Operační systémy Windows 3

Klíčové pojmy:

- Command Prompt Windows
- Argumenty příkazů
- Options
- Historie v cmd
- Místní proměnné
- Proměnné prostředí
- Proměnná Path
- Interní příkazy
- Externí příkazy
- Aliasy
- Řídící a speciální znaky

Příkazový řádek (cmd) Windows

CLI (Command Line Interface) v systému Windows je textové rozhraní, které umožňuje uživatelům zadávat příkazy přímo systému a tím provádět různé operace, jako je práce se soubory, správou systému, skriptováním a automatizací úloh. CLI v prostředí Windows zahrnuje **Příkazový řádek (Command Prompt, cmd.exe)** a pokročilejší nástroj **PowerShell**. V této části se zaměříme zejména na **příkazový řádek (cmd)** a jeho klíčové vlastnosti.

Příkazy

Příkazy jsou základní jednotkou, kterou uživatelé zadávají do CLI, aby provedli určité operace. Tyto příkazy mohou být:

- **Interní:** Příkazy zabudované přímo do příkazového shellu (např. `dir`, `cd`, `echo`).
- **Externí:** Samostatné programy nebo nástroje, které jsou spouštěny prostřednictvím CLI (např. `ping`, `ipconfig`, `notepad`).

Příkazy mohou být doplněny **argumenty** a **volbami (options)**, které specifikují, jak má být příkaz proveden.

Jak zadat příkaz v CLI

Abyste mohli zadat příkaz v CLI Windows, musíte nejdříve otevřít Příkazový řádek:

1. Otevřete Start nebo stiskněte Windows + R a zadejte cmd.
2. Po otevření se zobrazí konzolové okno s výzvou, kde můžete zadávat příkazy.
3. Napište příkaz a stiskněte Enter pro jeho vykonání.

Argumenty příkazů v cmd

Argumenty jsou hodnoty, které se přidávají za příkaz a určují, nad čím má příkaz pracovat. Například při použití příkazu copy potřebujete zadat **zdrojový soubor** a **cílovou složku** jako argumenty:

```
copy soubor.txt C:\CilovaSlozka
```

Zde jsou `soubor.txt` a `C:\CilovaSlozka` argumenty příkazu `copy`.

Options příkazů v cmd

Options (volby) jsou speciální **přepínače** nebo **modifikátory**, které mění chování příkazu. Obvykle se zapisují jako **lomítko** `/` **následované písmenem** nebo slovem. Například u příkazu `dir` můžete použít volbu `/w` pro zobrazení výstupu v šířkovém formátu:

```
dir /w
```

Nebo řekneme, že chceme nařídit příkazu `ping` provedení čtyř pokusů o odeslání:

```
ping google.com /n 4
```

Historie příkazů v cmd

V **CLI Windows** je možné procházet historii příkazů pomocí kláves **šipka nahoru** a **šipka dolů**. To vám umožňuje rychle znovu spustit příkazy, které jste již zadali, bez nutnosti je znovu psát.

CLI uchovává **historii příkazů** po dobu trvání aktuální relace, ale existují také nástroje, jako je **doskey**, které mohou tuto historii spravovat:

```
doskey /history
```

Tento příkaz zobrazí seznam všech příkazů, které byly zadány během aktuální relace.

Proměnné

Proměnné v cmd Windows umožňují ukládat hodnoty, které mohou být později použity v příkazech nebo skriptech. Existují dva hlavní typy proměnných:

- **Místní proměnné**: Tyto proměnné jsou platné pouze pro aktuální relaci příkazového řádku. Lze je definovat příkazem `set`:

  ```
  set mojePromenna=hodnota
  echo %mojePromenna%
  ```

  Zde `set` nastaví hodnotu proměnné `mojePromenna` na `hodnota` a příkaz `echo` zobrazí její obsah.

- **Proměnné prostředí**: Tyto proměnné jsou **globální** a jsou platné pro celý systém, případně uživatele. Ovlivňují chování celého systému a aplikací. Příklady proměnných prostředí jsou `PATH`, `TEMP`, `USERNAME`. Tyto proměnné mohou být spravovány pomocí příkazu `set` nebo nastaveny v systémových nastaveních.

  ```
  echo %PATH%
  ```

Vytvoření proměnné prostředí

Aby se **místní proměnná** stala **globální proměnnou** v systému Windows, je třeba ji převést na **proměnnou prostředí**, což znamená, že bude dostupná pro všechny uživatelské relace a aplikace v systému, nejen v rámci aktuálního příkazového řádku (CLI). To lze provést buď přímo pomocí příkazového řádku, nebo přes grafické uživatelské rozhraní (GUI).

1. **Přes příkazový řádek (cmd)**

Pomocí příkazu `setx` lze vytvořit proměnnou prostředí, která bude **globální** (dostupná i po restartu a ve všech budoucích relacích). **Na rozdíl od příkazu `set`, který platí jen pro aktuální relaci CLI**, `setx` uloží proměnnou do systémových proměnných.

Například máme místní proměnnou:

```
set mojePromenna=hodnota
```

Tato proměnná bude dostupná pouze v aktuálním okně příkazového řádku. Jakmile zavřete okno, proměnná se ztratí. Aby se proměnná stala globální (proměnnou prostředí, tedy dostupnou ve všech dalších aplikacích a relacích), použijete příkaz `setx`:

```
setx mojePromenna "hodnota"
```

Tímto způsobem se vytvoří globální proměnná `mojePromenna` s hodnotou `hodnota`, která bude dostupná i po restartu systému. Následně, když znovu otevřete příkazový řádek a zadáte:

```
echo %mojePromenna%
```

Zobrazí se hodnota.

2. **Přes GUI**

Globální proměnné prostředí lze také přidat přes Nastavení systému:

1. Otevřete vlastnosti systému:
   - Klikněte pravým tlačítkem na Tento počítač (nebo Tento počítač na ploše) a vyberte Vlastnosti.
   - Klikněte na Pokročilá nastavení systému (vlevo).
   - Otevře se okno Vlastnosti systému.
2. Přejděte na proměnné prostředí:
   - Klikněte na tlačítko Proměnné prostředí... ve spodní části okna.
3. Přidání globální proměnné:
   - V části Systémové proměnné klikněte na Nová....
   - Zadejte název proměnné (např. mojePromenna) a její hodnotu (např. hodnota).
   - Potvrďte kliknutím na OK.
4. Uložení a použití:
   - Poté, co proměnnou přidáte, bude dostupná ve všech nových oknech CLI a aplikacích. Pokud chcete, aby se změna projevila okamžitě, zavřete a znovu otevřete okno příkazového řádku.

Odstranění proměnné prostředí

Chcete-li **trvale** odstranit proměnnou prostředí, která byla přidána pomocí `setx` nebo v systémových nastaveních, použijte tento příkaz:

```
setx PROMENNA ""
```

Zde `PROMENNA` je název proměnné, kterou chcete odstranit. Přiřazením prázdné hodnoty (tedy "") dojde k odstranění proměnné.

Tento příkaz proměnnou odstraní a již nebude dostupná v systému po restartu. Změna se projeví až po zavření a znovuotevření okna příkazového řádku.

Pokud chcete proměnnou odstranit pouze z aktuální relace příkazového řádku (**nebude trvale odstraněna ze systému**), použijte příkaz `set`:

```
set PROMENNA=
```

Tímto způsobem bude proměnná `PROMENNA` vymazána, ale pouze pro aktuální relaci příkazového řádku. Po uzavření a opětovném otevření příkazového řádku nebo po restartu systému bude proměnná znovu dostupná.

Proměnná PATH

Proměnná `PATH` je klíčovou proměnnou prostředí, která určuje **složky**, ve kterých CLI vyhledává spustitelné soubory, když zadáte příkaz. Pokud chcete spustit program bez nutnosti uvádět celou cestu, musí být jeho cesta zahrnuta v proměnné `PATH`.

Pro zobrazení aktuální hodnoty `PATH`:

```
echo %PATH%
```

Pro přidání nové cesty do proměnné PATH:

```
set PATH=%PATH%;C:\NovaCesta
```

Tímto přidáte novou složku `C:\NovaCesta` k aktuálnímu `PATH`.

Interní příkazy

Interní příkazy jsou ty, které jsou přímo zabudované do shellu (cmd.exe). Nevyžadují žádné externí programy pro jejich spuštění. Příklady:

- `dir`: Zobrazí seznam souborů a složek.
- `cd`: Změní aktuální adresář.
- `copy`: Kopíruje soubory.
- `echo`: Zobrazuje zprávy nebo hodnoty proměnných.
- `set`: Správa proměnných.

Externí příkazy

Externí příkazy jsou samostatné programy nebo nástroje, které nejsou přímo integrovány do příkazového řádku. Aby mohly být spuštěny, musí být v cestě systému nebo zadány s plnou cestou. Příklady:

- `notepad`: Spouští aplikaci Poznámkový blok.
- `ping`: Testuje dosah síťových připojení.
- `ipconfig`: Zobrazuje konfiguraci sítě.

Aliasy

Aliasy jsou alternativní názvy pro příkazy, které můžete použít ke zkrácení dlouhých příkazů nebo vytvoření vlastních zkratek. V CLI Windows můžete používat aliasy pomocí nástroje **doskey**:

```
doskey dirlist=dir /B
```

Tento příkaz vytváří alias `dirlist`, který bude spouštět příkaz `dir /B`. Poté můžete zadat `dirlist` místo plného příkazu.

Řídící a speciální znaky

Při práci s **příkazovým řádkem Windows (cmd)** se často setkáváme s různými **speciálními a řídícími znaky**, které hrají důležitou roli při zadávání příkazů, manipulaci s proměnnými a řízení toku skriptů. Každý z těchto znaků má specifickou funkci a může být interpretován příkazovým interpretem různými způsoby v závislosti na kontextu.

1. Dvojité uvozovky (")

Používají se k **uzavření textu nebo řetězce**, který obsahuje mezery nebo speciální znaky. Vše, co je uvnitř dvojitých uvozovek, je považováno za jeden celek.

Příklad:

```
echo "Tento text má mezery"
```

Použití dvojitých uvozovek je důležité při práci s cestami, které obsahují mezery.

Příklad:

```
cd "C:\Program Files"
```

2. Jednoduché uvozovky (')

V příkazovém řádku cmd jednoduché uvozovky nemají speciální význam jako v některých jiných shellech (např. v Linuxu nebo PowerShellu). Používají se jen jako obyčejné znaky, které budou vypsány nebo interpretovány doslovně.

```
echo 'Tento text má jednoduché uvozovky'
```

3. Procento (%)

Používá se k **referencování proměnných prostředí** nebo pro práci s **lokálními proměnnými** v dávkových souborech.

Pro proměnné prostředí je syntaxe `%PROMENNA%`:

```
echo %USERNAME%
```

Tento příkaz vypíše aktuální uživatelské jméno uložené v proměnné prostředí `USERNAME`.

V dávkových souborech se procento také používá pro práci s **argumenty skriptů**. Například `%1` odkazuje na první argument skriptu:

```
echo První argument je: %1
```

4. Lomítko (/)

Lomítko se používá k označení voleb (options) **nebo** přepínačů **příkazů.**

Příklady:

`/w` pro formát zobrazení v příkazu `dir`:

```
dir /w
```

`/s` pro rekurzivní operace:

```
del /s *.txt
```

odstraní všechny soubory s příponou `.txt` v aktuální složce a ve všech podadresářích.

5. Zpětné lomítko (\)

Zpětné lomítko se používá k označení cest k souborům nebo adresářům ve Windows. Je to oddělovač adresářů v cestě.

Příklad:

```
cd C:\Windows\System32
```

Dvojité zpětné lomítko (`\\`) se používá k označení síťových cest**.**

```
net use t: \\Server\SdilenyAdresar
```

6. Svislítko (Pipe, |)

Svislítko umožňuje přesměrování výstupu jednoho příkazu jako vstupu do jiného příkazu**.** Používá se pro propojení příkazů, které na sebe navazují.

Příklad:

```
dir | find "soubor.txt"
```

Tento příkaz vyhledá řetězec `soubor.txt` ve výstupu příkazu `dir`.

7. Přesměrování (>, >>)

> přesměruje výstup příkazu do souboru. Pokud soubor existuje, bude přepsán.
>> přidá výstup příkazu na konec souboru, aniž by přepsal jeho obsah.

Příklad:

```
echo "Ahoj světe!" > soubor.txt
rem (vytvoří soubor nebo přepíše obsah)
echo "Další řádek" >> soubor.txt
rem (přidá další řádek)
```

8. Přesměrování vstupu (<)

Přesměruje obsah souboru jako vstup pro příkaz.

Příklad**:**

```
sort < soubor.txt
```

Načte obsah souboru `soubor.txt`, seřadí jeho řádky v abecedním pořadí a výsledek zobrazí v příkazovém řádku.

9. Sloučení (&)

Umožňuje spustit více příkazů v jedné řádce, přičemž se všechny příkazy provedou jeden po druhém.

```
echo Ahoj & echo Světe
```

10. Podmíněné sloučení (&&)

&& znamená, že druhý příkaz se provede pouze, pokud první příkaz proběhne úspěšně (tj. vrátí kód 0).

Příklad:

```
mkdir NovyAdresar && cd NovyAdresar
```

11. Logický operátor OR (||)

`||` znamená, že druhý příkaz se provede pouze, pokud první příkaz selže (vrátí chybu).

Příklad:

```
del soubor.txt || echo Soubor nebyl nalezen
```

12. Escape znak (^)

Escape znak (^) se používá k tomu, aby speciální znak byl považován za obyčejný znak a nikoli jako řídící znak příkazového řádku. Používá se, pokud chcete použít speciální znaky (`|`, `>`, `<`, `&`, `&&`) ve skriptu nebo příkazu bez jejich speciální funkce.

Příklad:

```
echo Toto je svislítko: ^|
```

13. Dvojtečka (:)

V dávkových souborech se dvojtečka používá k označení štítků (labels) pro skoky (např. s příkazem goto).

Příklad:

```
:start
echo Tento příkaz se provede
goto end
:end
```

14. Otazník (?)

Otazník se používá jako zástupný znak pro jeden libovolný znak při práci se soubory nebo složkami.

Příklad:

```
del soubor?.txt
```

Tento příkaz odstraní soubory jako `soubor1.txt`, `soubor2.txt` apod.

15. Hvězdička (*)

Hvězdička se používá jako zástupný znak pro libovolný počet znaků**.** Může zastupovat libovolný řetězec znaků.

Příklad:

```
del *.txt
```

Tento příkaz odstraní všechny soubory s příponou `.txt` v aktuální složce.

16. Zavináč (@)

Zavináč před příkazem potlačí zobrazení samotného příkazu před jeho vykonáním. Často se používá v dávkových souborech.

Příklad:

`echo off`: Potlačí výpis následujících příkazů, ale samotný příkaz echo off se zobrazí.
`@echo off`: Potlačí výpis všech příkazů včetně echo off, takže žádný z příkazů nebude viditelný, jen výstupy z echo nebo jiných příkazů.

Odkazy:

Microsoft.com

Wikipedia

Cisco Linux Essentials

Root.cz

Chatgpt