Operační systémy Windows 4

Windows PowerShell

Klíčové pojmy:

- Windows Powershell
- Argumenty příkazů
- Options
- Historie v powershell
- Místní proměnné
- Proměnné prostředí
- Proměnná Path
- Interní příkazy
- Externí příkazy
- Aliasy
- Nejpoužívanější cmdlety

WindowsPowershell

PowerShell je pokročilé příkazové řádkové rozhraní a skriptovací prostředí vyvinuté společností Microsoft. Navržené pro správu konfigurací a automatizaci administrativních úloh, PowerShell poskytuje robustní nástroje pro správu systémů, aplikací a sítí v prostředí Windows. Na rozdíl od tradičního příkazového řádku (cmd.exe), PowerShell využívá objektově orientovaný přístup založený na .NET Framework, což umožňuje komplexnější a flexibilnější skriptování.

Příkazy

Příkazy (cmdlets) jsou základními stavebními kameny PowerShellu. Cmdlety jsou malé, specializované příkazy, které provádějí specifické úkoly. Každý cmdlet má jasnou strukturu a následuje konvenci názvů **Verb-Noun** (sloveso-podstatné jméno), což usnadňuje jejich zapamatování a použití. Příkazy mohou být doplněny **argumenty** a **volbami (options)**, které specifikují, jak má být příkaz proveden.

Příklady cmdletů:

- **`Get-Process`**: Získá seznam běžících procesů na systému.
- **`Set-Date`**: Nastaví aktuální datum a čas.
- **`New-Item`**: Vytvoří nový soubor nebo složku.
- **`Remove-Item`**: Odstraní soubor nebo složku.

Jak zadat příkaz v PowerShell

Abyste mohli zadat příkaz v PowerShell, postupujte následovně::

- Otevření PowerShellu:
    - Klikněte na Start a napište PowerShell.
    - Vyberte Windows PowerShell nebo PowerShell (v závislosti na verzi Windows).

Pro administrátorská oprávnění klikněte pravým tlačítkem a vyberte Spustit jako správce.

- Zadání příkazu:

- V otevřeném okně PowerShellu napište cmdlet a stiskněte Enter.
  Například:
  `Get-Process`

Argumenty příkazů v PowerShell

Argumenty jsou hodnoty, které se přidávají k cmdletům, aby specifikovaly, na čem nebo jak mají být příkazy provedeny. Můžete je zadávat pozicemi nebo názvy.
Příklad:
Příkaz **`Get-Process`** může přijímat název procesu jako argument:

```
Get-Process -Name "notepad"
```

Nebo můžete použít pozici:

```
Get-Process notepad
```

Options (volby) příkazů v PowerShell

Volby (parameters) jsou speciální argumenty, které upravují chování cmdletů. Jsou označeny pomlčkou (-) a následuje jejich název. Některé volby mohou přijímat hodnoty.
Příklad:
Příkaz **`Get-Process`** s volbou **`-Name`** a hodnotou:

```
Get-Process -Name "chrome"
```

Historie uvnitř PowerShell

PowerShell uchovává historii příkazů, které jste zadali během aktuální relace. Tuto historii můžete prohlížet a znovu používat.

Prohlížení historie:

```
Get-History
```

Opakování předchozího příkazu:

```
Invoke-History -Id 5
```

(Nahradíte `5` číslem ID příkazu z historie)

Proměnné v PowerShell

Proměnné v PowerShell jsou kontejnery pro ukládání dat, která mohou být později použity v příkazech nebo skriptech. Proměnné začínají znakem **dolar ($)**.

Příklad:

```
$mojePromenna = "Hello, World!"
echo $mojePromenna
```

Místní proměnné

Místní proměnné jsou platné pouze v aktuální relaci PowerShellu nebo v rámci aktuálního skriptu. Jakmile zavřete relaci, proměnné zaniknou.

Příklad:

```
$localVar = "Lokální hodnota"
echo $localVar
```

Proměnné prostředí

Proměnné prostředí jsou globální proměnné, které jsou dostupné ve všech relacích PowerShellu a aplikacích běžících na systému. Umožňují konfiguraci systému a aplikací prostřednictvím sdílených nastavení.

Příklad zobrazení proměnné prostředí:

```
echo $env:USERNAME
```

Nastavení proměnné prostředí:

```
$env:MOJE_ENV_PROMENNA = "Hodnota"
```

Proměnná PATH

Proměnná PATH je klíčová proměnná prostředí, která určuje, kde PowerShell (a další příkazové interpretry) hledají spustitelné soubory. Přidání cesty do proměnné PATH umožňuje spouštět programy bez nutnosti zadávání jejich úplné cesty.

Zobrazení proměnné PATH:

```
echo $env:PATH
```

Přidání nové cesty do PATH:

```
$env:PATH += ";C:\NovaCesta"
```

Trvalé přidání do PATH (pro aktuálního uživatele):

```
[Environment]::SetEnvironmentVariable("PATH",
"$env:PATH;C:\NovaCesta", "User")
```

Trvalé přidání do PATH (pro všechny uživatele):

```
[Environment]::SetEnvironmentVariable("PATH",
"$env:PATH;C:\NovaCesta", "Machine")
```

Alias příkazy v PowerShell

Aliasy jsou zkrácené názvy pro cmdlety nebo jiné příkazy, které umožňují rychlejší a snadnější jejich zadávání. Aliasy jsou užitečné pro zkrácení dlouhých příkazů nebo pro přizpůsobení PowerShellu podle vašich potřeb.

Vytvoření aliasu:

```
Set-Alias ll Get-ChildItem
```

Použití aliasu:

```
ll
```

(Tento příkaz zobrazí obsah aktuální složky, stejně jako `Get-ChildItem`)

Interní a externí příkazy v PowerShell

PowerShell kombinuje **interní cmdlety** (integrované příkazy) a **externí příkazy** (spustitelné soubory nebo skripty). Stejně jako tradiční příkazový řádek (cmd.exe), PowerShell umožňuje použití obou typů příkazů.

Interní cmdlety:
Jsou přímo integrovány do PowerShellu a běží bez nutnosti externích programů.
Příklad: `Get-Process`, `Set-Location`, `New-Item`

Externí příkazy:
Jsou samostatné programy nebo skripty, které jsou spouštěny prostřednictvím PowerShellu. Musí být dostupné v cestě systému (PATH) nebo specifikována jejich úplná cesta.
Příklad: `ping.exe`, `ipconfig.exe`, `notepad.exe`

Doporučení pro učení

Zkoumejte cmdlety: Používejte příkaz `Get-Command` k objevování nových cmdletů a jejich funkcí.
```
Get-Command *Process*
```

Čtěte nápovědu: Každý cmdlet má podrobnou nápovědu, kterou můžete zobrazit pomocí `Get-Help`.
```
Get-Help Get-Process -Detailed
```

Experimentujte s proměnnými: Vytvářejte a manipulujte s proměnnými pro lepší pochopení jejich použití a dosahu.
```
$testVar = "Hello PowerShell"
Write-Output $testVar
```

Vytvářejte skripty: Začněte psát jednoduché PowerShell skripty pro automatizaci běžných úkolů a postupně přecházejte k složitějším skriptům využívajícím podmínky, smyčky a funkce.
```
# Simple script to list all .txt files in a directory
$path = "C:\Documents"
Get-ChildItem -Path $path -Filter *.txt
```

Používejte aliasy a funkce: Vytvářejte si vlastní aliasy a funkce pro zjednodušení a zrychlení práce s PowerShellem.
```
Set-Alias ll Get-ChildItem
```

Správa proměnné PATH: Naučte se bezpečně a efektivně upravovat proměnnou PATH, abyste mohli spouštět programy bez nutnosti zadávat jejich úplnou cestu.
```
# Přidání nové cesty do PATH
[Environment]::SetEnvironmentVariable("PATH",
"$env:PATH;C:\NovaCesta", "User")
```

Nejpoužívanější cmdlety

Cmdlety jsou základními stavebními kameny PowerShellu, které umožňují provádět různé operace pomocí jednoduchých příkazů. Níže je přehled některých z nejběžnějších cmdletů, které se často používají při správě systémů, práci se soubory, procesy a dalšími úkoly.

1. Get-Help

- **Popis:** Zobrazí nápovědu a dokumentaci k jiným cmdletům.
- **Použití:**
  `Get-Help Get-Process`
- **Příklad**: Získání podrobné nápovědy pro cmdlet `Get-Process`.

---

2. Get-Command

- **Popis:** Zobrazí seznam všech dostupných cmdletů, funkcí, skriptů a aliasů.
- **Použití:**
  `Get-Command`
- **Příklad:** Vyhledání všech cmdletů, které obsahují slovo "process":
  `Get-Command *Process*`

---

3. Get-Process

- **Popis:** Získá seznam běžících procesů na systému.
- **Použití:**

  ```
  Get-Process
  ```

- **Příklad:** Získání informací o procesu "notepad":

  ```
  Get-Process -Name notepad
  ```

---

4. Stop-Process

- **Popis:** Ukončí běžící procesy.
- **Použití:**

  ```
  Stop-Process -Name notepad
  ```

- **Příklad:** Ukončení procesu "notepad" pomocí ID procesu:

  ```
  Stop-Process -Id 1234
  ```

---

5. Get-Service

- **Popis:** Zobrazí seznam služeb na systému a jejich stav.
- **Použití:**

  ```
  Get-Service
  ```

- **Příklad:** Získání informací o službě "wuauserv" (Windows Update):

  ```
  Get-Service -Name wuauserv
  ```

---

6. Start-Service

- **Popis:** Spustí zastavenou službu.
- **Použití:**

  ```
  Start-Service -Name wuauserv
  ```

- **Příklad:** Spuštění služby "wuauserv".

---

7. Stop-Service

- **Popis:** Zastaví běžící službu.
- **Použití:**

  ```
  powershell
  Zkopírovat kód
  Stop-Service -Name wuauserv
  ```

- **Příklad:** Zastavení služby "wuauserv".

---

8. Set-ExecutionPolicy

- **Popis:** Nastaví politiku spuštění skriptů v PowerShellu.
- **Použití:**

  ```
  Set-ExecutionPolicy RemoteSigned
  ```

- **Příklad:** Umožní spuštění lokálních skriptů a skriptů podepsaných důvěryhodným vydavatelem.

---

9. Get-ChildItem

- **Popis:** Zobrazí seznam souborů a složek v aktuálním nebo specifikovaném adresáři.
- **Použití:**

  ```
  Get-ChildItem
  ```

- **Příklad:** Zobrazení všech `.txt` souborů ve složce `C:\Documents`:

```
Get-ChildItem -Path C:\Documents -Filter *.txt
```

---

10. Copy-Item

Popis: Cmdlet **`Copy-Item`** slouží k **kopírování souborů nebo složek** z jednoho umístění na jiné. Umožňuje také rekurzivní kopírování složek a jejich obsahu.

Syntaxe:

```
Copy-Item -Path <source> -Destination <destination> [-Recurse] [-
Force] [-Filter <filter>]
```

Příklad:

```
# Kopírování jednoho souboru
Copy-Item -Path "C:\Source\file.txt" -Destination
"D:\Backup\file.txt"

# Rekurzivní kopírování celé složky
Copy-Item -Path "C:\SourceFolder" -Destination "D:\BackupFolder" -
Recurse

# Kopírování všech .txt souborů s přepsáním existujících souborů
Copy-Item -Path "C:\Source\*.txt" -Destination "D:\Backup\" -Force
```

Vysvětlení:

První příklad kopíruje jednotlivý soubor `file.txt` z `C:\Source` do `D:\Backup`.
Druhý příklad rekurzivně kopíruje celou složku `SourceFolder` do `BackupFolder`, včetně všech podadresářů a souborů.
Třetí příklad kopíruje všechny soubory s příponou `.txt` z `C:\Source` do `D:\Backup` a přepíše existující soubory v cílovém umístění.

---

11. Move-Item

Popis: Cmdlet **`Move-Item`** slouží k **přesunutí souborů nebo složek** z jednoho umístění na jiné. Umožňuje také přejmenování souborů nebo složek během přesunu.

Syntaxe:

```
Move-Item -Path <source> -Destination <destination> [-Force] [-PassThru]
```

Příklad:

```
# Přesunutí jednoho souboru
Move-Item -Path "C:\Source\file.txt" -Destination "D:\Destination\file.txt"

# Přesunutí celé složky
Move-Item -Path "C:\OldFolder" -Destination "D:\NewFolder"

# Přejmenování souboru během přesunu
Move-Item -Path "C:\Source\oldname.txt" -Destination
"C:\Source\newname.txt"
```

Vysvětlení:

První příklad přesouvá soubor `file.txt` z `C:\Source` do `D:\Destination`.
Druhý příklad přesouvá celou složku `OldFolder` do `D:\NewFolder`.
Třetí příklad přejmenovává soubor `oldname.txt` na `newname.txt` ve stejném adresáři `C:\Source`.

---

12. Remove-Item

Popis: Cmdlet **`Remove-Item`** slouží k **odstranění souborů nebo složek**. Může být použit pro odstranění jednoho nebo více objektů a podporuje rekurzivní odstranění obsahu složek.

Syntaxe:

```
Remove-Item -Path <path> [-Recurse] [-Force] [-WhatIf] [-Confirm]
```

Příklad:

```
# Odstranění jednoho souboru
Remove-Item -Path "C:\Source\file.txt"

# Rekurzivní odstranění celé složky a jejího obsahu
Remove-Item -Path "C:\SourceFolder" -Recurse

# Odstranění všech .log souborů v adresáři s potvrzením
Remove-Item -Path "C:\Logs\*.log" -Confirm
```

Vysvětlení:

První příklad odstraňuje soubor `file.txt` z `C:\Source`.
Druhý příklad rekurzivně odstraňuje složku `SourceFolder` a veškerý její obsah.
Třetí příklad odstraňuje všechny soubory s příponou `.log` v adresáři `C:\Logs` a vyžaduje potvrzení před každým odstraněním.

---

13. New-Item

Popis: Cmdlet **`New-Item`** slouží k **vytvoření nových souborů nebo složek**. Může být použit pro vytvoření různých typů objektů v systému.

Syntaxe:

```
New-Item -Path <path> -ItemType <type> [-Value <value>] [-Force]
```

Příklad:

```
# Vytvoření nové složky
New-Item -Path "C:\NewFolder" -ItemType Directory

# Vytvoření nového textového souboru
New-Item -Path "C:\NewFolder\example.txt" -ItemType File -Value "Initial content."

# Vytvoření nového souboru s přepsáním existujícího
New-Item -Path "C:\NewFolder\example.txt" -ItemType File -Force
```

Vysvětlení:

První příklad vytváří novou složku `NewFolder` v kořenovém adresáři `C:\`.
Druhý příklad vytváří nový soubor `example.txt` ve složce `NewFolder` a zapisuje do něj počáteční obsah.
Třetí příklad vytváří nový soubor `example.txt` a přepíše existující soubor s tímto názvem, pokud již existuje.

---

14. Set-Location

Popis: Cmdlet **`Set-Location`** slouží k **změně aktuálního pracovního adresáře** v PowerShellu. Je ekvivalentní příkazu `cd` v tradičním příkazovém řádku.

Syntaxe:

```
Set-Location -Path <path>
```
Příklad:
```
# Přechod do složky C:\Projects
Set-Location -Path "C:\Projects"

# Přechod do složky s využitím aliasu
sl "D:\Documents"
```
Vysvětlení:
První příklad změní aktuální pracovní adresář na `C:\Projects`.
Druhý příklad ukazuje použití aliasu `sl`, který může být definován jako zkratka pro `Set-Location` (např. `Set-Alias sl Set-Location`).

---

15. Get-Content

Popis: Cmdlet **`Get-Content`** slouží k **získání obsahu souboru**. Umožňuje čtení textového obsahu souborů a jeho zobrazení v PowerShellu.
Syntaxe:
```
Get-Content -Path <path> [-Tail <number>] [-TotalCount <number>] [-Wait]
```
Příklad:
```
# Zobrazení obsahu souboru
Get-Content -Path "C:\Logs\log.txt"

# Zobrazení posledních 10 řádků souboru
Get-Content -Path "C:\Logs\log.txt" -Tail 10

# Sledujte soubor pro nové přidávání řádků (podobné tail -f v Unixu)
Get-Content -Path "C:\Logs\log.txt" -Wait
```
Vysvětlení:
První příklad zobrazuje celý obsah souboru `log.txt`.
Druhý příklad zobrazuje pouze posledních 10 řádků souboru.
Třetí příklad sleduje soubor a zobrazuje nové řádky, které se přidávají v reálném čase.

---

16. Set-Content

Popis: Cmdlet **`Set-Content`** slouží k **zápisu obsahu do souboru**. Přepisuje existující obsah souboru nebo vytváří nový soubor, pokud ještě neexistuje.
Syntaxe:
```
Set-Content -Path <path> -Value <value> [-Force]
```
Příklad:
```
# Přepsání obsahu souboru
Set-Content -Path "C:\Notes\note.txt" -Value "Toto je nový obsah souboru."

# Vytvoření nového souboru s obsahem
Set-Content -Path "C:\Notes\newfile.txt" -Value "Initial content."

# Přepsání souboru bez varování
Set-Content -Path "C:\Notes\existingfile.txt" -Value "Updated content." -
Force
```
Vysvětlení:
První příklad přepíše obsah existujícího souboru `note.txt` novým textem.
Druhý příklad vytvoří nový soubor `newfile.txt` s počátečním obsahem.
Třetí příklad přepíše existující soubor `existingfile.txt` bez varování díky přepínači `-Force`.

---

17. Add-Content

Popis: Cmdlet **`Add-Content`** slouží k **přidávání obsahu na konec souboru**. Umožňuje doplnění textu do existujícího souboru bez přepisování jeho stávajícího obsahu.

Syntaxe:

```
Add-Content -Path <path> -Value <value> [-Force]
```

Příklad:

```
# Přidání nového řádku do souboru
Add-Content -Path "C:\Logs\log.txt" -Value "Nová logovací událost."

# Přidání více řádků najednou
Add-Content -Path "C:\Notes\note.txt" -Value @("První řádek", "Druhý řádek")

# Přidání obsahu do chráněného souboru
Add-Content -Path "C:\Protected\file.txt" -Value "Dodatečný text." -Force
```

Vysvětlení:

První příklad přidává nový řádek `Nová logovací událost.` na konec souboru `log.txt`.
Druhý příklad přidává dva nové řádky do souboru `note.txt`.
Třetí příklad přidává text do chráněného souboru `file.txt` s použitím přepínače `-Force`, který umožňuje zápis i do souborů s omezenými oprávněními.

---

18. Get-Variable

Popis: Cmdlet **`Get-Variable`** slouží k **získání informací o proměnných** definovaných v aktuální relaci PowerShellu. Umožňuje zobrazit seznam všech proměnných nebo konkrétní proměnné.

Syntaxe:

```
Get-Variable [-Name <name>] [-ValueOnly] [-Scope <scope>]
```

Příklad:

```
# Získání všech proměnných
Get-Variable

# Získání konkrétní proměnné
Get-Variable -Name Path

# Získání hodnoty proměnné PATH
Get-Variable -Name Path -ValueOnly
```

Vysvětlení:

První příklad zobrazuje seznam všech proměnných definovaných v aktuální relaci.
Druhý příklad zobrazuje pouze proměnnou `Path`.
Třetí příklad zobrazuje pouze hodnotu proměnné `Path` bez dalších informací.

---

19. Set-Variable

Popis: Cmdlet **`Set-Variable`** slouží k **nastavení nebo úpravě hodnoty existující proměnné**. Může také vytvářet nové proměnné, pokud ještě neexistují.

Syntaxe:

```
Set-Variable -Name <name> -Value <value> [-Scope <scope>] [-Force]
```

Příklad:

```
# Nastavení hodnoty proměnné
Set-Variable -Name MyVar -Value "Hello, PowerShell!"
```

```
# Vytvoření nové proměnné s hodnotou
Set-Variable -Name Count -Value 10

# Přepsání existující proměnné bez potvrzení
Set-Variable -Name Count -Value 20 -Force
```

Vysvětlení:

První příklad nastavuje hodnotu proměnné `MyVar` na `"Hello, PowerShell!"`.

Druhý příklad vytváří novou proměnnou `Count` s hodnotou `10`.

Třetí příklad přepíše existující proměnnou `Count` na novou hodnotu `20` pomocí přepínače `-Force`, který umožňuje přepsání bez potvrzení.

---

20. Clear-Variable

Popis: Cmdlet **`Clear-Variable`** slouží k **vymazání hodnoty proměnné**. Proměnná zůstává definovaná, ale její hodnota je odstraněna (nastavena na `$null`).

Syntaxe:

```
Clear-Variable -Name <name> [-Scope <scope>] [-ErrorAction <action>]
```

Příklad:

```
# Vymazání hodnoty proměnné
Clear-Variable -Name MyVar

# Vymazání hodnoty proměnné s kontrolou
Clear-Variable -Name Count -ErrorAction SilentlyContinue
```

Vysvětlení:

První příklad vymaže hodnotu proměnné `MyVar`, aniž by odstranil samotnou proměnnou.

Druhý příklad vymaže hodnotu proměnné `Count` a potlačí případné chyby, pokud proměnná neexistuje, pomocí přepínače `-ErrorAction SilentlyContinue`.

Shrnutí Nejběžnějších Cmdletů

Cmdlet	Popis	Příklad Použití
`Copy-Item`	Kopíruje soubory nebo složky	`Copy-Item -Path "C:\Source\file.txt" -Destination "D:\Backup\"`
`Move-Item`	Přesouvá soubory nebo složky	`Move-Item -Path "C:\Source\file.txt" -Destination "D:\Destination\"`
`Remove-Item`	Odstraňuje soubory nebo složky	`Remove-Item -Path "C:\Source\file.txt" -Recurse`
`New-Item`	Vytváří nové soubory nebo složky	`New-Item -Path "C:\NewFolder" -ItemType Directory`
`Set-Location`	Změní aktuální pracovní adresář	`Set-Location -Path "C:\Projects"`
`Get-Content`	Získá obsah souboru	`Get-Content -Path "C:\Logs\log.txt"`

Cmdlet	Popis	Příklad Použití
`Set-Content`	Zapisuje obsah do souboru	`Set-Content -Path "C:\Notes\note.txt" -Value "New content"`
`Add-Content`	Přidává obsah na konec souboru	`Add-Content -Path "C:\Logs\log.txt" -Value "New log entry"`
`Get-Variable`	Získá informace o proměnných	`Get-Variable -Name Path`
`Set-Variable`	Nastaví nebo upraví hodnotu proměnné	`Set-Variable -Name Count -Value 10`
`Clear-Variable`	Vymaže hodnotu proměnné	`Clear-Variable -Name Count`

Odkazy:

Microsoft.com

Cisco Linux Essentials

Chatgpt