Лабораторная работа №3

Шулер М.А 1-МД-20

**4. Вывести содержимое каталога Windows**

Get-ChildItem "C:\Windows" -File |

Where-Object { $\_.Length -gt 10000 } |

Sort-Object Length |

Select-Object Name, Length, CreationTime, Attributes

**Get-ChildItem** - перечисляет содержимое каталога

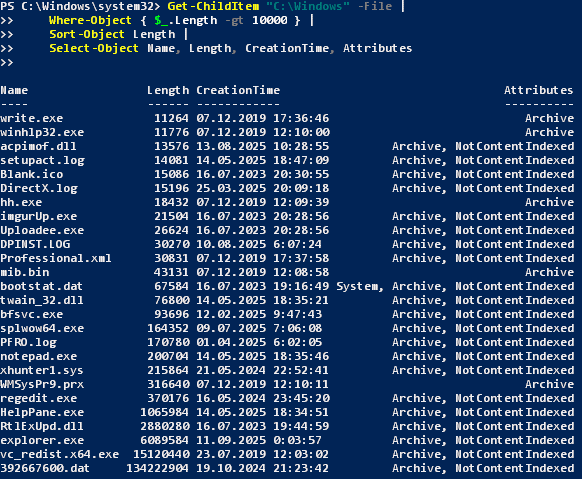
**| (конвейер)** - Передаём объекты (файлы) дальше

**Where-Object** - фильтрует объекты по условию.

**$\_** — текущий объект в конвейере

**Sort-Object –** сортировка

**Select-Object -** позволяет выбирать определённые свойства объектов



**5. Вывести список свойств процесса в файл и общее количество**

$props = (Get-Process | Get-Member -MemberType Property).Name

$props | Out-File "C:\Lab3\_process\_props.txt"

Write-Host "Количество свойств:" $props.Count

**Get-Process** - возвращает список всех процессов

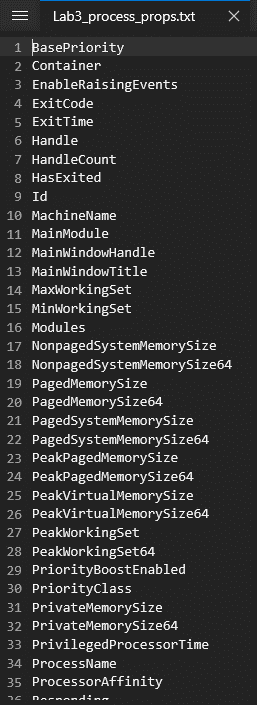
**Get-Member** - показывает все свойства и методы объекта, который поступает на вход

**-MemberType Property** - отбирает только свойства (без методов и событий)

**.Name** – только имена

**Out-File** - перенаправляет вывод в текстовый файл.





**6. Создать текстовый файл со списком процессов**

Get-Process |

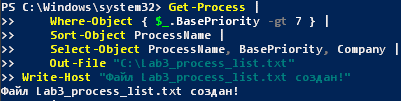
Where-Object { $\_.BasePriority -gt 7 } |

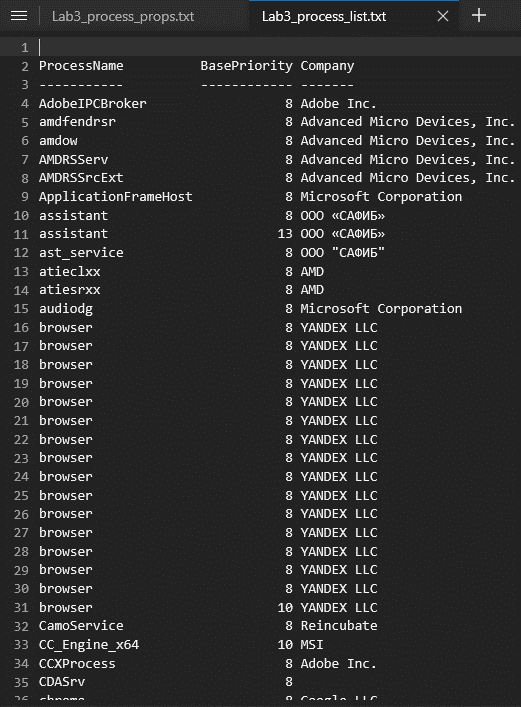
Sort-Object ProcessName |

Select-Object ProcessName, BasePriority, Company |

Out-File "C:\Lab3\_process\_list.txt"

Write-Host "Файл Lab3\_process\_list.txt создан!"





**7. Создать HTML-файл со списком процессов**

$processes = Get-Process |

Where-Object { $\_.BasePriority -gt 7 } |

Select-Object ProcessName, Id, BasePriority, Path |

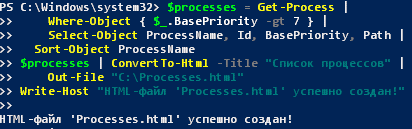
Sort-Object ProcessName

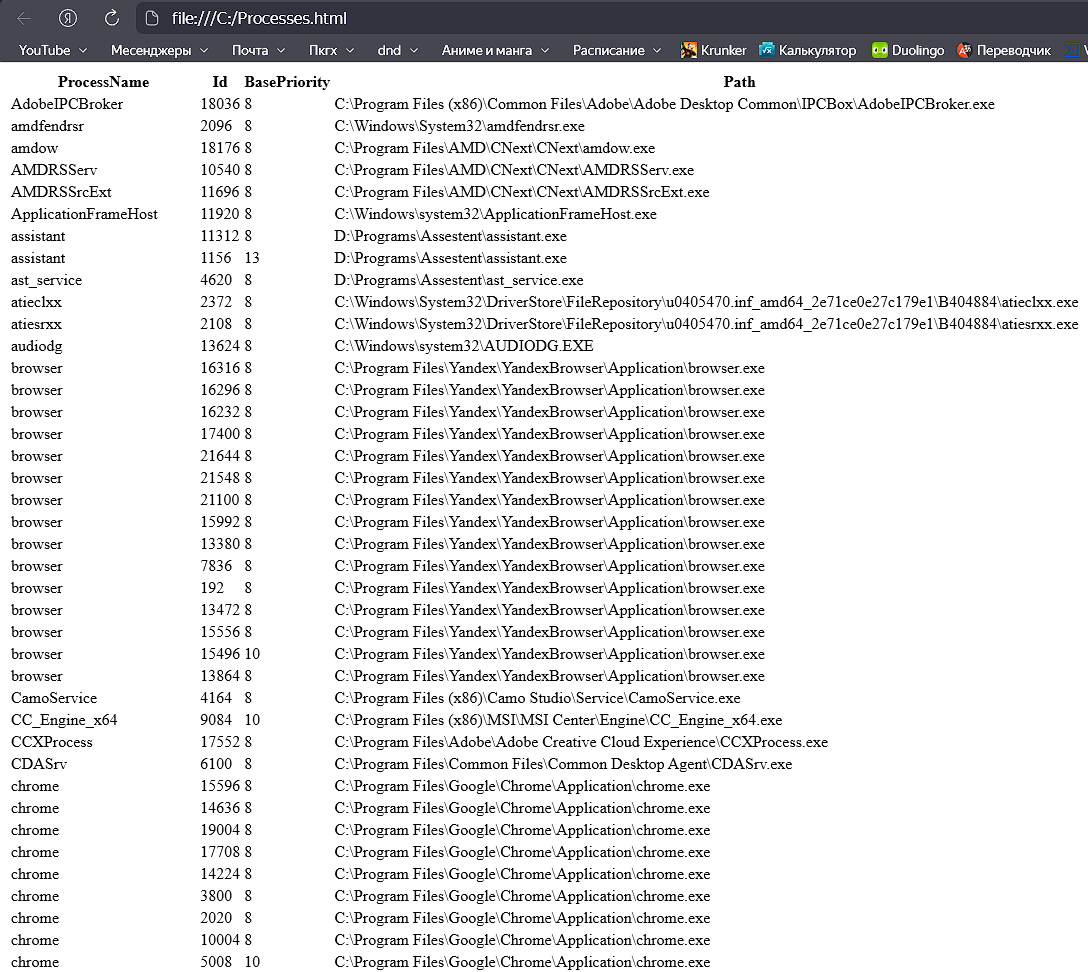
$processes | ConvertTo-Html -Title "Список процессов" |

Out-File "C: \Processes.html"

Write-Host "HTML-файл 'Processes.html' успешно создан!"

**ConvertTo-Html** — командлет в PowerShell, который преобразует объекты .NET в HTML, которые можно отобразить в веб-браузере.





**8. Найти суммарный объем всех bmp и jpg файлов в каталоге Windows и подкаталогах**

$totalSize = Get-ChildItem "C:\Windows" -Include \*.bmp, \*.jpg -Recurse -ErrorAction SilentlyContinue |

Measure-Object -Property Length -Sum

Write-Host "Суммарный объем графических файлов:" ($totalSize.Sum / 1MB) "MB"

**-Recurse** — чтобы обойти все подпапки

**-ErrorAction SilentlyContinue** — чтобы пропустить защищённые папки (без прав доступа)

**Measure-Object -Sum** — суммирует все значения

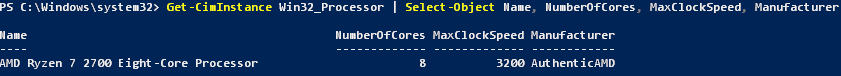
**.Sum** – только сумма



**9. Вывести сведения о ЦП компьютера**

Get-CimInstance Win32\_Processor | Select-Object Name, NumberOfCores, MaxClockSpeed, Manufacturer

**Get-CimInstance Win32\_Processor** — команда PowerShell, которая позволяет получить данные о процессоре



**10. Найти максимальное, минимальное и среднее время выполнения командлетов dir и ps**

$dirTimes = @()

$psTimes = @()

for ($i = 1; $i -le 5; $i++) {

$dirTime = (Measure-Command { Get-ChildItem C:\Windows })

$psTime = (Measure-Command { Get-Process })

$dirTimes += $dirTime.TotalMilliseconds

$psTimes += $psTime.TotalMilliseconds

}

Write-Host "`nРезультаты для Get-ChildItem:"

Write-Host " Минимум: $([Math]::Round(($dirTimes | Measure-Object -Minimum).Minimum,2)) мс"

Write-Host " Максимум: $([Math]::Round(($dirTimes | Measure-Object -Maximum).Maximum,2)) мс"

Write-Host " Среднее: $([Math]::Round(($dirTimes | Measure-Object -Average).Average,2)) мс"

Write-Host "`nРезультаты для Get-Process:"

Write-Host " Минимум: $([Math]::Round(($psTimes | Measure-Object -Minimum).Minimum,2)) мс"

Write-Host " Максимум: $([Math]::Round(($psTimes | Measure-Object -Maximum).Maximum,2)) мс"

Write-Host " Среднее: $([Math]::Round(($psTimes | Measure-Object -Average).Average,2)) мс"

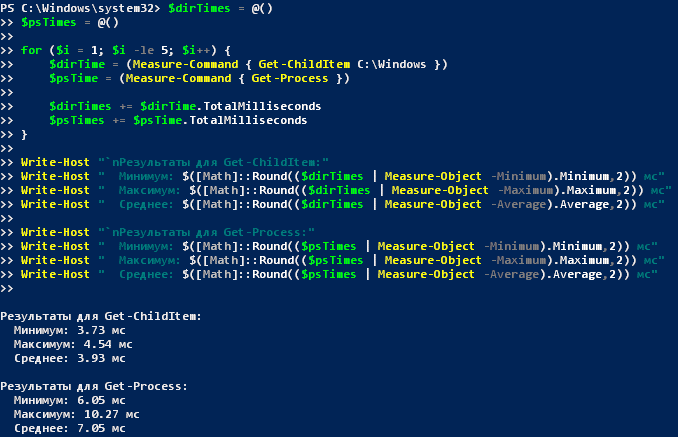
**Measure-Command** — встроенный PowerShell командлет для замера времени выполнения скрипта или команды

**.TotalMilliseconds** — преобразуем результат в миллисекунды

**Measure-Object -Minimum** — находит минимальное значение в массиве

**Measure-Object -Maximum** — максимальное

**Measure-Object -Average** — среднее



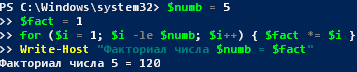
**11. Индивидуальные задания**

*11.1 — Факториал*

$numb = 5

$fact = 1

for ($i = 1; $i -le $numb; $i++) { $fact \*= $i }

Write-Host "Факториал числа $numb = $fact"

*11.2 — Минимум и максимум из файла*

"1`n5`n9`n3`n8" | Out-File "C:\nn.txt"

$data = Get-Content "C:\nn.txt" | ForEach-Object { [int]$\_ }

$min = ($data | Measure-Object -Minimum).Minimum

$max = ($data | Measure-Object -Maximum).Maximum

Write-Host "Минимум: $min, Максимум: $max"

**Get-Content "nn.txt"** — считываем строки файла

**ForEach-Object { [int]$\_ }** — преобразуем каждую строку в число

