

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:

\_\_\_\_\_

(подпись)

Викулова Е.Н.

(фамилия, и.,о.)

СТУДЕНТ:

\_\_\_\_\_

(подпись)

Сухоруков В.А.

(фамилия, и.,о.)

19-В-2

(шифр группы)

Работа защищена «\_\_» \_\_\_\_\_

С оценкой \_\_\_\_\_

Нижний Новгород 2021

## Оглавление

Цель работы .....	3
Ход работы .....	3
Командные оболочки Microsoft.....	3
Оболочка(интерпретатор) командной строки command.com/cmd.exe .....	3
Поддержка языков сценариев. Сервер сценариев Windows Script Host.....	4
Командная оболочка Microsoft PowerShell .....	5
Состав команд командного процессора cmd.exe .....	7
Основные команды работы с каталогами .....	7
Команды работы с файлами .....	8
Команды системного назначения .....	9
Перенаправление ввода-вывода .....	10
Конвейеры команд и фильтры.....	12
Примеры использования командного языка для решения практических задач.....	14
Преимущества оболочки Windows - cmd.exe.....	16
Состав и функциональные возможности MS PowerShell .....	17
Полезные командлеты Windows PowerShell .....	17
Работа с переменными .....	17
Форматирование в Windows PowerShell .....	19
Импорт и экспорт .....	20
Работа с сетью в Windows PowerShell .....	20
Работа с элементами.....	21
Пример практического использования MS PowerShell .....	22
Преимущества PowerShell .....	23

## 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).

```
Командная строка
Microsoft Windows [Version 10.0.17763.737]
(с) Корпорация Майкрософт (Microsoft Corporation), 2018. Все права защищены.

C:\Users\Валерий>cd C:\Study\Пимоспс\6 семестр

C:\Study\Пимоспс\6 семестр>md ЛР3

C:\Study\Пимоспс\6 семестр>tree
Структура папок
Серийный номер тома: 7C8E-3BE9
C:..
├── ЛР1
│   └── Код
│       ├── c_asm
│       └── tsr
├── ЛР2
└── ЛР3

C:\Study\Пимоспс\6 семестр>dir
Том в устройстве C не имеет метки.
Серийный номер тома: 7C8E-3BE9

Содержимое папки C:\Study\Пимоспс\6 семестр

16.03.2022  13:57    <DIR>        .
16.03.2022  13:57    <DIR>        ..
07.03.2022  12:40    <DIR>        ЛР1
16.03.2022  13:56    <DIR>        ЛР2
16.03.2022  13:57    <DIR>        ЛР3
               0 файлов             0 байт
               5 папок   49 140 715 520 байт свободно

C:\Study\Пимоспс\6 семестр>rd ЛР3

C:\Study\Пимоспс\6 семестр>
```

## Команды работы с файлами

- ❖ COPY CON — Создание файла. Синтаксис: (COPY 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\Объединения файлов. Синтаксис(копирование): (COPY что\_копируем куда\_копируем, есть возможность указать путь расположения). Синтаксис(объединение): (COPY имя\_файла + имя\_файла + имя\_файла... имя\_файла\_объединения, 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 в\_что\_переименовываем, есть возможность указать путь расположения).
- ❖ FC — Сравнение файлов. (FC имя\_файла имя\_файла имя\_файла..., есть возможность указать путь расположения).

```
C:\Study\ПимОСПС\6_семестр>copy con lr2.txt
Created from cmd^Z
Скопировано файлов:          1.

C:\Study\ПимОСПС\6_семестр>type lr2.txt
Created from cmd

C:\Study\ПимОСПС\6_семестр>move lr2.txt C:\Study\ПимОСПС\6_семестр\ЛР2
Перемещено файлов:          1.

C:\Study\ПимОСПС\6_семестр>ren C:\Study\ПимОСПС\6_семестр\ЛР2\lr2.txt file.txt

C:\Study\ПимОСПС\6_семестр>dir C:\Study\ПимОСПС\6_семестр\ЛР2
Том в устройстве C не имеет метки.
Серийный номер тома: 7C8E-3BE9

Содержимое папки C:\Study\ПимОСПС\6_семестр\ЛР2

16.03.2022  14:29    <DIR>        .
16.03.2022  14:29    <DIR>        ..
16.03.2022  14:27             16 file.txt
16.03.2022  14:26          120 851 ПимОСПС_ЛР2_19-В-2_Сухоруков .docx
                2 файлов             120 867 байт
                2 папок  49 118 838 784 байт свободно

C:\Study\ПимОСПС\6_семестр>del C:\Study\ПимОСПС\6_семестр\ЛР2\file.txt

C:\Study\ПимОСПС\6_семестр>dir C:\Study\ПимОСПС\6_семестр\ЛР2
Том в устройстве C не имеет метки.
Серийный номер тома: 7C8E-3BE9

Содержимое папки C:\Study\ПимОСПС\6_семестр\ЛР2

16.03.2022  14:30    <DIR>        .
16.03.2022  14:30    <DIR>        ..
16.03.2022  14:26          120 851 ПимОСПС_ЛР2_19-В-2_Сухоруков .docx
                1 файлов             120 851 байт
                2 папок  49 118 183 424 байт свободно
```



## 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.

```
C:\Study\ПимОСПС\6_семестр>date
Текущая дата: 16.03.2022
Введите новую дату (дд-мм-гг): 15-2-2021

C:\Study\ПимОСПС\6_семестр>
```

```
C:\Study\ПимОСПС\6_семестр>ver

Microsoft Windows [Version 10.0.17763.737]

C:\Study\ПимОСПС\6_семестр>vol
Том в устройстве C не имеет метки.
Серийный номер тома: 7C8E-3BE9
```

```
C:\Study\ПимОСПС\6_семестр>systeminfo

Имя узла: COMPUTER
Название ОС: Майкрософт Windows 10 Корпоративная LTSC
Версия ОС: 10.0.17763 Н/Д построение 17763
Изготовитель ОС: Microsoft Corporation
Параметры ОС: Изолированная рабочая станция
Сборка ОС: Multiprocessor Free
Зарегистрированный владелец: Валерий
Зарегистрированная организация:
Код продукта: 00425-00000-00002-AA092
Дата установки: 07.12.2020, 12:43:14
Время загрузки системы: 10.03.2022, 15:16:47
Изготовитель системы: Gigabyte Technology Co., Ltd.
Модель системы: AB350M-DS3H V2
Тип системы: x64-based PC
Процессор(ы): Число процессоров - 1.
[01]: AMD64 Family 23 Model 1 Stepping 1 AuthenticAMD ~3593 МГц
Американ Megatrends Inc. F51, 03.08.2020
Версия BIOS: C:\Windows
Папка Windows: C:\Windows\system32
Системная папка: \Device\HarddiskVolume1
Устройство загрузки: ru;Русский
Язык системы: ru;Русский
Язык ввода: Н/Д
Часовой пояс: Полный объем физической памяти: 16 334 МБ
Доступная физическая память: 12 262 МБ
Виртуальная память: Макс. размер: 24 358 МБ
Виртуальная память: Доступна: 17 819 МБ
Виртуальная память: Используется: 6 539 МБ
Расположение файла подкачки: C:\pagefile.sys
H:\pagefile.sys
Домен: WORKGROUP
Сервер входа в сеть: \COMPUTER
Исправление(я): Число установленных исправлений - 7.
[01]: KB5009468
[02]: KB4465065
[03]: KB4486153
[04]: KB4509095
[05]: KB4512577
[06]: KB4580325
[07]: KB4512578
Сетевые адаптеры: Число сетевых адаптеров - 2.
[01]: Realtek PCIe GbE Family Controller
Имя подключения: Ethernet
DHCP включен: Да
DHCP-сервер: 192.168.0.1
IP-адрес
[01]: 192.168.0.104
[02]: fe80::18ff:9a0e:5b75:cf8f
[02]: VirtualBox Host-Only Ethernet Adapter
Имя подключения: VirtualBox Host-Only Network
DHCP включен: Нет
IP-адрес
[01]: 192.168.56.1
[02]: fe80::b497:4421:7dc7:e00c
Требования Hyper-V: Расширения режима мониторинга виртуальной машины: Да
Виртуализация включена во встроенном ПО: Да
Преобразование адресов второго уровня: Да
Доступно предотвращение выполнения данных: Да
```

## 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.

```
cmd Командная строка
C:\Study\Пимоспс\6_семестр\ЛР2>ping google.com

Обмен пакетами с google.com [64.233.165.139] с 32 байтами данных:
Ответ от 64.233.165.139: число байт=32 время=22мс TTL=109
Ответ от 64.233.165.139: число байт=32 время=22мс TTL=109
Ответ от 64.233.165.139: число байт=32 время=22мс TTL=109
Ответ от 64.233.165.139: число байт=32 время=22мс TTL=109

Статистика Ping для 64.233.165.139:
    Пакетов: отправлено = 4, получено = 4, потеряно = 0
        (0% потеря)
Приблизительное время приема-передачи в мс:
    Минимальное = 22мсек, Максимальное = 22 мсек, Среднее = 22 мсек

C:\Study\Пимоспс\6_семестр\ЛР2>ping google.com >ping.txt

C:\Study\Пимоспс\6_семестр\ЛР2>ping vk.com

Обмен пакетами с vk.com [87.240.190.78] с 32 байтами данных:
Ответ от 87.240.190.78: число байт=32 время=20мс TTL=58
Ответ от 87.240.190.78: число байт=32 время=19мс TTL=58
Ответ от 87.240.190.78: число байт=32 время=19мс TTL=58
Ответ от 87.240.190.78: число байт=32 время=18мс TTL=58

Статистика Ping для 87.240.190.78:
    Пакетов: отправлено = 4, получено = 4, потеряно = 0
        (0% потеря)
Приблизительное время приема-передачи в мс:
    Минимальное = 18мсек, Максимальное = 20 мсек, Среднее = 19 мсек

C:\Study\Пимоспс\6_семестр\ЛР2>ping vk.com >>ping.txt
```



## 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.

```
C:\Study\laboratory-works>dir|sort /R
23.03.2022 12:23 <DIR> 5 Семестр
23.03.2022 11:51 <DIR> 6 Семестр
23.03.2022 11:45 <DIR> ..
23.03.2022 11:45 <DIR> .
23.03.2022 11:39 2 README.md
23.03.2022 11:38 <DIR> 4 Семестр
23.03.2022 11:38 <DIR> 3 Семестр
23.03.2022 11:38 <DIR> 2 семестр
23.03.2022 11:38 <DIR> 1 семестр
Том в устройстве C не имеет метки.
Содержимое папки C:\Study\laboratory-works
Серийный номер тома: 7C8E-3BE9
      8 папок 47 352 127 488 байт свободно
      1 файлов                2 байт
```

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 выполняются 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.

```
C:\Study\ПимОСПС> DIR & PAUSE & COPY /?
Том в устройстве C не имеет метки.
Серийный номер тома: 7C8E-3BE9

Содержимое папки C:\Study\ПимОСПС

23.03.2022  11:25    <DIR>          .
23.03.2022  11:25    <DIR>          ..
27.02.2022  12:08    <DIR>          5_семестр
16.03.2022  14:28    <DIR>          6_семестр
28.02.2022  14:11             104 448 cmd_PS_минимальный_набор.doc
14.11.2017  10:53             61 952 cmd_перенапр_фильтры.doc
17.02.2021  20:13    <DIR>          EXAMPLES
28.02.2022  10:44             45 568 LAB2_задание.doc
16.03.2022  14:25    <DIR>          New_Folder
28.09.2017  11:21             47 908 903 Введение в Windows PowerShell А.Попов.pdf
              4 файлов             48 120 871 байт
              6 папок             47 402 074 112 байт свободно
Для продолжения нажмите любую клавишу . . .
Копирование одного или нескольких файлов в другое место.

COPY [/D] [/V] [/N] [/Y | /-Y] [/Z] [/L] [/A | /B] источник [/A | /B]
      [+ источник [/A | /B] [+ ...]] [результат [/A | /B]]

источник      Имена одного или нескольких копируемых файлов.
/A            Файл является текстовым файлом ASCII.
/B            Файл является двоичным файлом.
/D            Указывает на возможность создания зашифрованного файла
результат     Каталог и/или имя для конечных файлов.
/V            Проверка правильности копирования файлов.
/N            Использование, если возможно, коротких имен при копировании
              файлов, чьи имена не удовлетворяют стандарту 8.3.
/Y            Подавление запроса подтверждения на перезапись существующего
              конечного файла.
/-Y           Обязательный запрос подтверждения на перезапись существующего
              конечного файла.
/Z            Копирование сетевых файлов с возобновлением.
/L            Если источник является символической ссылкой, копирование
              ссылки вместо реального файла, на который указывает ссылка.

Ключ /Y можно установить через переменную среды COPYCMD.
Ключ /-Y командной строки переопределяет такую установку.
По умолчанию требуется подтверждение, если только команда COPY
не выполняется в пакетном файле.

Чтобы объединить файлы, укажите один конечный и несколько исходных файлов,
используя подстановочные знаки или формат "файл1+файл2+файл3+...".
```

## 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
```

```
C:\Study>to_github Сети\ЛР1 Добавлена_ЛР1_по_Сетям
C:\Study\Сети\ЛР1\Сети_и_телекоммуникации_19-В-2_ЛР1_Сухоруков.docx
C:\Study\Сети\ЛР1\Сети_и_телекоммуникации_19-В-2_ЛР1_Сухоруков.pdf
Скопировано файлов: 2.
Для продолжения нажмите любую клавишу . . .
[main 98834183] Добавлена_ЛР1_по_Сетям
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 "6 Semestr/\320\241\320\265\321\202\320\270/\320\233\320\2401/\320\241\320\265\321\202\320\270_\320\270_\3
320\270\320\270_19-\320\222-2_\320\233\320\2401_\320\241\321\203\321\205\320\276\321\200\321\203\320\272\320\276\320\262.docx"
 create mode 100644 "6 Semestr/\320\241\320\265\321\202\320\270/\320\233\320\2401/\320\241\320\265\321\202\320\270_\320\270_\3
320\270\320\270_19-\320\222-2_\320\233\320\2401_\320\241\321\203\321\205\320\276\321\200\321\203\320\272\320\276\320\262.pdf"
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 12 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (7/7), 975.07 KiB | 195.01 MiB/s, done.
Total 7 (delta 2), reused 4 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Valery-S/laboratory-works
 846ed61b..98834183 main -> main
C:\Study\laboratory-works>
```

### *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\\*. \*" /s** 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.

```
PS C:\Users\Валерий> Get-Help -Name Get-Service

NAME
    Get-Service

SYNTAX
    Get-Service [[-Name] <string[]>] [-DependentServices] [-RequiredServices] [-Include <string[]>] [-Exclude <string[]>] [<CommonParameters>]

    Get-Service -DisplayName <string[]> [-DependentServices] [-RequiredServices] [-Include <string[]>] [-Exclude <string[]>] [<CommonParameters>]

    Get-Service [-DependentServices] [-RequiredServices] [-Include <string[]>] [-Exclude <string[]>] [-InputObject <ServiceController[]>] [<CommonParameters>]

ALIASES
    gsv

REMARKS
    Get-Help cannot find the Help files for this cmdlet on this computer. It is displaying only partial help.
    -- To download and install Help files for the module that includes this cmdlet, use Update-Help.
    -- To view the Help topic for this cmdlet online, type: "Get-Help Get-Service -Online" or
       go to https://go.microsoft.com/fwlink/?LinkID=2096496.
```

### 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.

```
PS C:\Users\Валерий> [int]$TestVar = 100
PS C:\Users\Валерий> Get-Variable

Name                           Value
----                           -
$                               100
?                               True
^                               [
args                            {}
ConfirmPreference              High
DebugPreference                SilentlyContinue
EnabledExperimentalFeatures    {}
Error                          {The term 'cla' is not recognized as a name of a cmdlet, function, script file, or executable p...
ErrorActionPreference          Continue
ErrorView                      ConciseView
ExecutionContext               System.Management.Automation.EngineIntrinsics
false                           False
FormatEnumerationLimit         4
HOME                            C:\Users\Валерий
Host                            System.Management.Automation.Internal.Host.InternalHost
InformationPreference           SilentlyContinue
input                           System.Collections.ArrayList+ArrayListEnumeratorSimple
IsCoreCLR                       True
IsLinux                         False
IsMacOS                         False
IsWindows                      True
MaximumHistoryCount             4096
MyInvocation                   System.Management.Automation.InvocationInfo
NestedPromptLevel               0
null
OutputEncoding                 System.Text.UTF8Encoding
PID                             13880
PROFILE                        C:\Users\Валерий\Documents\PowerShell\Microsoft.PowerShell_profile.ps1
ProgressPreference              Continue
PSBoundParameters               {}
PSCommandPath                   {}
PSCulture                       ru-RU
PSDefaultParameterValues        {}
PSEdition                      Core
PSEmailServer                   C:\Program Files\PowerShell\7
PSScriptRoot                    C:\Program Files\PowerShell\7
PSSessionApplicationName        wsman
PSSessionConfigurationName      http://schemas.microsoft.com/powershell/Microsoft.PowerShell
PSSessionOption                 System.Management.Automation.Remoting.PSSessionOption
PSStyle                         System.Management.Automation.PSStyle
PSUICulture                     ru-RU
PSVersionTable                  {PSVersion, PSEdition, GitCommitId, OS...}
PWD                             C:\Users\Валерий
ShellId                         Microsoft.PowerShell
StackTrace                      at System.Management.Automation.CommandDiscovery.LookupCommandInfo(String commandName, Comma...
TestVar                         100
true                            True
VerbosePreference               SilentlyContinue
WarningPreference               Continue
WhatIfPreference                False
```

## 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.

```
PS C:\Users\Валерий> Get-Process | Sort-Object | Format-Table
```

NPM(K)	PM(M)	WS(M)	CPU(s)	Id	SI	ProcessName
10	2,04	6,57	0,00	4112	0	AdobeUpdateService
13	3,20	9,90	0,00	4020	0	AGMSERVICE
12	2,93	10,38	0,00	4036	0	AGSSERVICE
19	10,82	26,19	0,17	4572	1	ApplicationFrameHost
13	15,18	19,41	21,84	13776	0	audiodg
18	58,37	107,44	6,41	14188	1	chrome
15	12,32	26,07	0,06	13592	1	chrome
19	7,60	20,07	6,28	13008	1	chrome
22	382,62	426,01	494,61	12860	1	chrome
19	63,36	113,98	11,12	12284	1	chrome
19	53,59	99,03	2,52	12184	1	chrome
16	29,14	64,33	2,02	11772	1	chrome
15	30,96	65,41	1,30	11360	1	chrome
16	52,89	88,45	2,59	13140	1	chrome
20	80,42	117,07	5,84	11212	1	chrome
10	2,25	7,32	0,00	10996	1	chrome
64	163,34	270,20	136,81	10388	1	chrome

```
PS C:\Users\Валерий> Get-Process | Sort-Object | Format-List
```

```
Id       : 4112
Handles  : 161
CPU      :
SI       : 0
Name     : AdobeUpdateService

Id       : 4020
Handles  : 220
CPU      :
SI       : 0
Name     : AGMSERVICE

Id       : 4036
Handles  : 219
CPU      :
SI       : 0
Name     : AGSSERVICE

Id       : 4572
Handles  : 311
CPU      : 0,171875
SI       : 1
Name     : ApplicationFrameHost

Id       : 13776
Handles  : 504
CPU      : 22,640625
SI       : 0
Name     : audiodg
```

## 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.

```
PS C:\Users\Валерий> "This is a test" | Export-Clixml -Path .\sample.xml
PS C:\Users\Валерий> Get-Content -Path .\sample.xml
<Objs Version="1.1.0.1" xmlns="http://schemas.microsoft.com/powershell/2004/04">
  <S>This is a test</S>
</Objs>
```

## 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.

```
PS C:\Users\Валерий> Get-NetIPInterface
```

ifIndex	InterfaceAlias	AddressFamily	NlMtu(Bytes)	InterfaceMetric	Dhcp	ConnectionState	PolicyStore
8	VirtualBox Host-Only Network	IPv6	1500	25	Enabled	Connected	ActiveStore
5	Ethernet	IPv6	1500	25	Enabled	Connected	ActiveStore
1	Loopback Pseudo-Interface 1	IPv6	4294967295	75	Disabled	Connected	ActiveStore
8	VirtualBox Host-Only Network	IPv4	1500	25	Disabled	Connected	ActiveStore
5	Ethernet	IPv4	1500	25	Enabled	Connected	ActiveStore
1	Loopback Pseudo-Interface 1	IPv4	4294967295	75	Disabled	Connected	ActiveStore

## 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.

```
PS C:\Users\Валерий> New-Item -Path . -Name "testfile1.txt" -ItemType "file" -Value "This is a text string"

Directory: C:\Users\Валерий

Mode                LastWriteTime         Length Name
----                -
-a---             24.03.2022   13:52             22 testfile1.txt

PS C:\Users\Валерий> New-Item -Path . -Name "logfiles" -ItemType "directory"

Directory: C:\Users\Валерий

Mode                LastWriteTime         Length Name
----                -
d---             24.03.2022   13:52             logfiles

PS C:\Users\Валерий> Set-Content -Path .\testfile1.txt -value qwerty
```

## 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.

```
PS C:\Users\Валерий> $source="C:\Study\laboratory-works"
PS C:\Users\Валерий> Get-ChildItem $source -recurse -force | where {$_.psIscontainer} | foreach {
>> $count = Get-ChildItem $_.fullname -recurse | where {$_.length} | Measure-Object -property length -Sum
>> $FileSize = '{0:F}' -f (((($count.Sum)/1024)/1024)
>> '$_' + $_.FullName + ' Files: ' + $count.count + ' Size: ' + $FileSize + ' MB' + ' ' | Out-File D:\Count.txt -Append
>> }
```

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

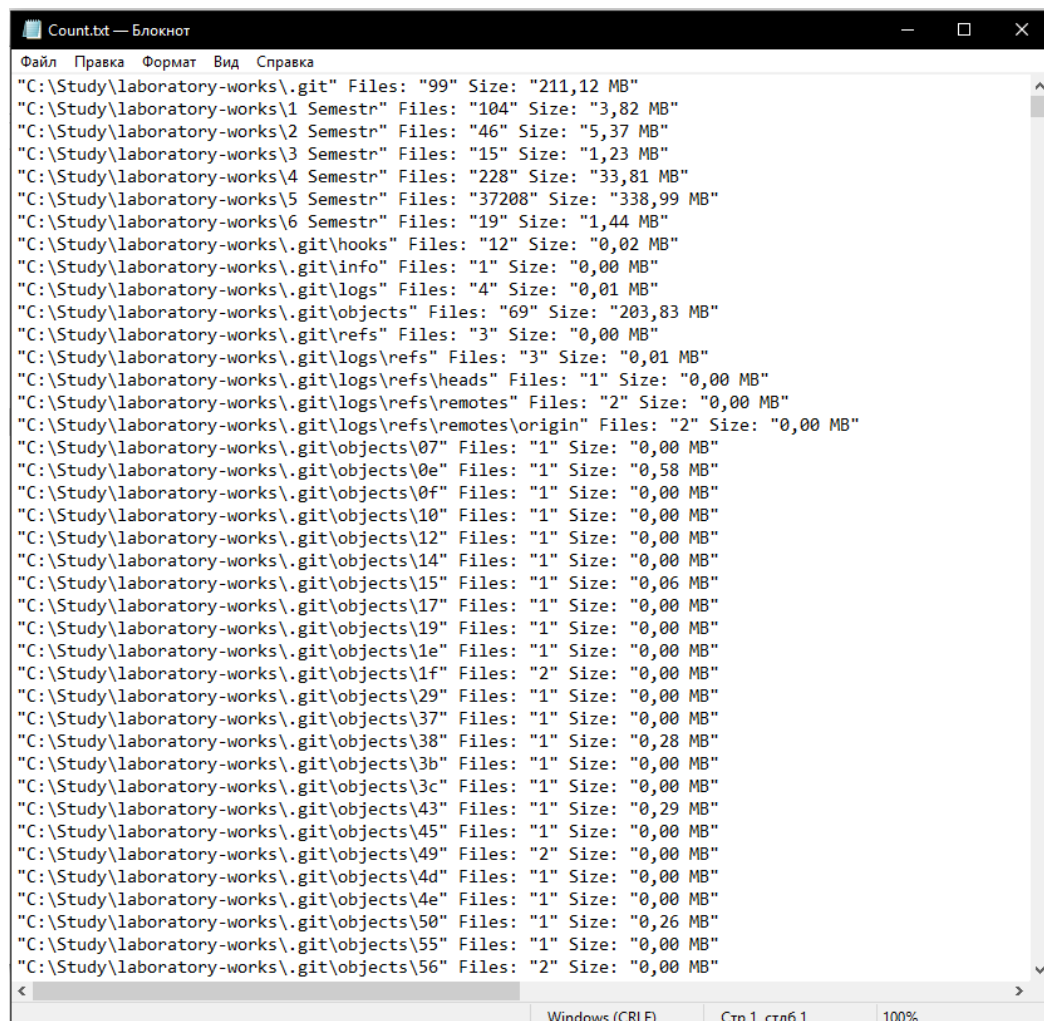
```
Get-ChildItem $source -recurse -force | where {$_.psIscontainer} |
foreach {
```

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

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

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

```
}
```



```
Count.txt — Блокнот
Файл Правка Формат Вид Справка
"C:\Study\laboratory-works\.git" Files: "99" Size: "211,12 MB"
"C:\Study\laboratory-works\1 Semestr" Files: "104" Size: "3,82 MB"
"C:\Study\laboratory-works\2 Semestr" Files: "46" Size: "5,37 MB"
"C:\Study\laboratory-works\3 Semestr" Files: "15" Size: "1,23 MB"
"C:\Study\laboratory-works\4 Semestr" Files: "228" Size: "33,81 MB"
"C:\Study\laboratory-works\5 Semestr" Files: "37208" Size: "338,99 MB"
"C:\Study\laboratory-works\6 Semestr" Files: "19" Size: "1,44 MB"
"C:\Study\laboratory-works\.git\hooks" Files: "12" Size: "0,02 MB"
"C:\Study\laboratory-works\.git\info" Files: "1" Size: "0,00 MB"
"C:\Study\laboratory-works\.git\logs" Files: "4" Size: "0,01 MB"
"C:\Study\laboratory-works\.git\objects" Files: "69" Size: "203,83 MB"
"C:\Study\laboratory-works\.git\refs" Files: "3" Size: "0,00 MB"
"C:\Study\laboratory-works\.git\logs\refs" Files: "3" Size: "0,01 MB"
"C:\Study\laboratory-works\.git\logs\refs\heads" Files: "1" Size: "0,00 MB"
"C:\Study\laboratory-works\.git\logs\refs\remotes" Files: "2" Size: "0,00 MB"
"C:\Study\laboratory-works\.git\logs\refs\remotes\origin" Files: "2" Size: "0,00 MB"
"C:\Study\laboratory-works\.git\objects\07" Files: "1" Size: "0,00 MB"
"C:\Study\laboratory-works\.git\objects\0e" Files: "1" Size: "0,58 MB"
"C:\Study\laboratory-works\.git\objects\0f" Files: "1" Size: "0,00 MB"
"C:\Study\laboratory-works\.git\objects\10" Files: "1" Size: "0,00 MB"
"C:\Study\laboratory-works\.git\objects\12" Files: "1" Size: "0,00 MB"
"C:\Study\laboratory-works\.git\objects\14" Files: "1" Size: "0,00 MB"
"C:\Study\laboratory-works\.git\objects\15" Files: "1" Size: "0,06 MB"
"C:\Study\laboratory-works\.git\objects\17" Files: "1" Size: "0,00 MB"
"C:\Study\laboratory-works\.git\objects\19" Files: "1" Size: "0,00 MB"
"C:\Study\laboratory-works\.git\objects\1e" Files: "1" Size: "0,00 MB"
"C:\Study\laboratory-works\.git\objects\1f" Files: "2" Size: "0,00 MB"
"C:\Study\laboratory-works\.git\objects\29" Files: "1" Size: "0,00 MB"
"C:\Study\laboratory-works\.git\objects\37" Files: "1" Size: "0,00 MB"
"C:\Study\laboratory-works\.git\objects\38" Files: "1" Size: "0,28 MB"
"C:\Study\laboratory-works\.git\objects\3b" Files: "1" Size: "0,00 MB"
"C:\Study\laboratory-works\.git\objects\3c" Files: "1" Size: "0,00 MB"
"C:\Study\laboratory-works\.git\objects\43" Files: "1" Size: "0,29 MB"
"C:\Study\laboratory-works\.git\objects\45" Files: "1" Size: "0,00 MB"
"C:\Study\laboratory-works\.git\objects\49" Files: "2" Size: "0,00 MB"
"C:\Study\laboratory-works\.git\objects\4d" Files: "1" Size: "0,00 MB"
"C:\Study\laboratory-works\.git\objects\4e" Files: "1" Size: "0,00 MB"
"C:\Study\laboratory-works\.git\objects\50" Files: "1" Size: "0,26 MB"
"C:\Study\laboratory-works\.git\objects\55" Files: "1" Size: "0,00 MB"
"C:\Study\laboratory-works\.git\objects\56" Files: "2" Size: "0,00 MB"
Windows (CRLF) Стр 1, столб 1 100%
```

## 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.