***Лабораторна робота №9***

***«Введення до оболонки Windows PowerShell. Основи роботи у командному рядку PowerShell»***

Нова оболонка **Windows PowerShell** була задумана розробниками **Microsoft** як більш потужне середовище для написання сценаріїв та роботи у командному рядку. Розробники **PowerShell** переслідували кілька цілей, головна з яких - створення середовища складання сценаріїв, яке найкращим чином підходило б для сучасних версій **ОС Windows** і було б більш функціональним, що розширюється і простим у використанні, ніж будь-який аналогічний продукт для будь-якої іншої ОС. В першу чергу це середовище повинно було підходити для вирішення завдань, що стоять перед системними адміністраторами, а також задовольняти вимогам розробників програмного забезпечення, надаючи їм засоби для швидкої реалізації інтерфейсів управління до створюваних додатків.

***Запуск оболонки. Виконання команд***

Для запуску оболонки слід натиснути на кнопку **Пуск (Start)**, відкрити меню **Всі програми (All Programs)**, вибрати елемент **Стандартні**, **Windows PowerShell** і **Windows PowerShell ISE**. Інший варіант запуску оболонки - пункт **Виконати ... (Run)** в меню **Пуск (Start)**, ввести ім'я файлу **powershell\_ise** і натиснути кнопку **OK**.

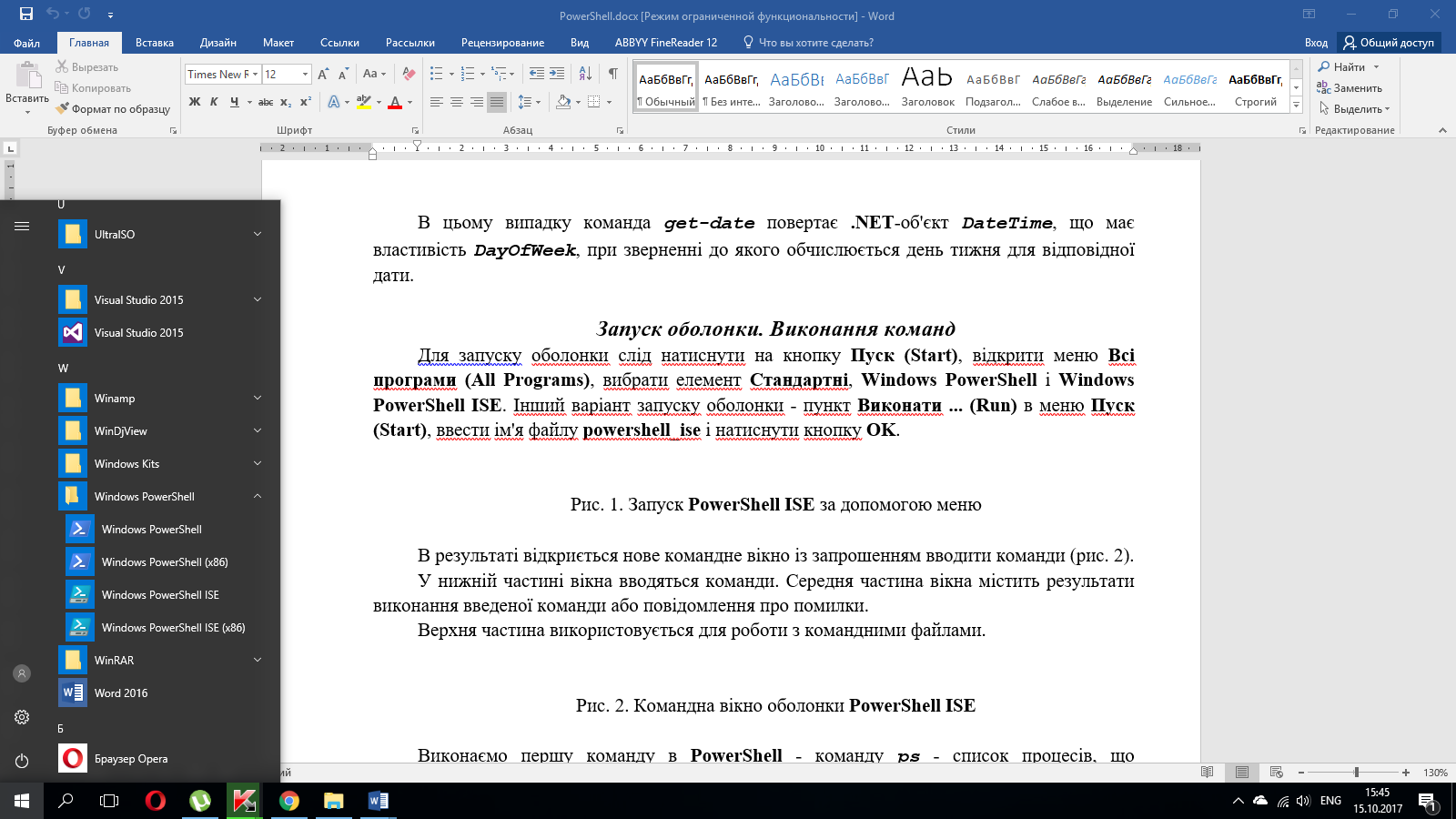


Рис. 1. Запуск **PowerShell ISE** за допомогою меню

В результаті відкриється нове командне вікно із запрошенням вводити команди (рис. 2).

У нижній частині вікна вводяться команди. Середня частина вікна містить результати виконання введеної команди або повідомлення про помилки.

Верхня частина використовується для роботи з командними файлами.

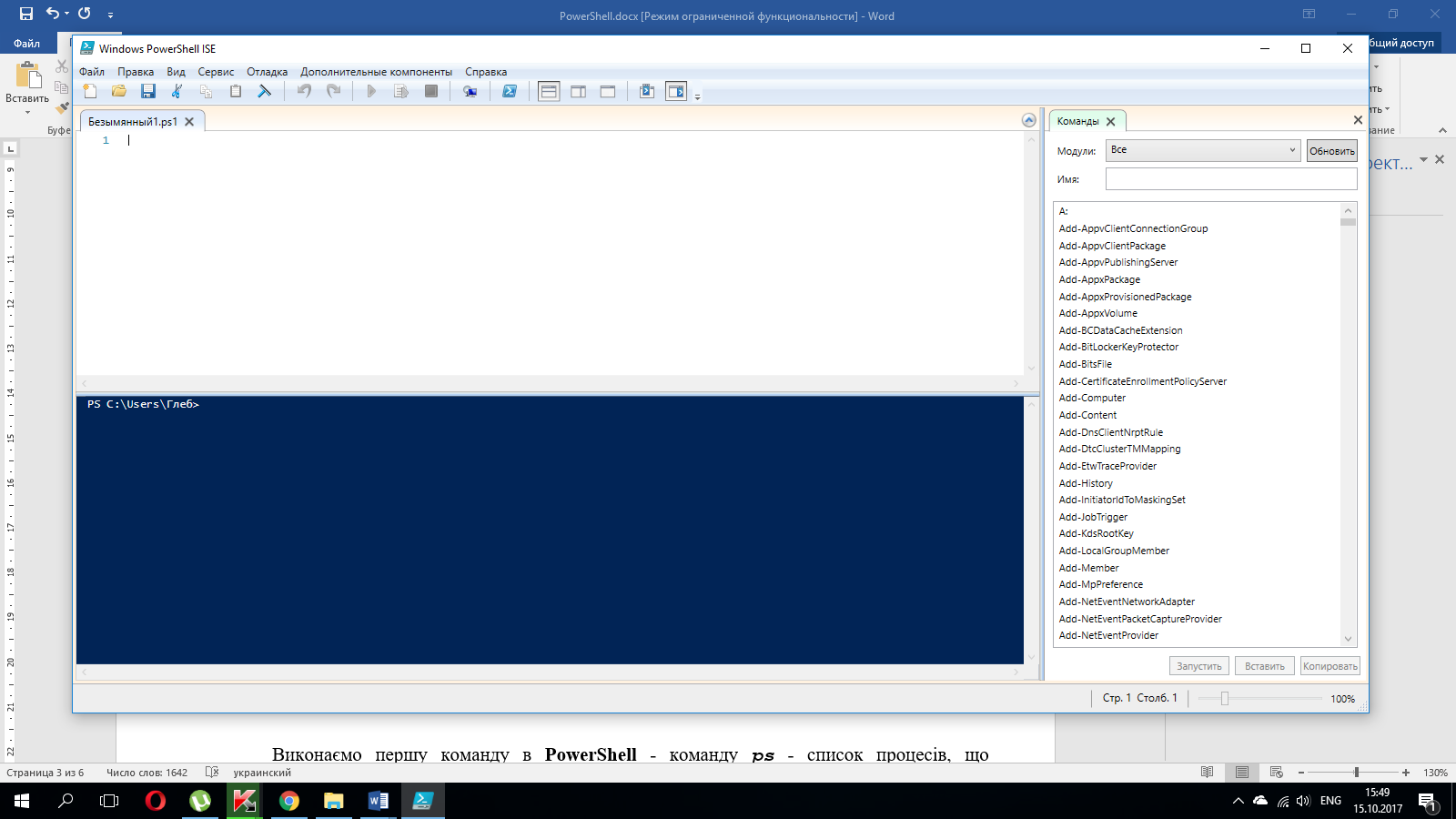


Рис. 2. Командна вікно оболонки **PowerShell ISE**

***Типи команд PowerShell***

В оболонці **PowerShell** підтримуються команди чотирьох типів: командлети, функції, сценарії і зовнішні виконувані файли.

Перший тип - так звані **командлети (cmdlet)**. Цей термін використовується поки тільки всередині **PowerShell**. Командлет - аналог внутрішньої команди інтерпретатора командного рядка - являє собою клас **.NET**, породжений від базового класу ***Cmdlet***; розробляються командлети за допомогою пакета **PowerShell Software Developers Kit (SDK)**. Єдиний базовий клас ***Cmdlet*** гарантує сумісний синтаксис всіх командлетів, а також автоматизує аналіз параметрів командного рядка і опис синтаксису командлетів для вбудованої довідки.

Даній тип команд компілюється в дінамічну бібліотеку (DLL) и підвантажуються до процесса **PowerShell** під час запуску оболонки (тобто самі по собі командлети не можуть буті запущені як додатки, але в них містяться виконувані об'єкти). Командлети - це аналог внутрішніх команд традіційніх оболонок.

Наступний тип команд - **функції**. Функція - це блок коду на мові **PowerShell**, що має назву і знаходиться в пам'яті до завершення поточного сеансу командної оболонки. Функції, як і командлети, підтримують іменовані параметри. Аналіз синтаксису функції проводитися один раз при її оголошенні.

**Сценарій** - це блок коду на мові **PowerShell**, що зберігається в зовнішньому файлі з розширенням ***ps1***. Аналіз синтаксису сценарію проводитися при кожному його запуску.

Останній тип команд - **зовнішні виконувані файли**, які виконуються звичайним чином операційною системою.

***Імена і синтаксис командлетів***

У **PowerShell** аналогом внутрішніх команд є командлети. Командлети можуть бути дуже простими або дуже складними, але кожен з них розробляється для вирішення однієї, вузької задачі. Робота з командлетами стає по-справжньому ефективною при використанні їх композиції (конвеєризації об'єктів між командлетами).

Команди **Windows PowerShell** слідують певним правилам іменування:

* Команди складаються з дієслова та іменника (завжди в однині), розділених тире. Дієслово задає певну дію, а іменник визначає об'єкт, над яким ця дія буде здійснено. Команди записуються на англійській мові. Приклад: ***Get-Help*** викликає інтерактивну довідку по синтаксису **Windows PowerShell**.
* Перед параметрами ставиться символ «***-***». Наприклад: ***Get-Help -Detailed***.
* У **Windows PowerShell** також включені псевдоніми багатьох відомих команд. Це спрощує знайомство і використання **Windows PowerShell**. Приклад: команди ***help*** (класичний стиль **Windows**) і ***man*** (класичний стиль **UNIX**) працюють так само, як і ***Get-Help***.

Наприклад, ***Get-Process*** (отримати інформацію про процес), ***Stop-Service*** (зупинити службу), ***Clear-Host*** (очистити екран консолі) і т.д. Щоб переглянути список командлетів, доступних в ході поточного сеансу, потрібно виконати командлет ***Get-Command***.

За замовчуванням командлет ***Get-Command*** інформує вас в трьох стовпцях: ***CommandType***, ***Name*** і ***Definition***. При цьому в стовпці ***Definition*** відображається синтаксис командлетів (три крапки (***...***) в стовпці синтаксису вказує на те, що дані обрізані).

**Зауваження.** Косі риски (***/*** і ***\***) разом з параметрами в оболонці **Windows PowerShell** не використовуються.

У загальному випадку синтаксис командлетів має наступну структуру:

***імя\_командлета -параметр1 -параметр2 аргумент1 аргумент2***

Тут ***параметр1*** - параметр (перемикач), що не має значення; ***параметр2*** - ім'я параметра, що має значення ***аргумент1***; ***аргумент2*** - параметр, який не має імені. Наприклад, командлет ***Get-Process*** має параметр ***Name***, який визначає ім'я процесу, інформацію про який потрібно вивести. Ім'я цього параметра вказувати необов'язково. Таким чином, для отримання відомостей про процес **Far** можна ввести або команду ***Get-Process -Name Far***, або команду ***Get-Process Far***.

***Псевдоніми команд***

Механізм псевдонімів, реалізований в оболонці **PowerShell**, дає можливість користувачам виконувати команди по їх альтернативних іменах (наприклад, замість команди ***Get-Childitem*** можна користуватися псевдонімом ***dir***). У **PowerShell** заздалегідь визначено багато псевдонімів, можна також додавати власні псевдоніми в систему.

Псевдоніми в **PowerShell** діляться на два типи. Перший тип призначений для сумісності імен з різними інтерфейсами. Псевдоніми цього типу дозволяють користувачам, які мають досвід роботи з іншими оболонками (**Cmd.exe** або **UNIX-оболонки**), використовувати знайомі їм імена команд для виконання аналогічних операцій в **PowerShell**, що спрощує освоєння нової оболонки, дозволяючи не витрачати зусиль на запам'ятовування нових команд **PowerShell**. Наприклад, користувач хоче очистити екран. Якщо у нього є досвід роботи з **Cmd.exe**, то він, природно, спробує виконати команду ***cls***. **PowerShell** при цьому виконає командлет ***Clear-Host***, для якого ***cls*** є псевдонімом і який виконує потрібні дії - очищення екрану. Для користувачів **Cmd.exe** в **PowerShell** визначені псевдоніми ***cd, cls, copy, del, dir, echo, erase, move, popd, pushd, ren, rmdir, sort, type***; для користувачів **UNIX** - псевдоніми ***cat, chdir, clear, diff, h, history, kill, lp, ls, mount, ps, pwd, r, rm, sleep, tee, write***.

Дізнатися, який саме командлет ховається за знайомим псевдонімом, можна за допомогою командлета ***Get-Alias***:

***PS C:\> Get-Alias cd***

***CommandType Name Definition***

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

***Alias cd Set-Location***

Псевдоніми другого типу (стандартні псевдоніми) в **PowerShell** призначені для швидкого введення команд. Такі псевдоніми утворюються з імен командлетів, яким вони відповідають. Наприклад, дієслово ***Get*** скорочується до ***g***, дієслово ***Set*** скорочується до ***s***, іменник ***Location*** скорочується до ***l*** і т.д. Таким чином, командлету ***Set-Location*** відповідає псевдонім ***sl***, а командлету ***Get-Location*** - псевдонім ***gl***.

Переглянути список всіх пседонімов, оголошених в системі, можна за допомогою командлета ***Get-Alias*** ​​без параметрів. Визначити власний псевдонім можна за допомогою командлета ***Set-Alias***.

***Довідкова система PowerShell***

У **PowerShell** передбачено кілька способів отримання довідкової інформації всередині оболонки.

Коротку довідку по одному командлету можна отримати за допомогою параметра ***?*** (Знак питання), зазначеного після імені цього командлета. наприклад:

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

Для отримання докладної інформації про командлети служить спеціальний командлет ***Get-Help***, який слід запускати з параметрами ***-Detailed*** або ***-Full***. Ключ ***-Full*** призводить до виведення всієї наявної довідкової інформаціі, а при використанні ключа ***-Detailed*** деяка технічна інформація опускається. В обох випадках будуть виведені докладні описи кожного з параметрів, підтримуваних даним командлетом.

Комана ***Get-Help*** дозволяє переглядати довідкову інформацію не тільки про різні командлети, а й про синтаксис мови **PowerShell**, про псевдоніми і т. д.

Наприклад, щоб прочитати докладну інформацію щодо використання масивів в **PowerShell**, потрібно виконати наступну команду: ***Get-Help about\_array***.

Командлет ***Get-Help*** виводить вміст розділу довідки на екран відразу цілком. Функції ***man*** і ***help*** дозволяють довідкову інформацію виводити поекранно (аналогічно команді ***more*** інтерпретатора **Cmd.exe**), наприклад: ***man about\_array***.

***Навігація в файловій системі***

В оболонці **cmd.exe** зміна поточного каталогу проводиться за допомогою команди ***cd***. У **PowerShell** команда ***cd*** має таке ж значення, при цьому вона є стандартним псевдонімом командлета ***Set-Location***. Наприклад, наступна команда робить поточним каталог ***C:\Windows:***

***PS C:\> cd C:\Windows:***

***PS C:\Windows>***

Як і в оболонці **cmd.exe** як шлях можна вказувати символи ***..*** (для 'переходу в батьківський каталог) і ***\*** (для переходу в кореневий каталог поточного диску). При цьому потрібно враховувати наступний нюанс. Людина, яка часто користувалась командою ***cd*** в оболонці ***cmd.exe***, швидше за все, буде машинально набирати команди типу ***cd..*** або ***cd\*** без додаткових проміжків. У **PowerShell** це викличе помилку.

Ця помилка пов'язана з тим. що в **PowerShell** параметри команди завжди повинні відділятися від імені самої команди проміжком. Тому останню команду потрібно виконувати наступним чином:

***PS C:\Windows> cd \***

***Отримання списку файлів і каталогів***

Нагадаємо, що в оболонці **cmd.exe** список файлів і каталогів формується за допомогою внутрішньої команди ***dir***. У **PowerShell** також можна використовувати команду ***dir***, яка є псевдонімом командлета ***Get-ChildItem***.

У шляху, який вказується для команди ***dir***, можна застосовувати групові символи підстановки. Наприклад, наступна команда виведе всі файли з розширенням ***log*** з каталогу ***C:\Windows***:

***PS С:\> dir С:\Windows\\*.log***

Параметр ***-Exclude*** дозволяє задати маску файлів, які не будуть оброблюватись командою ***dir***. Наприклад, наступна команда виведе всі файли з розширенням ***log*** з каталогу ***C:\Windows***, крім тих, чиє ім'я починається на букву ***d***:

***PS C:\> dir C:\Windows \\*.log -Exclude d\*.log***

Параметр ***-Name*** дозволяє виводити на екран тільки імена файлів (таким чином, цей параметр є аналогом ключа ***/b*** команди ***dir*** з **cmd.exe**), наприклад:

***PS C:\> dir C:\Windows \\*.log –Name***

Параметр ***-Recurse*** вмикає режим рекурсії, при якому командлет ***dir*** відображає не тільки вміст зазначеного каталогу, але і всіх його підкаталогів:

***PS С:\> dir 'Documents and Settings' –Recurse***

За замовчуванням командлет ***dir*** не «бачить" приховані файли. Якщо необхідно такі файли також включати в список, то потрібно вказати параметр ***-Force***:

***PS С:\> dir –Force***

***Створення файлів і каталогів***

Створити новий файл або каталог у **PowerShell** дозволяє командлет ***New-Item***. Шлях до створюваного елементу вказується у вигляді значення параметра ***-Path***, а в якості значення параметра ***-Type*** вказується "***directory***", якщо потрібно створити каталог, і "***file***", якщо потрібно створити файл. Наприклад, наступна команда створює на диску ***С:*** каталог з ім'ям ***test folder***:

***PS С:\> New-Item -Path C:\test\_folder -Type "directory"***

При створенні файлу в нього відразу можна записати рядок, вказавши його як значення параметра ***-Value***, наприклад:

***PS С:\> New-Item -Path C:\test\_file.txt -Type "file" -Value "Test"***

Якщо спробувати назвати створюваний файл ім'ям вже існуючого файлу:

***PS С:\> New-Item -Path C:\test\_file.txt -Type "file" -Value "Test"***

то виникне помилка.

Для перезапису існуючого файлу при створенні потрібно вказати параметр ***-Force***:

***PS С:\> New-Item -Path C:\test\_file.txt -Type "file" -Value "Test2" –Force***

***Читання і перегляд вмісту файлів***

В оболонці **cmd.exe** є команда ***type***, яка виводить вміст текстового файлу на екран. У **PowerShell** команда ***type*** є псевдонімом командлету ***Get-Content*** (інші псевдоніми цього ж командлету - ***cat*** і ***gc***), призначеного для порядкового зчитування вмісту текстового файлу з поверненням об'єкта для кожного рядка (при цьому рядки відображаються на екрані). Наприклад:

***PS С:\> Get-Content C:\Windows\win.ini***

Параметр ***-Encoding*** командлет ***Get-Content*** дозволяє явно вказувати кодування файлу для коректної обробки його вмісту. Допустимі значення даного параметра: ***Unicode***, ***Byte***, ***BigEndianUnicode***, ***UTF8***, ***UTF7***, ***Ascii***.

За замовчуванням командлет ***Get-Content*** зчитує всі рядки з файлу; їх кількість можна обмежити за допомогою параметра ***-TotalCount***. Наприклад, наступна команда зчитує перші п'ять рядків з файлу ***C:\Windows\win.ini***:

***PS С:\> Get-Content C:\Windows\win.ini -TotalCount 5***

***Запис файлів***

Записати дані у зовнішні файли можна за допомогою операторів перенаправлення (***>*** і ***>>***) і командлет ***Out-File***. При цьому командлет ***Out-File*** намагається форматувати записуються в файл об'єкти. Якщо потрібно просто записати в файл текстову інформацію (без додаткового форматування), то краще скористатися командлетом ***Set-Content***.

Дані для запису в файл можуть задаватися як значення параметра ***-Value***. Наприклад, наступна команда записує в файл ***C:\test.txt*** рядок ***"Рядок з PowerShell"***:

***PS C:\> Set-Content C:\test.txt -Value "Рядок з PowerShell"***

***Копіювання файлів і каталогів***

У **PowerShell** копіювання файлів і каталогів здійснюється командлетом ***Copy-Item***, які мають псевдонім ***copy***. Шлях до файлів, що копіюються, при цьому вказується в якості значення параметра ***-Path*** (даний параметр використовується за замовчуванням), а шлях до цільового каталогу, в який потрібно скопірювати файли, задається значенням параметра ***-Destination***. Наприклад, наступна команда скопіює файл ***styles.css*** з кореневого каталогу диска з: в каталог ***C:\test\_folder***:

***PS С:\> copy C:\Styles.css -Destination C:\test\_folder***

Для того щоб побачити результат виконання команди копіювання, потрібно вказати параметр ***-PassThru***:

***PS C:\> copy C:\styles.css -Destination C:\test\_folder –PassThru***

Якщо шлях до об'єктів, що копіюються, вказує на каталог, то за замовчуванням буде скопійований тільки цей каталог без свого вмісту (цим **PowerShell** відрізняється від більшості інших оболонок, в тому числі від ***cmd.exe***). Наприклад:

***PS С:\> copy C:\script -Destination C:\test\_folder -PassThru***

Параметр ***-Recurse*** дозволяє копіювати вміст вкладених каталогів, наприклад:

***PS С:\> copy C:\script -Destination C:\test\_folder -Recurse -PassThru***

Можна копіювати не всі файли з каталогу, а тільки відповідні певній масці. При цьому маску можна вказати всередині шляху для копіювання або в якості значення параметра ***-Include***. Наприклад, наступна команда копіює всі файли з розширенням ***psl*** з каталогу ***C:\script*** в папку ***C:\test\_folder***:

***PS C:\> copy C:\script\\*.psl -Destination C:\test\_folder –PassThru***

Однак якщо необхідно скопіювати і підкаталоги, то одним командлетом ***Copy-Item*** обійтися не вдасться. Попередньо необхідні файли потрібно отримати командлетом ***Get-ChildItem (dir***), а потім передати їх командлету ***Copy-Item*** по конвеєру. Наприклад, наступна команда копіює всі файли з розширенням ***psl*** з каталогу ***C:\script*** і всіх його підкаталогів в папку ***C:\test\_folder***:

***PS С:\> dir -Recurse -Include \*.psl c:\script\\* | copy -Destination C:\test\_folder -PassThru***

Команда ***copy*** оболонки **cmd.exe** дозволяла об'єднувати кілька файлів (конкатенація файлів). У **PowerShell** об'єднати файли можна за допомогою командлету ***Get-Content*** (псевдонім ***type***) і перенаправлення виведення в результуючий файл. Розглянемо приклад. Створимо файли ***l.txt*** і ***2.txt***:

***PS С:\> New-Item -Path C:\l.txt -Type "file" -Value "File 1"***

***PS C:\> New-Item -Path C:\2.txt -Type "file" -Value "File 2"***

Наступна команда об'єднує файли ***1.txt*** і ***2.txt*** у файл ***3.txt***:

***PS С:\> type l.txt, 2.txt > .\3.txt***

***Перейменування і переміщення файлів і каталогів***

Перейменувати файл або каталог можна за допомогою командлета ***Rename-Item*** (псевдонім ***ren***). Значення параметра ***-Path*** цього командлета задає шлях до елементів для перейменування, а значення параметра ***–NewName*** - нове ім'я. Імена цих параметрів можна опускати (в цьому випадку першим має зазначатися значення параметра ***-Path***). Наприклад, створимо файл ***c:\l.tmp*** і перейменуємо його у файл ***2.tmp***:

***PS С:\> New-Item -Path C:\l.tmp -Type "file"***

***PS C:\> ren l.tmp 2.tmp***

Для того щоб побачити результат дії командлету ***Rename-Item***, потрібно вказати параметр ***-PassThru***:

***PS С:\> ren 2.tmp 3.tmp –PassThru***

Командлет ***Rename-Item*** дозволяє лише перейменовувати файли або каталоги, а не переміщувати їх. Якщо потрібно перемістити файл або каталог в іншу папку, то слід скористатися командлетом ***Move-item*** (псевдонім ***move***). Значення параметра ***-Path*** даного командлету задає шлях до файлів або каталогів для переміщення (в цьому шляху допускається використання символів узагальнення), а значення параметра ***–Destination*** - шлях до каталогу. куди будуть переміщені ці файли або каталоги. Результат перемещення можна побачити на екрані, вказавши параметр ***-PassThru***. Наприклад, наступна команда перенесе у кореневий каталог диску ***С:*** каталог ***C:\test\_folder\folderl*** з усім його вмістом:

***PS C:\> Move-Item -Path C:\test\_folder\folderl C:\ -PassThru***

***Видалення файлів і каталогів***

Видаляти об'єкти файлової системи можна за допомогою командлета ***Remove-Item*** (псевдонім ***del***). Значення параметра ***-Path*** цього командлета задає шлях до файлів або каталогів, що видаляються (ім'я параметра в команді можна не вказувати). У шляху допускаються групові символи. Крім того, командлет ***Remove-Item*** має параметр ***-Include***, значення якого задає файли, на які діятиме команда, і параметр ***-Exclude***, що задає файли-виключення. які видалятися не будуть.

Наприклад, наступна команда видалить всі файли з розширенням ***psl*** в каталозі ***С:\test\_folder***:

***PS С:\> del С:\test\_folder\\*.psl***

Якщо спробувати видалити всі файли в каталозі, що має підкаталоги, то система видасть попередження:

***PS С:\> del С:\test\_folder\\****

***Завдання***

**Для виконання завдань використовувати командну оболонку Windows PowerShell!**

1. Переглянути вміст поточної директорії.
2. Переглянути вміст директорії N:\Scripts (або будь-якої іншої на Ваш розсуд), виводячи тільки імена файлів.
3. Створити в кореневому каталозі диску N каталог Work.
4. Скопіювати в каталог Work всі файли з розширенням .bat з ЛР №4.
5. Створити в каталозі Work підкаталог Sub і скопіювати в нього файли, що починаються з літери «t» з обраного на Ваш розсуд каталогу на диску N.
6. Створити в каталозі Sub копії bat-файлів, змінивши розширення на .txt, але залишивши ім'я.
7. Переглянути вміст каталогу Work.
8. Видалити файли з розширенням .bat з каталогу Work із запитом на підтвердження, а потім знову Переглянути вміст каталогу Work.
9. Створити підкаталог Temp в каталозі Sub.
10. У каталозі Temp створити файл mail.txt з Вашою електронною адресою і файл surname.txt з Вашим прізвищем.
11. Об'єднати файли mail.txt і surname.txt в all.txt і переглянути його вміст на екрані.
12. Знищити створені каталоги і їх вміст після закінчення роботи.

***Контрольні питання***

1. У чому полягає основна різниця оболонки Windows PowerShell від командного рядка?
2. Для чого призначена утиліта Windows PowerShell ISE?
3. На які типи поділяються команди PowerShell?
4. Що таке командлет? Яку структуру він має?
5. Псевдоніми яких стандартних команд Windows та Linux існують у PowerShell? Які командлети насправді використовуються при виклику даних команд?
6. Як вивести довідку про певну команду у PowerShell?