Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ОТЧЕТ

к лабораторной работе по дисциплине:

|  |
| --- |
| **Администрирование информационных систем** |
| Язык сценариев ОС Microsoft Windows «Power Shell» |

наименование темы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил студент группы: | ИСМб-19-1 |  |  |  | Малиновцев И.А. |
|  |  | подпись |  | Фамилия И.О |
| Проверил: |  |  |  |  | Хритова М.А. |
|  | должность |  | подпись |  | Фамилия И.О |

Иркутск 2022 г.

Содержание

[1 Введение 2](#_Toc119721181)

[2 Краткая информация о языке 4](#_Toc119721182)

[3 Описание всех действий и листинг скриптов 5](#_Toc119721183)

[3.1 Описание используемых команд 5](#_Toc119721184)

[3.1.1 Описание скрипта 1 5](#_Toc119721185)

[3.1.2 Описание скрипта 2 5](#_Toc119721186)

[3.1.3 Описание скрипта 3 5](#_Toc119721187)

[3.2 Листинг кода 6](#_Toc119721188)

[3.2.1 Скрипт 1 6](#_Toc119721189)

[3.2.2 Скрипт 2 8](#_Toc119721190)

[3.2.3 Скрипт 3 9](#_Toc119721191)

[4 Результаты работы скрипта 10](#_Toc119721192)

[4.1 Работа скрипта 1 10](#_Toc119721193)

[4.2 Работа скрипта 2 11](#_Toc119721194)

[4.3 Работа скрипта 3 13](#_Toc119721195)

[5 Заключение 15](#_Toc119721196)

1 Введение

**Цель работы:** Освоить язык сценариев «Power Shell».

**Задание:**

1. Изучить теоретическую информацию о языке сценариев Power Shell.
2. Написать 2-3 простейших сценария, оформив их в виде скрипта. Варианты сценариев студент придумывает сам.

Примеры сценариев:

* Копирование нескольких файлов на сменный носитель в определенное время (учитывать возможность изменения буквы диска).
* Изменения прав доступа по шаблону в папке студента.
* Отобразить список служб на компьютере пользователя.
* Отсортировать все службы на компьютере пользователя на запущенные и остановленные.
* Создать подкаталог текущего каталога и скопировать в него файлы с расширениями PAS, BAS, C и TXT, присвоив им атрибут «только для чтения»;
* В заданном каталоге выдать на экран список имен файлов с расширениями PAS,TXT, BAT.
* Отсортировать указанный файл текущего каталога по убыванию 2-го символа и выдать его на экран.

1. Написать отчет с приложением листинга скрипта.

**Требования к отчету:** Отчетный документ представляется в печатном или электронном виде, в форматах doc(x).

Отчет оформляется согласно действующему нормоконтролю и содержит следующие главы:

1. Введение

2. Краткая информация о языке.

3. Описание всех действий и листинг скрипта с комментариями.

4. Результаты работы скрипта, со скриншотами.

5. Заключение (с анализом возможностей Power Shell).

2 Краткая информация о языке

PowerShell — расширяемое средство автоматизации от Microsoft с открытым исходным кодом, состоящее из оболочки с интерфейсом командной строки и сопутствующего языка сценариев. Впервые публично язык был продемонстрирован на Профессиональной конференции разработчиков в сентябре 2003 года под кодовым названием Monad. Изначально средство выпущено как компонент Windows под названием Windows PowerShell. Открытый исходный код был выпущен с началом разработки PowerShell Core в августе 2016 года.

Команды, исполняемые в Windows PowerShell, могут быть в форме командлетов, которые являются специализированными классами .NET, созданными с целью предоставления функциональности в PowerShell в виде сценариев PowerShell (.PS1) или являются обычными исполняемыми файлами. Если команда является исполняемым файлом, то PowerShell запускает её в отдельном процессе; если это команда, то он исполняется внутри процесса PowerShell. PowerShell предоставляет интерфейс командной строки, в котором можно вводить команды и отображать выводимые ими данные в текстовом виде. Этот пользовательский интерфейс, базирующийся на стандартном механизме консоли Windows, предоставляет настраиваемый механизм автозавершения команд, но не обладает возможностью подсветки синтаксиса, хотя при желании её можно обеспечить. В PowerShell также можно создавать псевдонимы (англ. alias) для командлетов, которые при вызове преобразуются в оригинальные команды. Кроме того, поддерживаются позиционные и именованные параметры для командлетов. При выполнении командлета работа по привязке значений аргументов к параметрам выполняется самим PowerShell, но при вызове внешних исполняемых файлов аргументы передаются им для самостоятельного разбора.

3 Описание всех действий и листинг скриптов

3.1 Описание используемых команд

3.1.1 Описание скрипта 1

beep(440,500) – вызов консольного сигнала. Первое значение – высота звука, второе – продолжительность.

Write-Host – записывает пользовательские выходные данные в узел.

-BackgroundColor – определяет цвет фона. Значение по умолчанию отсутствует.

-ForegroundColor – определяет цвет текста. Значение по умолчанию отсутствует.

Read-Host – считывает строку ввода из консоли.

Switch – передает управление в оператор внутри своего тела.

3.1.2 Описание скрипта 2

Read-Host – считывает строку ввода из консоли.

New-Object – cоздает экземпляр объекта Microsoft .NET Framework или COM.

-ComObject – задает программный идентификатор (ProgID) COM-объекта.

Word.Application – представляет приложение Microsoft Word.

Get-ChildItem – получает элементы и дочерние элементы в одном или нескольких указанных расположениях.

-Path – указывает путь к одному или нескольким расположениям.

-Filter – указывает фильтр для уточнения параметра Path .

ForEach-Object – выполняет операцию над каждым элементом в коллекции входных объектов.

Documents.Open – открывает указанный документ и добавляет его в коллекцию документов.

FullName – возвращает полное имя типа, включая пространство имен, но не сборку.

DirectoryName – получает строку, представляющую полный путь к каталогу.

SaveAs – сохраняет указанный документ с новым именем или форматом.

Quit() – заканчивает работу с объектом MS Word.

Close() – заканчивает работу с документом.

3.1.3 Описание скрипта 3

Get-NetIPConfiguration - получает конфигурацию IP-сети.

Select – выбирает объекты или свойства объектов.

-ExpandProperty – указывает свойство для выбора и указывает, что следует попытаться расширить это свойство.

Out-File – отправляет вывод в файл.

3.2 Листинг кода

3.2.1 Скрипт 1

#Проигрывание мелодий с помощью beep

function playStarWars{ #Метод проигрывания музыкальной темы из Star Wars с помощью beep (консольный сигнал)

[console]::beep(440,500) #beep: первое значение - высота звука, второе - продолжительность

[console]::beep(440,500)

[console]::beep(440,500)

[console]::beep(349,350)

[console]::beep(523,150)

[console]::beep(440,500)

[console]::beep(349,350)

[console]::beep(523,150)

[console]::beep(440,1000)

[console]::beep(659,500)

[console]::beep(659,500)

[console]::beep(659,500)

[console]::beep(698,350)

[console]::beep(523,150)

[console]::beep(415,500)

[console]::beep(349,350)

[console]::beep(523,150)

[console]::beep(440,1000)

}

function playKind { #Метод проигрывания музыкальной темы из фильмов о НЛО

[console]::beep(900,400)

[console]::beep(1000,400)

[console]::beep(800,400)

[console]::beep(400,400)

[console]::beep(600,1600)

}

function playMissionImpossible { #Метод проигрывания музыкальной темы из фильма Миссия невыполнима

[console]::beep(784,150)

Start-Sleep -m 300

[console]::beep(784,150)

Start-Sleep -m 300

[console]::beep(932,150)

Start-Sleep -m 150

[console]::beep(1047,150)

Start-Sleep -m 150

[console]::beep(784,150)

Start-Sleep -m 300

[console]::beep(784,150)

Start-Sleep -m 300

[console]::beep(699,150)

Start-Sleep -m 150

[console]::beep(740,150)

Start-Sleep -m 150

[console]::beep(784,150)

Start-Sleep -m 300

[console]::beep(784,150)

Start-Sleep -m 300

[console]::beep(932,150)

Start-Sleep -m 150

[console]::beep(1047,150)

Start-Sleep -m 150

[console]::beep(784,150)

Start-Sleep -m 300

[console]::beep(784,150)

Start-Sleep -m 300

[console]::beep(699,150)

Start-Sleep -m 150

[console]::beep(740,150)

Start-Sleep -m 150

[console]::beep(932,150)

[console]::beep(784,150)

[console]::beep(587,1200)

Start-Sleep -m 75

[console]::beep(932,150)

[console]::beep(784,150)

[console]::beep(554,1200)

Start-Sleep -m 75

[console]::beep(932,150)

[console]::beep(784,150)

[console]::beep(523,1200)

Start-Sleep -m 150

[console]::beep(466,150)

[console]::beep(523,150)

}

#Отрисовываем консольное меню

Write-Host #отступ строки (пустой)

Write-Host "Меню" -BackgroundColor Black -ForegroundColor Red #Надпись Меню в консоли с покраской заднего фона в черный и текст в красный

Write-Host #отступ строки (пустой)

#Вторая часть меню с пунктами выбора (варианты вводимых значений с зеленым цветом текста)

Write-Host "1. Имперский марш (Star Wars)" -ForegroundColor Green

Write-Host "2. Контакт третьего рода" -ForegroundColor Green

Write-Host "3. Миссия невыполнима" -ForegroundColor Green

Write-Host "0. Выход" -ForegroundColor Green

Write-Host

function readCommand {

$choice = Read-Host "Введите числовое значение для работы" #Выводим текст о с просьбой ввода и считываем вводимое значение

Switch($choice){ #Провиряем с помощью Switch введенный результат (если введено числовое значение из пункта меню - вызываем соответствующий метод.

#Иначе - просим ввести корректное значение)

1{playStarWars}

2{playKind}

3{playMissionImpossible}

0{Write-Host "До скорой встречи!"; exit}

default {

Write-Host "Ошибка выбора! Попробуйте еще раз!" -ForegroundColor Red

readCommand

}

}

}

readCommand

3.2.2 Скрипт 2

#Конвертер Word to Pdf

function docToPdf{ #Метод конвертации файла doc и docx в pdf

$documents\_path = Read-Host "Путь к файлу Word" #Считываем путь к файлу

$word\_app = New-Object -ComObject Word.Application #Инициализируем объект приложения MS Word

Get-ChildItem -Path $documents\_path -Filter \*.doc? | ForEach-Object { #Проверяем файл из пути на совпадение формата

#(.doc? обозначает, что ищем объект .doc с опциональным символом после [.docx])

$document = $word\_app.Documents.Open($\_.FullName) #С помощью иниализируемого объекта MS Word открываем файл по введенному пути

$pdf\_filename = "$($\_.DirectoryName)\$($\_.BaseName).pdf" #Подготавливаем конструкцию для будущего файла pdf

#(используем директорию оригинального файла и его имя, но уже с расширением .pdf)

$document.SaveAs([ref] $pdf\_filename, [ref] 17) #Сохраняем файл (первый параметр - путь для сохраняемого файла, второй - кодировка)

$document.Close() #Заканчиваем работу с документов

}

Write-Host "Работа закончена." -BackgroundColor Black -ForegroundColor Green #Вывод сообщения о конце работы

$word\_app.Quit() #Заканчиваем работу с объектом

}

docToPdf

3.2.3 Скрипт 3

#Работаем с сетью и сохраняем в txt

#Get-NetIPConfiguration - Получаем информацию о конфигурации сети и передаем ее.

#Из полученной информации получает конфигурации IP-адреса забираем информацию о конфигурации адреса IPv4 и передаем ее.

#Полученную информацию сохраняем с помощью Out-File в корневую директорию со скриптом в файл config.txt

Get-NetIPConfiguration | Select -ExpandProperty IPV4Address | Out-File -FilePath config.txt

4 Результаты работы скрипта

4.1 Работа скрипта 1

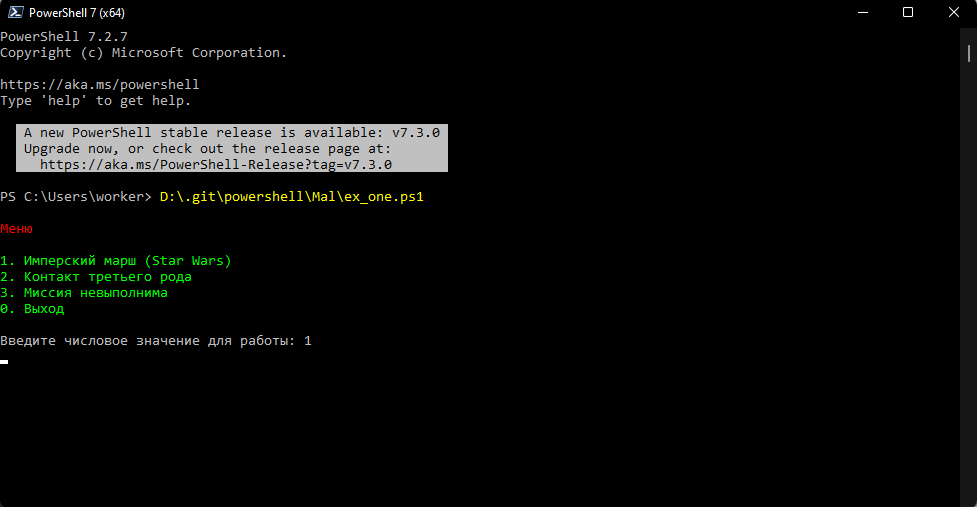


Рисунок 1 – Проигрывание мелодии

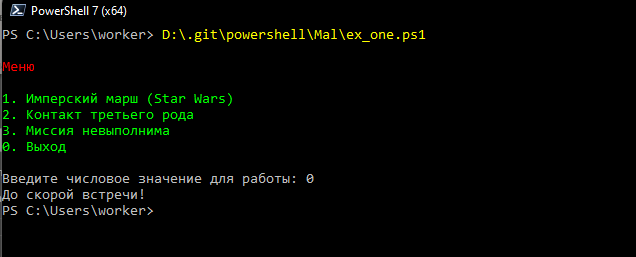


Рисунок 2 – Выход из скрипта

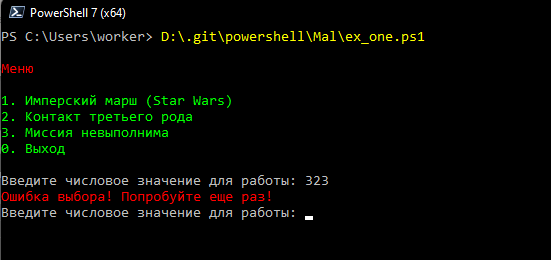


Рисунок 3 – Ввод некорректных данных

4.2 Работа скрипта 2

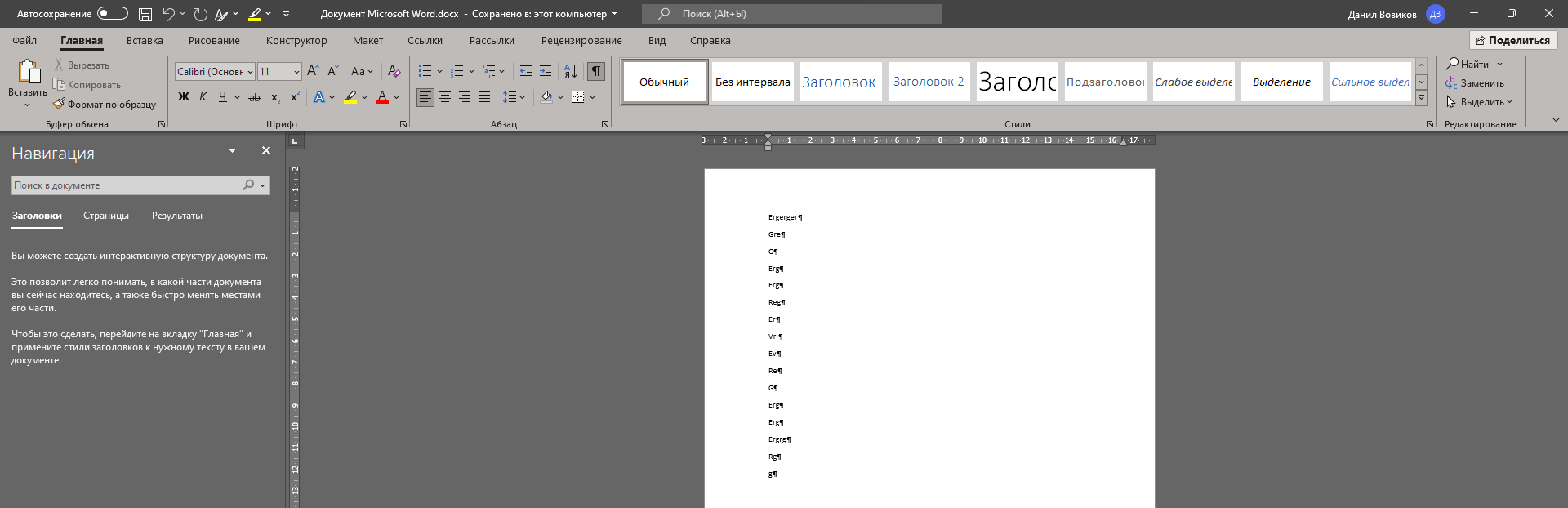


Рисунок 4 – Файл MS Word

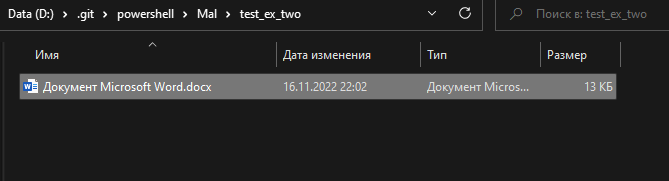


Рисунок 5 – Расположение файла MS Word

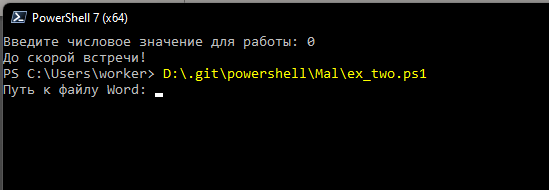


Рисунок 6 – Вызов скрипта

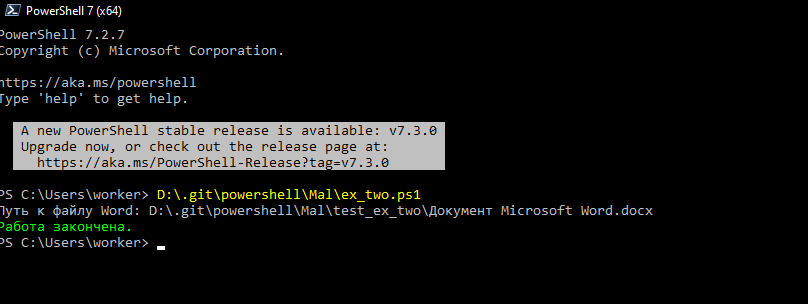


Рисунок 7 – Завершение работы скрипта

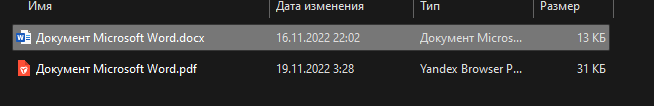


Рисунок 8 – Результат работы скрипта

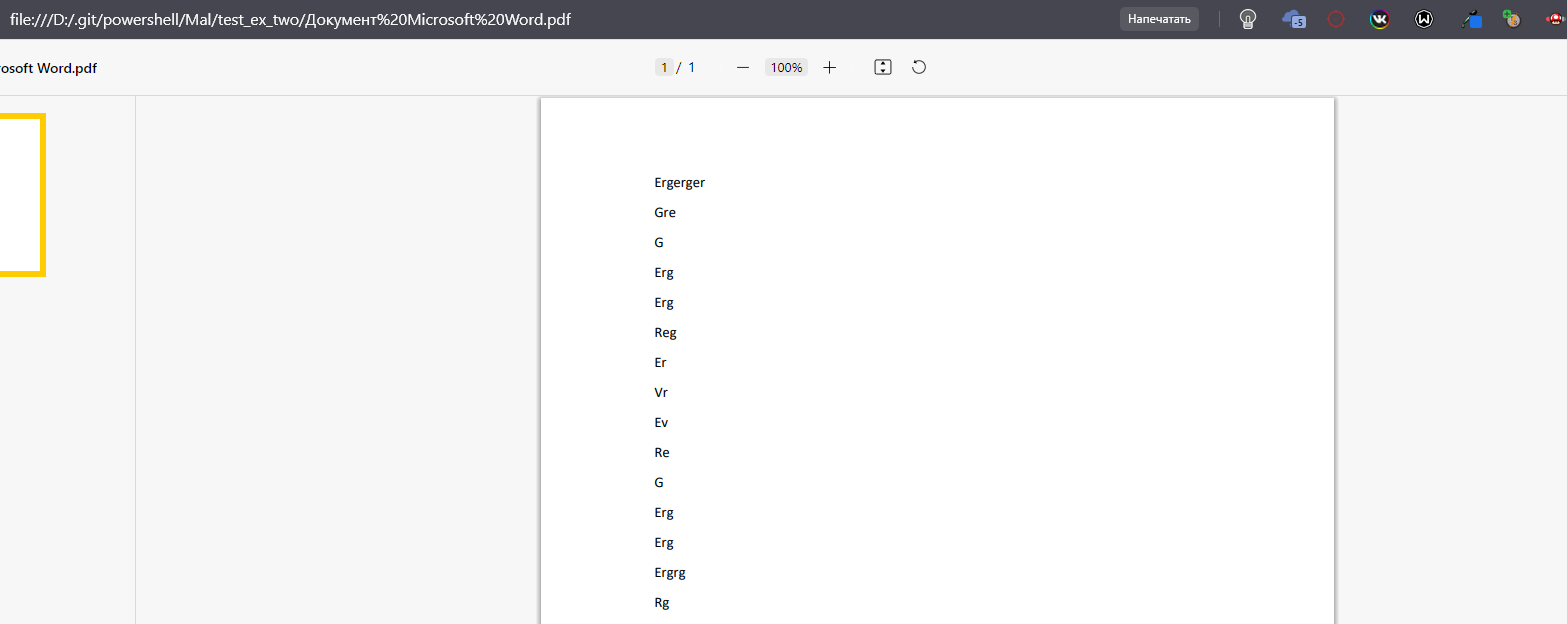


Рисунок 9 – Результат работы скрипта

4.3 Работа скрипта 3

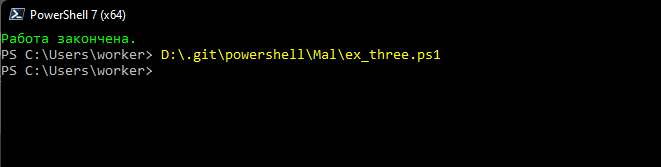


Рисунок 10 – Вызов скрипта

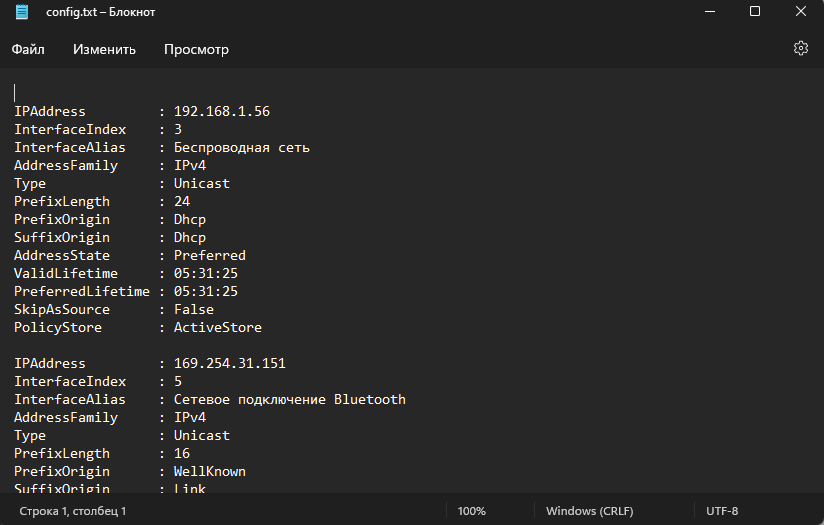


Рисунок 11 – Результат работы скрипта

5 Заключение

Несмотря на возможность выполнения большинства функций командной строки в Windows PowerShell, функционал приложений не совпадает.

В то время как Cmd использует команды DOS, PowerShell использует команды, известные как командлеты, которые представляют собой независимые программные объекты, позволяющие получить доступ к основным параметрам администрирования в Windows. Эти параметры недоступны из Cmd, что означает, что пользователи должны перемещаться по графическому интерфейсу, чтобы получить к ним доступ.

Cmd – это просто система оболочки, которая ограничивает пользователей выполнять только простые и базовые сценарии для выполнения командного файла. Однако PowerShell – это и оболочка, и среда сценариев, которая поддерживает создание больших и сложных файлов для управления системой Windows.

Командная строка не предназначена для системного администрирования, и пользователям требуются дополнительные утилиты для расширения ее функциональности. Напротив, в Windows PowerShell есть множество командлетов, которые упрощают жизнь системным администраторам. Он позволяет выполнять широкий спектр задач и автоматизацию, включая управление пользователями и разрешениями, конфигурации безопасности, извлечение данных и администрирование Active Directory.

Главной особенностью среды PowerShell, отличающей ее от всех других оболочек командной строки, является то, что единицей обработки и передачи информации здесь является объект, а не строка текста.