Дисципліна «Операційні системи»

Лабораторна робота №8

(експрес лабораторна робота)

Тема: «Основи об'єктно-орієнтованого керування об'єктами ОС в командному інтерпретаторі *MS PowerShell*»

Викладач: Олександр А. Блажко,

доцент кафедри ІС Одеської політехніки, blazhko@ieee.org

(для всіх груп під час додаткового *Online*-заняття)

Мета роботи: отримання навичок в об'єктно-орієнтованому керуванні об'єктами Unix-подібної ОС з використанням командного інтерпретатора *MS PowerShell*

1 Теоретичні відомості

1.1 Особливості *MS PowerShell*

1.1.1 Проблеми керування ОС

ОС Windows впродовж багатьох років орієнтується на графічний інтерфейс користувача (*Graphical User Interface, GUI*), більш орієнтований на персональне використання комп'ютеру

Активний розвиток розподілених (кластерних) систем та хмарних середовищ виявив основиний недолік *GUI* :

– значне зростання трудомісткості *GUI*-операцій для великої кількості *Windows*-серверів

– відсутність зручних засобів автоматизації GUI-операцій

Microsoft PowerShell – оболонка програмного рядку:

– автоматизація виконання задач та керування процесами налаштування програм;

– можливість роботи в різних ОС (*Windows, Linux, Mac OS*);

– *MIT*-ліцензія для *ОС Linux, Mac OS,* для ОС *Windows – Proprietary*.

1.1.2 *MS PowerShell* та об'єктно-орієнтоване керування

Відомо, що об'єктно-орієнтоване програмування надає:

– абстрагування – структурування опису сутностей предметної галузі як:

– класи як підмножина сутностей предметної галузі, які вливають на досягнення мети інформаційної системи;

– атрибути класу як підмножина властивостей сутності, які вливають на досягнення мети інформаційної системи;

– успадкування (наслідування) – успадкування атрибутів та алгоритмів їх обробки одного класу іншим класом, який ієрархічно споріднений з цим класом;

– інкапсуляція – об'єднання атрибутів структур даних опису сутностей реального світу із описом алгоритмів їх обробки у вигляді методів (процедур/функцій);

– поліморфізм – уніфікація назв методів, коли для різних класів використовуються семантично схожі назви методів.

Основний недолік використання команд оболонки *Unix*-подібних ОС – складність запам'ятовування різних команд та різних опцій цих команд.

Одним із засобів зменшення такої складності є впровадження об'єктно-орієнтованого керування через введення спеціального формату команд керування об'єктами ОС – так званих *Cmdlets* («командлети») у вигляді: дієслово-іменник, де дієслово визначає поліморфну операцію, іменник визначає об'єкт ОС.

Приклади операцій: *Get, Set, Select, Move, Remove, Rename, Write, Stop*

Приклади об'єктів: *Item, ChildItem, Output, Location, String, Process*

Приклади *Cmdlets: Get-Item* – отримати дані про файл, *Get-ChildItem* – отримати зміст каталогу, *Get-Location* – отримати поточну локацію,

Get-Process – отримати процеси, *Set-Location* – встановити локацію

1.2 Встановення *MS PowerShell*

В 2016 році *Microsoft* відкрило частину програмного коду *PowerShell*, що дозволило встановлювати *PowerShell* на ОС *Linux* різних дистрибуцій та на ОС *MacOS*.

Програмні пакети *PowerShell* розміщено у *Github*-репозиторії за посиланням:

https://github.com/PowerShell/PowerShell

Для встановлення поточної версії використовується клонування командою:

git clone https://github.com/PowerShell/PowerShell.git

Для встановлення *PowerShell*, наприклад, в ОС *Ubuntu* необхідно виконати дії, наведені за посиланням https://learn.microsoft.com/en-us/powershell/scripting/install/install-ubuntu?view=powershell-7.3

Після встановлення *powershell* для його запуску виконайте команду:

pwsh

1.3 Основи роботи в командному інтерпретаторі *MS PowerShell*

В таблиці 1 наведено приклади відповідності між командами *Bash*-оболонки та *PowerShell*-оболонки.

Таблиця 1 – Приклади відповідності між командами *PowerShell* та *Bash*

№	Дія	*Bash*-оболонка	*PowerShell*-оболонка
1.	Очистити екран псевдотерміналу	*clear*	*Clear-Host*
2.	Отримати ім’я поточного користувача	*whoami*	*[Environment]::UserName*
3.	Отримати шлях до поточного каталогу перебування	*pwd*	*Get-Location*
4.	Отримати поточну дату	*date*	*Get-Date*
5.	Отримати атрибути класу, пов`язаного з поточною датою	*date --help*	*Get-Date | Get-Member -MemberType Property, ScriptProperty*
6.	Отримати поточну дату, яка містить день, місяць та рік.	*date +'%d %m %Y'*	*Get-Date | Select-Object Day,Month,Year*
7.	Створити псевдонім для команди отримання поточної дати	*alias my_date="date"*	*New-Alias -Name my_date Get-Date*
8.	Змінити псевдонім на складний псевдонім для команди отримання поточної дати, яка містить день, місяць та рік (попередньо створити функцію з отримання дати)	*alias my_date="date +'%d %m %Y'"*	*Function f_my_date {Get-Date | Select-Object Day,Month,Year}* *Set-Alias -Name my_date -Value f_my_date*
9.	Отримати зміст поточного каталогу в псевдотабличному вигляді	*ls -l*	*Get-ChildItem*

Таблиця 1 – продовження

№	Дія	*Bash*-оболонка	*PowerShell*-оболонка
10.	Отримати атрибути класу, пов`язаного зі змістом поточного каталогу	*ls --help*	*Get-ChildItem | Get-Member -MemberType Property, ScriptProperty*
11.	Отримати зміст поточного каталогу у псевдотабличному вигляді лише із назвами файлів, їх власниками, групами власників та правами доступу до файлів	*ls -l | awk '{print $9,$3,$4,$1}'*	*Get-ChildItem | Select-Object Name,User,Group,UnixMode*
12.	Повторити попередню дію, отримуючи рядки таблиці у порядку убування назв файлів	*ls -l | awk '{print $9,$3,$4,$1}' | sort -r*	*Get-ChildItem | Select-Object Name, User,Group,UnixMode | Sort-Object -Property Name -Descending*
13.	Створити каталог	*mkdir my_dir*	*New-Item -ItemType directory -Path my_dir*
14.	Перейти до каталогу	*cd my_dir*	*Set-Location my_dir*
15.	Створити файл	*touch my_file*	*New-Item -ItemType file -Path my_file*
16.	Передати рядок на *stdout*-потік	*echo 'Hello'*	*Write-Output 'Hello'*
17.	Передати рядок до файлу	*echo 'Hello' > my_file*	*Write-Output 'Hello' > my_file*
18.	Отримати зміст файлу	*cat my_file*	*Get-Content my_file*
19.	Отримати рядок файлу за шаблоном	*grep -E '^H' my_file*	*Select-String -Pattern '^H' my_file*
20.	Створити символічний зв’язок	*ln -s my_file my_file_link*	*New-Item -ItemType symboliclink -Path my_file_link -Target my_file*

Таблиця 1 – продовження

№	Дія	*Bash*-оболонка	*PowerShell*-оболонка
21.	Скопіювати файл */etc/group* до поточного каталогу, змінивши його назву на *group_copy*	*cp /etc/group ./group_copy*	*Copy-Item -Path /etc/group -Destination ./group_copy*
22.	Отримати перші 5-ть рядків файлу		
23.	Видалити файл *group_copy*	*rm ./group_copy*	*Remove-Item ./group_copy*
24.	Видалити каталог *my_dir* зі всіма файлами каталогу (рекурсивне видалення)	*rm ./my_dir/ -r*	*Remove-Item ./my_dir/ -Recurse*
25.	Отримати всі процеси ОС	*ps -A*	*Get-Process*
26.	Отримати 5-ть перших процесів зі списку всіх процесів ОС	*ps -A | head -6*	*Get-Process | Select -First 5*
27.	Отримати всі процеси ОС, які мають назву *bash*	*ps -A | grep "bash"*	*Get-Process | Where-Object {$_.ProcessName -eq "bash"}*
28.	Отримати всі процеси ОС, для яких наприкінці назви є символ *d*	*ps -A | grep -E "d$"*	*Get-Process | Where-Object {$_.ProcessName -match "d$"}*

2 Завдання

Увага! Ця лабораторна робота виконується лише під час проведення лабораторного *Online*-заняття. Якщо ви не змогли за техінчних причин її виконати на лабораторному занятті, тоді у вас буде можливість це зробити на спеціальній *Online*-консультації, про яку буде повідомлено викладачем.

2.1 Підготовка до виконання завдань

2.1.1 Налаштування *GitHub*-репозиторію

Результати роботи будуть оформлені лише у файлі *README.md*, тому для підготовки репозиторію пропонується виконати наступні дії на веб-сервісі *GitHub*:

1) створити нову *Git*-гілку з назвою «*Laboratory-work-8*»;
2) перейти до роботи зі створеною гілкою;
3) створити каталог з назвою «*Laboratory-work-8*» та файлом *README.md*;
4) додати до файлу *README.md* рядок тексту із темою лабораторної роботи *«Основи об'єктно-орієнтованого керування об'єктами ОС в командному інтерпретаторі MS PowerShell»* як заголовок 2-го рівня *Markdown*-форматування.

2.1.2 Початок роботи із виконання завдань

Використовуючи оболонку командного рядку вашого локального комп'ютера, наприклад, *Git-Bash*, виконати наступні дії:

1) встановити з'єднання з віддаленим *Linux*-сервером з IP-адресою = 46.175.148.116, логіном, наданим вам викладачем, та паролем, зміненим вами у попередній роботі.
2) перейти до каталогу вашого *Git*-репозиторію;
3) запустити оболонку *PowerShell*.

2.2 Робота в командному інтерпретаторі *MS PowerShell*

Примітка: під час виконнаня завдань достатньо створити 5-ть знімки екрану, які будуь містити результати 6-ти рішень з підписами «Рис. 1 – рішення завдань 1-4», «Рис. 2 – рішення завдань 5-8», «Рис. 3 – рішення завдань 9-12», «Рис. 4 – рішення завдань 13-16», «Рис. 5 – рішення завдань 17-20», «Рис. 6 – рішення завдань 21-24»

Виконати наступні завдання, використовуючи *PowerShell cmdlets*:

1. Очистити екран псевдотерміналу.
2. Отримати ім'я поточного користувача.
3. Отримати шлях до поточного каталогу.
4. Отримати поточну дату.
5. Отримати поточну дату, яка містить лише мілісекунди та наносекунди.

6. Створити псевдонім «ваше прізвище*_my_time*», наприклад, *blazhko_my_time*, для команди отримання поточної дати.

7. Змінити створений раніше псевдонім *blazhko_my_time* на псевдонім, який буде отримувати з поточної дати лише мілісекунди та наносекунди. (попередньо створити функцію з назвою «ваше прізвище_», яка реалізує складний псевдонім).

8. Видалити створений раніше псевдонім.

9. Отримати зміст поточного каталогу у псевдотабличному вигляді лише із назвами файлів, їх власниками, правами доступу до файлів та дати створення файлів.

10. Повторити попередню дію, отримуючи рядки таблиці у порядку убування назви.

11. Створити каталог «ваше прізвище*_my_dir*», наприклад, *blazhko_my_dir*.

12. Перейти до створеного каталогу.

13. Створити файл «ваше прізвище*_my_file*», наприклад, *blazhko_my_file*.

14. Передати рядок «ваше прізвище», наприклад, «*Blazhko*» до створеного раніше файлу.

15. Переглянути зміст створеного раніше файлу.

16. На раніше створений файл створити символічний зв’язок «ваше прізвище*_my_link*», наприклад, *blazhko_my_link.*

17. Скопіювати файл */etc/group* до поточного каталогу, змінивши його назву на «ваше прізвище*_group*», наприклад, *blazhko_group*.

18. Отримати з попередньо скопійованого файлу лише останні *N* рядків, де *N* – номер вашого варіанту.

19. Отримати з попередньо скопійованого файлу рядки з інформацію про групи користувачів, для яких перша літера імені співпадає з першою літерою вашого прізвища латиницею.

20. Видалити створений раніше файл-символічний зв’язок.

21. Перейти до каталогу, який знаходиться вище за рівнем поточного каталогу.

22. Видалити створений раніше каталог (рекурсивне видалення).

23. Отримати перші *N* процесів зі списку всіх процесів ОС, де *N* – номер вашого варіанту.

24. Отримати всі процеси ОС, які мають у назві будь-яку цифру.

2.2 Підготовка процесу *Code Review* для надання рішень завдань лабораторної роботи на перевірку викладачем

2.2.1 На веб-сервісі *GitHub* зафіксувати зміни у файлі *README.md*

2.2.2 Виконати запит *Pull Request* з назвою «*Laboratory-work-8*».