Terminal do Windows

Tópicos de Informática para Automação

Mário Antunes

September 29, 2025

Universidade de Aveiro

Bem-vindo à Linha de Comandos do Windows! 💻

Mais do que uma Caixa Azul

A **Interface de Linha de Comandos (CLI)** é a sua ligação direta, baseada em texto, ao sistema operativo Windows.

- **Porquê usá-la?**
  - **Poder e Velocidade:** Execute tarefas complexas e aceda a configurações não disponíveis na GUI.
  - **Automação:** Crie *scripts* para tarefas repetitivas com ficheiros *Batch* ou *PowerShell*.
  - **Eficiência:** Utiliza o mínimo de recursos do sistema em comparação com ferramentas gráficas.
  - **Padrão da Indústria:** Essencial para programadores, profissionais de TI e administradores de sistemas em Windows.

Os Interpretadores: CMD vs. PowerShell

O Windows oferece dois interpretadores de linha de comandos principais.

- **Command Prompt (CMD):**
  - O interpretador legado do Windows, originário do MS-DOS.
  - Simples, fiável para operações básicas de ficheiros.
  - A sua linguagem de *scripting* (*Batch*) é básica e menos poderosa.
- **PowerShell:**
  - O interpretador moderno, poderoso e recomendado.
  - Trata tudo como um **objeto**, e não apenas texto, permitindo uma manipulação de dados mais avançada.
  - **Vamos mostrar exemplos para ambos, destacando as vantagens do PowerShell.**

O Sistema de Ficheiros do Windows

O sistema de ficheiros começa com letras de unidade (ex: `C:`), e não com uma única raiz (`/`).

- `C:\`: A raiz da unidade principal, onde o Windows está tipicamente instalado.
- `C:\Users`: Os seus ficheiros pessoais estão aqui (ex: `C:\Users\Student`). É o equivalente a `/home`.
- `C:\Windows\System32`: Contém programas e binários essenciais do sistema. É o equivalente mais próximo do `/bin` do Linux.

- `C:\Program Files`: Local de instalação padrão para aplicações de 64 bits.
- **O Registry:** Ao contrário da configuração baseada em texto do Linux em `/etc`, grande parte da configuração central do Windows é armazenada numa base de dados hierárquica chamada *Registry*.

Ficheiros Ocultos & Atributos

No Windows, “oculto” é um atributo de ficheiro, não apenas uma convenção de nome. Por defeito, os ficheiros ocultos não são mostrados.

Command Prompt (`dir`)

```
# Ver apenas ficheiros ocultos
$ dir /a:h

# Ver TODOS os ficheiros (incluindo ocultos)
$ dir /a
```

PowerShell (`ls` ou `Get-ChildItem`)

```
# Ver todos os ficheiros, incluindo ocultos e de sistema
$ ls -Force
```

O modificador `-Force` diz ao PowerShell para mostrar itens que normalmente estariam ocultos.

Navegação Básica: Mudar de Diretório (`cd`)

Mover-se no sistema de ficheiros é fundamental. O comando `cd` funciona em ambos, mas com uma diferença crucial.

Interpretador	Comando	Descrição
CMD	`cd C:\Users\Student`	Muda o diretório.
	`D:`	Para mudar de unidade, escreva a letra da unidade.
PowerShell	`cd C:\Users\Student`	Muda o diretório.
	`cd D:`	Muda a unidade diretamente.

Ponto-chave: O `cd` do PowerShell é mais intuitivo, pois lida com mudanças de caminho e de unidade com um único comando.

Navegação Básica: A Orientar-se

Tarefa	Command Prompt (CMD)	PowerShell
Mostrar localização atual	`cd` (sem argumentos)	`Get-Location` (alias: `pwd`)
Subir um nível	`cd ..`	`cd ..`
Ir para a pasta pessoal	`cd %USERPROFILE%`	`cd ~`

Listar Conteúdo de Diretórios (`dir`, `ls`)

`dir` e `Get-ChildItem` (com o alias `ls`) são os seus olhos no terminal.

Command Prompt (`dir`)

```
# Listagem simples
$ dir

# Formato largo, menos detalhe
$ dir /w
```

PowerShell (`Get-ChildItem` ou `ls`)

```
# Listagem simples (como o ls do Linux)
$ ls

# Uma vista mais detalhada (como o ls -l do Linux)
$ ls | Format-List
```

Criar Diretórios (`mkdir`)

Ambas as *shells* usam `mkdir` (ou `md`), mas a do PowerShell é mais poderosa.

Command Prompt

No CMD, tem de criar cada nível de um caminho de diretórios aninhado, um por um.

```
$ mkdir Projetos
$ mkdir Projetos\TIA
```

PowerShell

O PowerShell pode criar todo o caminho de diretórios pais automaticamente, semelhante a `mkdir -p` no Linux.

```
# Este único comando cria tanto 'Projetos' como 'TIA'
$ mkdir Projetos\TIA
```

Criar Ficheiros

O Windows não tem um equivalente direto ao `touch`, por isso usamos outros métodos.

Command Prompt

Usa redirecionamento. `echo.` cria uma linha em branco, que é redirecionada para um novo ficheiro.

```
# Cria um ficheiro vazio
$ echo. > notes.txt

# Cria um ficheiro com conteúdo (sobrescreve)
$ echo A minha primeira linha. > notes.txt

# Anexa conteúdo a um ficheiro
$ echo A minha segunda linha. >> notes.txt
```

PowerShell

Usa o *cmdlet* `New-Item` para ficheiros vazios e `Set-Content` para conteúdo.


```
# Cria um ficheiro vazio
```

Editar Ficheiros com o Notepad

O Windows não tem um editor de terminal moderno integrado como o `nano` ou o `vim`.

A sua principal ferramenta para editar ficheiros a partir da CLI é lançar um editor gráfico como o **Notepad**.

```
# Funciona tanto no CMD como no PowerShell
$ notepad o_meu_ficheiro.txt
```

Este comando abrirá `o_meu_ficheiro.txt` na aplicação Notepad. Se o ficheiro não existir, o Notepad perguntará se o deseja criar.

Obter Informação do Sistema (Parte 1)

Tarefa	Command Prompt (CMD)	PowerShell
Utilizador atual	`whoami` ou `echo %USERNAME%`	`whoami` ou `$env:USERNAME`
Data/Hora	`date /t && time /t`	`Get-Date`

Obter Informação do Sistema (Parte 2)

Tarefa	Command Prompt (CMD)	PowerShell
Informação Geral do Sistema	`systeminfo`	`Get-ComputerInfo`
Processos em Execução	`tasklist`	`Get-Process` (alias: `ps`)

Utilizadores & Privilégios (Administrador)

O Windows tem dois níveis principais de utilizador: **Utilizador Padrão** e **Administrador**.

- **Administrador** é o equivalente ao utilizador `root` do Linux.
- Não existe um equivalente direto ao `sudo`. Para executar um único comando com privilégios elevados, tem de abrir um **novo terminal elevado**.

Como Elevar Privilégios:

1. Procure por “cmd” ou “powershell” no Menu Iniciar.
2. Clique com o botão direito do rato no ícone e selecione **“Executar como administrador”**.

Qualquer comando executado nesta nova janela terá plenos direitos administrativos.

Gestão de Pacotes: Winget

O Windows moderno inclui o **Windows Package Manager (`winget`)**, uma ferramenta de linha de comandos para instalar *software*. Funciona tanto no CMD como no PowerShell.

```
# Procurar uma aplicação (ex: 7zip)
$ winget search 7zip

# Instalar uma aplicação
$ winget install 7zip.7zip

# Listar aplicações instaladas
$ winget list

# Desinstalar uma aplicação
$ winget uninstall 7zip.7zip
```

Alternativa: Durante anos, o padrão da comunidade tem sido o **Chocolatey**, que continua a ser uma alternativa muito poderosa.

Automação com o Agendador de Tarefas

O equivalente ao `cron` no Windows é o **Agendador de Tarefas** (*Task Scheduler*). Pode geri-lo através de uma GUI ou da linha de comandos.

Command Prompt (`schtasks`)

Cria uma tarefa para executar um *script* todos os dias às 8h.

```
$ schtasks /create /sc daily /tn "A Minha Tarefa" /tr "C:\Scripts\o_meu_script.bat" /st 08:0
```

PowerShell (`*-ScheduledTask`)

O PowerShell oferece uma forma muito mais estruturada de criar tarefas.

```
$action = New-ScheduledTaskAction -Execute "C:\Scripts\OMeuScript.ps1"
$trigger = New-ScheduledTaskTrigger -Daily -At 8am
Register-ScheduledTask -Action $action -Trigger $trigger -TaskName "A Minha Tarefa"
```

O Poder do Pipe |

O *pipe* envia o *output* de um comando para outro. No PowerShell, isto é mais poderoso porque envia **objetos estruturados**, e não apenas texto.

Command Prompt (*Pipe* de Texto)

Encontra o processo "explorer" a partir da lista de texto completa.

```
$ tasklist | findstr /i "explorer"
```

PowerShell (*Pipe* de Objetos)

Obtém objetos de processo, filtra-os e seleciona propriedades específicas.

```
# Obter o objeto de processo para o "explorer"
$ Get-Process | Where-Object { $_.ProcessName -eq "explorer" }

# Obter o processo e selecionar apenas o seu nome e uso de CPU
$ Get-Process "explorer" | Select-Object Name, CPU
```

Variáveis de Ambiente

Variáveis que armazenam configurações do sistema. A sintaxe é diferente em cada *shell*.

Command Prompt

```
# Ver uma variável usando %VAR%
$ echo %PATH%

# Definir uma variável para a sessão atual
$ set MINHAVAR=Ola
```

PowerShell

```
# Ver uma variável usando $env:VAR
$ echo $env:Path

# Definir uma variável para a sessão atual
$ $env:MINHAVAR="Ola"
```

Nota: Para tornar uma alteração de variável **permanente**, deve usar o comando `setx` ou editar as Propriedades do Sistema na GUI.

Introdução ao Scripting

- **Scripts Batch (`.bat`, `.cmd`):** A linguagem de *scripting* tradicional para o CMD. Simples, mas limitada.
- **Scripts PowerShell (`.ps1`):** Uma linguagem de *scripting* moderna e completa. Poderosa e versátil.

Como Executar Scripts

1. Guarde o seu código num ficheiro de texto com a extensão correta (`.bat` ou `.ps1`).
2. Navegue para o diretório no seu terminal.
3. Execute o *script*:
    - **CMD:** `o_meu_script.bat`
    - **PowerShell:** `./o_meu_script.ps1`

Segurança do PowerShell: Por defeito, a execução de *scripts* PowerShell está desativada. Poderá precisar de executar `Set-ExecutionPolicy RemoteSigned` num PowerShell

Exemplo de Script 1: Olá Utilizador

Batch (`ola.bat`)

```
@echo off
REM Define uma variável e imprime-a
set USERNAME=Estudante
echo Ola, %USERNAME%!
```

PowerShell (`ola.ps1`)

```
# Define uma variável e imprime-a
$Username = "Estudante"
Write-Host "Ola, $Username!"
```

Exemplo de Script 2: Se Ficheiro Existe

Batch (verifica_ficheiro.bat)

```
@echo off
REM Verifica se um ficheiro existe no diretório atual
if exist "notes.txt" (
    echo "O ficheiro notes.txt foi encontrado."
) else (
    echo "O ficheiro notes.txt NAO foi encontrado."
)
```

PowerShell (verifica_ficheiro.ps1)

```
# Verifica se um ficheiro existe no diretório atual
if (Test-Path "./notes.txt") {
    Write-Host "O ficheiro notes.txt foi encontrado."
} else {
    Write-Host "O ficheiro notes.txt NAO foi encontrado."
}
```

Exemplo de Script 3: Ciclo por Ficheiros

Batch (`lista_ficheiros.bat`)

```
@echo off
REM Lista todos os ficheiros .txt no diretório atual
echo Encontrados os seguintes ficheiros de texto:
for %%F in (*.txt) do (
    echo - %%F
)
```

PowerShell (`lista_ficheiros.ps1`)

```
# Lista todos os ficheiros .txt no diretório atual
Write-Host "Encontrados os seguintes ficheiros de texto:"
foreach ($file in Get-ChildItem "*.txt") {
    Write-Host "- $($file.Name)"
}
```

Pensamentos Finais: CMD vs. PowerShell

- **Use o CMD quando:** Precisa de executar comandos muito simples e antigos ou ficheiros *batch* legados.
- **Use o PowerShell quando:** Quer realizar tarefas administrativas, automatizar fluxos de trabalho complexos ou gerir sistemas Windows de forma eficiente. É o futuro da linha de comandos do Windows.

Para qualquer trabalho sério, **aprender PowerShell é altamente recomendado**. É mais poderoso, consistente e oferece um controlo muito superior sobre o sistema operativo Windows.

Apoio & Mais Recursos 📚

Guarde estas páginas nos seus favoritos para referência rápida.

- **CMD Cheat Sheets:**
  - StationX CMD Cheat Sheet
  - Columbia University CMD Cheatsheet
- **PowerShell Cheat Sheets:**
  - Microsoft PowerShell Language Reference
  - StationX PowerShell Cheat Sheet