Практика 1. Взаимодействия с ОС через командный интерфейс

Шарапов Ю.А.

Команды CMD

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

Команды PowerShell

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

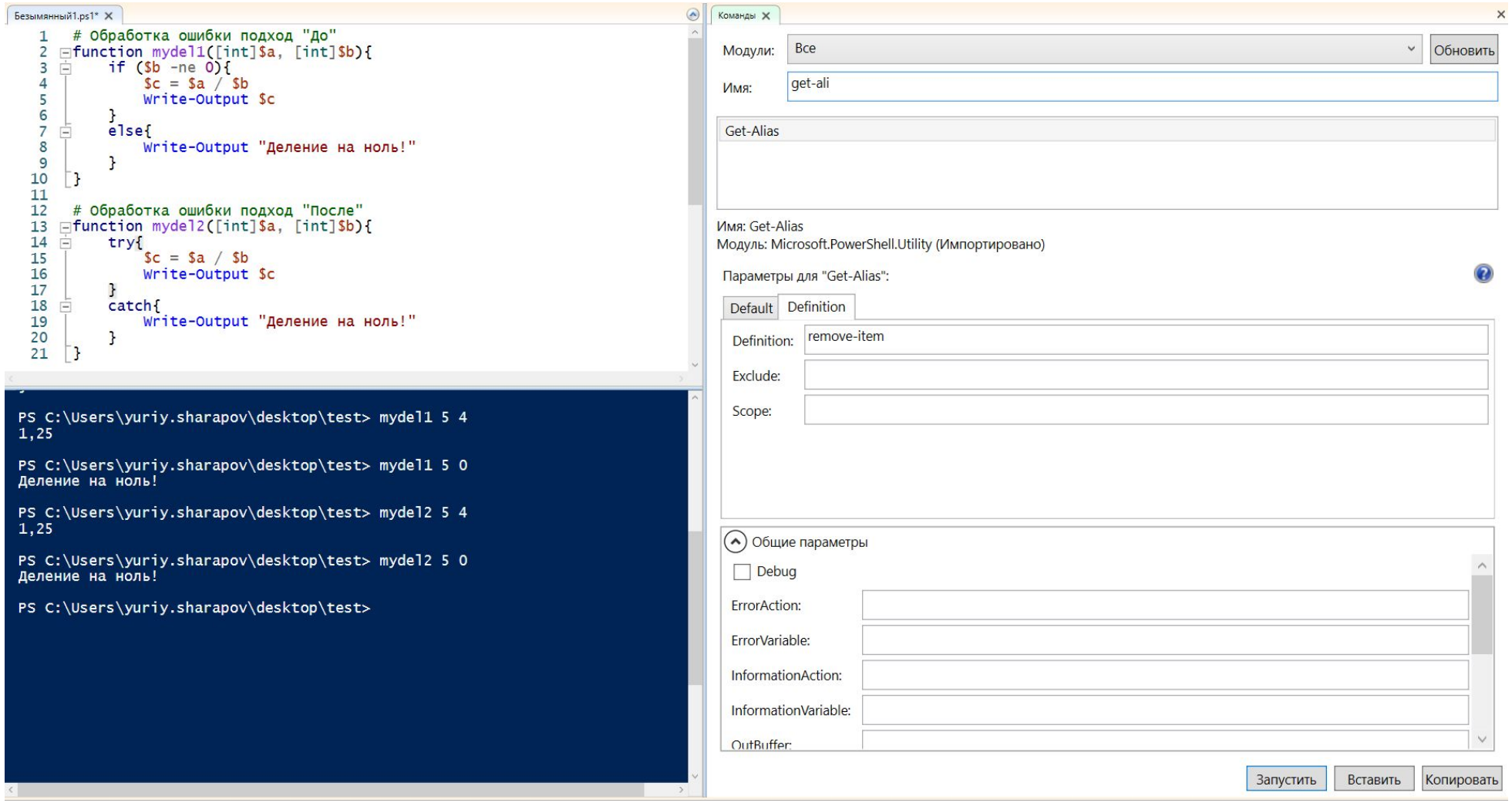
Найти алиас, соответствующий командлету **`Get-ChildItem`**

```
Get-Alias -Definition
Get-ChildItem
```

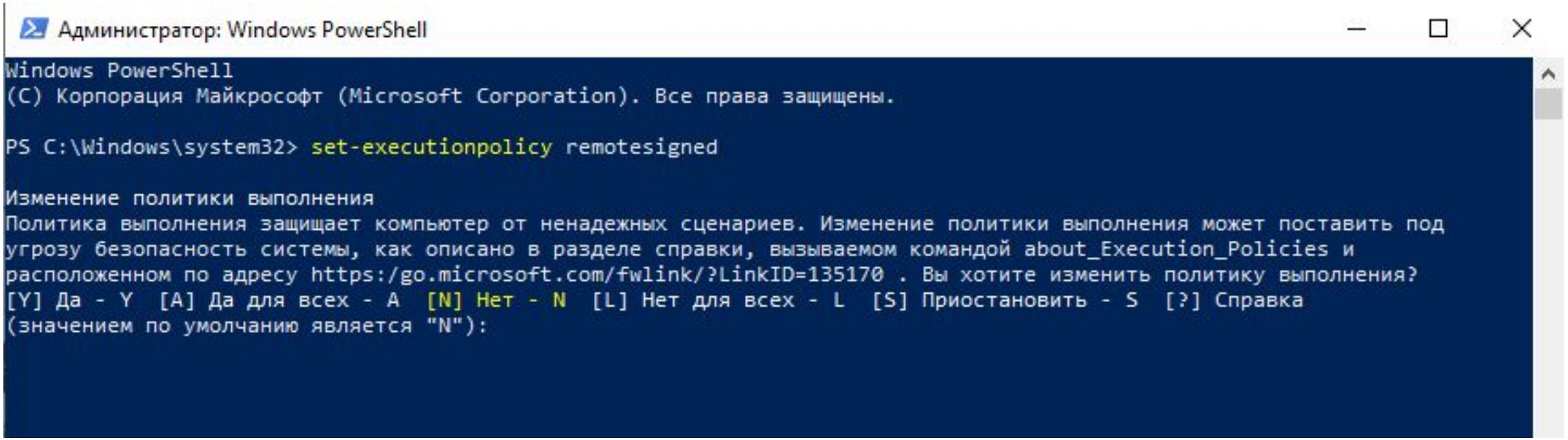
Программные инструменты Windows PowerShell

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



Включить выполнение скриптов в PowerShell (из под администратора)



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
квадрата
    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
  Rename-Item $files[$i].Name $newname
```

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

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