МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра вычислительные системы и технологии

Лабораторная работа № 2

Командный язык ОС Windows

и командная оболочка *Windows PowerShell*

ОТЧЕТ

по лабораторной работе

по дисциплине

Принципы и методы  
организации системных программных средств

РУКОВОДИТЕЛЬ:

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

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Сапожников В.О.

19-ИВТ-3

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

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

Нижний Новгород 2021

**Цель работы**

Изучить состав команд и функциональные возможности *cmd.exe* и MS *PowerShell*, провести сравнение.

**Задание 1**

Ознакомиться с эволюцией, достоинствами, недостатками, особенностями командных оболочек от *Microsoft.*

Любую ОС можно представить в виде ядра системы, которое имеет доступ к аппаратуре и управляет файлами и процессами и оболочки (командного интерпретатора) с утилитами, которые позволяют пользователю получить доступ к функциональности ядра ОС. В ОС Windows очень долгое время в качестве командной оболочки использовалась *cmd.exe*, которая расширяет возможности оболочки command.com ОС MS-DOS, а та в свою очередь представляет собой значительно упрощенный и урезанный вариант оболочки UNIX-систем.

Windows всегда проигрывала UNIX-системам в количестве и функциональных возможностях стандартных утилит командной строки, а так же в качестве и полноте встроенной справочной информации по командам оболочки. Данную проблему пытались решить при помощи утили Windows Resource Kit, проблему автоматизации в ОС решали при помощи Windows Script Host (WSH), оболочки WMI Command-line (WMIC).

И того ОС система Windows 3 разных инструмента для автоматизации работы: оболочки командной строки *cmd.exe*, WMIC и WSH, каждая из которых имела ряд серьезных недостатков.

|  |  |  |  |
| --- | --- | --- | --- |
| **Требования** | **Cmd.exe** | **WSH** | **WMIC** |
| Работа во всех версиях ОС без установки дополнительного ПО | **Да** | **Да** | **Нет** (только Windows XP и выше) |
| Интеграция с командной строкой | **Да** | **Нет** | **ДА** |
| Согласованный и непротиворечивый синтаксис команд и утилит | **Нет** | **Нет** | **Да** |
| Поддержка псевдонимов для длинных команд | **Нет** | **Нет** | **Да** |
| Автоматическое завершение команд и имен файлов при вводе их с клавиатуры | **Частично** | **Нет** | **Нет** |
| Наличие подробной справки по командам с примерами | **Частично** | **Нет** | **Да** |
| Возможность автоматического выполнение сценариев | **Да** (язык командных файлов) | **Да** (язык сценариев VBScript, Jscript и т.д.) | **Частично** (команды можно встраивать в командные файлы) |
| Доступ и использование всех технологий и возможностей поддерживаемых ОС | **Нет** (нет прямого доступа к объектам COM, WMI, ADSI, .NET) | **Да** | **Нет** (доступ только к объектам WMI) |

*PowerShell* разрабатывалась как более мощная среда выполнения сценариев и работы из командной строки. В первую очередь данная среда должна была подходить для решения задач, стоящих перед системными администраторами, а также удовлетворять требованиям разработчиков ПО, предоставляя им средства для быстрой реализации интерфейсов управления создаваемыми приложениями.

Для достижения этих целей были решены следующие задача:

* **Обеспечение прямого доступа из командной строки к объектам COM, WMI и .NET**. В новой оболочке присутствуют команды, позволяющие в интерактивном режиме работать с COM-объектами, а также с экземплярами классов, определенных в информационных схемах WMI и .NET.
* **Организация работы с произвольными источниками данных в командной строке по принципу файловой системы.** Например, навигация по системному реестру или хранилищу цифровых сертификатов выполняется из командной строки с помощью аналога команды cd интерпретатор *cmd.exe*.
* **Разработка интуитивно понятной унифицированной структуры встроенных команд, основанной на их функциональном назначении.** В новой оболочке имена всех внутренних команд (в *PowerShell* они называются командлетами) соответствуют шаблону “глагол-существительное” (Get-Process, Clear-Host, Stop-Service). Для одинаковых параметров внутренних команд используются стандартные имена, структура параметров во всех командах идентична, все команды обрабатываются одним синтаксическим анализатором. В результате облегчается изучение и запоминание команд.
* **Обеспечение возможности расширения встроенного набора команд.** Внутренние команды *PowerShell* могут дополняться командами, создаваемыми пользователем. При этом они полностью интегрируются в оболочку, информация о них может быть получена из стандартной справочной системы PowerShell.
* **Организация поддержки знакомых команд из других оболочек.** В *PowerShell* на уровне псевдонимов собственных внутренних команд поддерживаются наиболее используемые стандартные команды из оболочек *cmd.exe* и UNIX-систем (например, введя команду ls или dir пользователь получит ожидаемый результат: список всех файлов в текущем каталоге).
* **Разработка дополнительной встроенной справочной системы для внутренних команд.** Для большинства внутренних команд в справочной системе дано подробное описание и примеры использования. В любом случае встроенная справка по любой внутренней команде будет содержать краткое описание всех ее параметров.
* **Реализация автоматического завершения при вводе с клавиатуры имен команд, их параметров, а также имен файлов и папок.** Данная возможность, значительно упрощает и ускоряет ввод команд с клавиатуры.

Одной из ключевых особенностей *PowerShell* среди других командных оболочек является ориентация на объекты. При создании *PowerShell* разработчики решили не изобретать ничего нового и воспользоваться унифицированной объектной моделью .NET. Данный выбор был сделан по следующим причинам:

Во-первых, платформа .NET повсеместно используется при разработке программного обеспечения для Windows и предоставляет, в частности, общую информационную систему с помощью которой разные компоненты операционной системы могут обмениваться данными друг с другом.

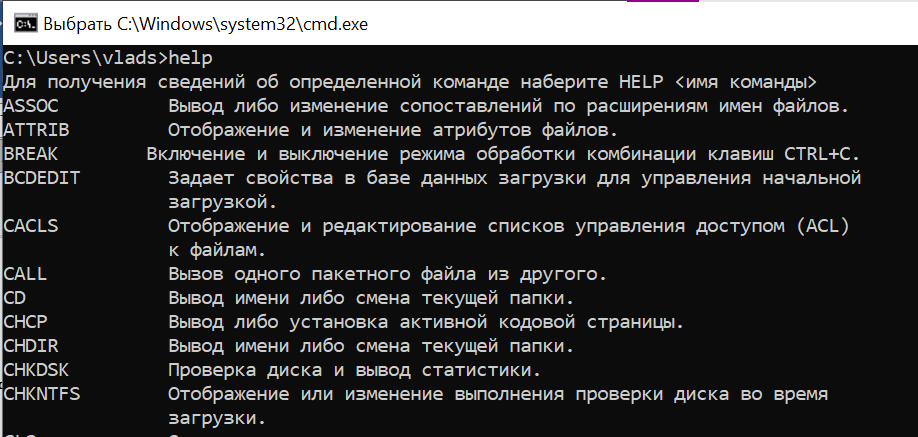
Во-вторых, объектная модель .NET является самодокументированной: каждый объект .NET содержит информации о своей структуре. При интерактивной работе это очень полезно, так как появляется возможность непосредственно из командной строки выполнять запрос к определенному объекту и увидеть описание его свойств и методов, т.е. понять, какие именно манипуляции можно проделать с объектом, не изучая дополнительную документацию с его описанием.

В-третьих, работая в оболочке с объектами, можно с помощью их свойств и методов легко получать нужные данные, не занимаясь разбором и анализом символьной информации, как это происходит во всех традиционных оболочках командной строки, ориентированных на текст.

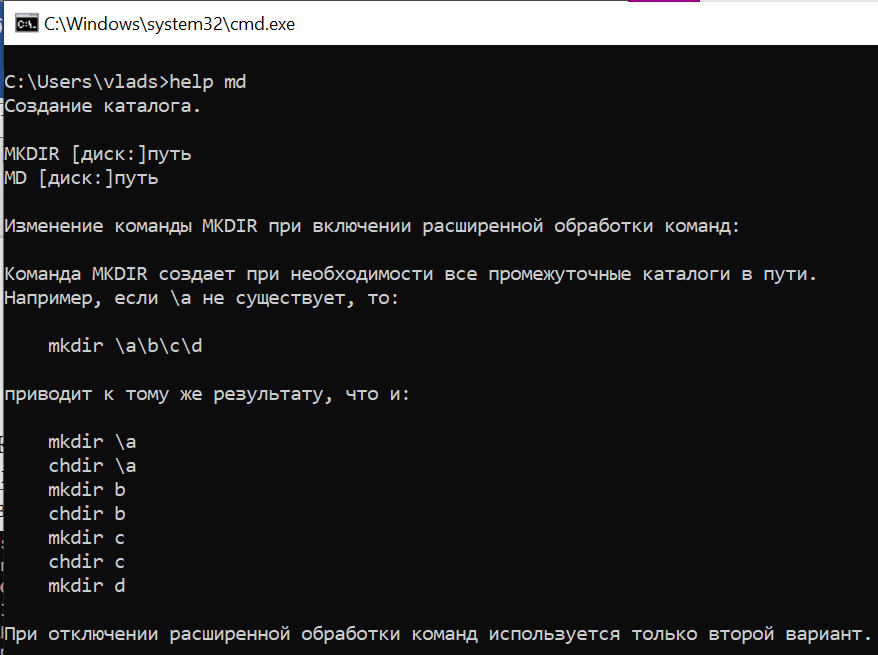
**Задание 2**

Изучить состав команд командного процессора *cmd.exe.*

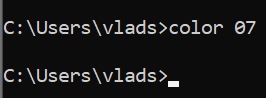
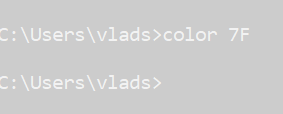
Команда help без параметров дает общие сведения об основных командах командой строки

****

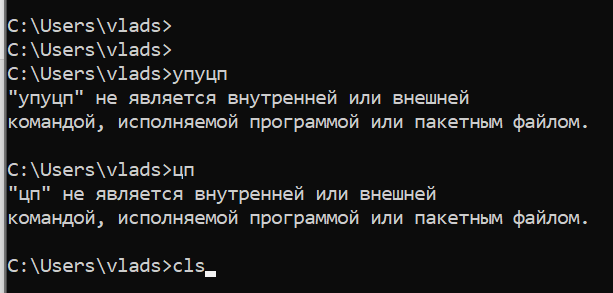
Используя данную команду с параметром – именем другой команды help <команда>, можно получить детальный сведения о переданной команде.



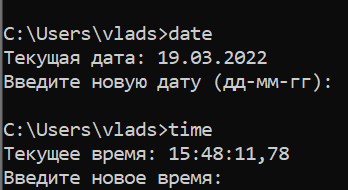
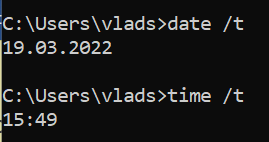
Команда color <цвет1><цвет2> используется для задания цветов вывода консоли. Атрибуты цветов задаются в виде двух шестнадцатеричных цифр, первая задает цвет фона, вторая определяет цвет переднего плана.



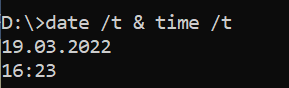
Команда cls – очистка содержимого экрана



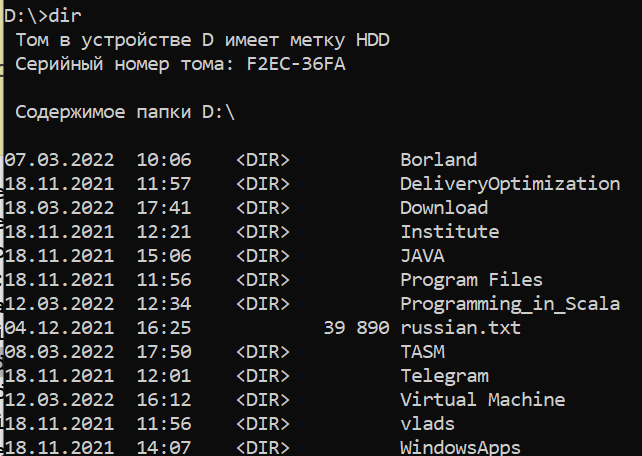
Команды date и time без параметров отображают текущую дату и время соответственно и запрашивают ввод новой даты, времени. При задании ключа /t данным командам отображаются текущие даты и время без запроса ввода.

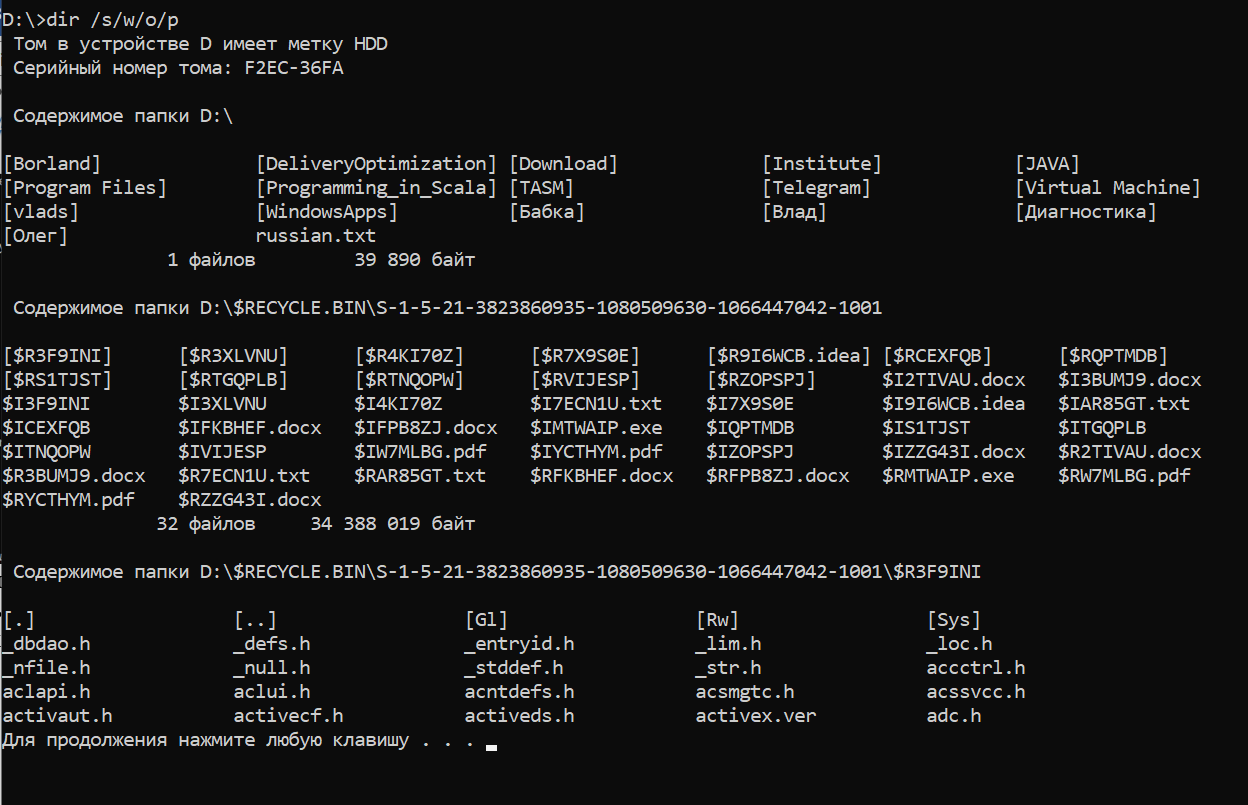
Командная оболочка *cmd.exe* так же поддерживает выполнение сразу нескольких команд. При этом команды перечисляются через оператор & И

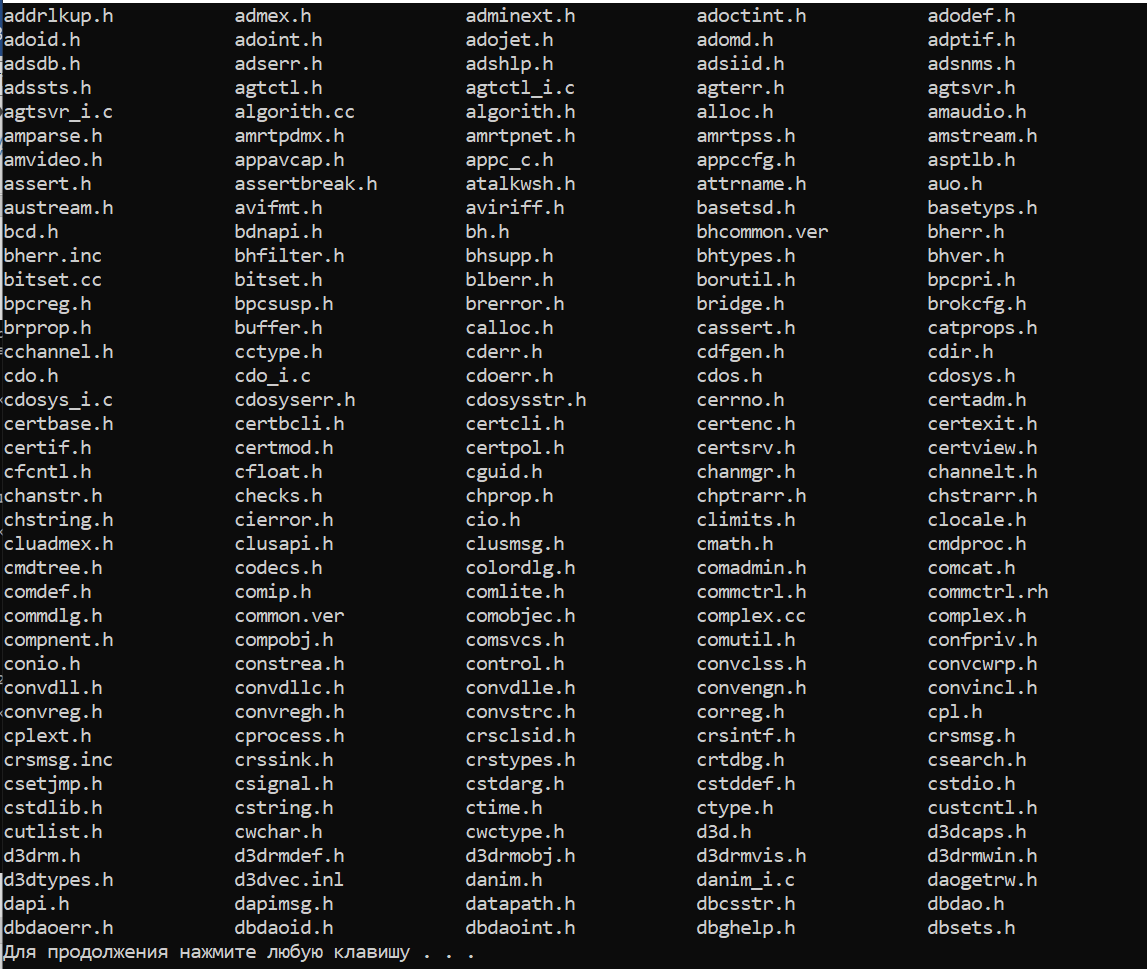


Команда dir выводит список файлов и подкаталогов в указанном каталоге.



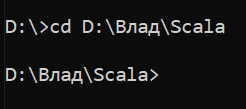
Данная команда имеет множество флагов и параметров. Например, для того, чтобы осуществить циклический обход всех папок внутри каталога, отобразить список в виде столбцов, отсортировать по списку отображаемых файлов и выполнить постраничный вывод, команде dir необходимо передать следующие ключи: /s/w/o/p





и т.д.

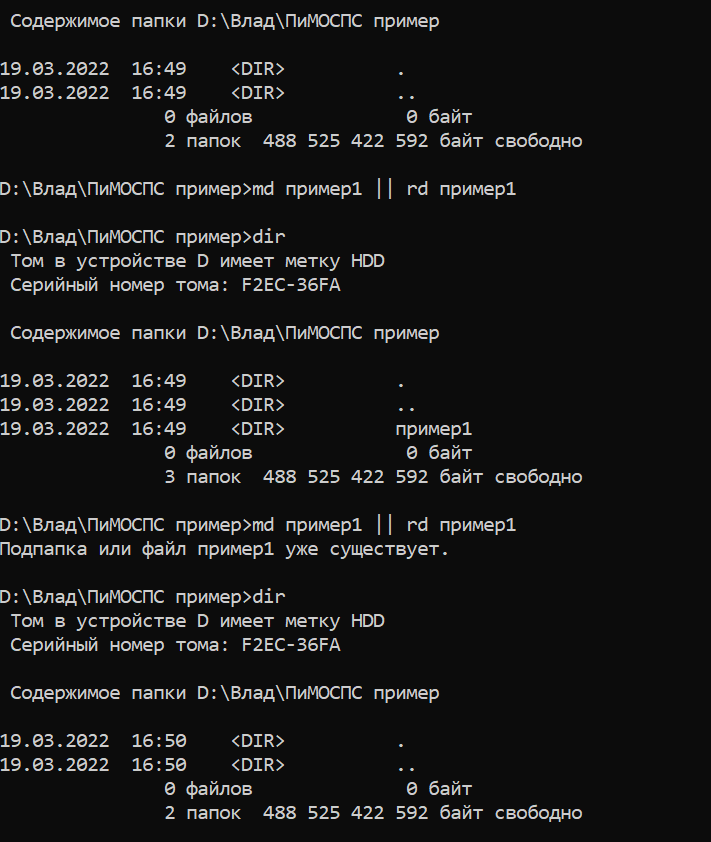
Команда cd выводит имя или изменяет текущий каталог.



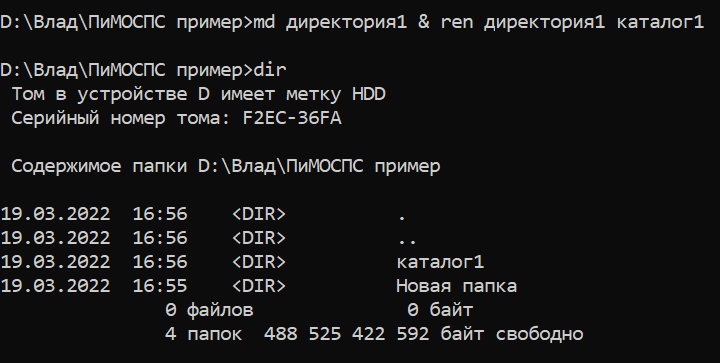
Команда md – создает каталог

Команде rd – удаляет каталог

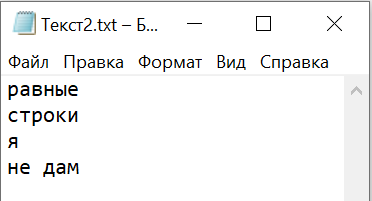
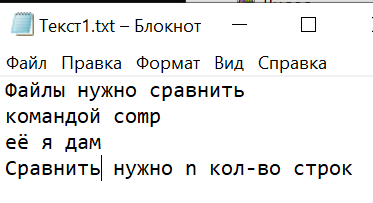
Условный оператор || запускает команду, стоящую за символом ||, только если команда, стоящая перед символом || не была выполнена. Пример: создаем каталог под именем пример1, если такой каталог есть, то удаляем его.

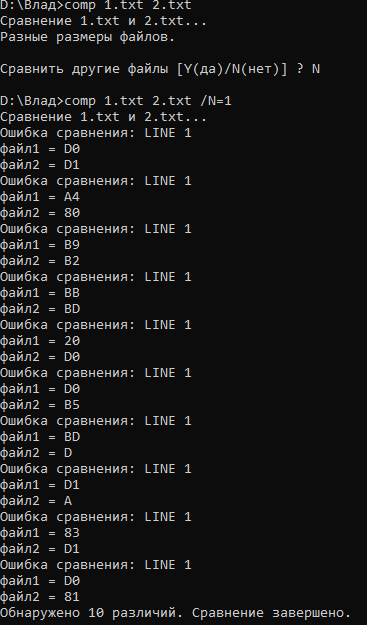


Команда ren <имя1> <имя2> выполняет переименование файла с <имя1> на <имя2>. Пример: создаем директорию1, а затем переименовываем директорию1 в каталог1

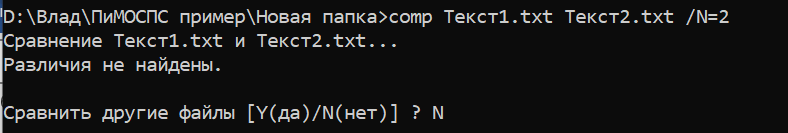


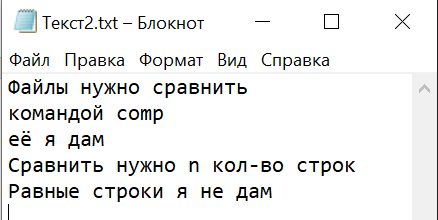
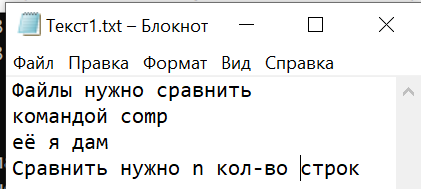
Команда comp выполняет сравнение содержимого двух файлов или двух наборов файлов.



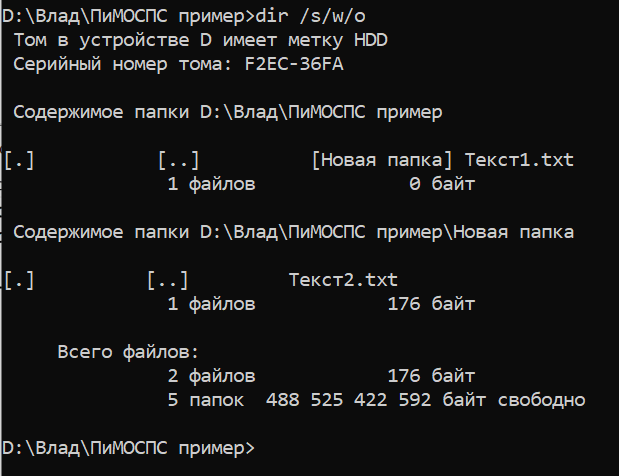


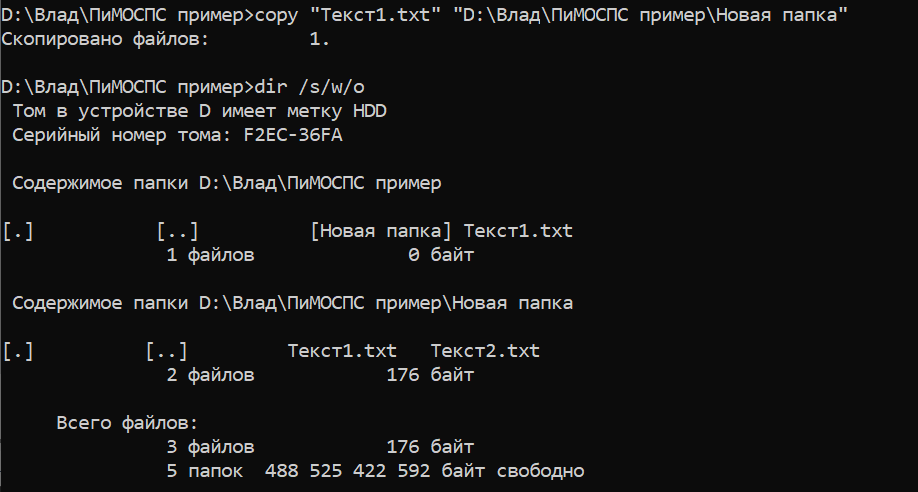
При передаче ключа /N=число можно сравнить первые n строк в файлах



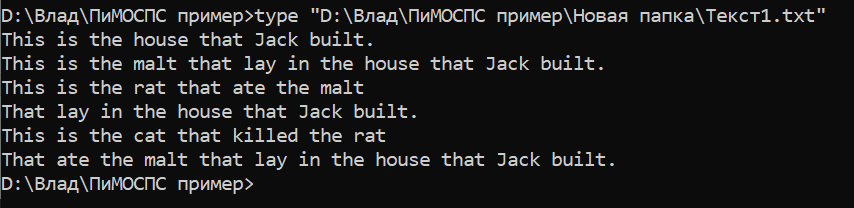


Команда copy выполняет копирование одного или нескольких файлов в другое место.

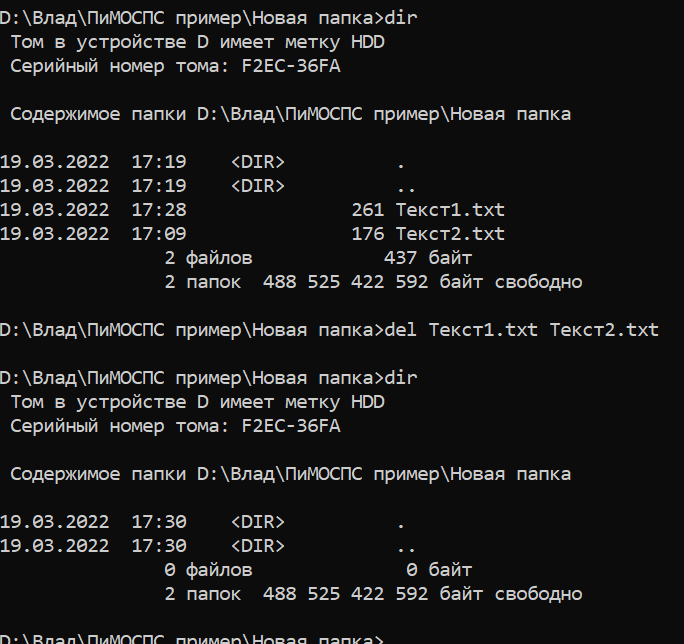




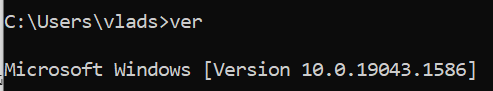
Команда type выводит содержимое одного или нескольких текстовых файлов



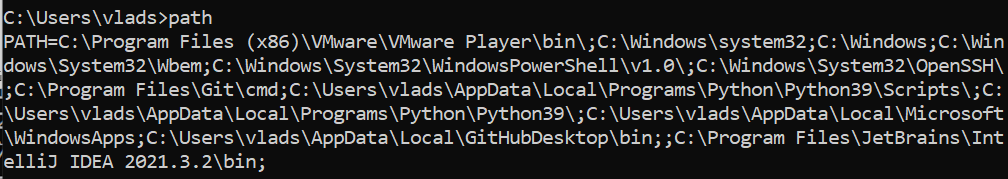
Команда del выполняет удаление одного или нескольких файлов



Команда ver выводит версию windows



Команда path выполняет вывод или задание пути поиск исполняемых файлов.

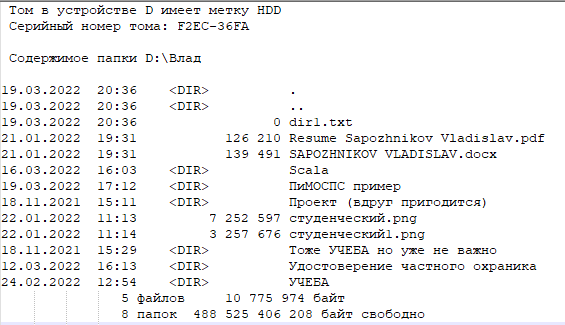


Команда оболочка *cmd.exe* так же поддерживает перенаправление ввода вывода.

|  |  |
| --- | --- |
| **Оператор перенаправления** | **Описание** |
| > | Записывает данные на выходе команды вместо командной строки в файл или на устройство, например, на принтер. |
| < | Читает поток входных данных команды из файла, а не с клавиатуры. |
| >> | Добавляет выходные данные команды в конец файла, не удаляя при этом существующей информации из файла. |
| >& | Считывает данные на выходе одного дескриптора как входные данные для другого дескриптора. |
| <& | Считывает входные данные одного дескриптора как выходные данные другого дескриптора. |
| | | Считывает входные данные одной команды и записывает их на вход другой команды. Эта процедура известна под название конвейеризация. |

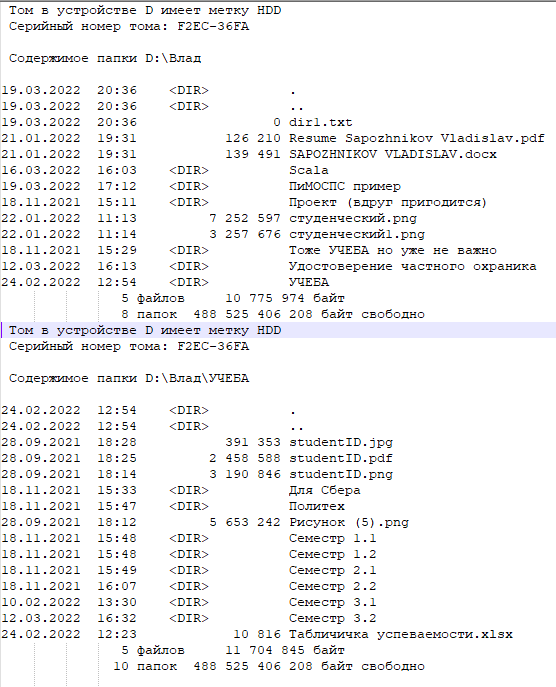
Запись результат работы команды dir в файл 1.txt





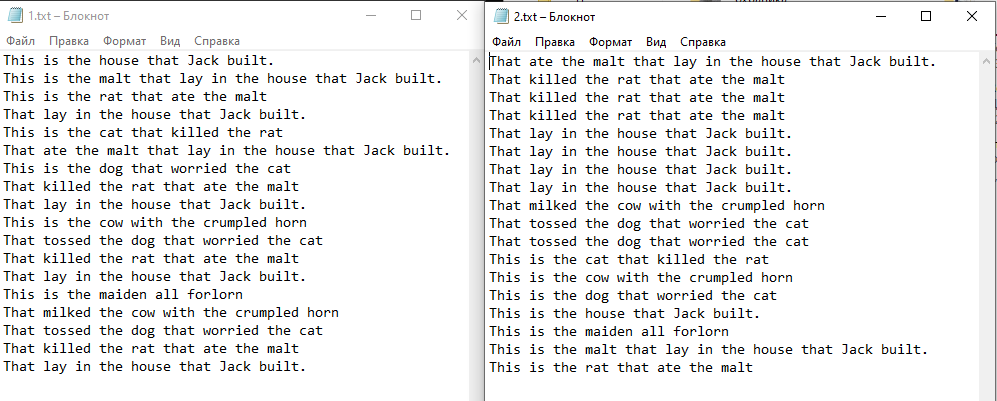
Приписывание результата выполнении команды dir УЧЕБА в файл 1.txt



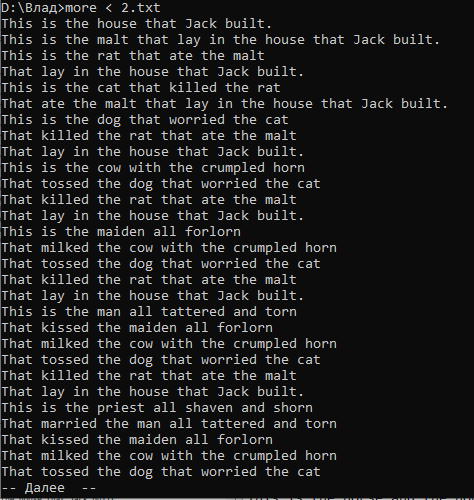


Сортировка файла 1.txt в лексическом порядке и запись результата в 2.txt

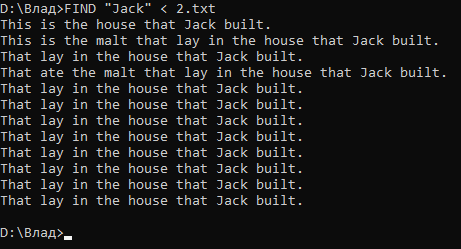




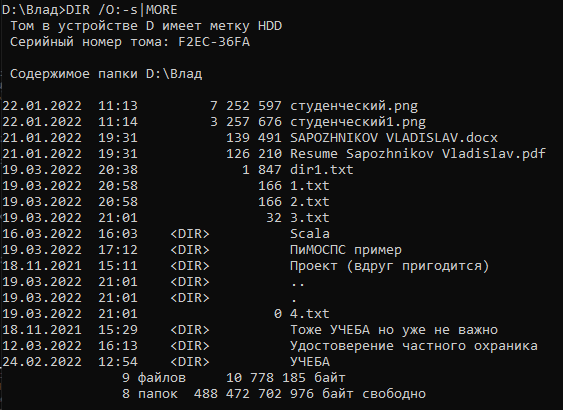
More – последовательный вывод данных по частям размером в один экран.



Команда find выполняет поиск текстовой строки в одном или нескольких файлах.



Сортировка по размеру от большего к меньшему и передача команде more



Так же командный интерпретатор *cmd.exe* может выполнять командные файлы.

Командным файлом называется текстовый файл, содержащий последовательность команд для исполнения командным интерпретатором. После запуска командного файла программ-интерпретатор читает его строка за строкой и последовательно исполняет команды.

Пример просто *.bat* файла для записи данных о системе в файл переданный в качестве аргументапрограмме *.rtf:*

@echo off

if /%1/ == // goto error1

title SYSTEM INFORMATION

date /T> %1

time /T>> %1

systeminfo>> %1

goto end

type %1

:error1

echo No args

:end

info.rtf

19.03.2022

Имя узла: DESKTOP-3RV1MEH

Название ОС: Майкрософт Windows 10 Домашняя для одного языка

Версия ОС: 10.0.19043 Н/Д построение 19043

Изготовитель ОС: Microsoft Corporation

Параметры ОС: Изолированная рабочая станция

Сборка ОС: Multiprocessor Free

Зарегистрированный владелец: vlad.sapozhnickof@gmail.com

Зарегистрированная организация:

Код продукта: 00327-60000-00000-AA448

Дата установки: 17.11.2021, 22:41:41

Время загрузки системы: 18.03.2022, 19:47:54

Изготовитель системы: LENOVO

Модель системы: 10130

Тип системы: x64-based PC

Процессор(ы): Число процессоров - 1.

[01]: Intel64 Family 6 Model 60 Stepping 3 GenuineIntel ~3401 МГц

Версия BIOS: LENOVO IEKT17AUS, 04.09.2013

Папка Windows: C:\Windows

Системная папка: C:\Windows\system32

Устройство загрузки: \Device\HarddiskVolume7

Язык системы: ru;Русский

Язык ввода: ru;Русский

Часовой пояс: (UTC+03:00) Москва, Санкт-Петербург

Полный объем физической памяти: 8 140 МБ

Доступная физическая память: 3 156 МБ

Виртуальная память: Макс. размер: 11 084 МБ

Виртуальная память: Доступна: 3 560 МБ

Виртуальная память: Используется: 7 524 МБ

Расположение файла подкачки: C:\pagefile.sys

Домен: WORKGROUP

Сервер входа в сеть: \\DESKTOP-3RV1MEH

Исправление(я): Число установленных исправлений - 7.

[01]: KB5010472

[02]: KB5000736

[03]: KB5011487

[04]: KB5006753

[05]: KB5007273

[06]: KB5011352

[07]: KB5005699

Сетевые адаптеры: Число сетевых адаптеров - 4.

[01]: Realtek PCIe GbE Family Controller

Имя подключения: Ethernet

DHCP включен: Да

DHCP-сервер: 192.168.1.1

IP-адрес

[01]: 192.168.1.142

[02]: fe80::dcc7:84ea:d1b6:c4e

[02]: VirtualBox Host-Only Ethernet Adapter

Имя подключения: VirtualBox Host-Only Network

DHCP включен: Нет

IP-адрес

[01]: 192.168.56.1

[02]: fe80::b9a4:53ba:8cb3:a574

[03]: VMware Virtual Ethernet Adapter for VMnet1

Имя подключения: VMware Network Adapter VMnet1

DHCP включен: Да

DHCP-сервер: 192.168.206.254

IP-адрес

[01]: 192.168.206.1

[02]: fe80::18a6:3484:8612:499c

[04]: VMware Virtual Ethernet Adapter for VMnet8

Имя подключения: VMware Network Adapter VMnet8

DHCP включен: Да

DHCP-сервер: 192.168.183.254

IP-адрес

[01]: 192.168.183.1

[02]: fe80::21f8:9bb6:71e4:8402

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

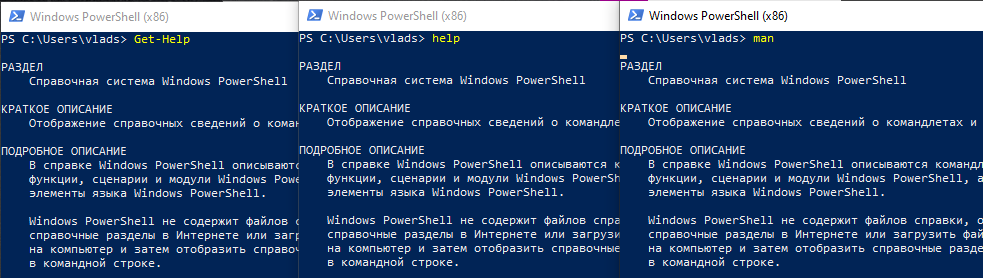
Таким образом командная оболочка cmd.exe предоставляет все необходимые инструменты для выполнения повседневных задач обычным пользователем. Однако, как говорилось ранее, по функциональным возможностям cmd.exe сильно уступает терминалу в UNIX-системах. Конечно, это можно объяснить тем, что Windows ориентирована на широкую аудиторию, включающую рядовых пользователей, в то время как UNIX-системы используются для конкретных целей квалифицированными специалистами. Но, с другой стороны, такие ограничения делают ОС Windows менее привлекательной для пользования её разработчиками ПО и системными администраторами, а в дальнейшем могут мешать развитию ОС в принципе.

**Задание 3**

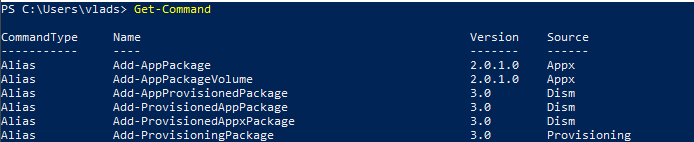
Изучить состав и функциональные возможности *MS PowerShell*.

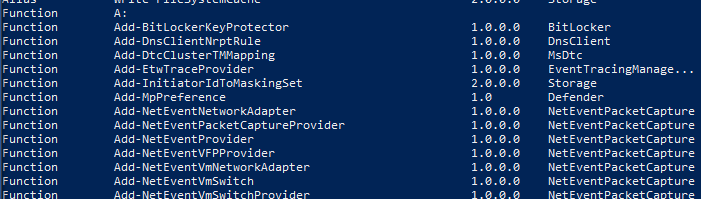
Как и *cmd.exe, PowerShell* обладает большой справочной информацией о самой оболочке и о её командах (в *PowerShell* они называются командлетами это сочетание глагола и существительного, представляющих команду и объект, к которому она применяется).

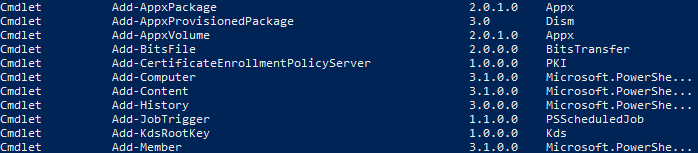
Так же *PowerShell* поддерживает систему синонимов. Для части самых распространенных команд внутри оболочки уже встроены синонимы созвучные командам аналогам из *cmd.exe* и *UNIX*-систем. Например: командлет Get-help имеет синонимы help – из *cmd.exe* и man – из *UNIX*-систем.



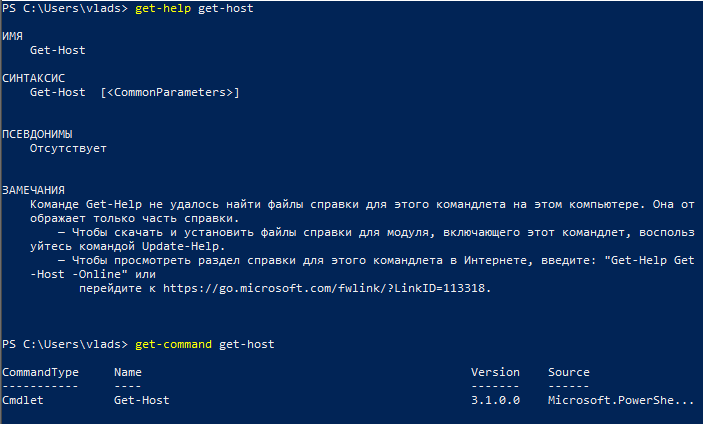
Для получения списка доступных псевдонимов, командлетов и функций в виде Type Name Version Source можно при помощи командлета Get-Command.



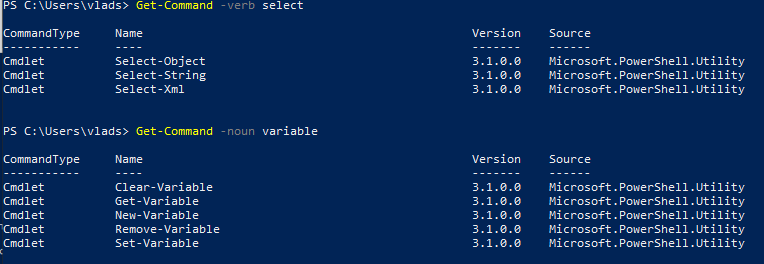




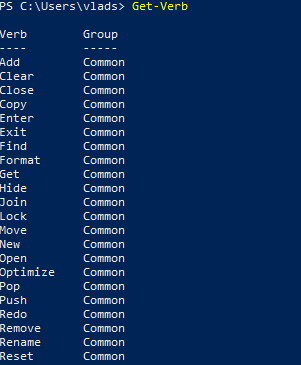
Так же командлеты Get-Command и Get-Help можно использовать к отдельным объектам.



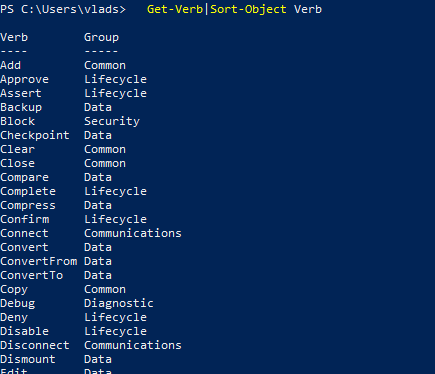
Если мы забыли часть команды, а точнее глагол или существительное, то при помощи команды Get-Command можно получить список командлетов, функций и объектов с указанным глаголом и указав -verb или список команд указав существительное и -noun



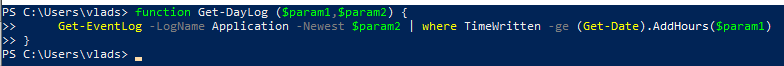
Получить список всех глаголов можно при помощи Get-Verb. Причем список будет сформирован следующим образом: группы будут отсортированы в алфавитном порядке, а уже внутри группу по лафитному порядку будут отсортированы глаголы.

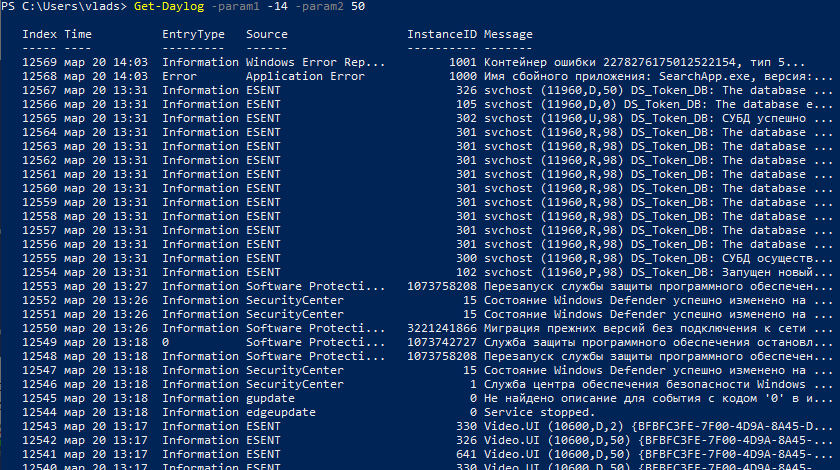


Если мы хотим получить фильтрацию по глаголом, а затем по группам, то следует передать результат Get-Verb на вход Sort-Object (*PowerShell* так же поддерживает конвейеризацию)

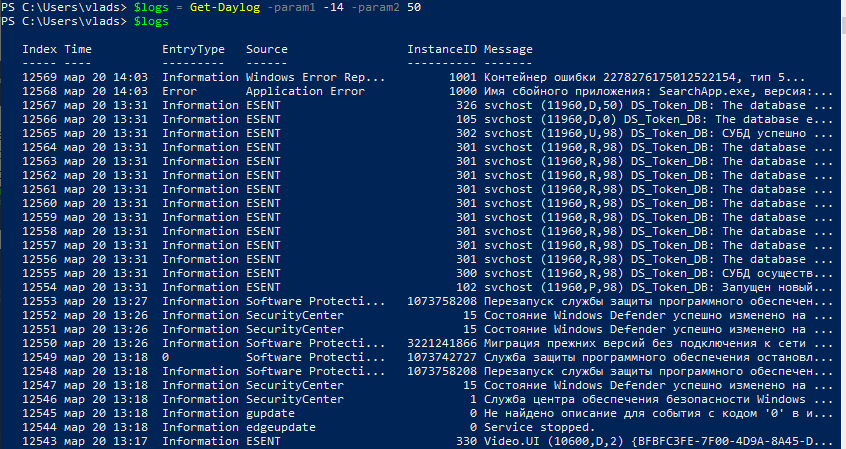


*PowerShell* позволяет создавать свои функции. Например, создадим функцию получения логов, принимающую два параметра: кол-во последних часов, за которые необходимо получить логи и само кол-во логов.





Помимо функций можно создавать и обычные переменные.

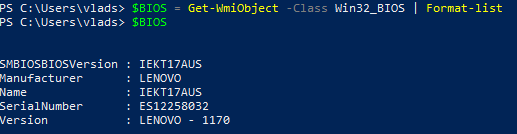


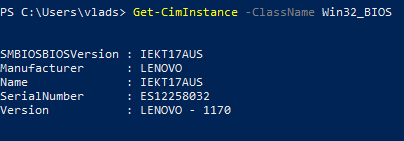
А также и удалять их



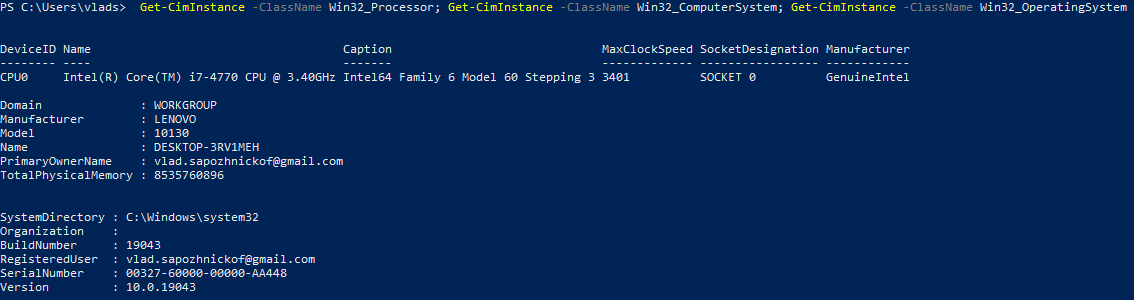
Все основные сведения о системе хранятся в Инструментах Управления Windows - Windows Management Instrumentation (WMI). Для доступа к объектам WMI используются команды Get-WmiObject или Get-CimInstance. Классы WMI имею префикс Win32.

Пример: получения сведения о BIOS



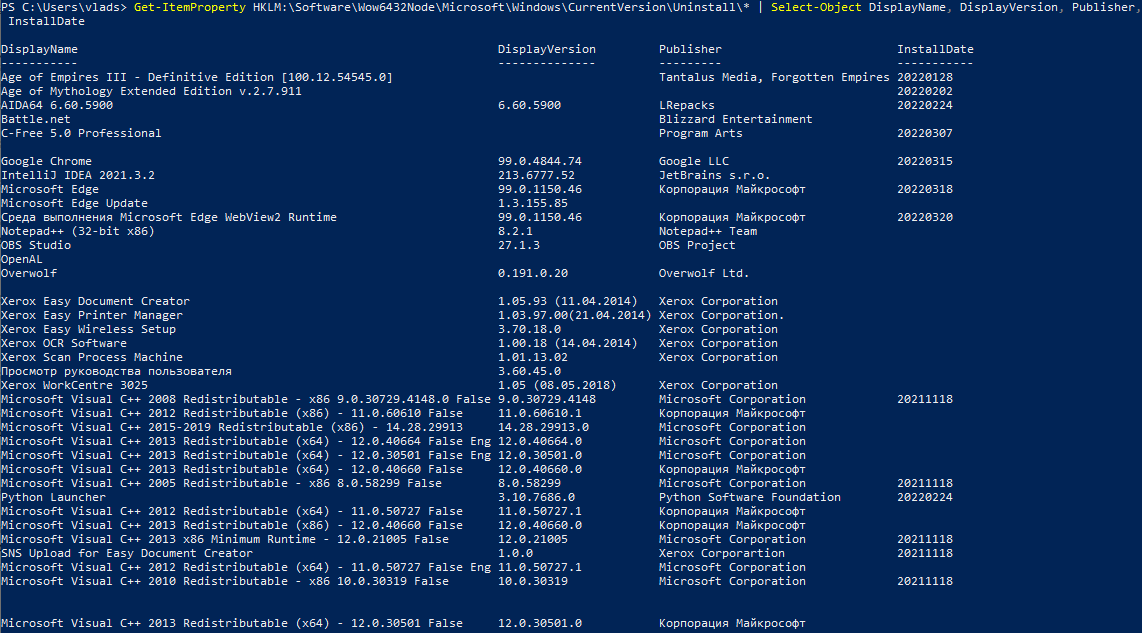


Получить сведения о процессоре, компьютерной системе и ОС можно обратившись к классам Win32\_Processor, Win32\_ComputerSystem, Win32\_OperatingSystem.

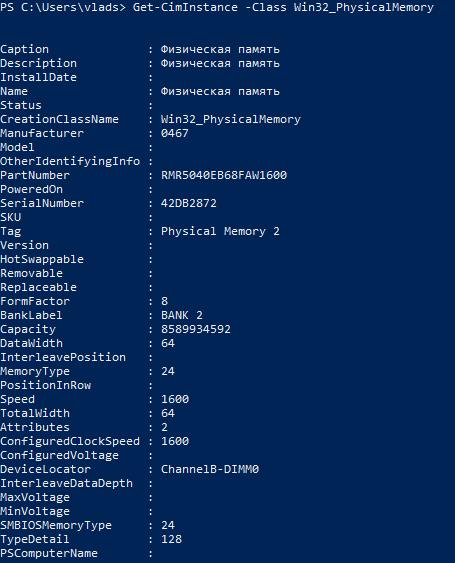


Получение списка установленных программ в отдельный файл programs-list.txt. Для это мы обращаемся к программному обеспечению локального компьютера, к списку программ доступных на удаление (Get-ItemProperty получение свойства указанного элемента):

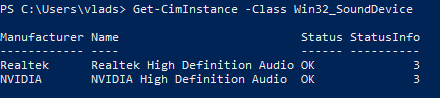
Get-ItemProperty HKLM:\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\\* | Select-Object DisplayName, DisplayVersion, Publisher, InstallDate



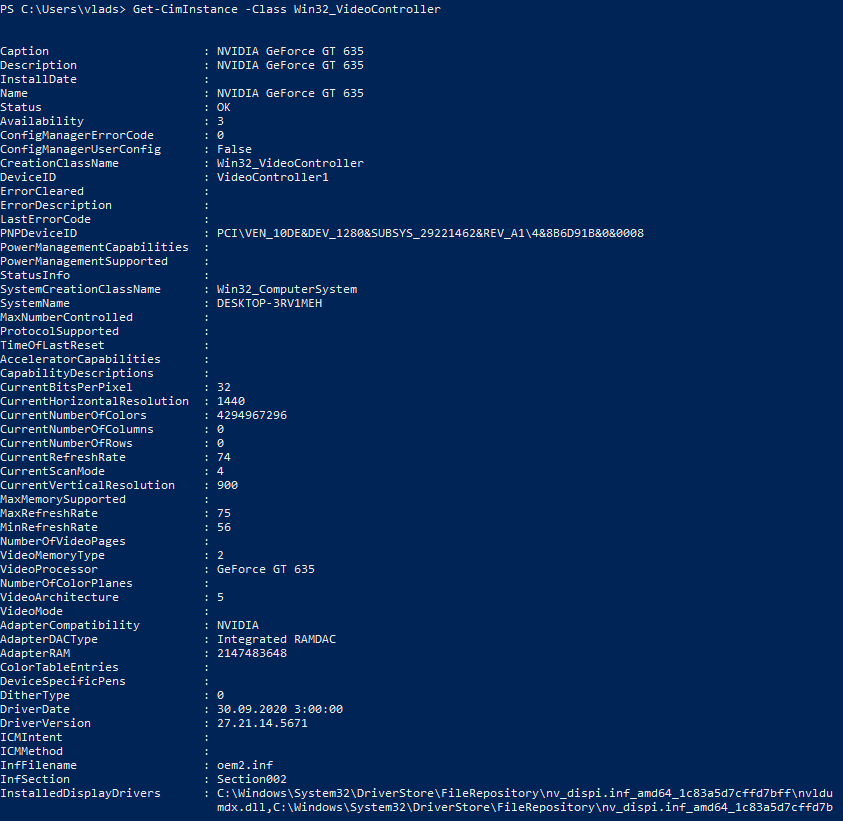
Сведения о физической памяти можно получить, обратившись к классу Win32\_PhysicalMemory



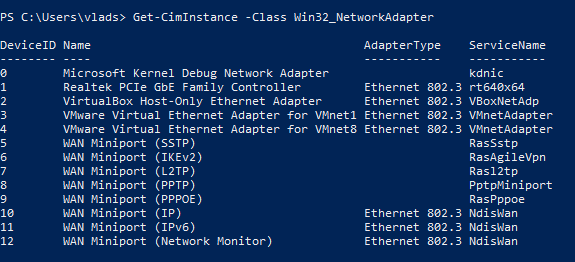
Сведения о звуковой карте можно получить, обратившись к классу Win32\_SoundDevice



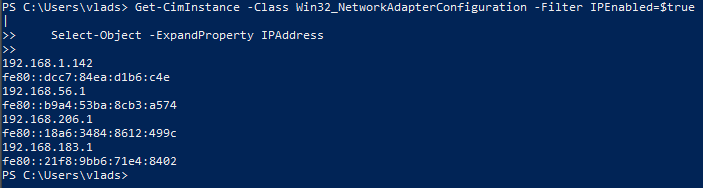
Сведения о видеокарте можно получить, обратившись к классу Win32\_VideoController



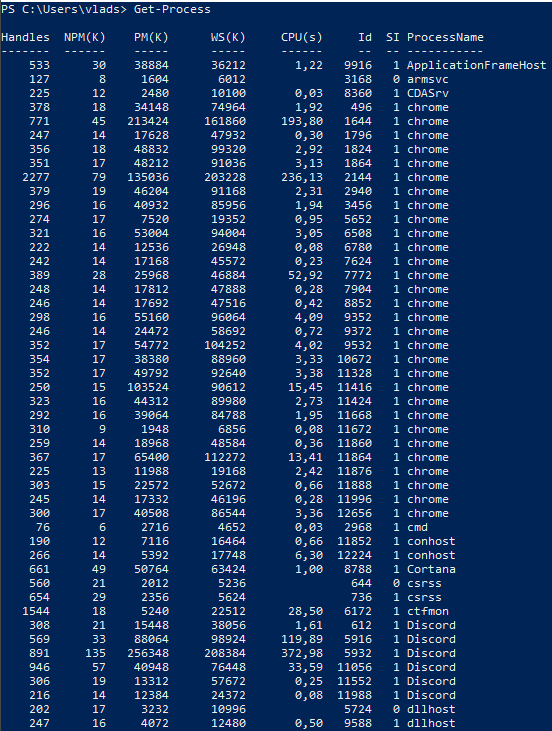
Сведения о сетевых адаптерах можно получить обратившись к классу Win32\_NetworkAdapter



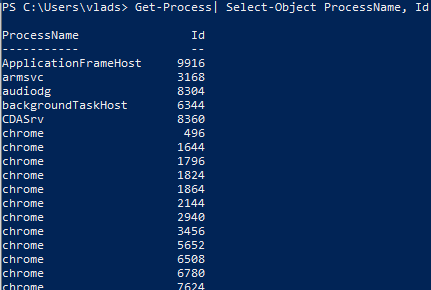
Список IP адресов можно получить, обратившись к классу Win32\_NetworkAdapterConfiguration



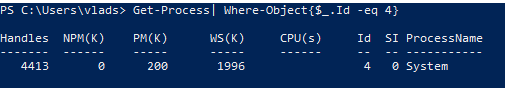
PowerShell так же поддерживает работу с процессами. Получить список всех активных процессов можно при помощи команды Get-Process



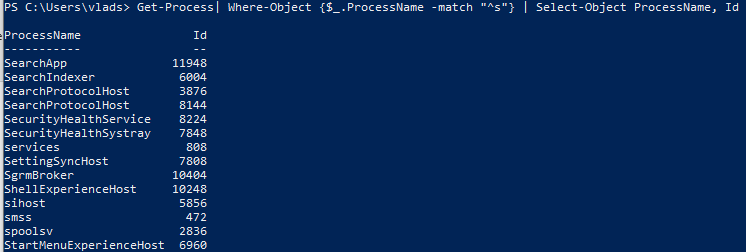
Поскольку данная команда возвращает массив, то мы можем выполнить выборку по отдельным полям каждого объекта массива.



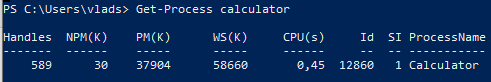
Можно выполнять и условную выборку. Например: выбрать процесс с id = 4



Выборка процессов начинающих на s



Выбрать один конкретный процесс - калькулятор

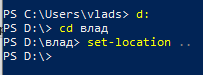


Остановить процесс калькулятор

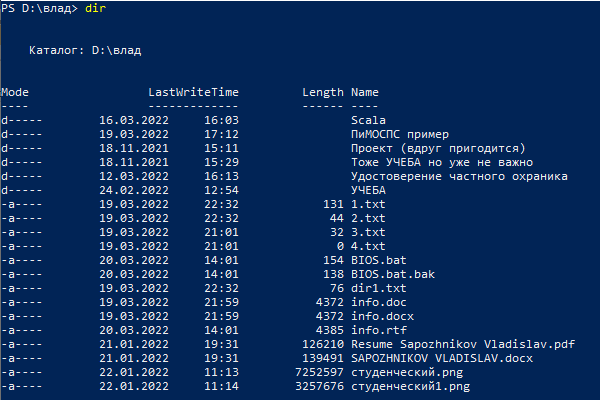


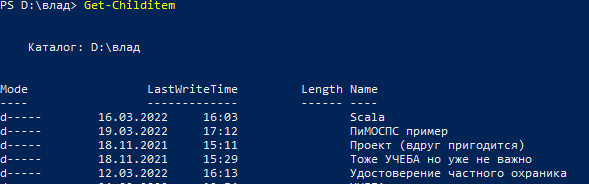
Ну и конечно же *PowerShell* поддерживает работу с файловой системой.

Для перемещения по каталог используется командлет Set-Location или её псевдоним cd

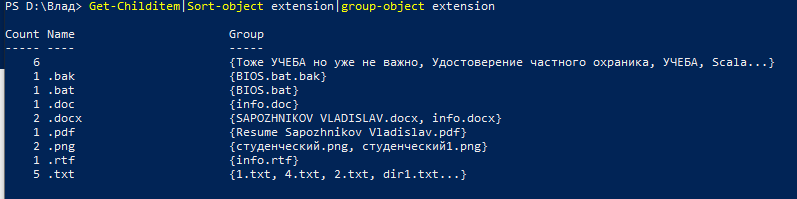


Командлета Get-ChildItem (dir) выводит список файлов и подкаталогов текущего каталога

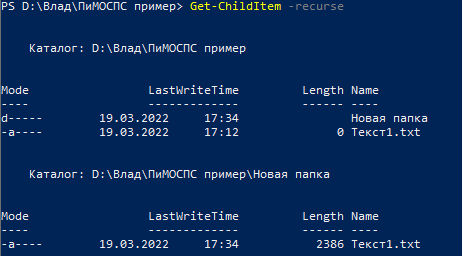




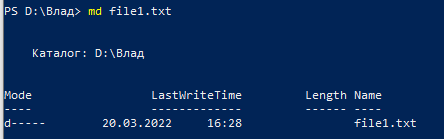
Получение файлов и подкаталогов, сгруппированных по типу разрешения. Для этого результат командлеты Get-ChildItem подается на вход Sort-object, которая сортирует их по расширению, затем производится группировка по расширению.

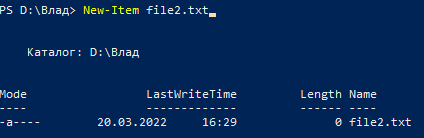


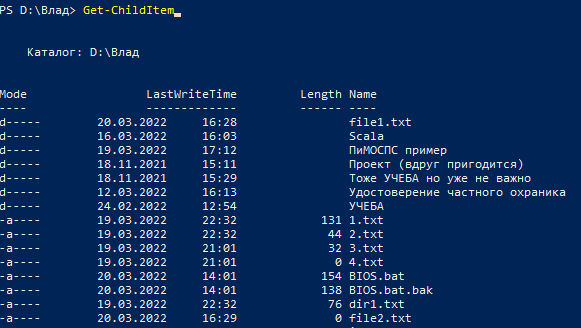
Для рекурсивного обхода всех подкаталогов можно указать флаг -recurce командлеты Get-ChildItem



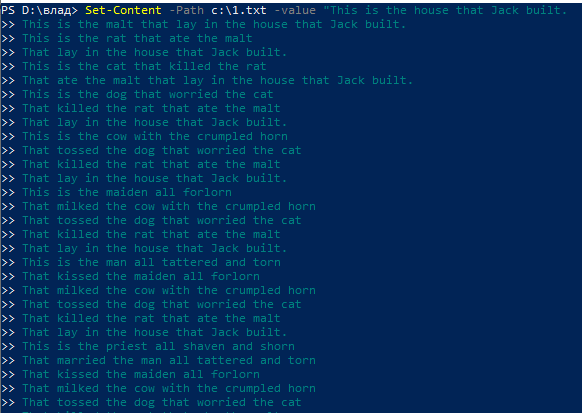
Для создания новых файлов используется командлета New-Item (md)

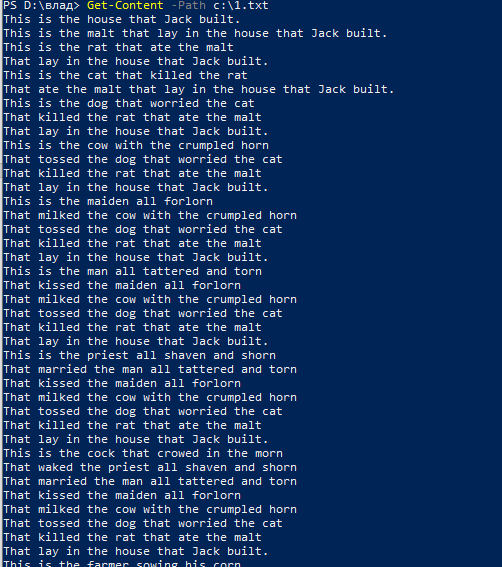




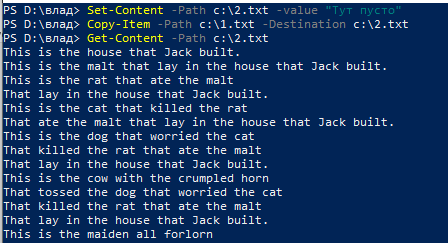


При помощи командлеты Set-Content можно делать запись в файл. Get-Content – получить содержимое файла.

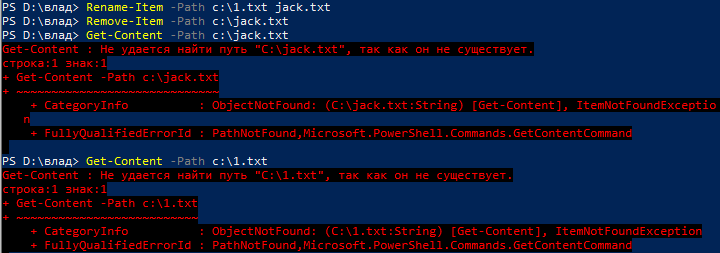




Для копирования файлов используется командлета Copy-Item



Командлеты Rename-Item и Remove-Item используются для переименования и удаления файлов соответственно.



Как можно заметить *PowerShell* ориентирована на объекты, что делает её команды для простыми для понимания и запоминания. Используемая объектная модель .NET используется при разработке ПО под ОС Windows и является самодокументированной. *PowerShell* обеспечивает прямой доступ к COM, WMI и .NET объектам, работает по принципу файловой системы, обеспечивает возможность расширения встроенного набора команд.

*PowerShell* является более совершенным инструментом по сравнению с *cmd.exe* и обязательная к освоению продвинутым пользователям данной ОС.