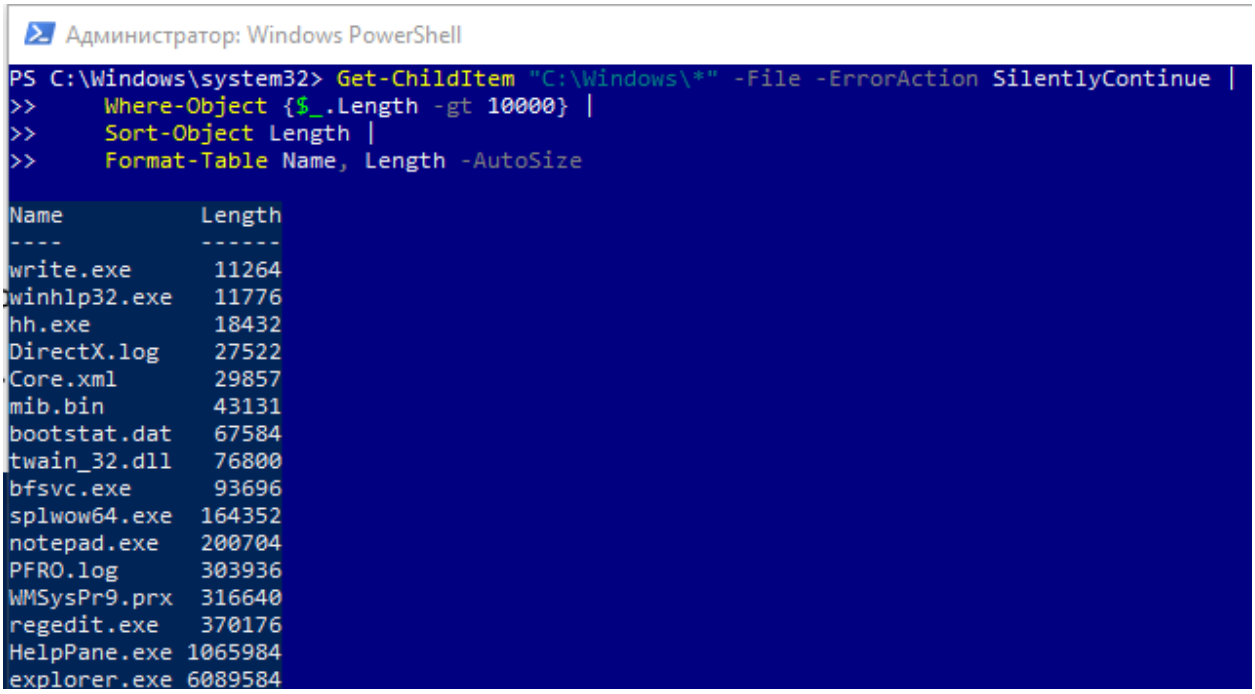
Лабораторная работа №3

1. Вывести файлы от 10000 размера



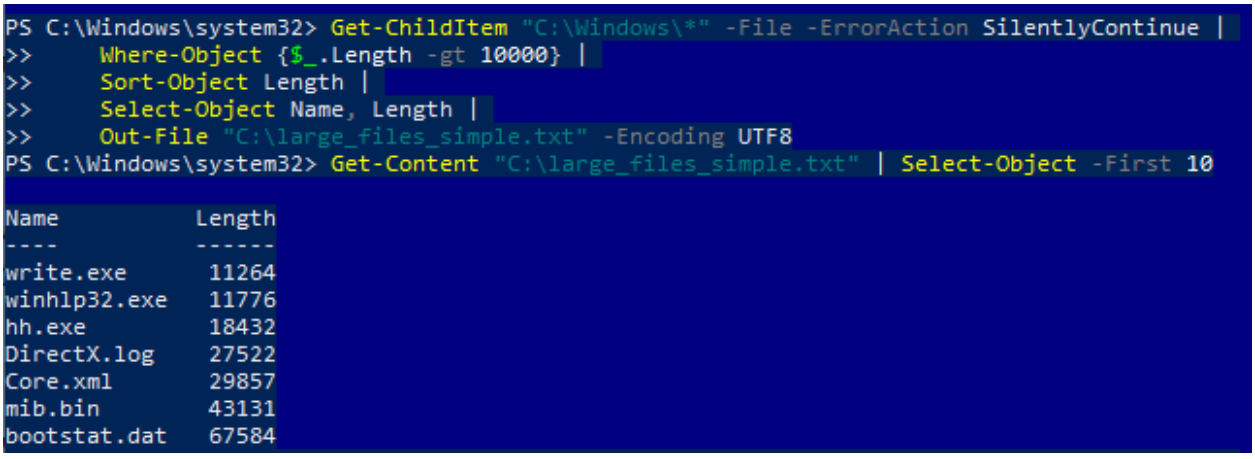
Get-ChildItem – получение содержимого папки

ErrorAction SilentlyContinue – игнорирует ошибки

Where-Object {$_.Length -gt 10000} – фильтрует объекты больше 10к байт

Sort-Object Length – сортировка по размеру

Format-Table – формирует таблицу



Теперь все это записываем в файл C:\large_files_simple.txt

Get-Content – считываем содержание файла

2. Вывести в текстовый файл список процессов и показать их кол-во

```
PS C:\Windows\system32> Get-Process | Out-File "C:\all_processes.txt" -Encoding UTF8
PS C:\Windows\system32> $processCount = (Get-Process).Count
>> Write-Host "Общее количество процессов: $processCount" -ForegroundColor Green
Общее количество процессов: 237
PS C:\Windows\system32> Write-Host "`nПример первых 5 процессов:" -ForegroundColor Yellow
>> Get-Process | Select-Object -First 5 | Format-Table ProcessName, Id, CPU -AutoSize

Пример первых 5 процессов:

ProcessName        Id       CPU
-----------        --       ---
AggregatorHost   6880  3,328125
amdfendrsr       1912         0
amdow           14464   0,03125
AMDRSServ       14380 11,640625
AMDRSSrcExt      3008    0,1875
```

Get-Process – получает процессы

Записываем их в файл "C:\all_processes.txt"

.Count – получает количество

$processCount – переменная для кол-ва

Выводим первые 5 на экран

3. Создать текстовый файл, содержащий список выполняемых процессов, упорядоченный по возрастанию указанного параметра

```
PS C:\Windows\system32> $filteredProcesses = Get-Process |
>>     Select-Object ProcessName, BasePriority, Company |
>>     Where-Object {$_.BasePriority -gt 7} |
>>     Sort-Object ProcessName
PS C:\Windows\system32> $filteredProcesses | Out-File "C:\high_priority_processes.txt" -Encoding UTF8
PS C:\Windows\system32> Write-Host "Процессы с BasePriority > 7 сохранены в файл!" -ForegroundColor Green
>> Write-Host "Количество процессов: $($filteredProcesses.Count)" -ForegroundColor Yellow
>> Write-Host "Файл: C:\high_priority_processes.txt" -ForegroundColor Cyan
>>
Процессы с BasePriority > 7 сохранены в файл!
Количество процессов: 198
Файл: C:\high_priority_processes.txt
PS C:\Windows\system32> Write-Host "`nПервые 5 процессов:" -ForegroundColor Magenta
>> $filteredProcesses | Select-Object -First 5 | Format-Table -AutoSize

Первые 5 процессов:

ProcessName    BasePriority Company
-----------    ------------ -------
AggregatorHost            8
amdfendrsr                8 Advanced Micro Devices, Inc.
amdow                     8 Advanced Micro Devices, Inc.
AMDRSServ                 8 Advanced Micro Devices, Inc.
AMDRSSrcExt               8 Advanced Micro Devices, Inc.
```

Select-Object – выбирает нужные колонки

Where-Object {$_.BasePriority -gt 7} – фильтр БейсПриорити больше 7

Sort-Object ProcessName – сортировка по имени

Сохраняем в C:\high_priority_processes.txt

$filteredProcesses – переменная с отфильтрованными процессами

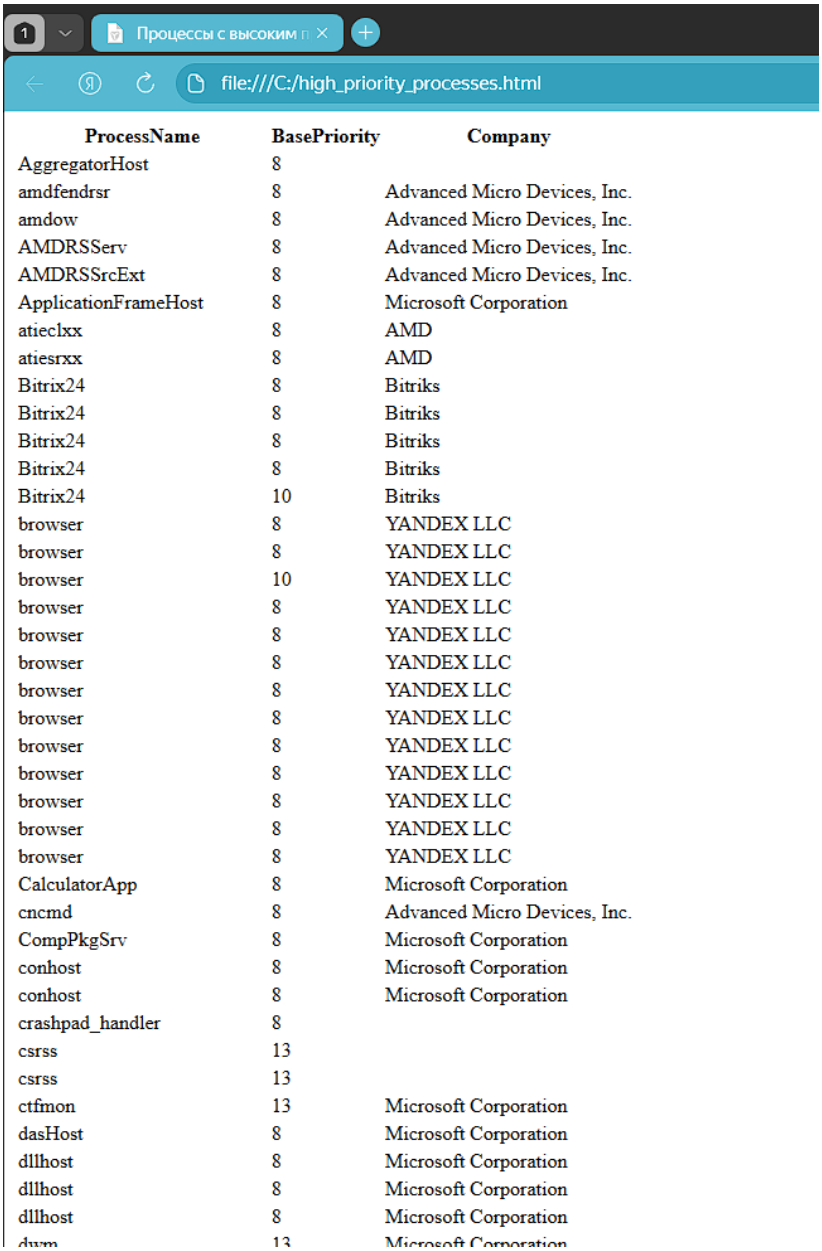
4. Создать HTML-файл, содержащий список выполняемых процессов, упорядоченный по возрастанию указанного параметра

```
PS C:\Windows\system32> Get-Process |
>>     Select-Object ProcessName, BasePriority, Company |
>>     Where-Object {$_.BasePriority -gt 7} |
>>     Sort-Object ProcessName |
>>     ConvertTo-Html -Title "Процессы с высоким приоритетом (BasePriority > 7)" |
>>     Out-File "C:\high_priority_processes.html"
PS C:\Windows\system32> Start-Process "C:\high_priority_processes.html"
PS C:\Windows\system32>
```

ConvertTo-Html – конвертирует данные в формат html

Out-File "C:\high_priority_processes.html" – сохраняет файл

Результат:



ProcessName	BasePriority	Company
AggregatorHost	8	
amdfendrsr	8	Advanced Micro Devices, Inc.
amdow	8	Advanced Micro Devices, Inc.
AMDRSServ	8	Advanced Micro Devices, Inc.
AMDRSSrcExt	8	Advanced Micro Devices, Inc.
ApplicationFrameHost	8	Microsoft Corporation
atieclxx	8	AMD
atiesrxx	8	AMD
Bitrix24	8	Bitriks
Bitrix24	8	Bitriks
Bitrix24	8	Bitriks
Bitrix24	8	Bitriks
Bitrix24	10	Bitriks
browser	8	YANDEX LLC
browser	8	YANDEX LLC
browser	10	YANDEX LLC
browser	8	YANDEX LLC
browser	8	YANDEX LLC
browser	8	YANDEX LLC
browser	8	YANDEX LLC
browser	8	YANDEX LLC
browser	8	YANDEX LLC
browser	8	YANDEX LLC
browser	8	YANDEX LLC
browser	8	YANDEX LLC
browser	8	YANDEX LLC
CalculatorApp	8	Microsoft Corporation
cncmd	8	Advanced Micro Devices, Inc.
CompPkgSrv	8	Microsoft Corporation
conhost	8	Microsoft Corporation
conhost	8	Microsoft Corporation
crashpad_handler	8	
csrss	13	
csrss	13	
ctfmon	13	Microsoft Corporation
dasHost	8	Microsoft Corporation
dllhost	8	Microsoft Corporation
dllhost	8	Microsoft Corporation
dllhost	8	Microsoft Corporation
dwm	13	Microsoft Corporation

5. Найти суммарный объем всех графических файлов

```
PS C:\Windows\system32> $files = Get-ChildItem "C:\Users\Ignat\Pictures\Saved Pictures\*" -Include "*.bmp", "*.jpg", "*.jpeg"
, "*.png" -File
>> $totalSize = ($files | Measure-Object -Property Length -Sum).Sum
>>
>> Write-Host "Количество графических файлов: $($files.Count)"
>> Write-Host "Суммарный размер: $totalSize байт"
Количество графических файлов: 2
Суммарный размер: 1652329 байт
PS C:\Windows\system32>
```

Include "*.bmp", "*.jpg", "*.jpeg", "*.png" -File – ищет эти форматы

Сохраняем результат в переменную $files

Measure-Object -Property Length -Sum – измеряет суммарный размер

Сумму сохраняем в $totalSize

6. Вывод информации о ЦП

```
PS C:\Windows\system32>  Get-CimInstance Win32_Processor | Format-List


Caption            : Intel64 Family 6 Model 151 Stepping 5
DeviceID           : CPU0
Manufacturer       : GenuineIntel
MaxClockSpeed      : 2500
Name               : 12th Gen Intel(R) Core(TM) i5-12400
SocketDesignation  : U3E1
```

7. Найти максимальное, минимальное и среднее значение времени выполнения команд

```
PS C:\Windows\system32> $dirStats = 1..5 | %{(Measure-Command {dir C:\Windows >$null}).TotalMill
iseconds} | measure -Min -Max -Average
>> $psStats = 1..5 | %{(Measure-Command {Get-Process >$null}).TotalMilliseconds} | measure -Min
-Max -Average
>>
>> Write-Host "DIR: Мин=$([math]::Round($dirStats.Minimum,2))ms, Макс=$([math]::Round($dirStats.
Maximum,2))ms, Сред=$([math]::Round($dirStats.Average,2))ms"
>> Write-Host "PS: Мин=$([math]::Round($psStats.Minimum,2))ms, Макс=$([math]::Round($psStats.Max
imum,2))ms, Сред=$([math]::Round($psStats.Average,2))ms"
DIR: Мин=1.26ms, Макс=3.06ms, Сред=1.87ms
PS: Мин=1.64ms, Макс=5.04ms, Сред=2.57ms
PS C:\Windows\system32>
```

Measure-Command {dir C:\Windows >$null - замеряет время выполнения dir

> $null - скрывает вывод команды

.TotalMilliseconds - берет время в мс

| measure -Min -Max -Average - считает минимум, максимум и среднее

Сохраняем статистику в переменную $dirStats

Аналогично для ps