***Лабораторна робота №10***

***«Управління виведенням команд у Windows PowerShell. Перегляд структури, фільтрація та сортування об’єктів»***

***Конвеєризація і управління виведенням команд Windows PowerShell***

Раніше було розглянуто поняття конвеєризації (або композиції) команд інтерпретатора **Cmd.exe**, коли вихідний потік однієї команди перенаправлявся у вхідний потік іншої, об'єднуючи тим самим дві команди разом. Подібні конвеєри команд використовуються в більшості оболонок командного рядка і є засобом, що дозволяє передавати інформацію між різними процесами. Механізм композиції команд представляє, ймовірно, найбільш цінну концепцію, яка використовується в інтерфейсах командного рядка. Конвеєри не тільки знижують зусилля, прикладені при введенні складних команд, але і полегшують відстеження потоку роботи в командах.

В оболонці **PowerShell** також дуже широко використовується механізм конвеєризації команд, проте тут по конвеєру передається не потік тексту, як у всіх інших оболонках, а **об'єкти**. При цьому з елементами конвеєра можна виробляти різні маніпуляції: фільтрувати об'єкти за певним критерієм, сортувати і групувати об'єкти, змінювати їх структуру (нижче ми докладніше розглянемо операції фільтрації і сортування елементів конвеєра).

***Конвеєризація об'єктів в PowerShell***

Конвеєр в **PowerShell** - це послідовність команд, розділених між собою знаком ***|*** (Вертикальна риса). Кожна команда в конвеєрі отримує об'єкт від попередньої команди, виконує певні операції над ним та передає наступній команді в конвеєрі. З точки зору користувача, об'єкти упаковують пов'язану інформацію в форму, в якій інформацією простіше маніпулювати як єдиним блоком і з якої при необхідності витягуються певні елементи.

Передача даних між командами у вигляді об'єктів має велику перевагу над звичайним обміном інформацією за допомогою потоку тексту. Адже команда, яка бере потік тексту від іншої утиліти, повинна його проаналізувати, розібрати і виділити потрібну їй інформацію, а це може бути непросто, так як зазвичай виведення команди більше орієнтоване на візуальне сприйняття людиною (це природно для інтерактивного режиму роботи), а не на зручність подальшого синтаксичного розбору.

При передачі по конвеєру об'єктів цієї проблеми не виникає, тут потрібна інформація витягується з елемента конвеєра простим зверненням до відповідної властивості об'єкта. Однак виникає нове запитання: яким чином дізнатися, які саме властивості є у об'єктів, що передаються по конвеєру? Адже при виконанні того чи іншого командлету ми на екрані бачимо тільки одну або кілька колонок відформатованого тексту.

**Приклад.** Запустимо командлет ***Get-Process***, який виводить інформацію про запущені в системі процеси:

***PS C:\> Get-Process***

***Handles NPM(K) PM(K) WS(K) CPU(s) Id SI ProcessName***

***------- ------ ----- ----- ------ -- -- -----------***

***80 4 876 4184 1780 0 AsLdrSrv***

***3372 44 61296 65988 1812 0 AvastSvc***

***989 24 17316 36228 6,80 8124 3 AvastUI***

***107 5 2592 10488 0,33 3184 3 conhost***

***---------------------------------------------------------------***

***746 21 65836 65372 63,73 2576 3 WINWORD***

***117 4 900 5268 2224 0 WmiApSrv***

***260 6 1492 7456 4956 0 WUDFHost***

Фактично на екрані ми бачимо тільки зведену інформацію (результат форматування отриманих даних), а не повне представлення вихідного об'єкта. З цієї інформації незрозуміло, скільки точно властивостей є у об'єктів, що генеруються командою ***Get-Process***, і які імена мають ці властивості. Наприклад, ми хочемо знайти всі "завислі" процеси, які не відповідають на запити системи. Чи можна це зробити за допомогою командлета ***Get-Process***, яку властивість потрібно перевіряти у виведених об'єктів?

Для відповіді на ці запитання потрібно навчитися досліджувати структуру об'єктів **PowerShell**, дізнаватися, які властивості і методи є у цих об'єктів.

***Перегляд структури об'єктів***

Для аналізу структури об'єкта, що повертається певною командою, найпростіше направити цей об'єкт по конвеєру на командлет ***Get-Member*** (псевдонім ***gm***), наприклад:

***PS C:\> Get-Process | Get-Member***

***TypeName: System.Diagnostics.Process***

***Name MemberType Definition***

***---- ---------- ----------***

***Handles AliasProperty Handles = Handlecount***

***Name AliasProperty Name = ProcessName***

***NPM AliasProperty NPM = NonpagedSystemMemorySize***

***PM AliasProperty PM = PagedMemorySize***

***VM AliasProperty VM = VirtualMemorySize***

***WS AliasProperty WS = WorkingSet***

***-----------------------------------------------------------------***

***Responding Property System.Boolean Responding {get;}***

***-----------------------------------------------------------------***

Тут ми бачимо ім'я .NET-класу, екземпляри якого повертаються в ході роботи досліджуваного командлету (в нашому прикладі це клас ***System.Diagnostic.Process***), а також повний список елементів об'єкта (зокрема, цікавить нас властивість ***Responding***, що визначає "завислі" процеси ). При цьому на екран виводиться дуже багато елементів, переглядати їх незручно. Командлет ***Get-Member*** дозволяє перерахувати тільки ті елементи об'єкта, які є його властивостями. Для цього використовується параметр ***-MemberType*** зі значенням ***Properties***:

***PS C:\> Get-Process | Get-Member -MemberType Property***

***TypeName: System.Diagnostics.Process***

***Name MemberType Definition***

***---- ---------- ----------***

***BasePriority Property System.Int32 BasePriority {get;}***

***ExitCode Property System.Int32 ExitCode {get;}***

***ExitTime Property System.DateTime ExitTime {get;}***

***Handle Property System.IntPtr Handle {get;}***

***HandleCount Property System.Int32 HandleCount {get;}***

***HasExited Property System.Boolean HasExited {get;}***

***Id Property System.Int32 Id {get;}***

***. . .***

***Responding Property System.Boolean Responding {get;}***

Процесам ОС відповідають об'єкти, що мають дуже багато властивостей, на екран же при роботі командлет ***Get-Process*** виводяться лише кілька з них (способи відображення об'єктів різних типів задаються файлами в форматі XML, що знаходяться в каталозі, де встановлений файл **powershell.exe**).

Розглянемо найбільш часто використовувані операції над елементами конвеєра: фільтрації і сортування.

***Фільтрація об'єктів в конвеєрі***

У **PowerShell** підтримується можливість фільтрації об'єктів в конвеєрі, тобто видалення з конвеєра об'єктів, які задовольняють певній умові. Дану функціональність забезпечує командлет ***Where-Object***, що дозволяє перевірити кожен об'єкт, що знаходиться в конвеєрі, і передати його далі по конвеєру, тільки якщо об'єкт задовольняє умовам перевірки.

Наприклад, для виведення інформації про "завислі" процеси (об'єкти, які повертаються командлетом ***Get-Process***, у яких властивість ***Responding*** дорівнює ***False***) можна використовувати наступний конвеєр:

***Get-Process | Where-Object {-not $\_.Responding}***

Інший приклад - залишимо в конвеєрі тільки ті процеси, у яких значення ідентифікатора (властивість ***Id***) більше ***1000***:

***Get-Process | Where-Object {$\_.Id -gt 1000}***

У блоках сценаріїв командлет ***Where-Object*** для звернення до поточного об'єкту конвеєра і вилучення потрібних властивостей цього об'єкта використовується спеціальна змінна ***$\_***, яка створюється оболонкою **PowerShell** автоматично. Дана змінна використовується і в інших командлетах, які виробляють обробку елементів конвеєра.

Умова перевірки в ***Where-Object*** задається в вигляді блоку сценарію - однієї або декількох команд **PowerShell**, укладених у фігурні дужки ***{}***. Результатом виконання даного блоку сценарію має бути значення логічного типу: ***True*** або ***False***). Як можна зрозуміти з прикладів, в блоці сценарію використовуються спеціальні оператори порівняння.

**Зауваження.** У **PowerShell** для операторів порівняння не використовуються звичайні символи ***>*** або ***<***, так як в командному рядку вони зазвичай означають перенаправлення введення/виведення.

Основні оператори порівняння наведені в табл. 1.

Табл 1. Оператори порівняння в **PowerShell**

|  |  |  |
| --- | --- | --- |
| **Оператор** | **Значення** | **Приклад (повертається значення True)** |
| ***-eq*** | дорівнює | ***10 -eq 10*** |
| ***-ne*** | не дорівнює | ***9 -ne 10*** |
| ***-lt*** | менше | ***3 -lt 4*** |
| ***-le*** | менше або дорівнює | ***3 –le 4*** |
| ***-gt*** | більше | ***4 -gt 3*** |
| ***-ge*** | більше або дорівнює | ***4 -ge 3*** |
| ***-like*** | порівняння на збіг з урахуванням символи узагальнення в тексті | ***"file.doc" –like "f\*.doc"*** |
| ***-notlike*** | порівняння на розбіжність з урахуванням символи узагальнення в тексті | ***"file.doc" –notlike "f\*.rtf"*** |
| ***-contains*** | містить | ***1,2,3 –contains 1*** |
| ***-notcontains*** | не містить | ***1,2,3 –notcontains 4*** |

Оператори порівняння можна з'єднувати один з одним за допомогою логічних операторів (табл. 2).

Табл. 2. Логічні оператори в **PowerShell**

|  |  |  |
| --- | --- | --- |
| **Оператор** | **Значення** | **Приклад (повертається значення True)** |
| ***-and*** | логічне І | ***(10 -eq 10) –and (1 –eq 1)*** |
| ***-or*** | логічне АБО | ***(9 -ne 10) –or (3 –eq 4)*** |
| ***-not*** | логічне НЕ | ***-not (3 –gt 4)*** |
| ***!*** | логічне НЕ | ***!(3 -gt 4)*** |

***Сортування об'єктів***

Сортування елементів конвеєра - ще одна операція, яка часто застосовується при конвеєрної обробці об'єктів. Дану операцію здійснює командлет ***Sort-Object***: йому передаються імена властивостей, за якими потрібно провести сортування, а він повертає дані, впорядковані за значеннями цих властивостей.

Наприклад, для виведення списку запущених в системі процесів, впорядкованого за витраченому процесорного часу (властивість ***cpu***), можна скористатися наступним конвеєром:

***PS C:\> Get-Process | Sort-Object cpu***

Для сортування в зворотному порядку використовується параметр ***Descending***:

***PS C:\> Get-Process | Sort-Object cpu –Descending***

У розглянутих нами прикладах конвеєри складалися з двох командлетів. Це не обов'язкова умова, конвеєр може об'єднувати і більшу кількість команд, наприклад:

***Get-Process | Where-Object {$ \_. Id -gt 1000} | Sort-Object cpu –Descending***

У **PowerShell** є командлет ***Select-Object***, за допомогою якого можна виділяти вказану кількість об'єктів з початку або з кінця конвеєра, вибирати унікальні об'єкти з конвеєра, а також виділяти певні властивості в об'єктах, що проходять по конвеєру.

Для виділення з конвеєра декількох перших або останніх об'єктів слід скористатися відповідно параметрами ***-First*** або ***–Last*** командлета ***Select-Object***. Наприклад, наступний конвеєр команд виведе на екран інформацію про п'ять процесів, що використовують найбільший обсяг пам'яті:

***PS С:\> Get-Process | Sort-Object WS | Select-Object -Last 5***

***Handles NPM(K) PM(K) WS(K) VM(M) CPU(s) Id ProcessName***

***------- ------ ----- ----- ----- ------ -- -----------***

***1025 51 81192 136852 417 94,84 3360 chrome***

***815 328 264024 154796 453 936 sqlservr***

***274 45 135896 209136 932 25,06 5064 chrome***

***328 40 94880 236864 962 52,09 6232 chrome***

***749 628 414860 244072 974 2024 msmdsrv***

Перший командлет в конвеєрі (***Get-Process***) повертає масив об'єктів, відповідних запущеним в системі процесам. Другий командлет ***Sort-Object*** впорядковує об'єкти, проходять по конвеєру, за значенням властивості ***WS*** (обсяг пам'яті, займаної процесом). Нарешті, третій командлет ***Select-Object*** вибирає з упорядкованого масиву об'єкта останні п'ять елементів.

Припустимо тепер, що нам потрібно отримати список запущених в системі процесів, в якому були б вказані тільки імена процесів і їх ідентифікатори. Якщо ви не пам'ятаєте назви потрібних властивостей, то можна за допомогою командлета ***Get-Member*** знову переглянути структуру об'єктів, що повертаються командою ***Get-Process***:

***PS С:\> Get-Process | Get-Member -MemberType Property***

***Name MemberType Definition***

***---- ---------- ----------***

***BasePriority Property System.Int32 BasePriority {get;}***

***---------------------------------------------------------------------***

***Id Property System.Int32 Id {get;}***

***---------------------------------------------------------------------***

***ProcessName Property System.String ProcessName {get;}***

***---------------------------------------------------------------------***

Отже, в підсумкових об'єктах нам потрібно залишити тільки властивості ***ProcessName*** і ***Id***. Це можна зробити, вказавши імена потрібних властивостей в якості параметрів командлет ***Select-Object***:

***PS С:\> Get-Process | Select-Object ProcessName, Id***

***ProcessName Id***

***----------- --***

***AcroRd32 5808***

***AcroRd32 5952***

***AGSService 1364***

***--------------------------------------------------------------***

***Використання змінних***

У змінних зберігаються всі можливі значення, навіть якщо вони є об'єктами. Імена змінних в **PowerShell** завжди повинні починатися з символу «***$***». Можна зберегти список процесів в змінної, це дозволить в будь-який час отримувати доступ до списку процесів. Присвоїти значення змінної легко:

***$a = Get-Process | Sort-Object CPU***

Вивести вміст змінної можна, просто надрукувавши в командному рядку ***$a***.

***Виконання довільних дій над об’єктами у конвеєрі***

Командлет ***ForEach-Object*** дозволяє виконати певний блок сценарію (код на мові **PowerShell**) для кожного об'єкта в конвеєрі. Іншими словами, за допомогою цього командлета можна виробляти довільні операції над елементами конвеєра. Для прикладу давайте підрахуємо загальний обсяг файлів, що зберігаються у певному каталозі диска. Для цього оголосимо змінну ***$TotalLength*** і присвоїмо її значення нулю:

***PS D:\Gleb> $TotalLength=0***

Тепер виконаємо команду ***dir*** і результат її роботи передамо по конвеєру командлету ***ForEach-Object***:

***PS D:\Gleb> dir | ForEach-Object {$TotalLength+=$\_.Length}***

У блоці сценарію командлет ***ForEach-Object*** до поточного значення змінної ***$TotalLength*** додається значення властивості ***Length*** проходить через конвеєр об'єкта (розмір відповідного цьому об'єкту файлу). В результаті у змінній ***$TotalLength*** буде зберігатися загальний розмір файлів в байтах:

***PS D:\Gleb> $TotalLength***

***20336777***

**Командлет Group-Object – групування об’єктів**

Об'єкти, що проходять по конвеєру, можна згрупувати за значенням певних властивостей за допомогою командлета ***Group-Object***. В одну групу будуть потрапляти об'єкти, що мають однакові значення зазначених властивостей (властивості можуть бути обчислювані).

Розглянемо приклад. Командлет ***Get-Process*** генерує об'єкти, що мають властивості ***Company*** (назва компанії-розробника певного модуля, запущеного в операційній системі в якості процесу). Виконаємо групування цих об'єктів за значенням властивості ***Company***:

***PS С:\> Get-Process | Group-Object Company***

***Count Name Group***

***----- ---- -----***

***94 {System.Diagnostics.Process (AGSService)…***

***1 Avast Software s.r.o. {System.Diagnostics.Process (AvastUI)}***

***6 Google Inc. {System.Diagnostics.Process (chrome)…***

***------------------------------------------------------------------------***

***Завдання***

**Для виконання завдань можливо знадобиться дізнатись назви властивостей відповідних об’єктів, що повертаються тим чи іншим командлетом!**

1. Вивести інформацію про 10 процесів, які споживають найбільший процесорний час.
2. Створити список всіх служб (командлет ***Get-Service***) та відсортуйте їх по статусу.
3. Вивести всі працюючі на даний момент служби, назва яких починається на літеру «S».
4. Вивести список файлів поточного каталогу, залишивши при цьому тільки його назву та розмір.
5. Вивести інформацію про те, скільки файлів, що мають відповідне розширення, є у поточному каталозі, та відсортуйте її.
6. Видалити у поточному каталозі всі файли, що мають розмір менше ніж 4Кб.

***Контрольні питання***

1. Яким чином використовуються конвеєри у PowerShell?
2. Для чого призначений командлет Where-Object?
3. Яким чином задається умова для перевірки у командлеті Where-Object?
4. Які оператори порівняння існують у PowerShell? Наведіть їх.
5. Які логічні оператори існують у PowerShell? Наведіть їх.
6. Який командлет призначений для сортування об’єктів? Як задати режим сортування об’єктів?