МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

ОТЧЕТ

по лабораторной работе №2

«Командный язык ОС Windows и командная оболочка Windows PowerShell»

по дисциплине

«Принципы и методы организации системных

программных средств»

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_ Викулова Е.Н.\_\_

(подпись) (фамилия, и.,о.)

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Сухоруков В.А.\_\_\_

(подпись) (фамилия, и.,о.)

\_\_\_\_\_\_19-В-2\_\_\_\_\_\_\_\_

(шифр группы)

Работа защищена «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород 2021

Оглавление

[Цель работы 3](#_Toc99364266)

[Ход работы 3](#_Toc99364267)

[Командные оболочки Microsoft 3](#_Toc99364268)

[Оболочка(интерпретатор) командной строки command.com/cmd.exe 3](#_Toc99364269)

[Поддержка языков сценариев. Сервер сценариев Windows Script Host. 4](#_Toc99364270)

[Командная оболочка Microsoft PowerShell 5](#_Toc99364271)

[Состав команд командного процессора cmd.exe 7](#_Toc99364272)

[Основные команды работы с каталогами 7](#_Toc99364273)

[Команды работы с файлами 8](#_Toc99364274)

[Команды системного назначения 9](#_Toc99364275)

[Перенаправление ввода-вывода 10](#_Toc99364276)

[Конвейеры команд и фильтры 12](#_Toc99364277)

[Примеры использования командного языка для решения практических задач. 14](#_Toc99364278)

[Преимущества оболочки Windows - cmd.exe 16](#_Toc99364279)

[Состав и функциональные возможности MS PowerShell 17](#_Toc99364280)

[Полезные командлеты Windows PowerShell 17](#_Toc99364281)

[Работа с переменными 17](#_Toc99364282)

[Форматирование в Windows PowerShell 19](#_Toc99364283)

[Импорт и экспорт 20](#_Toc99364284)

[Работа с сетью в Windows PowerShell 20](#_Toc99364285)

[Работа с элементами 21](#_Toc99364286)

[Пример практического использования MS PowerShell 22](#_Toc99364287)

[Преимущества PowerShell 23](#_Toc99364288)

# Цель работы

1. Ознакомиться с эволюцией, достоинствами, недостатками, особенностями командных оболочек от Microsoft.
2. Изучить состав команд командного процессора cmd.exe.
3. Изучить состав и функциональные возможности MS PowerShell.

# Ход работы

## Командные оболочки Microsoft

### Оболочка(интерпретатор) командной строки command.com/cmd.exe

Во всех версиях операционной системы Windows поддерживается интерактивная оболочка командной строки (command shell) и по умолчанию устанавливается определенный набор утилит командной строки (количество и состав этих утилит зависит от версии операционной системы). Механизм работы оболочек в разных системах одинаков: в ответ на приглашение, выдаваемое находящейся в ожидании оболочкой, пользователь вводит некоторую команду, 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 директивного программирования, в качестве операторов которого могут использоваться исполняемые программы.

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

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

*COMMAND.COM* - это интерпретатор командной строки по умолчанию для MS- DOS , Windows 95 , Windows 98 , Windows 98SE и Windows Me . В случае DOS это пользовательский интерфейс по умолчанию.

Начиная с версии Windows NT, оболочка командной строки представляется интерпретатором *CMD.EXE*, который расширяет возможности оболочки command.com операционной системы MS-DOS.

Функциональность командного интерпретатора command.com была позаимствована из операционной системы CP/M, оболочка которой представляла собой значительно упрощенный и урезанный вариант оболочки Unix-систем.

Таким образом, оболочка командной строки MS-DOS изначально уступала Unix-оболочкам по удобству работы и развитости языка сценариев. В командной оболочке Windows (cmd.exe), несмотря на все сделанные улучшения, не удалось преодолеть данное отставание ни в режиме интерактивной работы (например, в cmd.exe отсутствует поддержка псевдонимов для длинных названий команд и не реализовано автоматическое завершение команд при вводе их с клавиатуры), ни в синтаксисе или возможностях языка командных файлов. Ситуация усугублялась тем, что Windows всегда проигрывала Unix-системам в количестве и функциональных возможностях стандартных (не требующих дополнительной установки) утилит командной строки, а также в качестве и полноте встроенной справочной системы по командам оболочки.

Оболочка командной строки cmd.exe и командные файлы – наиболее универсальные и простые в изучении средства автоматизации работы в Windows, доступные во всех версиях операционной системы, которые, однако, существенно проигрывают аналогичным инструментам в Unix-системах и не обеспечивают доступ к объектным моделям, поддерживаемым операционной системой (COM, WMI, .NET).

### Поддержка языков сценариев. Сервер сценариев Windows Script Host.

Следующим шагом в развитии средств и технологий автоматизации в операционной системе Windows стало появление сервера сценариев Windows Script Host (WSH). Этот инструмент разработан для всех версий Windows и позволяет непосредственно в операционной системе выполнять сценарии на полноценных языках сценариев (по умолчанию, VBScript и JScript).

По сравнению с командными файлами интерпретатора cmd.exe сценарии WSH имеют несколько преимуществ:

* VBScript и JScript – это полноценные алгоритмические языки, имеющие встроенные функции и методы для обработки символьных строк, выполнения математических операций, обработки исключительных ситуаций и т.д
* WSH поддерживает несколько собственных объектов, свойства и методы которых позволяют решать некоторые часто возникающие повседневные задачи администратора операционной системы: работа с сетевыми ресурсами, переменными среды, системным реестром, ярлыками и специальными папками Windows, запуск и управление работой других приложений.
* Из сценариев WSH можно обращаться к службам любых приложений-серверов автоматизации (например, программ из пакета Microsoft Office), которые регистрируют в операционной системе свои объекты.
* Сценарии WSH позволяют работать с объектами информационной модели Windows Management Instrumentation (WMI), обеспечивающей программный интерфейс управления всеми компонентами операционной модели, а также с объектами службы каталогов Active Directory Service Interface (ADSI)

Слабые места WSH:

* WSH – это только среда выполнения сценариев, а не оболочка; WSH не интегрирован с командной строкой, то есть отсутствует режим, в котором можно было вводить команды с клавиатуры и сразу видеть результат их выполнения.
* В операционной системе по умолчанию нет полноценной подробной справочной информации по объектам WSH и языкам VBScript.
* Сценарии WSH представляют собой довольно серьезную потенциальную угрозу с точки зрения безопасности, известно большое количество вирусов, использующих WSH для выполнения деструктивных действий.

Таким образом, можно дать следующую общую оценку: сценарии WSH – это универсальный инструмент, который в любой версии операционной системы Windows позволяет решать задачи автоматизации практически любой степени сложности, но требует при этом большой работы по изучению самих языков сценариев и ряда смежных технологий управления операционной системой (WMI, ADSI и т.п.).

### Командная оболочка Microsoft PowerShell

 С одной стороны функциональности и гибкости языка оболочки cmd.exe было явно недостаточно, а с другой стороны сценарии WSH, работающие с объектными моделями ADSI и WMI, оказались слишком сложными для пользователей среднего уровня и начинающих администраторов.

Перед разработчиками новой оболочки, получившей название Windows PowerShell (предварительное название – *Monad*), стояли следующие основные цели и задачи, которые были успешно решены:

* применение командной строки в качестве основного интерфейса администрирования;
* реализация модели ObjectFlow (элементом обмена информации является объект);
* переработка существующих команд, утилит и оболочки;
* интеграция командной строки, объектов COM, WMI и .NET;
* работа с произвольными источниками данных в командной строке по принципу файловой системы.

Самая важная идея, заложенная в PowerShell, состоит в том, что в командной строке вывод результатов команды представляет собой не текст (последовательности байтов), а объект (данные вместе со свойственными им методами). В силу этого работать в PowerShell становится проще, чем в традиционных оболочках, так как не нужно выполнять никаких манипуляций по выделению нужной информации из символьного потока.

Кроме того, разработчики старались собрать в PowerShell все лучшие аспекты других оболочек командной строки из разных операционных систем. По их словам, сильное влияние на PowerShell оказали следующие продукты:

* BASH, KSH (конвейеризация или композиция команд);
* AS400/VMS (стандартные названия команд, ускоряющие изучение);
* TCL/WSH (поддержка встраиваемости и нескольких языков);
* PERL, PYTHON (выразительность и стиль).

PowerShell одновременно является и оболочкой командной строки (пользователь работает в интерактивном режиме) и средой выполнения сценариев, которые пишутся на специальном языке PowerShell.

Интерактивный сеанс в PowerShell похож на работу в оболочке Unix-систем: все команды в PowerShell имеют подробную встроенную справку (для большинства команд приводятся примеры их использования), поддерживается функция автоматического завершения названий команд и их параметров при вводе с клавиатуры, для многих команд имеются псевдонимы, аналогичные названиям Unix-утилит.

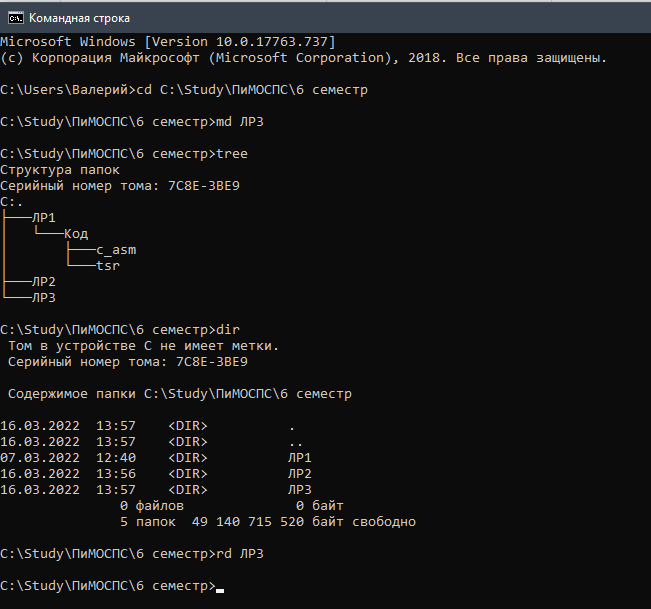
В целом, оболочка PowerShell намного удобнее и мощнее своих предшественников (cmd.exe и WSH), а основным недостатком, сдерживающим распространение нового инструмента, является тот факт, что PowerShell работает не во всех версиях операционной системы Windows. Оболочкой можно пользоваться только на версиях не ниже Windows XP Service Pack 2 с установленным пакетом .NET Framework 2.0.

Источник - <https://intuit.ru/studies/courses/1059/225/lecture/27283>

## Состав команд командного процессора cmd.exe

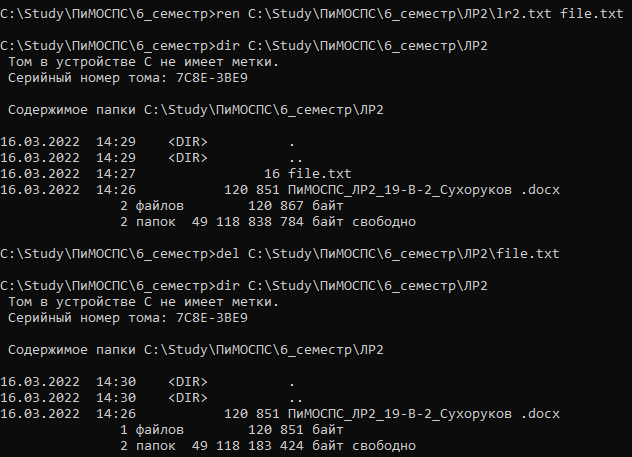
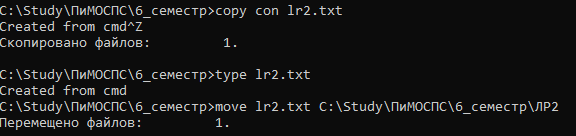
### Основные команды работы с каталогами

* MD — Создание каталога. Синтаксис: (MD имя каталога, есть возможность указать путь расположения).
* RD — Удаление каталога. Синтаксис: (RD имя каталога, есть возможность указать путь расположения). Примечание, RD без ключей позволяет удалить только пустой каталог.
* CD — Смена текущего каталога. Синтаксис: (CD имя каталога, есть возможность указать путь расположения). CD\ — Переход в корневой каталог. CD.. — Переход в родительский каталог.
* DIR — Просмотр каталога в виде списка. Синтаксис: (DIR имя каталога, есть возможность указать путь расположения).
* TREE — Вывод каталогов в графическом представлении. Синтаксис: (TREE имя каталога, есть возможность указать путь расположения).
* MOVE — Перемещение\Переименование каталога. Синтаксис: (MOVE что\_перемещаем куда\_перемещаем, есть возможность указать путь расположения).
* XCOPY — Копирование структур каталогов. Синтаксис: (XCOPY что\_копируем куда\_копируем, есть возможность указать путь расположения).



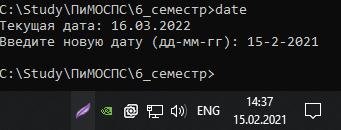
### Команды работы с файлами

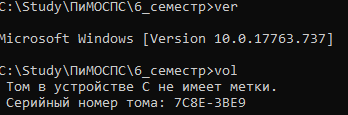
* COPY CON — Создание файла. Синтаксис: (COPY CON имя\_файла\_с\_расширением, есть возможность указать путь расположения).
* TYPE — Вывод содержимого файла на экран. Синтаксис: (TYPE имя\_файла\_с\_расширением, есть возможность указать путь расположения).
* DEL — Удаление файла. Синтаксис: (DEL имя\_файла\_с\_расширением, есть возможность указать путь расположения).
* COPY — Копирование файла\Обьединения файлов. Синтаксис(копирование): (COPY что\_копируем куда\_копируем, есть возможность указать путь расположения). Синтаксис(обьединение): (COPY имя\_файла + имя\_файла + имя\_файла… имя\_файла\_обьединения, есть возможность указать путь расположения).
* MOVE — Перемещение\Переименование файла. Синтаксис: (MOVE что\_перемещаем куда\_перемещаем, есть возможность указать путь расположения).
* REN — Переименование файлов. Синтаксис: (REN что\_переименовываем вочто\_переименовываем, есть возможность указать путь расположения).
* FC — Сравнение файлов. (FC имя\_файла имя\_файла имя\_файла…, есть возможность указать путь расположения).

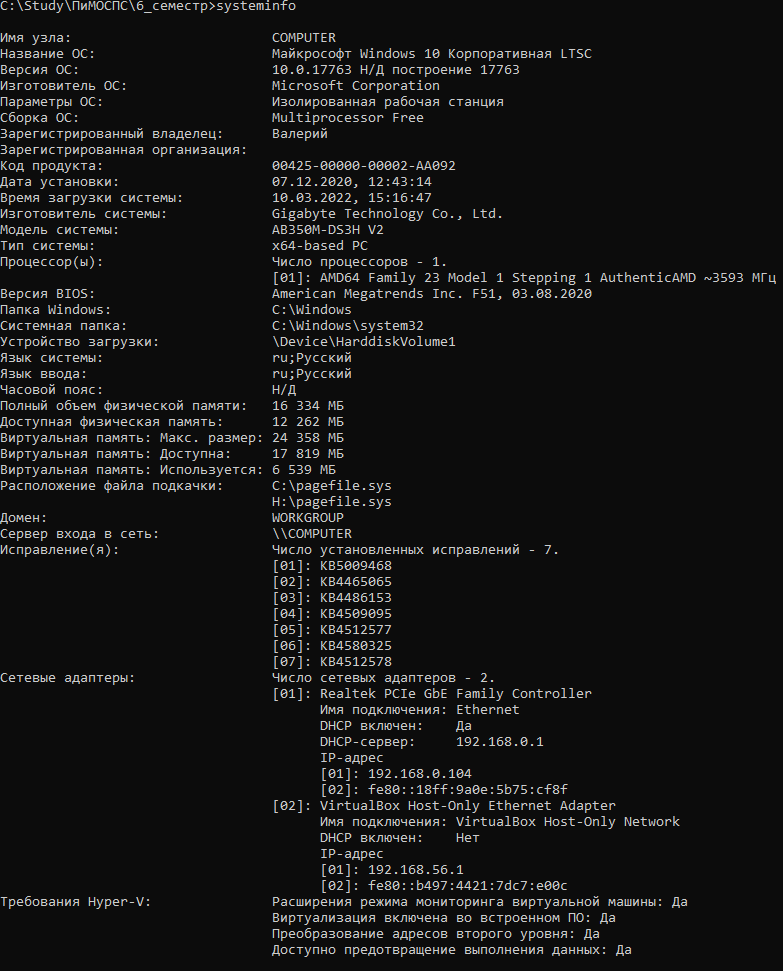


### Команды системного назначения

* CLS — Отчистка экрана.
* DATE — Просмотр и вывод на изменения текущей даты.
* VER — Вывод информации о текущей версии Операционной системы.
* VOL — Вывод информации о томе-логическом разделе диска.
* SYSTEMINFO — Вывод информации о конфигурациях системы.
* EXIT — Выход из командной строки.







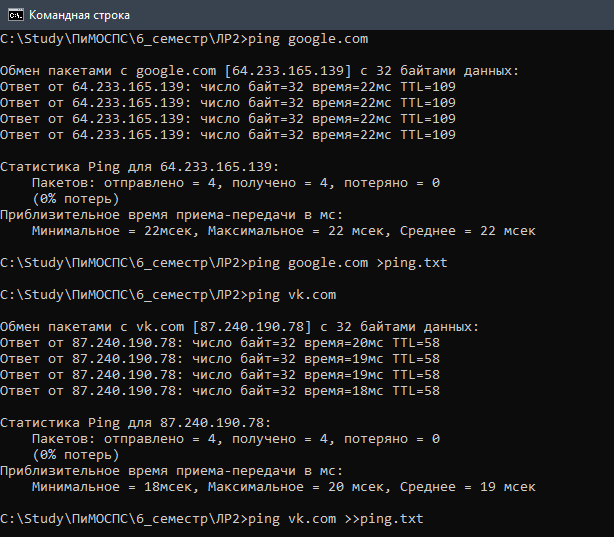
### Перенаправление ввода-вывода

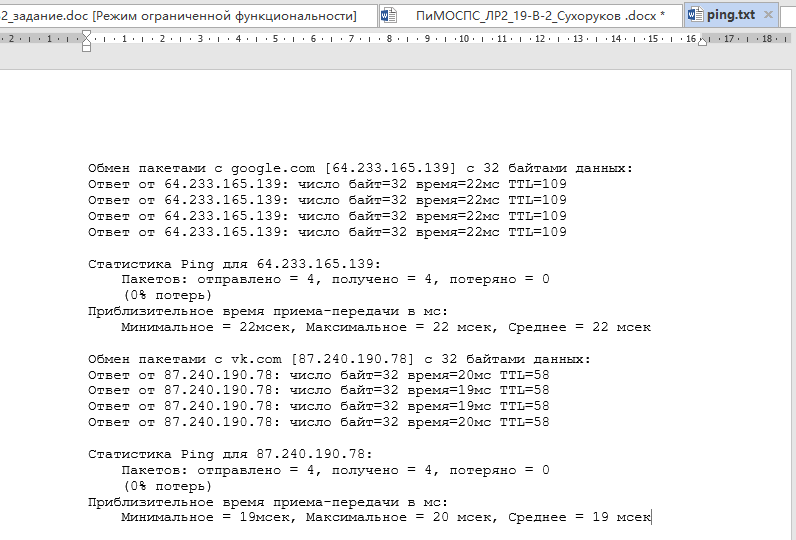
С помощью переназначения устройств ввода/вывода одна программа может направить свой вывод на вход другой или перехватить вывод другой программы, используя его в качестве своих входных данных. Таким образом, имеется возможность передавать информацию от процесса к процессу при минимальных программных издержках. Практически это означает, что для программ, которые используют стандартные входные и выходные устройства, операционная система позволяет:

* выводить сообщения программ не на экран (стандартный выходной поток), а в файл или на принтер (перенаправление вывода);
* читать входные данные не с клавиатуры (стандартный входной поток), а из заранее подготовленного файла (перенаправление ввода);
* передавать сообщения, выводимые одной программой, в качестве входных данных для другой программы (конвейеризация или композиция команд).

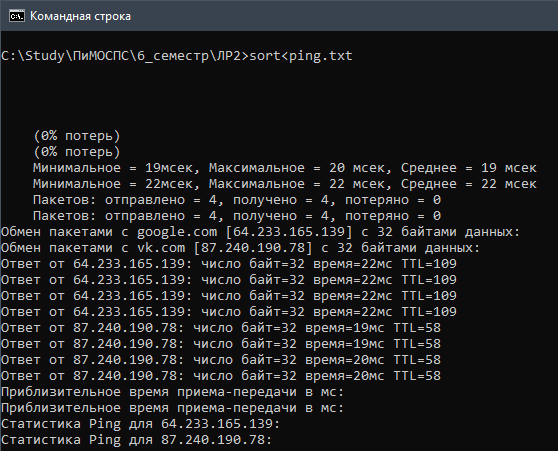
Из командной строки эти возможности реализуются следующим образом. Для того, чтобы перенаправить текстовые сообщения, выводимые какой-либо командой из командной строки, в текстовый файл, нужно использовать конструкцию **команда> имя\_файла.** Если при этом заданный для вывода файл уже существовал, то он перезаписывается (старое содержимое теряется), если не существовал создается. Можно также не создавать файл заново, а дописывать информацию, выводимую командой, в конец существующего файла. Для этого команда перенаправления вывода должна быть задана так: **команда>> имя\_файла**. С помощью символа **<**можно прочитать входные данные для заданной команды не с клавиатуры, а из определенного (заранее подготовленного) файла: **команда <имя\_файла.**

Команда **PING** позволяет выполнить отправку управляющего сообщения типа Echo Request (тип равен 8 и указывается в заголовке ICMP-сообщения) адресуемому узлу и интерпретировать полученный от него ответ в удобном для анализа виде.





Команда **SORT** используется для сортировки в алфавитном порядке строк текстового файла или стандартного вывода.



### Конвейеры команд и фильтры

Вывод первой команды является вводом для второй, вывод второй - вводом для третьей и т.д.

process1|process2|…process

**Фильтром** называется программа, принимающая на вход информационный поток, преобразующая его по определенному алгоритму и выводящая результат работы (команды find,sort,more)

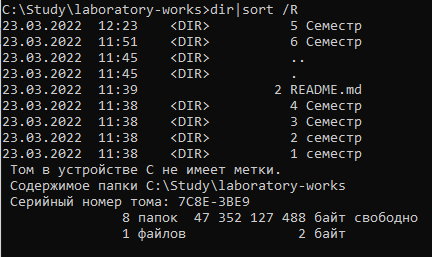
**Команда MORE** – постраничное отображение содержимого текстового файла (внешняя команда, файл …\Windows\System32\more.exe).

**Команда SORT** – сортировка строк текстового файла в алфавитном порядке.

* /R – сортировать в обратном порядке (от Z к A, от 9 к 0).

**Команда FIND** – поиск заданной последовательности символов (string) в одном или нескольких текстовых файлах (внешняя команда, файл …\Windows\System32\find.exe).

* /V – выводить только строки, не содержащие string;
* /C – вместо строк выводить их порядковые номера;
* /N – перед каждой выводимой строкой выводить и ее номер.

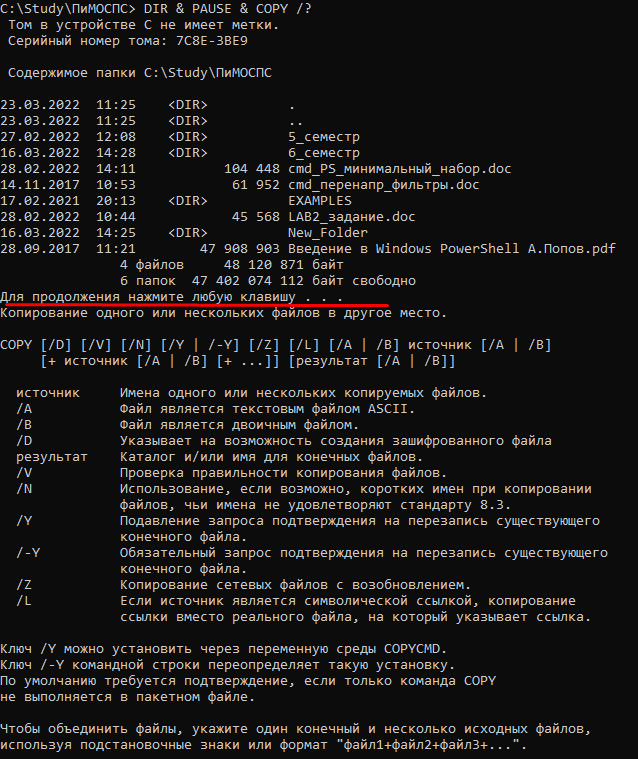


В данном примере обе команды запускаются одновременно, но команда sort приостанавливает работу до получения выходных данных команды dir. Команда sort использует выходные данные команды dir в качестве своих входных данных, а затем свои выходные данные отправляет в дескриптор 1 (STDOUT).

Условное выполнение и группировка команд

* Команда1 & Команда2 – сначала выполнятся Команда1, а уже потом Команда2
* Команда1 && Команда2 - Команда2 будет выполняться только в том случае, если произошло успешное выполнение Команды1.
* Команда1 || Команда2 - Команда2 будет выполняться только в том случае, если Команда1 не смогла выполниться.

Команда **DIR & PAUSE & COPY /?**  сначала выведет на экран содержимое текущего каталога, а после нажатия любой клавиши – выведена встроенная справка команды COPY.



### Примеры использования командного языка для решения практических задач.

#### Загрузка на github

@echo off

rem загрузка новых лабораторных на github

rem копирование папки с лабораторной в папку github

xcopy /y /o /e "C:\Study\%1\\*.\*" "C:\Study\laboratory-works\6 Semestr\%1\\*.\*"

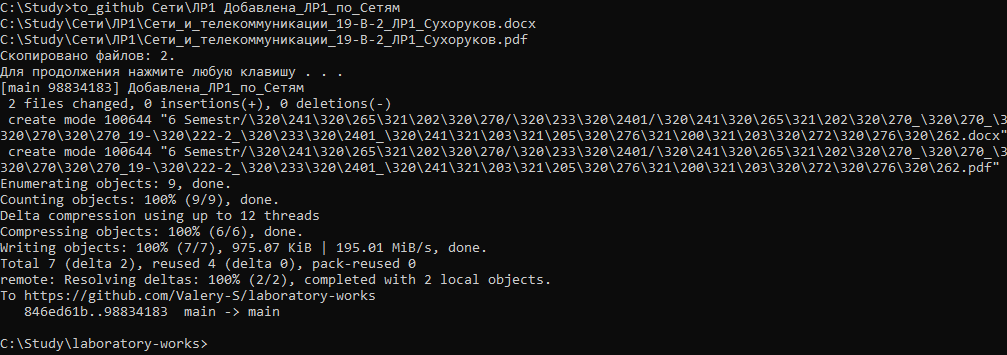
rem добавление изменений на сервер

cd C:\Study\laboratory-works

git add .

git commit -am %2

git push



#### Копирование данных с флешки при её подключении

@echo off

:test

if exist f:\ goto go

goto test

:go

xcopy "F:\\*.\*" "C:\flash\\*.\*"

**:test** обозначает начало действия батника

**if exist f:\ goto go** проверяет наличие в компьютере диска **F:\**, если он есть, то переходим к части **:go** (вставленная флешка получает для обозначения первую свободную в системе букву латинского алфавита)

**goto test** если диск **G:\** не был найден, возвращаемся к началу части **:test**

**:go** обозначает начало действия второй части батника

**xcopy "F:\\*. \*" "C:\flash\\*. \*"** копирует всё содержимое диска **G:\** в папку на диске **C:\**

### Преимущества оболочки Windows - cmd.exe

* С помощью cmd возможно создание сценариев автоматизации и пакетных файлов, т.е. выполнение одной или нескольких операций без вмешательства пользователя. Это отличный инструмент для создания сценариев.
* Управление данными и файлами. Преимущества cmd становятся очевидны, когда требуется выполнять однотипные операции над множеством объектов. Одним из важных преимуществ командной строки является непосредственная возможность управлять файлами и данными. К данным возможностям относятся: копирование, удаление, перемещение и т.д. При этом, не забывайте, что вы можете автоматизировать данный процесс.
* Администрирование компьютера. Быстрое получение текущей информации сокращает время диагностики компьютера.
* Администрирование сети. Многие команды администрирования сети не имеют графических эквивалентов (например – команда ping, pathping, tracert). Cmd очень удобна для контроля сетевой активности. Вы можете создавать службы, запускающиеся при старте оперативной системы, можете использовать команды администрирования сети, не имеющие графических эквивалентов.

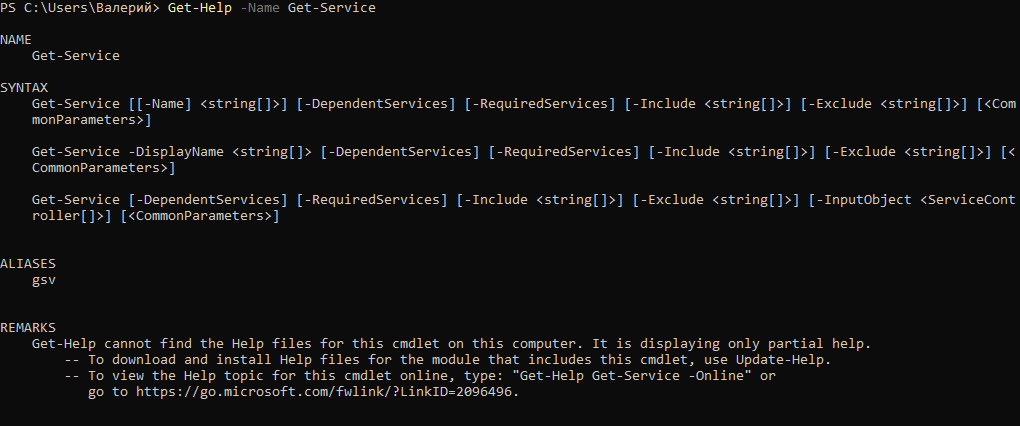
## Состав и функциональные возможности MS PowerShell

**Командлет** — это команда Windows PowerShell, предназначенная для работы с объектами и выполняющая единственную функцию. Командлеты можно идентифицировать по формату имени — глаголу и существительному, разделенным дефисом (-), например, Get-Help, Get-Process и Start-Service.

Большинство командлетов Windows PowerShell очень просты, и предполагается, что они будут использоваться вместе с другими командлетами. Например, командлеты категории "get" только возвращают данные, командлеты "set" только задают или изменяют значения элементов данных, командлеты "format" только форматируют данные, а командлеты "out" только направляют вывод в указанное место назначения.

### Полезные командлеты Windows PowerShell

* **Get-Help** – показывает справку по командлету, функции и общую справку по Windows PowerShell. Справка бывает нескольких типов: краткая, детальная, полная и вывод только примеров;
* **Update-Help** — загружает и устанавливает новые файлы справки, т.е. обновление справки;
* **Get-Command** – командлет поиска нужной команды, можно искать как по глаголу, так и по существительному, также возможно использование маски, если Вы не знаете точное наименование глагола или существительного;
* **Get-Alias** – показывает псевдонимы, все или конкретной команды;
* **Get-PSDrive** – показывает подключенные диски;
* **Get-Member** – выводит свойства и методы, которые есть у объекта;
* **Get-Service** - получает службы на локальном или удаленном компьютере.
* **Get-History** — возвращает список команд, введенных в ходе текущей сессии.



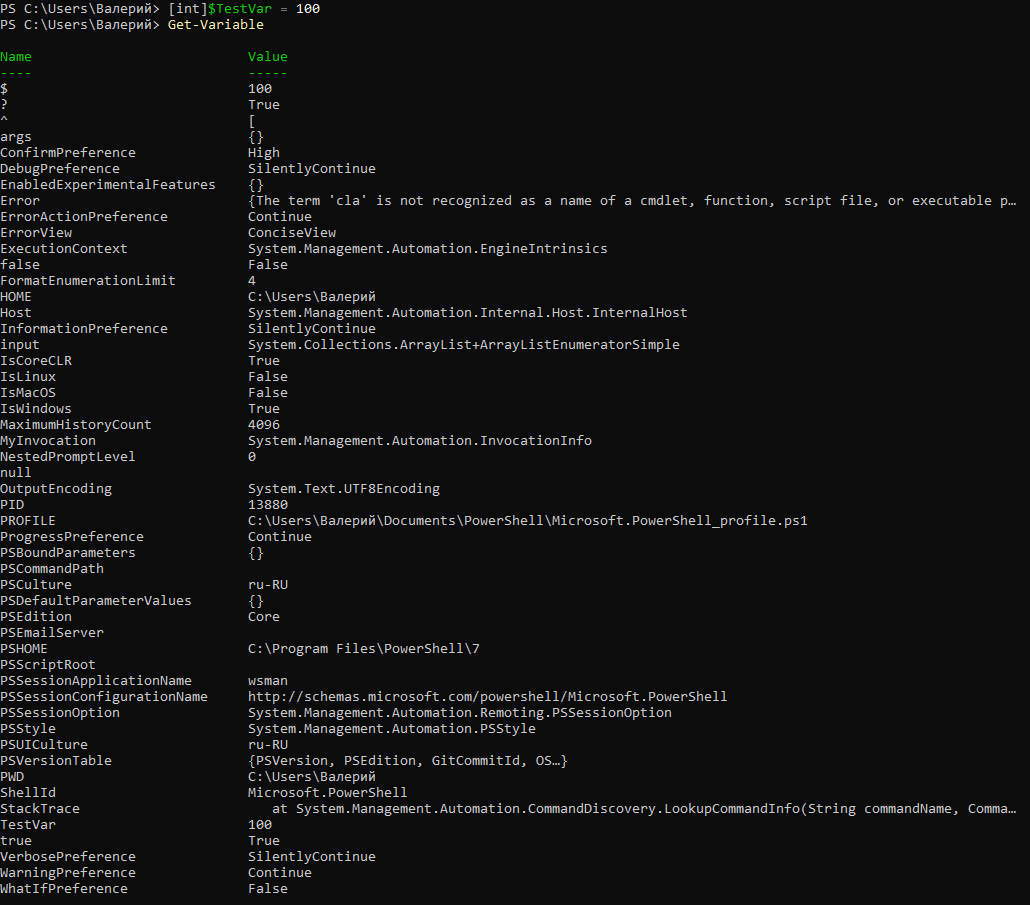
### Работа с переменными

В Windows PowerShell есть как встроенные переменные, так и переменные, которые может создавать пользователь. Переменные в PowerShell можно объявлять с указанием типа и без, при объявлении переменной ее можно сразу инициализировать. Еще одной особенностью переменных в PowerShell является то, что они могут менять свой тип в зависимости от значения этой переменной, но это только в том случае если мы принудительно при объявлении переменной не указали ее тип.

Для того чтобы создать переменную, необходимо перед ее названием указать знак доллара ($). В случае необходимости можно принудительно указать тип данных переменной в квадратных скобках перед ее названием. Например, ниже мы создаем переменную TestVar с целочисленным типом данных (int) и сразу ее инициализируем.

[int]$TestVar = 100

* **Get-Variable** – выводит список переменных и их значения (или одну указанную переменную);
* **New-Variable** – создает новую переменную;
* **Set-Variable** – задает значение переменной. Если переменная с указанным именем не существует, то она будет создана;
* **Clear-Variable** — удаляет значение переменной;
* **Remove-Variable** — удаляет переменную и ее значение.



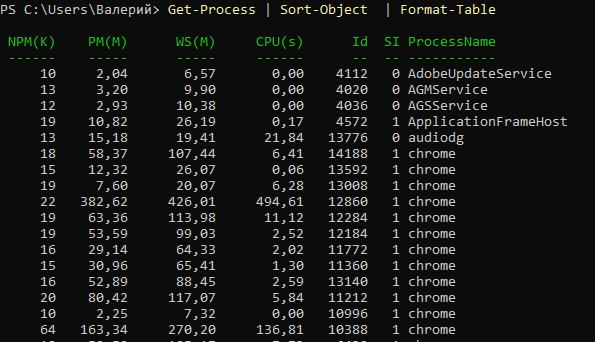
### Форматирование в Windows PowerShell

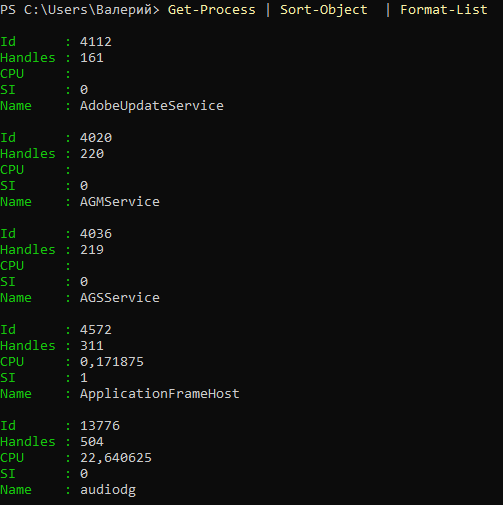
В Windows PowerShell существует набор командлетов, которые предназначены для форматирования вывода результата работы командлета. Они позволяют пользователю отобразить результат в том виде, в котором ему удобно просматривать данный результат.

* **Format-List** – вывод результата команды в формате списка свойств, где на каждой новой строке отдельное свойство;
* **Format-Table** — вывод результата команды в виде таблицы;
* **Format-Wide** — вывод результата команды в виде широкой таблицы, в которой отображается только одно свойство каждого объекта;
* **Format-Custom** – в данном случае форматирование вывода происходит с использованием пользовательского представления.

GET-PROCESS – отображает запущенные процессы.

Sort-Object - сортирует объекты в порядке возрастания или убывания на основе значений свойств объекта.

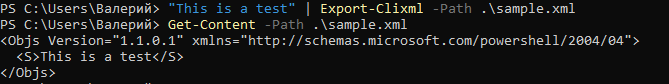




### Импорт и экспорт

PowerShell позволяет импортировать и экспортировать данные в разных распространенных форматах, например, CSV или XML, а также перенаправлять вывод результата работы команды во внешний файл или на принтер.

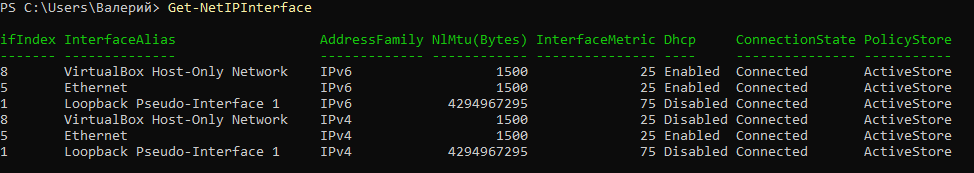
* **Export-Csv** – экспорт данных в формат CSV;
* **Import-Csv** – импортирует данные из CSV файла;
* **Export-Clixml** — экспорт данных в формат XML;
* **Import-Clixml** — импортирует файл CLIXML и создает соответствующие объекты в оболочке Windows PowerShell;
* **Out-File** — посылает вывод результата работы командлета во внешний файл (например, в TXT);
* **Out-Printer** — вывод результата работы команды на принтер;
* **Import-Module** — добавляет модули в текущей сессии.



### Работа с сетью в Windows PowerShell

Для администрирования сети в Windows PowerShell существуют такие командлеты как:

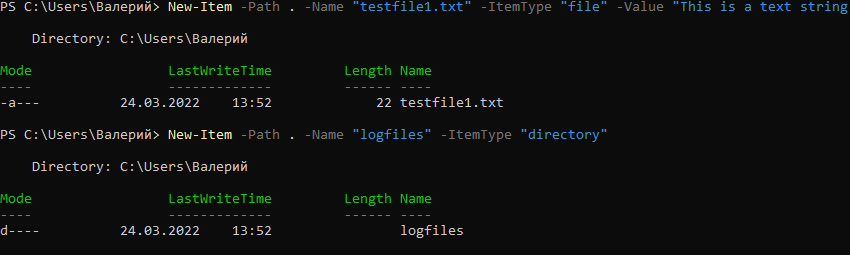
* **Disable-NetAdapter** – командлет отключает сетевой адаптер;
* **Enable-NetAdapter** – данный командлет включает сетевой адаптер;
* **Rename-NetAdapter** — переименовывает сетевой адаптер;
* **Restart-NetAdapter** — перезапускает сетевой адаптер;
* **Get-NetIPAddress** – выводит информацию о конфигурации IP-адреса;
* **Set-NetIPAddress** — изменяет конфигурацию IP-адреса;
* **New-NetIPAddress** — создает и настраивает IP-адрес;
* **Remove-NetIPAddress** — удаляет IP-адрес и его конфигурацию;
* **Get-NetRoute** — выводит таблицу маршрутизации IP;
* **Set-NetRoute** — изменяет таблицу маршрутизации IP;
* **New-NetRoute** — создает запись в таблице маршрутизации IP;
* **Remove-NetRoute** — удаляет одну или несколько записей (IP маршрутов) из таблицы маршрутизации IP;
* **Get-NetIPInterface** — выводит информацию о свойствах интерфейса IP;
* **Get-NetTCPSetting** — показывает информацию о настройках и конфигурации TCP;
* **Test-Connection** – командлет посылает ICMP пакеты к одному или нескольким компьютерам, т.е. «пингует» компьютеры.



### Работа с элементами

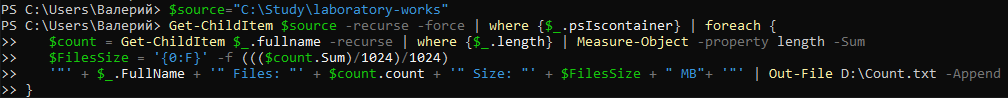
В Windows PowerShell есть командлеты, которые умеют работать с элементами, под элементами здесь можно понимать: файлы, папки, ключи реестра и так далее.

* **Clear-Item** — очищает содержимое элемента, но не удаляет сам элемент;
* **Copy-Item** – копирует элемент;
* **Get-Item** — получает элемент в указанном месте;
* **Invoke-Item** — выполняет действие по умолчанию над указанным элементом;
* **Move-Item** – перемещает элемент;
* **New-Item** – создает новый элемент;
* **Remove-Item** – удаляет указанные элементы;
* **Rename-Item** — переименовывает элемент в пространстве имен поставщика Windows PowerShell;
* **Set-Item** — изменяет элемент;
* **Set-Content** –записывает в файл
* **Get-ChildItem** — возвращает элементы и дочерние элементы в одном или нескольких определенных местах;
* **Get-Location** – выводит информацию о текущем местонахождении.

### Пример практического использования MS PowerShell

 Вывести в виде .txt-файла количество файлов и размер каталогов по адресу C:\Study\laboratory-works, помещая результат в файл D:\Count.txt.



$source=" C:\Study\laboratory-works "

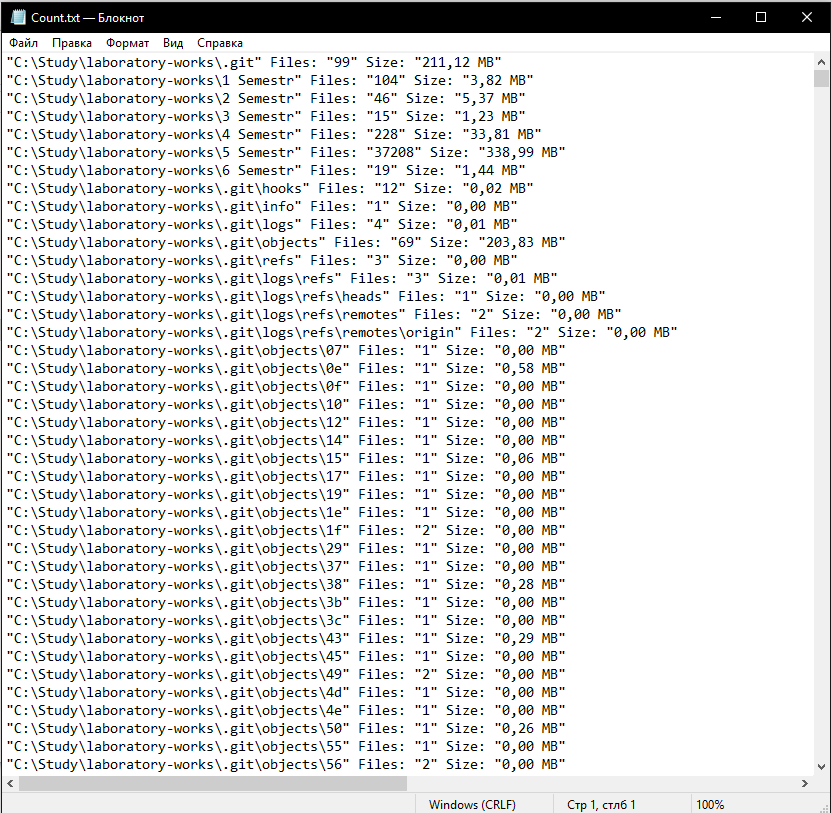
Get-ChildItem $source -recurse -force | where {$\_.psIscontainer} | foreach {

   $count = Get-ChildItem $\_.fullname -recurse | where {$\_.length} | Measure-Object -property length –Sum

   $FilesSize = '{0:F}' -f ((($count.Sum)/1024)/1024)

    '"' + $\_.FullName + '" Files: "' + $count.count + '" Size: "' + $FilesSize + " MB "+ '"' | Out-File D:\Count.txt –Append

}



### Преимущества PowerShell

* Объектно-ориентированный язык сценариев.
* Интерактивно и позволяет программистам пробовать новые вещи на консоли, а затем интегрировать их в более сложные сценарии.
* Поддерживает .Net объекты и формы. Может получить доступ ко всем библиотекам .Net и, следовательно, поддерживает такие языки, как C # или VB.
* Он поддерживает автоматизацию, которая также является важным фактором.