Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития

Кафедра инфокоммуникаций

**«ОБОЛОЧКА КОМАНДНОЙ СТРОКИ WINDOWS POWERSHELL 2.0»**

**ОТЧЕТ**

**по лабораторной работе №6**

**дисциплины**

**«Операционные системы»**

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Выполнил:  Мизин Глеб Егорович  2 курс, группа ПИЖ-б-о-21-1,  09.03.04 «Программная инженерия», направленность (профиль) «Разработка и сопровождение программного обеспечения», очная форма обучения  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | | Проверил:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | |  | |

Отчет защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_ Дата защиты\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ставрополь, 2022 г.

Задание№1: увеличить ширину окна оболочки до максимальной, увеличить высоту окна и задать цвет фона и цвет шрифта (рекомендуется синий фон и белый шрифт).

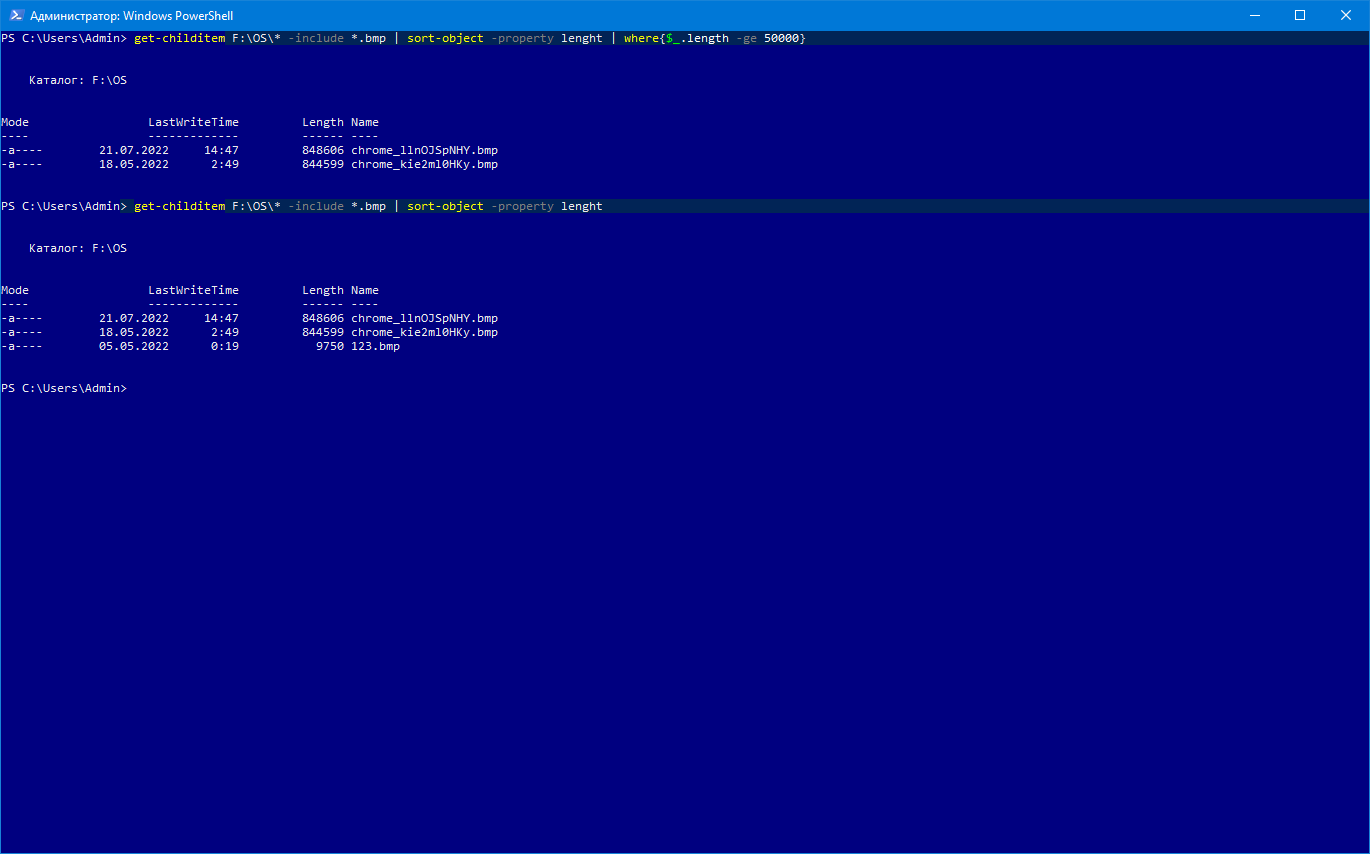


Рисунок 1 – Изменение размера и внешнего вида окна

Задание №2: вывести содержимое каталога Windows (для бригад 5 и 10 – и подкаталогов) по указанному в табл. 5 формату на экран и в текстовый файл.



get-childitem F:\OS\\* -include \*.bmp | sort-object -property lenght | where{$\_.length -ge 50000}

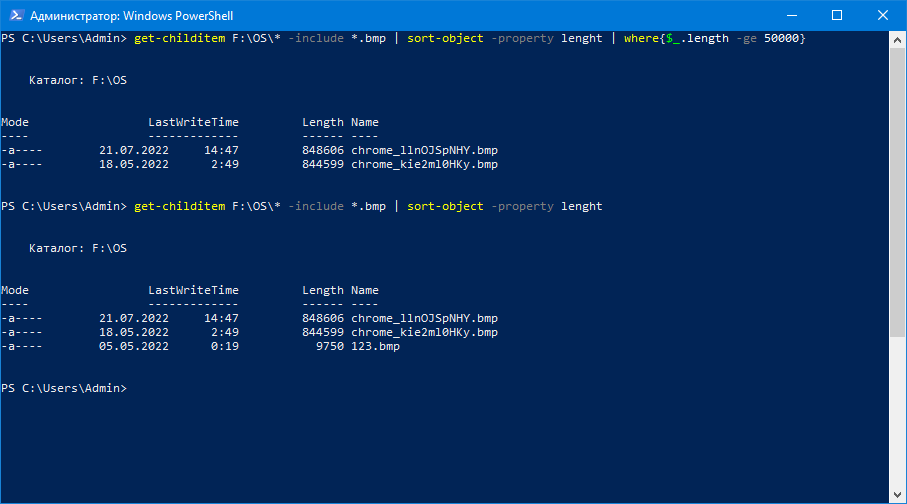


Рисунок 2 – Результат работы команды

Задание№3: вывести в текстовый файл список свойств процесса, возвращаемый командлетом Get-process и на экран – их общее количество.

get-process | get-member -membertype property | out-file -filepath F:\OS\6\_lab\_Test.txt

$a = get-process

$a.length

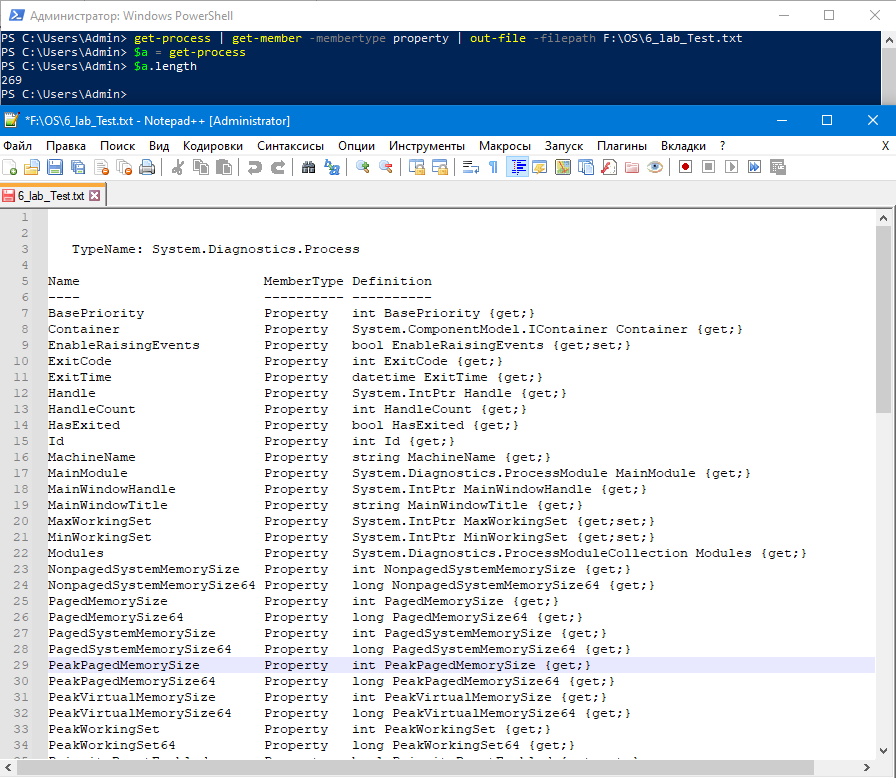
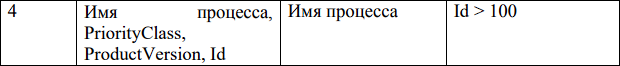


Рисунок 3 – Вывод свойств процесса и их количества

Задание №4: создать текстовый файл, содержащий список выполняемых процессов, упорядоченный по возрастанию указанного в табл.6 параметра. Имена параметров процессов указаны в (табл. 6).



get-process|where{$\_.Id -gt 100}|sort -property ProcessName|select Name, PriorityClass, ProductVersion, Id|ft -auto|out-File -filepath F:\OS\6LabTest2.txt

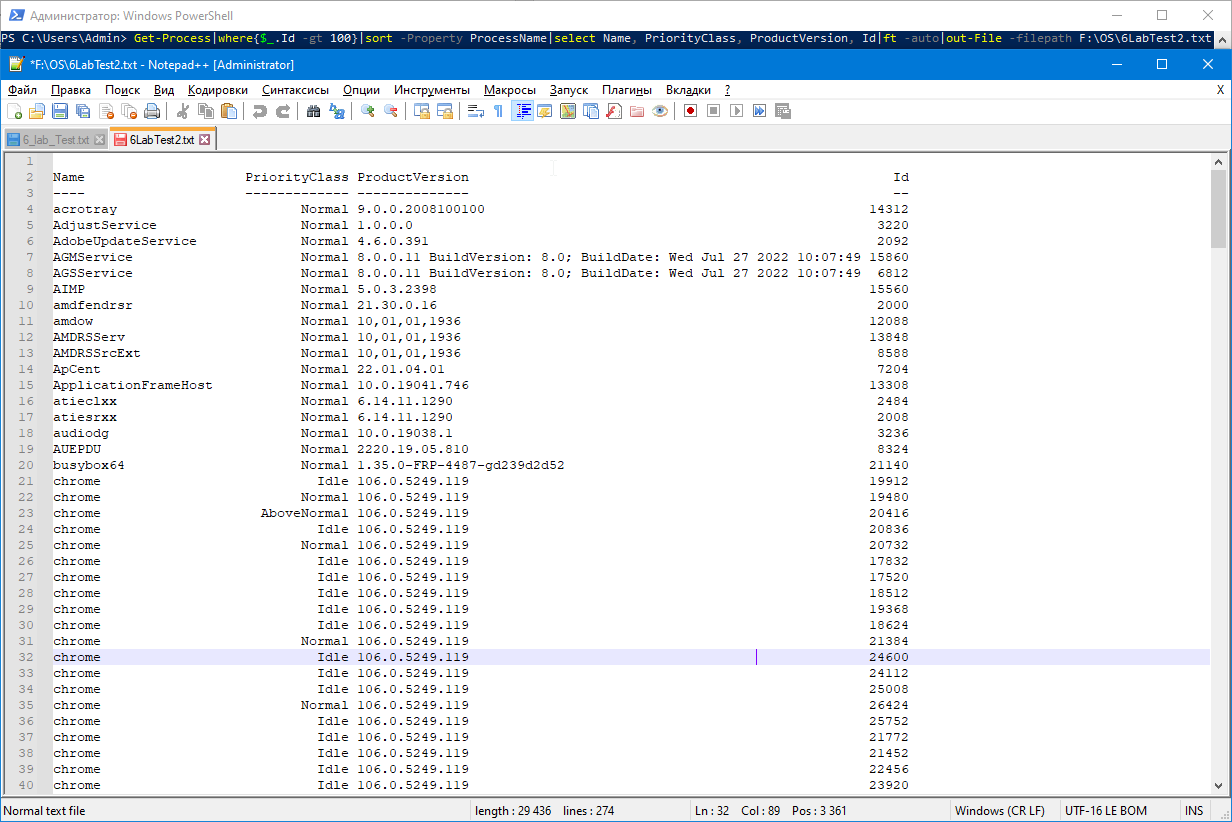


Рисунок 4 – Результат работы команды

Задание №5: создать HTML-файл, содержащий список выполняемых процессов, упорядоченный по возрастанию указанного в табл.5 параметра. Имена параметров процессов указаны в табл. 5.

get-process |sort -property ProcessName| Convertto-html -Property Name, PriorityClass, ProductVersion, Id > testLab6.htm

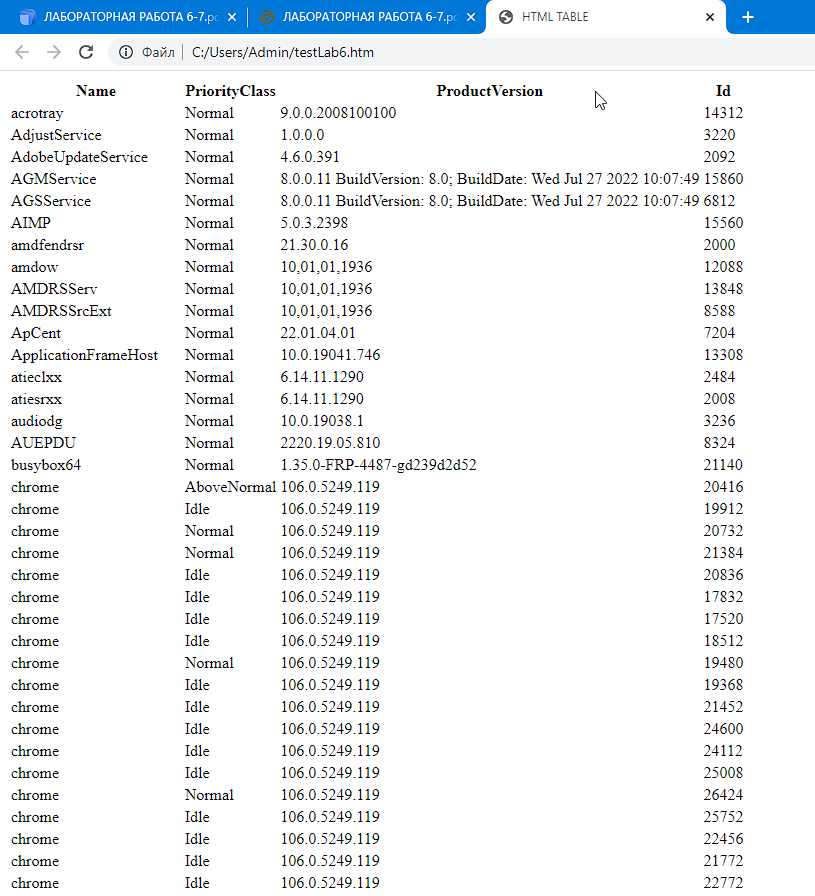


Рисунок 5 – Результат выполненной команды

Задание №6: найти суммарный объем всех графических файлов (bmp, jpg), находящихся в каталоге Windows и всех его подкаталогах.

$summa=0; ForEach ($f in dir C:\Windows\\* -include \*.bmp,\*.jpg,\*.gif -recurse) {$summa +=$f.length}; $summa

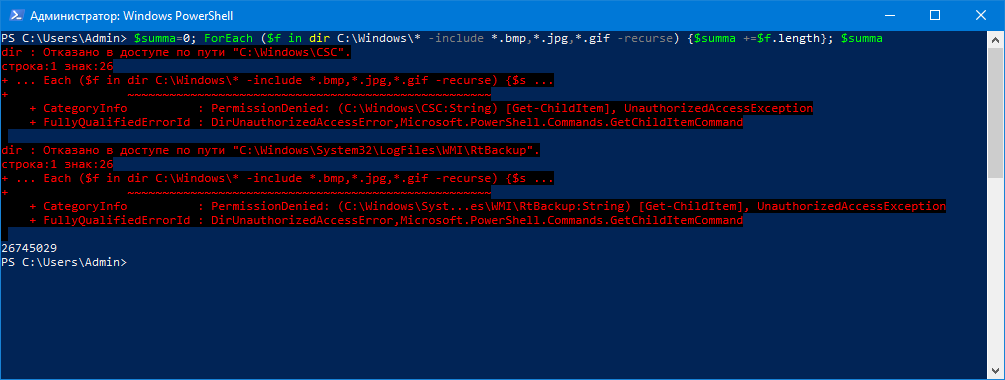


Рисунок 6 – Результат выполнения команды

Задание №7: вывести на экран сведения о ЦП компьютера.

get-wmiobject -Class Win32\_Processor

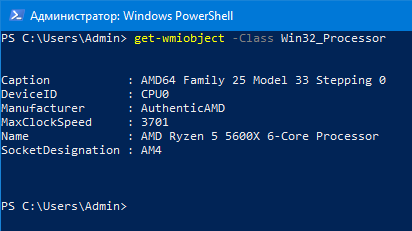


Рисунок 7 – результат работы команды

Задание №8: найти максимальное, минимальное и среднее значение времени выполнение командлетов dir и ps

$n = 10

1..$n | foreach {Measure-Command {dir}} | Measure-Object TotalMilliseconds -Max -Min -Average

1..$n | foreach {Measure-Command {ps}} | Measure-Object TotalMilliseconds -Max -Min -Average

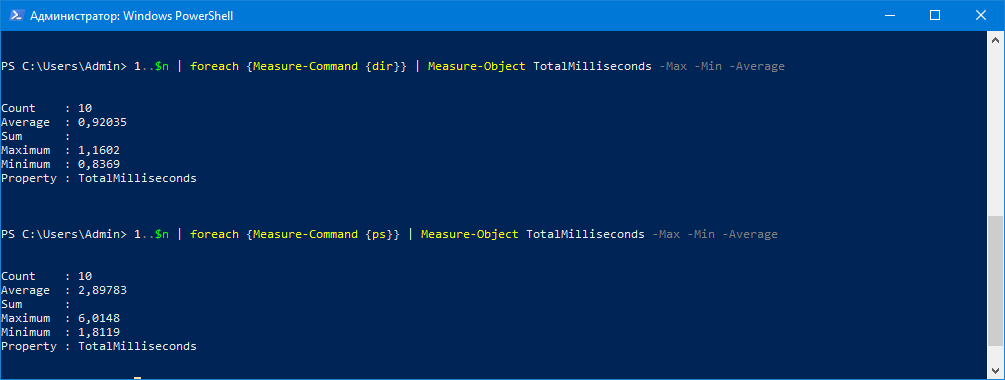


Рисунок 8 – Результаты выполнения команд

Задание №9: выполнить индивидуальные задания для студентов согласно (табл. 7).



Get-ChildItem F:\OS -File | sort Length |select -Last 1

Get-ChildItem F:\OS -File | sort Length -Descending |select -Last 3

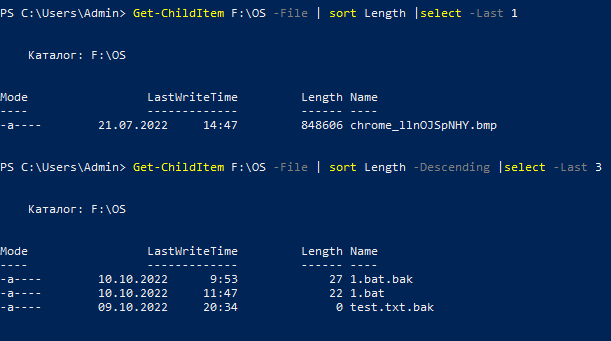


Рисунок 9 – Результаты работы команд

**Контрольные вопросы**

1. Типы команд PowerShell (PS).

Первый тип – так называемые командлеты (cmdlet). Командлет – аналог внутренней команды интерпретатора командной строки - представляет собой класс .NET, порожденный от базового класса Cmdlet

Следующий тип команд – функции. Функция – это блок кода на языке PowerShell, имеющий название и находящийся в памяти до завершения текущего сеанса командной оболочки. Функции, как и командлеты, поддерживают именованные параметры.

Сценарий – это блок кода на языке PowerShell, хранящийся во внешнем файле с расширением ps1. Анализ синтаксиса сценария производится при каждом его запуске.

Последний тип команд – внешние исполняемые файлы, которые выполняются обычным образом операционной системой.

2. Имена и структура командлетов.

Синтаксис командлетов имеет следующую структуру: имя\_командлета –параметр1 -параметр2 аргумент1 аргумент2

3. Псевдонимы команд.

Псевдонимы в PowerShell делятся на два типа. Первый тип предназначен для совместимости имен с разными интерфейсами. Псевдонимы этого типа позволяют пользователям, имеющим опыт работы с другими оболочками (Cmd.exe или Unix-оболочки), использовать знакомые им имена команд для выполнения аналогичных операций в PowerShell, что упрощает освоение новой оболочки, позволяя не тратить усилий на запоминание новых команд PowerShell.

Псевдонимы второго типа (стандартные псевдонимы) в PowerShell предназначены для быстрого ввода команд. Такие псевдонимы образуются из имен командлетов, которым они соответствуют.

4. Просмотр структуры объектов.

Для анализа структуры объекта, возвращаемого определенной командой, проще всего направить этот объект по конвейеру на командлет Get-Member (псевдоним gm)

5. Фильтрация объектов в конвейере. Блок сценария.

В PowerShell поддерживается возможность фильтрации объектов в конвейере, т.е. удаление из конвейера объектов, не удовлетворяющих определенному условию. Данную функциональность обеспечивает командлет Where-Object, позволяющий проверить каждый объект, находящийся в конвейере, и передать его дальше по конвейеру, только если объект удовлетворяет условиям проверки.

Условие проверки в Where-Object задается в виде блока сценария – одной или нескольких команд PowerShell, заключенных в фигурные скобки {}. Результатом выполнения данного блока сценария должно быть значение логического типа: True (истина) или False (ложь).

6. Какую информацию выводит команда Get-Help \* ?

Get-Help \* перечисляет все команды Windows PowerShell

7. Командлеты для форматирования выводимой информации.

Format-Table: форматирует вывод команды в виде таблицы, столбцы которой содержат свойства объекта (также могут быть добавлены вычисляемые столбцы). Поддерживается возможность группировки выводимых данных

Format-List: вывод форматируется как список свойств, в котором каждое свойство отображается на новой строке. Поддерживается возможность группировки выводимых данных

Format-Custom: для форматирования вывода используется пользовательское представление (view)

Format-Wide: форматирует объекты в виде широкой таблицы, в которой отображается только одно свойство каждого объекта

8. Перенаправление выводимой информации.

В оболочке PowerShell имеются несколько командлетов, с помощью которых можно управлять выводом данных. Эти командлеты начинаются со слова Out, их список можно получить с помощью команлета: PS C:\> Get-Command out-\* | Format-Table Name

9. Управляющие инструкции PS.

If …ElseIf … Else

Циклы While и Do … While

Цикл For и ForEach

10.Назначение регулярных выражений.

Регулярные выражения — это специальный мини-язык, служащий для разбора (parsing) текстовых данных. С его помощью можно разделять строки на компоненты, выбирать нужные части строк для дальнейшей обработки, производить замены и т. Д

11.Сохранение данных в текстовом файле и html-файле.

Командлет Out-File позволяет направить выводимые данные вместо окна консоли в текстовый файл. Аналогичную задачу решает оператор перенаправления (>), однако командлет Out-File имеет несколько дополнительных параметров, с помощью которых можно более гибко управлять выводом: задавать тип кодировки файла (параметр Encoding), задавать длину выводимых строк в знаках (параметр Width), выбирать режим перезаписи файла (параметр Append).

Для преобразования данных в формат html служит командлет Convertto-html.

12.Получение справочной информации в PS.

Краткую справку по одному командлету можно получить с помощью параметра «?» (вопросительный знак), указанного после имени этого командлета.

В Windows PowerShell можно также использовать команду Get-Help. Get-Help выводит на экран справку об использовании справки

Get-Help \* перечисляет все команды Windows PowerShell

Get-Help команда выводит справку по соответствующей команде

Get-Help команда -Detailed выводит подробную справку с примерами команды

13.Как создать массив в PS?

$a=1,5,7

$b=10..15

$f = Get-Content c:\data\numb.txt –TotalCount 25

14.Как объединить два массива?

$a = 1, 2

$b = 3, 4

$c = $a+$b

15.Как увеличить размер созданного в PS массива?

$a = 1, 2

$a += 5,6

16.Как ввести данные в массив?

$a=1,5,7

$b=10..15

$f = Get-Content c:\data\numb.txt –TotalCount 25

17.Использование командлета Out-Null.

Командлет Out-Null служит для поглощения любых своих входных данных. Это может пригодиться для подавления вывода на экран ненужных сведений, полученных в качестве побочного эффекта выполнения какой-либо команды.

18.Оператор PowerShell –match.

Оператор PowerShell –match позволяет сравнивать текст слева от него, с регулярным выражением справа. В случае если текст подпадает под регулярное выражение, оператор выдаёт True, иначе – False.

19.Использование символа ^ в командлетах.

^ (крышка) является якорем начала строки вне группы символов

^ как символ отрицания используется лишь в начале группы символов.

20.Использование символа $ в командлетах.

Имена переменных в PowerShell всегда должны начинаться с символа «$»

$ (знак доллара) - обозначает конец строки

21.Количественные модификаторы (квантификаторы).

Количественные модификаторы (квантификаторы) – это модификаторы, которые приписываются к любой группе справа, и определяют количество вхождений этой группы. Например, количественный модификатор {4} означает 4 вхождения

22.Использование групп захвата.

Как следует из названия, группы можно использовать для группировки. К группам захвата, как и к символам и символьным группам, можно применять количественные модификаторы

23.Командлеты для измерения свойств объектов.

Для измерения времени выполнения командлетов PS служит командлет Measure-Command