**Лабораторная работа №3**

**РАБОТА В СРЕДЕ КОМАНДНОЙ ОБОЛОЧКИ  
MICROSOFT POWERSHELL**

PowerShell включена во все версии OC Microsoft Windows, начиная с Vista.

Операционную систему можно представить в виде совокупности ядра системы, которое имеет доступ к аппаратуре и управляет файлами и процессами, и некоторого набора вспомогательных модулей, в том числе оболочки (командного интерпретатора) и утилит, которые позволяют пользователю получить доступ к функциональности ядра операционной системы. Механизм работы оболочек в разных системах одинаков: в ответ на приглашение («подсказку», prompt), выдаваемое находящейся в ожидании оболочкой, пользователь вводит некоторую команду (функциональность этой команды может быть реализована либо самой оболочкой, либо определенной внешней утилитой), оболочка выполняет ее, при необходимости выводя на экран какую-либо информацию, после чего снова выводит приглашение и ожидает ввода следующей команды.

С технической точки зрения оболочка представляет собой построчный интерпретатор простого языка сентенциального (директивного) программирования, в качестве операторов которого могут использоваться исполняемые программы.

Наряду с интерактивным режимом работы оболочки, как правило, поддерживают и пакетный режим, в котором система последовательно выполняет команды, записанные в текстовом файле-сценарии. Оболочка Windows не является исключением, с точки зрения программирования язык командных файлов Windows может быть ***охарактеризован следующим образом***:

* реализация сентенциальной (директивной) парадигмы программирования;
* выполнение в режиме построчной интерпретации;
* наличие управляющих конструкций;
* поддержка нескольких видов циклов (в том числе специальных циклов для обработки текстовых файлов);
* наличие оператора присваивания (установки значения переменной);
* возможность использования внешних программ (команд) операционной системы в качестве операторов и обработки их кодов возврата;
* наличие нетипизированных переменных, которые декларируются первым упоминанием (значение переменных могут интерпретироваться как числа и использоваться в выражениях целочисленной арифметики).

Запуск оболочки осуществляется по одному из трех вариантов:

* нажать кнопку Пуск, открыть Все программы, найти и выбрать Windows PowerShell;
* нажать кнопку Пуск, выбрать пункт Выполнить, ввести имя файла powershell, нажать кнопку ОК;
* в командной строке интерпретатора команд cmd.exe ввести имя файла powershell, нажать клавишу Enter.

После запуска PowerShell открывается командное окно оболочки с приглашением ввода команд (рис. 1).

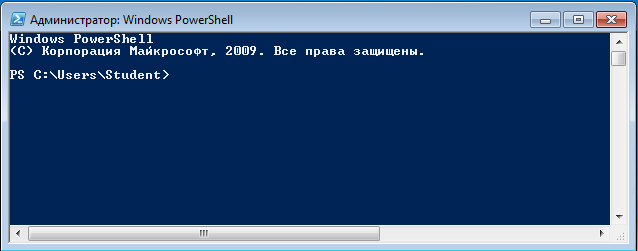


Рис. 1.

В начале сроки приглашения стоят буквы PS, указывающие принадлежность к оболочке PowerShell.

Для выхода из среды PowerShell автономного компьютера следует набрать команду exit и нажать клавишу Enter. Можно просто закрыть окно оболочки, но такой способ не является корректным, т.к. данные проведенного пользователем процесса будут потеряны.

Для того чтобы прочитать встроенную справку для определенной команды, нужно вызвать эту команду с ключом /? (например, xcopy /?).

**Типы команд в PowerShell**

В оболочке PowerShell поддерживаются ***команды четырех типов***: командлеты, функции, сценарии и внешние исполняемые файлы.

**Командлеты**

Имена командлетов всегда соответствуют шаблону «глагол-существительное», где глагол задает определенное действие, а существительное определяет объект, над которым это действие будет совершено. Это значительно упрощает запоминание и использование командлетов. Например, для получения ин формации о процессе служит командлет Get-Process, для остановки запущенной службы – командлет Stop-Service, для очистки экрана консоли – командлет Clear-Host и т. д.

Чтобы просмотреть список командлетов, доступных в ходе текущего сеанса,нужно выполнить командлет Get-Command (рис. 2).

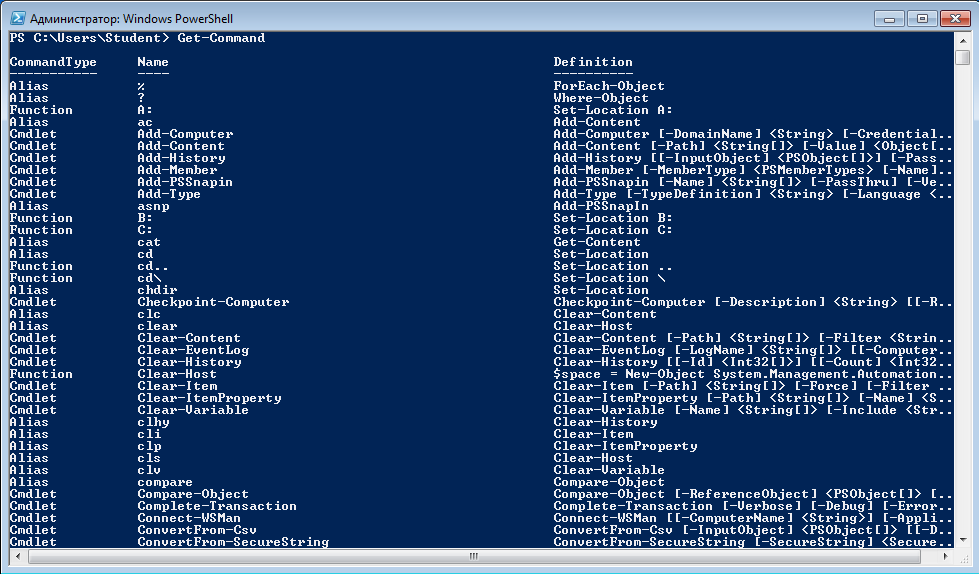


Рис. 2.

Командлеты могут иметь параметры – элементы, предоставляющие командлетам дополнительную информацию. Данная информация либо определяет элементы, с которыми должна работать команда, либо определяет, каким образом будет работать командлет. Параметры командлетов могут быть трех разных типов; обратиться к ним можно по имени, перед которым ставится дефис (-), или по позиции (в последнем случае интерпретация параметра будет выполняться в зависимости от его местоположения в командной строке).

В общем случае синтаксис командлетов имеет следующую структуру:

имя\_командлета –параметр1 -параметр2 аргумент1 аргумент2

Здесь –параметр1 – параметр, не имеющий значения (подобные параметры часто называют переключателями); –параметр2 – имя параметра, имеющего значение аргумент1; аргумент2 – параметр, не имеющий имени (или аргумент).

**Работа с директориями**

Командлет Get-ChildItem отображает содержимое директории. Для краткости команд вместо командлета Get-ChildItem мы будем применять его ***псевдоним*** dir (хотя это и не обязательно).

Пусть при выполнении лабораторной работы №2 создан пользователь Student. Его домашняя директория находится в каталоге **C:\Users**. Чтобы посмотреть содержимое домашней директории пользователя Student, необходимо выполнить команду dir (рис. 3).

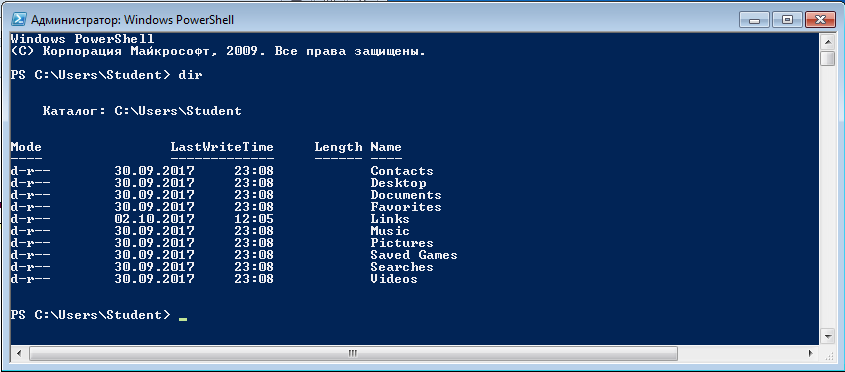


Рис. 3.

В качестве примера переключателя рассмотрим параметр -Recurse командлета   
Get-ChildItem. Переключатель -Recurse, если он указан, распространяет действие команды не только на определенный каталог, но и на все его подкаталоги.

Фрагмент вывода результата использования –Recurse (рис. 4):

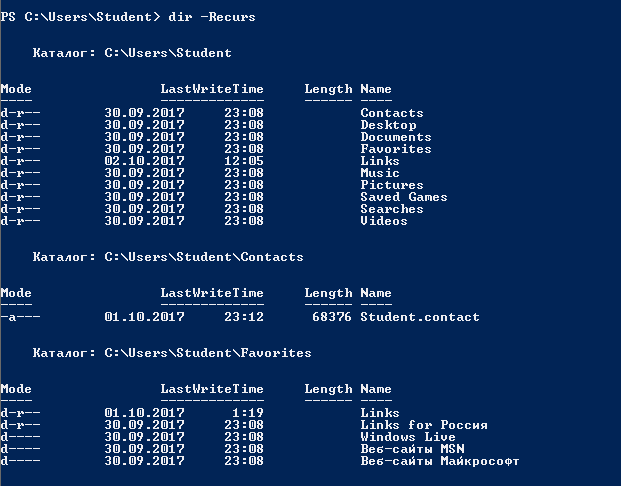


Рис. 4.

Можно задать другой каталог, например, **D:\**. Пусть этот диск представлен каталогами (рис. 5), и в каталоге **myNotes** хранятся текстовые файлы DbConnection\_h.txt и dir.txt (рис. 6). Тогда результат выполнения командлета dir (рекурсивно) представлен на рис. 7.

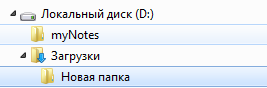


Рис. 5.

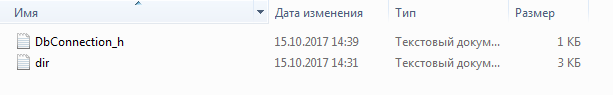


Рис. 6.

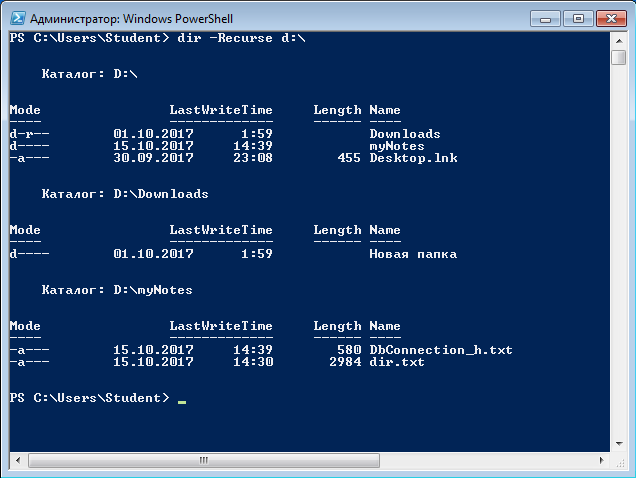


Рис. 7.

Следующий командлет выведет информацию обо всех файлах, которые находятся в каталоге **D:\myNotes** или его подкаталогах и имеют имя, удовлетворяющее маске \*i\*.txt (рис. 8).

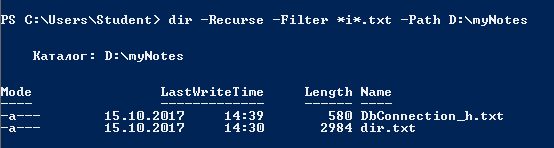


Рис. 8.

Для аргумента **D:\myNotes** имя параметра –Path может быть опущено (рис. 9):

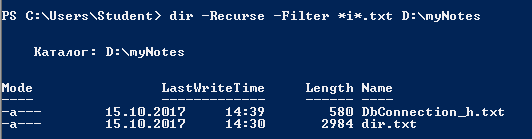


Рис. 9.

Для вывода на экран содержимого текстового файла необходимо воспользоваться командой type. Выведем теперь на экран содержимое файла **D:\myNotes\DbConnection\_h**.**txt**, воспользовавшись для этого командой type (рис. 10):

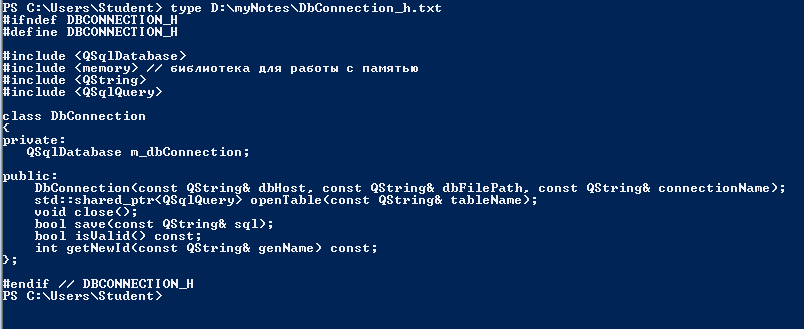


Рис. 10.

**Работа с дисками PowerShell**

Помимо обычных локальных или сетевых дисков файловой системы (C:, D: и т. д.) оболочка PowerShell поддерживает специальные (виртуальные) диски PowerShell, связанные с хранилищами данных разных типов. Например, корневому разделу реестра HKEY\_ LOCAL\_MACHINE соответствует диск **HKLM**:, псевдонимам, доступным в текущем сеансе работы, соответствует диск **Alias**:, а хранилищу сертификатов цифровых подписей – диск **Cert:**.

В отличие от обычных локальных или сетевых дисков файловой системы диски PowerShell доступны только из оболочки PowerShell, обратиться к ним из Проводника Windows нельзя. Имена дисков PowerShell могут содержать более одного символа.

Для получения списка дисков PowerShell, доступных в текущем сеансе работы, нужно воспользоваться командлетом Get-PSDrive (рис. 11):

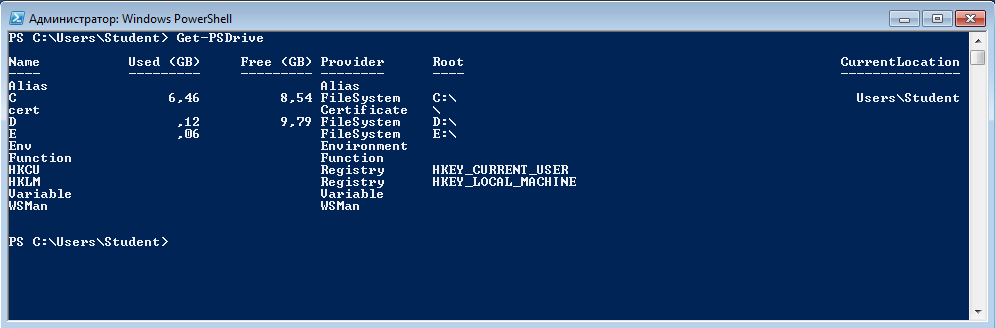


Рис. 11.

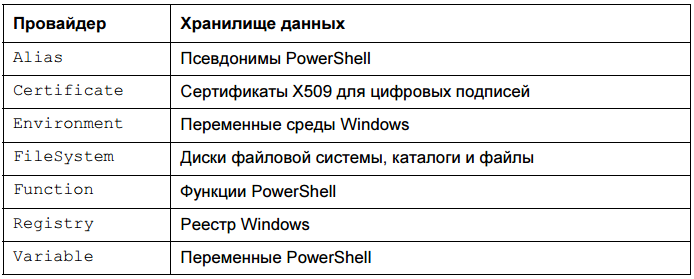
Командлет Get-PSDrive для каждого диска сообщает точное обозначение диска (Root), его имя (Name), имя провайдера (Provider), поддерживающего этот диск, и текущая локализация (CurrentLocation).

**Провайдеры PowerShell**

В оболочку PowerShell по умолчанию включено несколько встроенных провайдеров, которые можно использовать для доступа к различным хранилищам данных (табл. 1).

Таблица 1.

Встроенные провайдеры PowerShell



Дополнительно к встроенным, можно создавать собственные провайдеры PowerShell и устанавливать провайдеры, созданные другими разработчиками (например, для доступа к каталогам **Active** **Directory** или к почтовым ящикам **Microsoft Exchange**).

Для просмотра списка зарегистрированных в оболочке PowerShell провайдеров нужно воспользоваться командлетом Get-PSProvider (рис. 12):

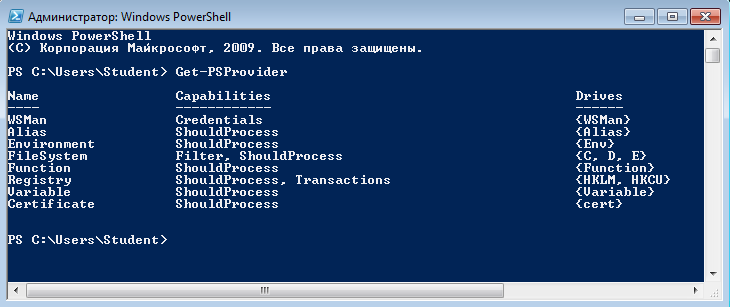


Рис. 12

**Навигация по дискам PowerShell**

Основное назначение провайдеров заключается в том, что они обеспечивают доступ к разнородным данным привычным согласованным образом. Используемая при этом модель представления данных основана на дисках файловой системы.

Предлагаемые провайдером данные можно просматривать и изменять так, как если бы они хранились в виде каталогов и файлов на жестком диске. Навигация по различным дискам PowerShell и просмотр содержимого этих дисков осуществляется с помощью одних и тех же базовых командлетов. Работая с файловой системой, мы используем понятие ***текущего*** или ***рабочего*** каталога. К файлам в рабочем каталоге можно обращаться по имени, не указывая полного пути к ним.

В оболочке PowerShell понятие рабочего (текущего) каталога распространяется и на диски PowerShell. Узнать путь к текущему каталогу можно с помощью командлета Get-Location (псевдоним pwd данного командлета соответствует команде UNIX-оболочек с аналогичной функциональностью) (рис. 13, 14):

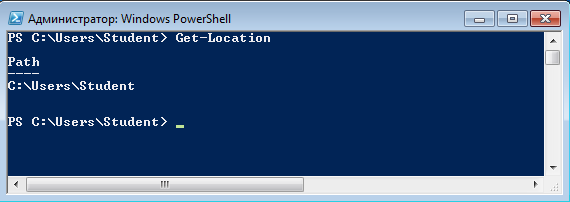


Рис. 13

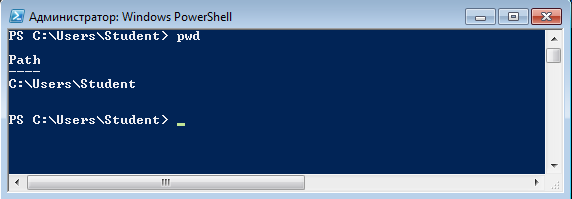


Рис. 14.

Для смены текущего каталога (в том числе для перехода на другой диск PowerShell) используется командлет Set-Location (псевдонимы cd, chdir, sl), рис. 15, 16.

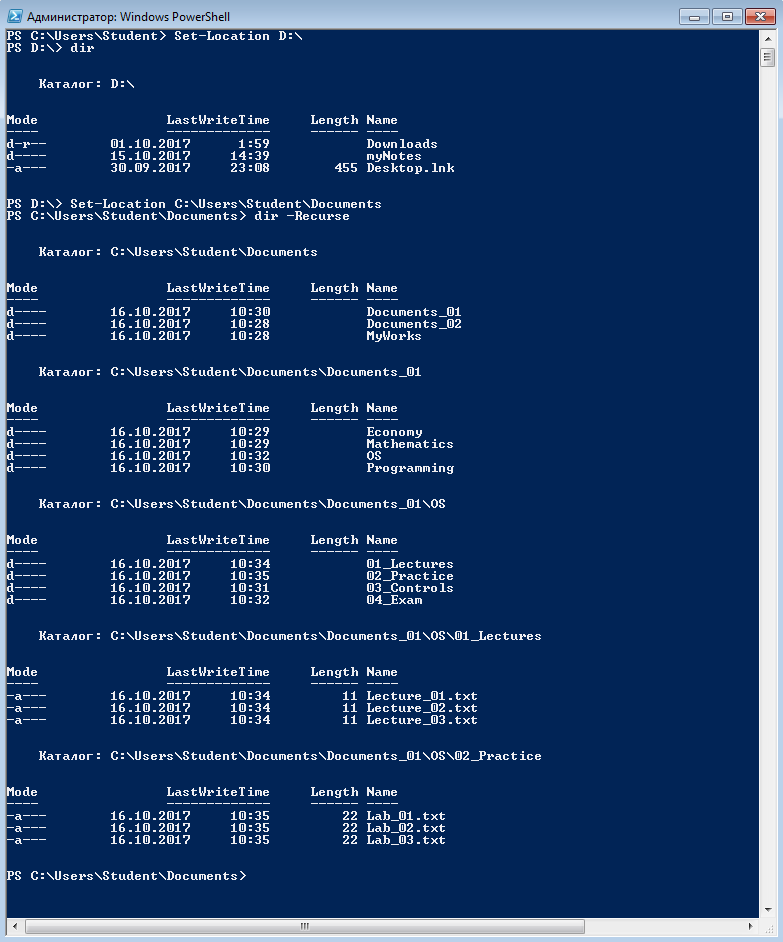


Рис. 15.

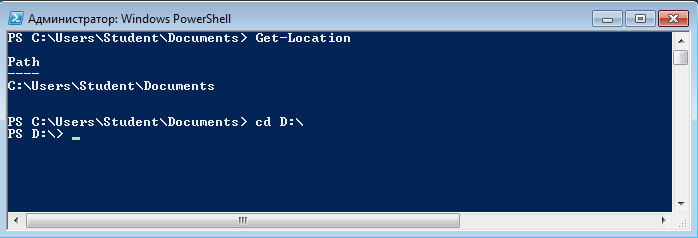


Рис. 16

При вводе командлета Set-Location на экран явно не выводится отзыв о его выполнении. При необходимости можно использовать параметр PassThru, выводящий после выполнения команды Set-Location путь к текущему каталогу (вместо командлета Set-Location в примере используются его псевдонимы cd), рис. 17:

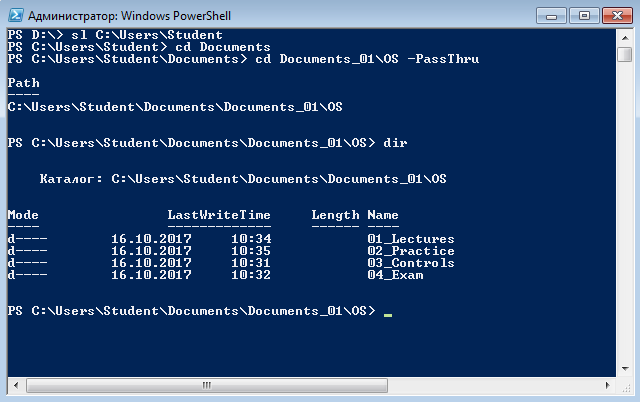


Рис. 17

В оболочке **cmd.exe**, как и в оболочках UNIX-систем, поддерживаются абсолютные и относительные пути. Первые задают полный путь, тогда как вторые указываются относительно рабочего каталога. При этом текущему каталогу соответствует путь . (точка), родительскому каталогу текущего каталога – путь .. (две точки), а корневому каталогу текущего диска – путь \ (обратная косая черта). В PowerShell данная нотация сохраняется (рис. 18, 19).

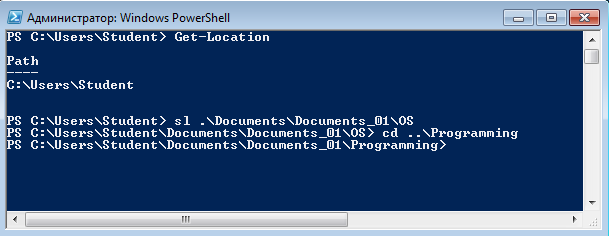


Рис. 18.

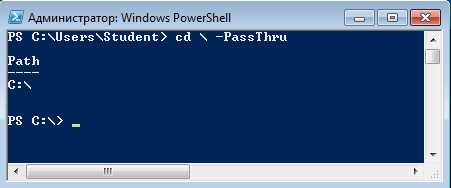


Рис. 19.

**Просмотр содержимого дисков и каталогов**

Для просмотра элементов и контейнеров, находящихся на определенном диске PowerShell, можно воспользоваться командлетом Get-ChildItem (псевдонимы dir и ls). На рис. 20 представлен результат навигации по каталогам с последующим отображением содержимого каталога **C:\Users\Student\Documents\Documents\_01**.

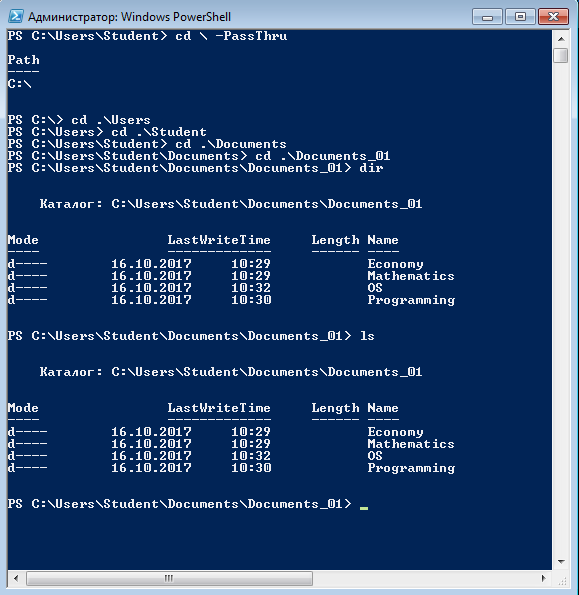


Рис. 20.

**Создание дисков**

Помимо использования стандартных дисков PowerShell, с помощью командлета New-PSDrive можно создавать собственные пользовательские диски. Для этого в командлете New-PSDrive нужно указать три параметра: -Name (имя создаваемого диска PowerShell), -PSProvider (название провайдера, например FileSystem для диска файловой системы или Registry для диска, соответствующего раздела реестра) и путь к корневому каталогу нового диска. Например, можно создать диск для определенной папки на жестком диске для того, чтобы обращаться к ней не по «настоящему» длинному пути, а просто по имени диска. Создадим диск PowerShell с именем **MyPrograms** в каталоге **C:\Users\Student\Documents\Documents\_01\Programming** (рис. 21):

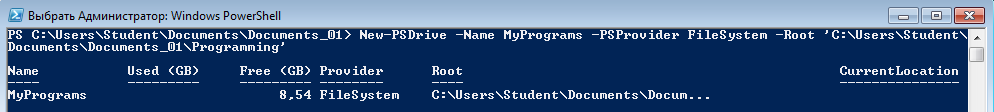


Рис. 21.

Посмотрим содержимое каталога **C:\Users\Student\Documents\Documents\_01\Programming**, используя полный путь к каталогу, и используя имя диска, только что созданного и ассоциированного с этим каталогом (рис. 22):

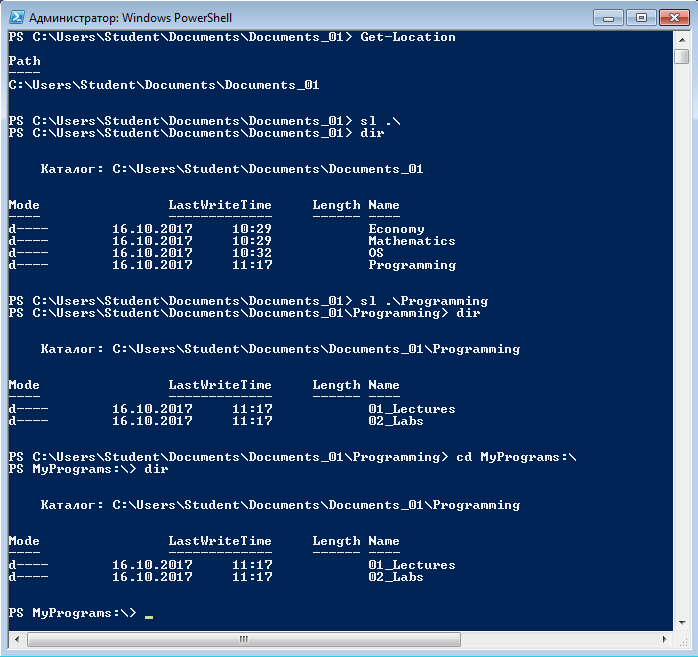


Рис. 22.

**Получение списка файлов и каталогов**

Напомним, что в PowerShell можно использовать команду dir, которая является псевдонимом командлета Get-Childitem. Если выполнить эту команду без параметров, то будет выведено содержимое текущего каталога.

В пути, который указывается для команды dir, можно применять подстановочные знаки. Например, следующая команда выведет все файлы с расширением log из каталога **с:\windows** (рис. 23):

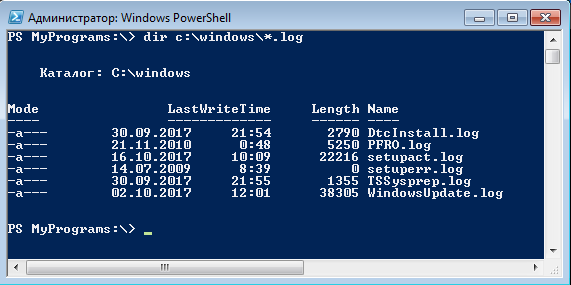


Рис. 23.

Параметр -Exclude позволяет задать маску файлов, которые не будут обрабатываться командой dir. Например, следующая команда выведет все файлы с расширением log из каталога **c:\windows**, кроме тех, чье имя начинается на букву s (рис. 24):

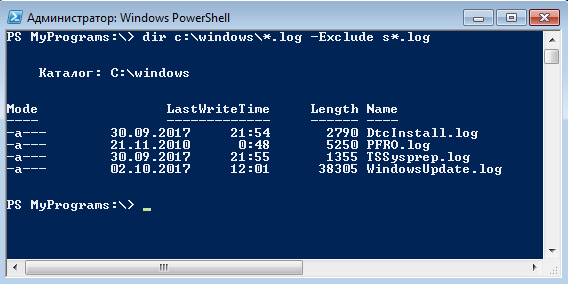


Рис. 24

Параметр -Name позволяет выводить на экран только имена файлов (рис. 25):

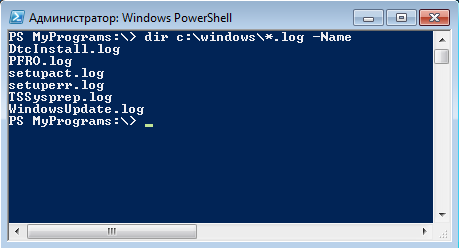


Рис. 25

Напомним, что параметр -Recurse включает режим рекурсии, при котором командлет dir отображает не только содержимое указанного каталога, но и всех его подкаталогов.

Иногда бывает нужно сформировать список, состоящий только из подкаталогов или только из файлов. В PowerShell построить список подкаталогов можно путем фильтрации объектов с помощью командлета Where-Object по значению логического свойства PSIsContainer (если объект соответствует каталогу файловой системы, то данное свойство равно $True, если объект соответствует файлу, то свойство PSIsContainer равно $False). Следующая команда выведет список подкаталогов каталога **C:\Users\Student\Documents\Documents\_01**(рис. 26):

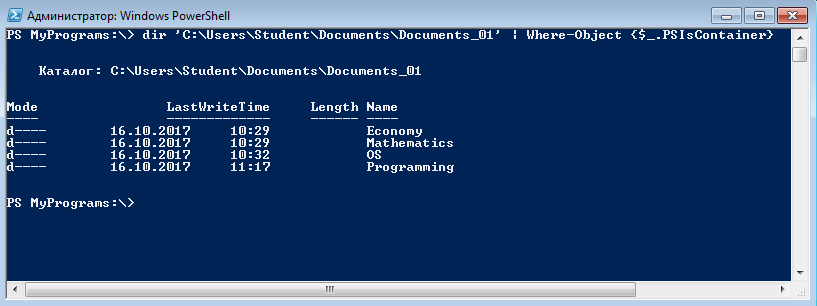


Рис. 26.

Напомним, что в этом примере **OS** и **Programming** имеют вложенные каталоги.

Сортировку списка в PowerShell можно выполнить по любому свойству соответствующих объектов с помощью командлета SortObject, например (рис. 27):

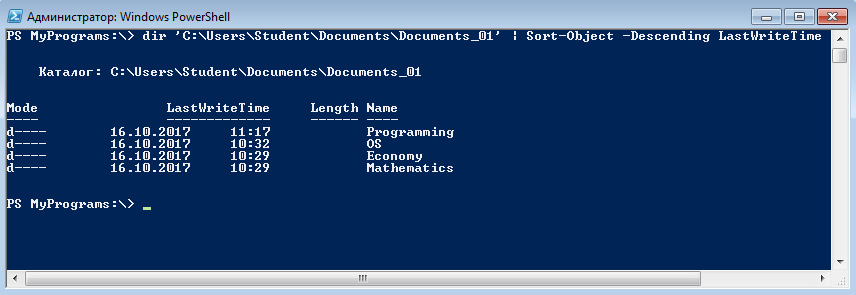


Рис. 27.

**Определение размера каталогов**

Любому каталогу файловой системы в PowerShell соответствует объект типа System.IO.DirectoryInfo, который не имеет свойства, в котором хранился бы размер этого каталога. Поэтому для определения размера каталога нужно подсчитать общий размер файлов, записанных в данном каталоге и всех его подкаталогах. Сделать это можно путем суммирования с помощью командлета Measure-Object значений свойства Length объектов-файлов, поставляемых по конвейеру командлетом dir (Get-ChildItem). Например, следующая команда показывает размер в байтах каталога **C:\Users\Student\Documents\Documents\_01\OS** (свойство Sum) и количество файлов и подкаталогов в этом каталоге (свойство Count) (рис. 28):

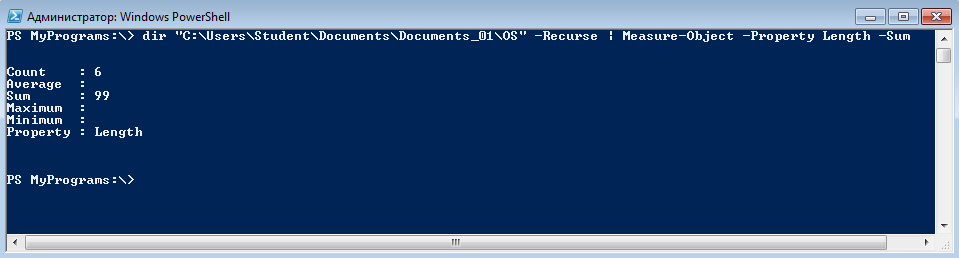


Рис. 28.

Если нужно определить размер в мегабайтах или гигабайтах, то значение свойства Sum следует разделить на соответствующую константу-суффикс (1Mb или 1Gb), (рис. 29):

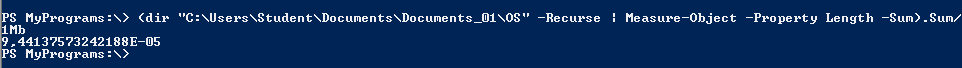


Рис. 29

**Создание файлов и каталогов**

Создать новый файл или каталог в PowerShell позволяет командлет   
New-Item. Путь к создаваемому элементу указывается в виде значения параметра -Path, а в качестве значения параметра -ItemType указывается "directory", если нужно создать каталог, и "file", если нужно создать файл. Например, следующая команда создает на диске **C:\Users\Student\Documents\** каталог с именем **test\_folder** (рис. 30):

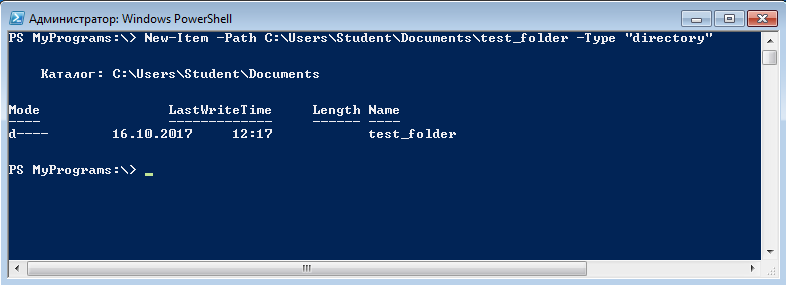


Рис. 30.

При создании файла в него сразу можно записать строку, указав ее в качестве значения параметра -Value, например (рис. 31):

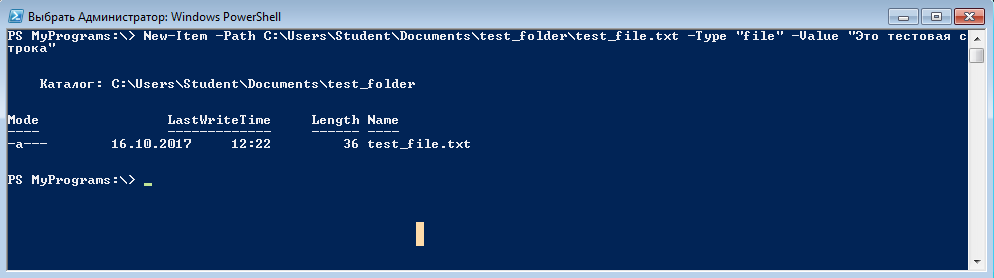


Рис. 31.

Если попытаться назвать создаваемый файл именем уже существующего файла, то возникнет ошибка.

**Чтение и просмотр содержимого файлов**

В оболочке cmd.exe имеется команда type, которая выводит содержимое текстового файла на экран. В PowerShell команда type является псевдонимом командлета Get-Content (другие псевдонимы этого же командлета – cat и gc), предназначенного для построчного считывания содержимого текстового файла с возвращением объекта для каждой строки (при этом строки отображаются на экране). Например (рис. 32):

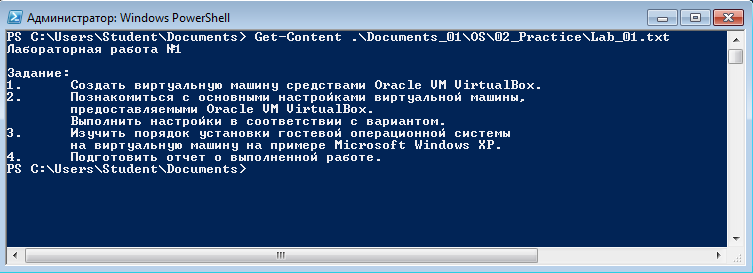


Рис. 32.

По умолчанию командлет Get-Content считывает все строки из файла; их количество можно ограничить с помощью параметра -TotalCount. Например, следующая команда считывает первые пять строк из файла (рис. 33):

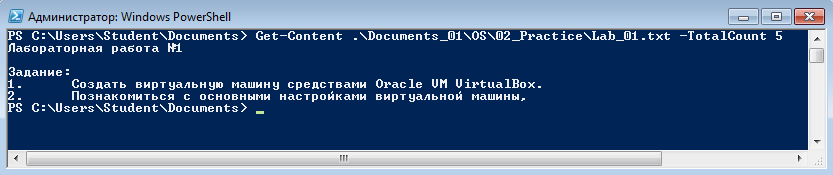


Рис. 33.

**Запись файлов**

Если нужно просто записать в файл текстовую информацию (без дополнительного форматирования), то можно воспользоваться командлетом   
Set-Content.

Данные для записи в файл могут задаваться в качестве значения параметра   
-Value. Например, следующая команда записывает в файл **C:\Users\Student\Documents\test\_folder\test\_file.txt** строку "Строка из PowerShell" (рис. 34):

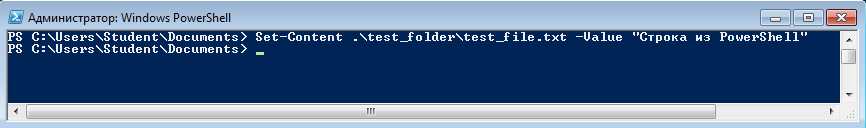


Рис. 34.

Если файл, в который производится запись, уже существует, то командлет   
Set-Content заменит его содержимое. В случае, когда требуется добавить строки в конец существующего файла, следует воспользоваться командлетом Add-Content.

**Копирование файлов и каталогов**

В PowerShell копирование файлов и каталогов осуществляется командлетом Copy-Item, имеющим псевдоним copy. Путь к копируемым файлам при этом указывается в качестве значения параметра -Path (данный параметр используется по умолчанию), а путь к целевому каталогу, в который нужно скопировать файлы, задается значением параметра -Destination. Например, следующая команда скопирует файл test\_file.txt из каталога **C:\Users\Student\Documents\test\_folder\** в каталог **C:\Users\Student\Documents\Documents\_02\newTest** (рис. 35):

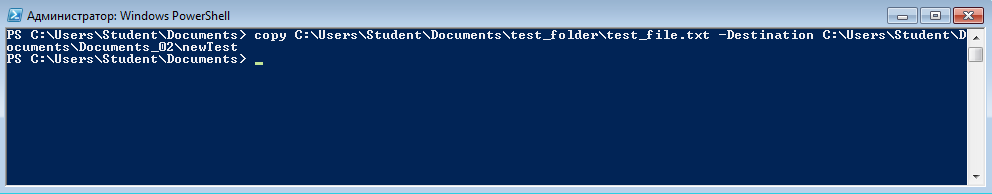


Рис. 35

Для того чтобы увидеть результат выполнения команды копирования, нужно указать параметр -PassThru (рис. 36):

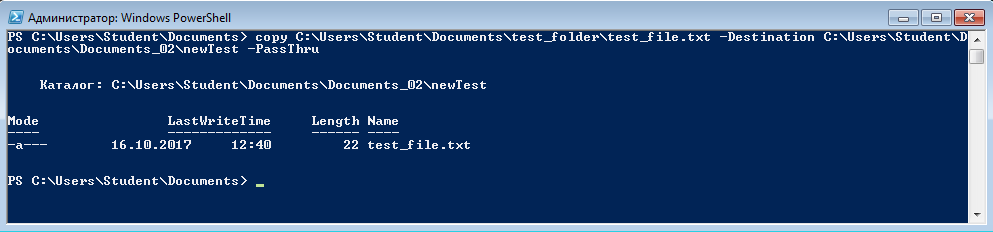


Рис. 36.

Параметр -Recurse позволяет копировать содержимое вложенных каталогов.

Можно копировать не все файлы из каталога, а только соответствующие определенной маске. При этом маску можно указать внутри пути для копирования или в качестве значения параметра -Include.

**Переименование и перемещение файлов и каталогов**

Переименовать файл или каталог можно с помощью командлета   
Rename-Item (псевдоним ren). Значение параметра -Path этого командлета задает путь к элементам для переименования, а значение параметра -NewName — новое имя. Имена этих параметров можно опускать (в этом случае первым должно указываться значение параметра -Path). Например, создадим файл **C:\Users\Student\Documents\test\_folder\1.tmp** и переименуем его в файл **2.tmp**. Результат представлен на рис. 37.

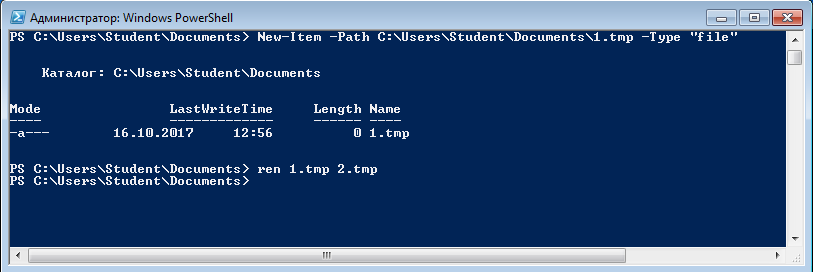


Рис. 37.

**Удаление файлов и каталогов**

Удалять объекты файловой системы можно с помощью командлета   
Remove-Item (псевдоним del). Значение параметра -Path этого командлета задает путь к удаляемым файлам или каталогам (имя параметра в команде можно не указывать). В пути допускаются подстановочные символы. Кроме того, командлет Remove-Item имеет параметр -Include, значение которого задает файлы, на которые будет действовать команда, и параметр -Exclude, задающий файлы-исключения, которые удаляться не будут.

Удаление файла **C:\Users\Student\Documents\test\_folder\1.tmp**. представлено на рис. 38.



Рис. 38.