

Практика 1.

Взаимодействия с ОС

через командный

интерфейс

Шарапов Ю.А.

Команды CMD

Команда	Описание
<code>dir</code>	Вывести содержимое папки
<code>mkdir</code>	Создать папку
<code>cd</code>	Перейти в папку
<code>NULL > file.txt</code>	Создать файл
<code>echo Hello, world! >> file.txt</code>	Записать текст в файл
<code>rename</code>	Переименовать файл
<code>move</code>	Переместить файл
<code>del</code>	Удалить файл
<code>rmdir</code>	Удалить папку
<code>cls</code>	Очистить экран консоли

Команды PowerShell

Команда	Описание
<code>Get-ChildItem</code>	Вывести содержимое папки
<code>New-Item dir1 -ItemType Directory</code>	Создать папку
<code>Cd</code>	Перейти в папку
<code>New-Item file.txt -ItemType File</code>	Создать файл
<code>Set-Content file.txt 'Hello, world!' -Force</code>	Записать текст в файл
<code>Move-Item file.txt file1.txt</code>	Переименовать файл
<code>Move-Item file.txt dir1</code>	Переместить файл
<code>Remove-Item file.txt</code>	Удалить файл
<code>Remove-Item -Path C:\temp\DeleteMe -Recurse</code>	Удалить папку
<code>clear</code>	Очистить экран консоли

PowerShell

1. Командная оболочка с языком сценариев, изначально созданная на основе платформы .NET Framework, а позднее — на .NET Core
2. Работает с классами .NET, у которых есть свойства и методы
3. Дает доступ к объектам COM, WMI и ADSI

Командлеты (cmdlets)

1. Это специализированные классы .NET, в которые заложена разнообразная функциональность
2. Именуются они по принципу «Действие-Объект» (например, `Get-Help` – «Показать справку»)

Полный список командлетов в системе можно получить, выполнив команду в оболочке PowerShell

```
Get-Command -CommandType cmdlet
```

Алиасы

1. Для часто используемых командлетов и внешних утилит в Windows PowerShell есть короткие синонимы – алиасы
2. Полный список синонимов можно посмотреть с помощью командлета **Get-Alias**

Например, **dir** – алиас для командлета **Get-ChildItem**

Найти командлет, соответствующий алиасу **ls**

Get-Alias

ls

Найти алиас, соответствующий командлету **Get-ChildItem**

Get-Alias-Definition

Get-Childitem

Программные инструменты Windows PowerShell

1. **Windows Powershell** – эмулятор консоли с командной оболочкой
2. **Windows PowerShell ISE** – полноценная среда разработки с редактором кода, который поддерживает вкладки, подсветку синтаксиса; конструктором команд; встроенным отладчиком

Windows PowerShell ISE

The screenshot displays the Windows PowerShell ISE interface. The main editor window contains the following PowerShell code:

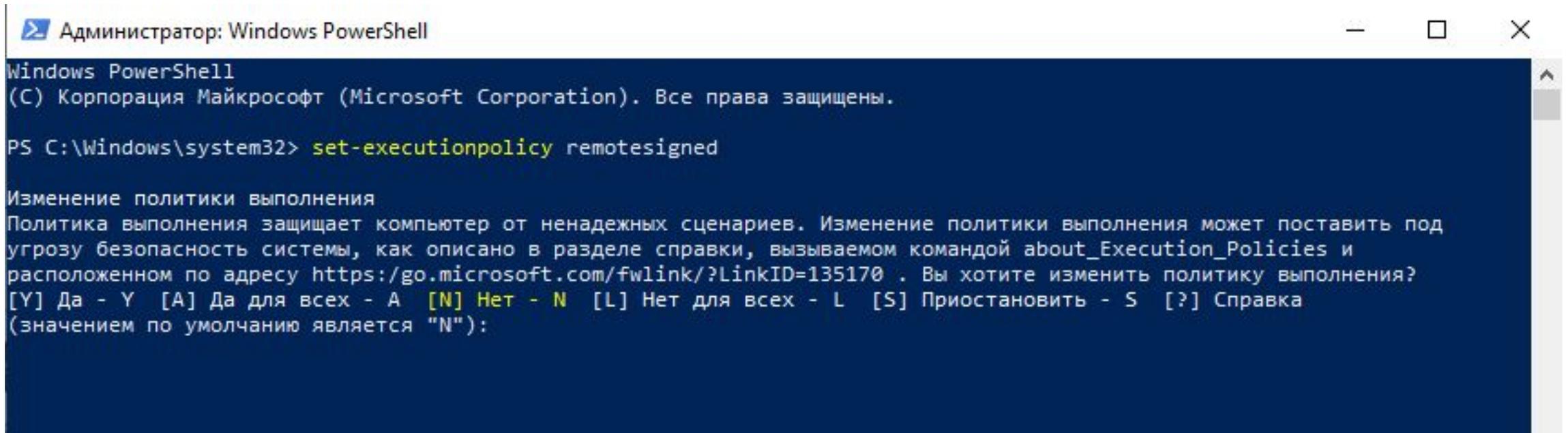
```
1 # Обработка ошибки подход "До"
2 function mydel1([int]$a, [int]$b){
3     if ($b -ne 0){
4         $c = $a / $b
5         Write-Output $c
6     }
7     else{
8         Write-Output "Деление на ноль!"
9     }
10 }
11
12 # Обработка ошибки подход "После"
13 function mydel2([int]$a, [int]$b){
14     try{
15         $c = $a / $b
16         Write-Output $c
17     }
18     catch{
19         Write-Output "Деление на ноль!"
20     }
21 }
```

The console window shows the execution results:

```
PS C:\Users\uriy.sharapov\desktop\test> mydel1 5 4
1,25
PS C:\Users\uriy.sharapov\desktop\test> mydel1 5 0
Деление на ноль!
PS C:\Users\uriy.sharapov\desktop\test> mydel2 5 4
1,25
PS C:\Users\uriy.sharapov\desktop\test> mydel2 5 0
Деление на ноль!
PS C:\Users\uriy.sharapov\desktop\test>
```

The Command pane on the right shows the 'Get-Alias' command. The 'Definition' tab is active, displaying the definition: 'remove-item'. The 'Scope' is set to 'Default'. The 'Общие параметры' (General parameters) section is expanded, showing options for Debug, ErrorAction, ErrorVariable, InformationAction, InformationVariable, and OutBuffer.

Включить выполнение скриптов в PowerShell (из под администратора)



```
Администратор: Windows PowerShell
Windows PowerShell
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

PS C:\Windows\system32> set-executionpolicy remotesigned

Изменение политики выполнения
Политика выполнения защищает компьютер от ненадежных сценариев. Изменение политики выполнения может поставить под угрозу безопасность системы, как описано в разделе справки, вызываемом командой about_Execution_Policies и расположенном по адресу https://go.microsoft.com/fwlink/?LinkID=135170 . Вы хотите изменить политику выполнения?
[Y] Да - Y [A] Да для всех - A [N] Нет - N [L] Нет для всех - L [S] Приостановить - S [?] Справка
(значением по умолчанию является "N"):
```

CMD vs PowerShell

Возможности	CMD	PowerShell
Обработка исключений	—	+
Параллельное присваивание	—	+
Аргументы по умолчанию	—	+
Именованные параметры	—	+
Лямбда-функции	—	+
Байт-код	—	+

PowerShell. Обработка ошибок

```
# обработка ошибки подход "до"  
function mydel1([int]$a, [int]$b){  
  if ($b -ne 0){  
    $c = $a / $b  
    write-output $c  
  }  
  else{  
    write-output "деление на ноль?"  
  }  
}
```

```
# обработка ошибки подход "после"  
(обработка исключений)  
function mydel2([int]$a, [int]$b){  
  try{  
    $c = $a / $b  
    write-output $c  
  }  
  catch{  
    write-output "деление на ноль?"  
  }  
}
```

PowerShell. Аргументы по умолчанию и именованные параметры

```
# Вычисление площади  
function square ([float] $a,  
[float] $b=0) {  
    if ($b -eq 0) {  
        # Вычисляем площадь  
квдрата  
        return $a * $a  
    }  
    else {  
        return $a * $b  
    }  
}
```



```
# Вызовы функции  
square 5 6  
# Пример: аргумент b задан по  
умолчанию  
square 5  
# Использование именованных
```

PowerShell. Лямбда-функции

```
# Лямбда-функция (функция без имени)  
# Можно функцию присвоить в переменную  
$F = {  
    param([float]$a, [float]$b=0)  
    if ($b -eq 0) {  
        # Вычисляем площадь квадрата  
        return $a * $a  
    }  
    else{  
        return $a * $b  
    }  
}  
  
# Вызовы функции-переменной  
# Вариант вызова 1  
& $F 7 8
```

PowerShell. Скрипт с параметрами

```
# Указываем входные  
параметры файла скрипта  
param([float]$a, [float]$b)
```

```
# Вычисление площади  
if ($b -eq 0) {  
    # Вычисляем площадь  
    квадрата  
    return $a * $a  
}  
else {  
    return $a * $b  
}
```

Скрипт: изменить расширение файлов в текущей папке

Использование конвейера команд:

результат выполнения командлета `Get-ChildItem` подается на вход следующему командлету `Rename-Item`

```
Get-Childitem *.txt | Rename-Item -NewName { $_.Name  
-replace '.txt', '.log' }
```

Аналог с использованием алиасов

```
dir *.txt | ren -NewName { $_.Name -replace '.txt', '.log' }
```

Скрипт: изменить название файлов в текущей папке

```
# Получаем список файлов текущей директории
```

```
$files = Get-Childitem
```

```
Write-Output $files.Count
```

```
For ($i=0; $i -lt $files.Count; $i++) {
```

```
Write-Output $files[$i].Name
```

```
# Демонстрация механизма параллельного присваивания
```

```
$name, $ext = $files[$i].BaseName,  
$files[$i].Extension
```

```
# Формируем новое имя для текущего файла
```

```
$newname = $name + "999" + $ext
```

```
Write-Output $newname
```

```
# Заменяем имя текущего файла
```

```
Renome-Item $files[$i].Name $newname
```

Скрипт: изменить название файлов в текущей папке (через конвейер)

```
dir | ren -newName { $_.BaseName + "999" +  
    $_.Extension }
```