не интерактивны и выполняют серию команд без вмешательства пользователя. Тем не менее их можно разрабатывать так, чтобы пользователь запрашивал ввод данных или получал параметры. Это увеличивает гибкость скриптов.
- *Политика выполнения скриптов.* К скриптам PowerShell могут применяться политики выполнения, определяющие возможность их запуска. Такие политики помогают исключить случайное выполнение вредоносных скриптов. При работе с командами в консоли по умолчанию аналогичной политики выполнения нет, поскольку каждая команда выполняется отдельно.
- *Обработка ошибок.* Скрипты PowerShell могут включать в себя механизмы обработки ошибок, позволяя управлять ошибками и исключениями. При вводе команд в консоли возможности обработки ошибок более ограниченны, и в случае сбоя зачастую приходится прибегать к ручному вмешательству или отладке.

Таким образом, скриптинг в PowerShell — это создание повторно используемых, структурированных последовательностей команд, призванных автоматизировать задачи. В свою очередь, работа с командами в консоли PowerShell носит более интерактивный характер и обычно подходит при решении разовых задач. Благодаря скриптам PowerShell системные администраторы и IT-специалисты получают мощный инструмент, позволяющий оптимизировать и автоматизировать задачи по управлению Windows.

1.2. ПОДХОДИТ ЛИ ЭТА КНИГА ВАМ

Книга предназначена для тех, кто уже умеет работать с PowerShell из командной строки и создавать повторно используемые скрипты. Основное внимание в ней уделяется не только самому процессу, но и синтаксису, поэтому хорошо, если вы уже умеете писать скрипты и просто хотите улучшить свои навыки или оценить их уровень. Следовательно, эта книга *не является* вводным руководством по PowerShell. Чтобы ваша дальнейшая работа с ней была успешной, попробуйте ответить на следующие вопросы, не раздумывая над ответами слишком долго.

- Какую команду вы бы использовали для запроса всех экземпляров `Win32_LogicalDisk` с удаленного компьютера? (**Подсказка:** если вы ответили `Get-WmiObject`, то имейте в виду, что ваши знания устарели и вам нужно освежить их, чтобы эта книга оказалась для вас полезной.)
- Какими двумя способами PowerShell может передавать данные из одной команды в другую в рамках конвейера?
- Грамотно составленные команды PowerShell не выводят текст. А что они выводят? Какие команды можно использовать, чтобы улучшить внешний вид вывода на экране?

- Как бы вы выяснили варианты правильного использования команды `Get-WinEvent`, если бы прежде с ней не сталкивались?
- Какие политики выполнения скриптов имеются и что означает каждая из них?

Мы не будем приводить ответы на эти вопросы — если вы в них не уверены, значит, вам стоит начать с чтения книги «Изучаем PowerShell за месяц, занимаясь один час в день». После того как вы ее прочтете и выполните приведенные в ней упражнения, можете переходить к нашей книге. Кроме того, мы предполагаем, что вы уже имеете опыт работы с операционной системой Windows, поскольку наши примеры будут иметь отношение именно к ней.

1.3. ЧТО ВАМ НУЖНО ДЛЯ РАБОТЫ С ЭТОЙ КНИГОЙ

Теперь быстро пробежимся по списку всего, что вам потребуется для эффективной работы с книгой.

1.3.1. Версия PowerShell

Мы писали книгу, используя PowerShell 7.2, но 99 % ее материала применимо и к более ранним версиям Windows PowerShell. Скачайте PowerShell по ссылке https://docs.microsoft.com/en-us/PowerShell/. Обратите внимание: не нужно устанавливать новые версии оболочки, если вы предварительно не изучили этот вопрос. Многие серверные приложения (например, Exchange Server) очень привередливы в отношении версии PowerShell, с которой они будут работать, и установка неподходящей может привести к тому, что все выйдет из строя. Кроме того, имейте в виду, что каждая версия PowerShell совместима только с конкретными версиями Windows — для этой книги мы используем Windows 11 и macOS.

Несмотря на использование PowerShell 7.2 (или более новой версии по мере их выхода), примеры из книги будут работать и в Windows PowerShell (5.1), хотя в ней мы ничего не тестировали. Функциональность, о которой мы будем говорить, настолько фундаментальна и стабильна, что не меняется даже после выхода новых версий оболочки. Для знакомства с новейшими возможностями PowerShell, относящимися к ее конкретным версиям, мы используем материалы с сайта https://PowerShell.org. А эта книга полностью посвящена основам, которые всегда остаются неизменными.

1.3.2. Права администратора

Вам потребуется возможность запустить на своем компьютере консоль PowerShell и ваш редактор от имени администратора (что будет показано в меню Пуск). В основном это нужно для того, чтобы вы смогли проработать примеры, которые приводятся в книге. Если вы не знаете, как запустить PowerShell от имени администратора, то, скорее всего, вам будет сложно читать эту книгу.

1.3.3. Редактор скриптов

Наконец, вам потребуется редактор скриптов. В клиентские версии Windows входит Integrated Script Editor (ISE), который будет работать только с Windows PowerShell. Мы советуем удалить его, поскольку команда PowerShell прекратила поддержку этого инструмента с момента выхода Windows 7. На сегодня Microsoft рекомендует бесплатный кросс-платформенный редактор Visual Studio Code (VS Code). Скачайте его по адресу https://code.visualstudio.com/, и в главе 2 мы покажем, как настроить этот редактор для работы с PowerShell.

ПРИМЕЧАНИЕ

VS Code и PowerShell являются кросс-платформенными. Все приведенные в этой книге принципы и практики применимы к PowerShell, работающей не только в Windows, но и в других системах. Тем не менее на момент написания книги примеры выполняются на компьютерах, работающих только под управлением Windows. В связи с этим рекомендуем использовать именно эту операционную систему (ОС). В противном случае вам придется переводить все наши рабочие примеры в форму, которая будет работать в других ОС.

1.4. КАК РАБОТАТЬ С КНИГОЙ

Предполагается, что вы будете читать по главе в день и для этого вам должно быть вполне достаточно одного часа. (Исключением станет специальная бонусная глава, о которой мы скажем отдельно, когда дойдем до нее.) Уделите дополнительное время — возможно, день или два — выполнению упражнений, приведенных в конце некоторых глав. *Не поддавайтесь* соблазну прочесть несколько глав за раз, даже если у вас больше одного часа свободного времени. Поясним почему. Мы будем приводить множество новых фактов, а вашему мозгу требуется время (и сон!), чтобы качественно упорядочить эти факты, связать их с уже известными вам сведениями и начать преобразовывать все это в *знания*. Когнитивисты (ученые, изучающие мышление) выяснили, что человеческий мозг способен усвоить за день определенный объем информации, и при составлении каждой главы мы старались учитывать эти ограничения. Так что просто читайте по одной главе в день. Попробуйте осваивать таким образом хотя бы по три-четыре главы в неделю, чтобы удерживать в сознании нить повествования. При этом не забывайте выполнять упражнения.

СОВЕТ

Рекомендуем повторять главу и приведенные в ней упражнения в течение двух-трех дней. Так вы лучше запомните материал. Не нужно спешить и читать по нескольку глав за один-два дня.

К слову, об упражнениях — *не нужно* просто их пролистывать и переходить сразу к нашим решениям. Опять же, когнитивисты утверждают, что человеческий мозг эффективнее всего усваивает материал, когда не просто узнает новые факты,

а сразу же применяет полученные знания. Вы можете посчитать какое-то упражнение слишком трудным, но благодаря этой сложности ваш мозг сосредоточится и начнет сопоставлять факты. Прежде чем вы захотите найти легкий ответ и откроете наше решение, вернитесь к предыдущим главам и перечитайте их. Такой поиск ответа позволит информации закрепиться. Да, потребуется приложить чуть больше усилий, но в итоге они окупятся. Если же вы выберете ленивый подход, то просто обманете сами себя, чего мы вам точно не желаем.

1.5. ОЖИДАНИЯ

Прежде чем переходить к основному повествованию, мы хотим убедиться, что вы знаете, чего ожидать. Вы уже можете представить, что выбранная тематика довольно обширна и мы могли бы добавить в книгу очень много материала. Но она была написана так, чтобы вы могли прочесть ее за месяц, уделяя этому час в день, и потому мы были вынуждены ограничить ее объем. Мы постарались предоставить базовую информацию, которую необходимо знать каждому, кто хочет начать писать скрипты и создавать простые инструменты PowerShell. Эта книга не задумывалась как всеобъемлющее руководство.

1.6. КАК ОБРАТИТЬСЯ ЗА ПОМОЩЬЮ

Вы можете обращаться на онлайн-форумы Manning, доступные по адресу http://mng.bz/rjgE, но мы советуем использовать и форум PowerShell, расположенный на https://PowerShell.org. Мы отслеживаем разделы вопросов и ответов на этих форумах, но самое главное: там вы найдете сотни единомышленников, которые делятся своими знаниями и тоже просят совета. Если вы работаете с PowerShell, то очень важно, чтобы вы познакомились с коллегами и единомышленниками, а со временем стали полноценным членом сообщества. Ресурс PowerShell.org предлагает видео с различными советами и рекомендациями, бесплатные электронные книги. Кроме того, вы можете принимать участие в ежегодных живых конференциях и использовать множество других возможностей.

ИТОГИ ГЛАВЫ

Надеемся, к этому моменту вы готовы погрузиться в книгу и начать писать скрипты или, что еще лучше, *создавать инструменты*. Вы должны установить и настроить все необходимые программы, а также иметь представление о том, сколько времени сможете уделять чтению каждую неделю.