目录

[一、启动 PowerShell 1](#_Toc200754392)

[二、基础命令与语法 1](#_Toc200754393)

[（一）命令格式 1](#_Toc200754394)

[（二）获取帮助 1](#_Toc200754395)

以下是一份系统的 PowerShell 入门教程，从基础到进阶，助你快速上手：

名词

power [ˈpaʊə(r)]  
shell [ʃel]

# 一、启动 PowerShell

（一）Windows 系统

- \*\*开始菜单\*\*：点击“开始”，搜索“PowerShell” ，选“Windows PowerShell”（普通权限）或“Windows PowerShell（管理员）”（高权限操作，如修改系统设置 ）。

- \*\*运行窗口\*\*：按 `Win + R` ，输入 `powershell` 回车启动；需管理员权限则输 `powershell` 后按 `Ctrl + Shift + Enter` 。

#### （二）Linux、macOS 系统

打开终端，输入 `pwsh` 启动。未安装的话：

- Ubuntu/Debian：

```bash

sudo apt - get update

sudo apt - get install - y powershell

```

- macOS：

`brew install --cask powershell`（需先装 Homebrew ）

# 二、基础命令与语法

## （一）命令格式

遵循 \*\*动词 - 名词\*\* 规范，直观表意，如 `Get-Process`（获取进程）、`Start-Service`（启动服务 ）。

## （二）获取帮助

用 `Get-Help` 查命令详情，例：

- 基础帮助：

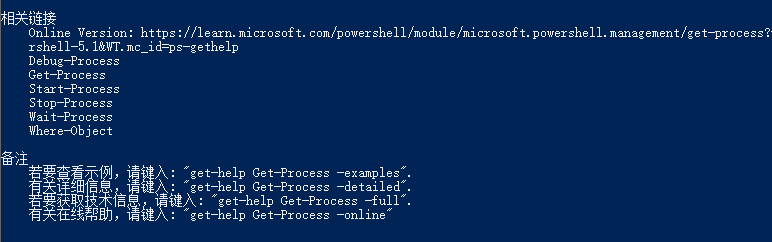
Get-Help Get-Process

- 详细帮助：

Get-Help Get-Process -Detailed

- 示例帮助：

Get-Help Get-Process -Examples Get-Help Get-Process -Detailed



#### （三）常用命令

- \*\*系统信息\*\*：

- 进程：`Get-Process`（查看运行进程，含 CPU、内存占用 ）

- 服务：`Get-Service`（列出系统服务及状态 ）

- 环境变量：`Get-ChildItem Env:`（查看系统/用户环境变量 ）

- \*\*文件/文件夹操作\*\*：

- 列目录：`Get-ChildItem`（类似 `ls`/`dir` ，显示文件、文件夹 ）

- 建文件夹：`New-Item -ItemType Directory -Path "C:\NewFolder"`（指定路径创建目录 ）

- 复制文件：`Copy-Item -Path "C:\Source.txt" -Destination "D:\TargetFolder"`（拷贝文件到目标位置 ）

- \*\*网络操作\*\*：

- 测试连接：`Test-NetConnection -ComputerName "www.example.com"`（检测与目标主机网络连通性 ）

- 查网络适配器：`Get-NetAdapter`（查看网卡信息，如名称、IP ）

### 三、管道与筛选（高效处理数据）

#### （一）管道（`|` ）

将前一命令输出，作为后一命令输入，实现数据流转。例：

`Get-Process | Sort-Object -Property CPU -Descending | Select-Object -First 5`

\*\*解析\*\*：先获所有进程 → 按 CPU 使用率降序排序 → 选前 5 个进程，快速定位高负载进程。

#### （二）筛选数据（`Where-Object` ）

按需过滤结果，例：

`Get-Process | Where-Object { $\_.CPU -gt 100 }`

\*\*解析\*\*：筛选出 CPU 使用率大于 100 的进程（`$\_` 代表管道传递的当前对象 ）。

### 四、变量与脚本（自动化核心）

#### （一）变量

以 `$` 开头，存储数据供复用。例：

```powershell

$message = "Hello, PowerShell!"

Write-Output $message # 输出变量内容

```

支持多变量赋值：`$a = $b = 100`（`$a`、`$b` 均为 100 ）。

#### （二）编写运行脚本

1. \*\*创建脚本\*\*：用文本编辑器（如 VS Code ）新建 `.ps1` 文件（如 `MyScript.ps1` ），写入代码：

```powershell

$processes = Get-Process # 获取进程

foreach ($proc in $processes) { # 遍历进程

Write-Output $proc.Name # 输出进程名称

}

```

2. \*\*运行脚本\*\*：

- 同目录下：在 PowerShell 中执行 `.\MyScript.ps1`

- 不同目录：用完整路径，如 `C:\Scripts\MyScript.ps1`

\*\*注意\*\*：默认 Windows 限制脚本执行，需修改策略（以当前用户为例 ）：

`Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser`

（`RemoteSigned` 允许运行本地创建、远程签名的脚本 ）

### 五、模块与函数（扩展功能、复用代码）

#### （一）模块

封装 cmdlet、函数等，扩展 PowerShell 能力。例：

`Import-Module -Name "ActiveDirectory"`（导入 AD 模块，管理域环境 ）

可通过 `Get-Module -ListAvailable` 查看已安装模块。

#### （二）函数

定义可复用代码块。例：

```powershell

function Show-Greeting {

Write-Output "Hello from PowerShell!"

}

Show-Greeting # 调用函数，输出问候语

```

支持参数化：

```powershell

function Add-Numbers {

param(

[int]$a,

[int]$b

)

return $a + $b

}

Add-Numbers -a 5 -b 3 # 输出 8

```

### 六、常见问题解决

1. \*\*权限不足\*\*：以管理员身份重新启动 PowerShell 再操作。

2. \*\*命令未找到\*\*：确认是否遗漏模块，用 `Import-Module` 加载对应模块后重试。

3. \*\*脚本无法运行\*\*：检查执行策略，按前文方法设置 `RemoteSigned` 等宽松策略。

通过上述内容实践，你能掌握 PowerShell 基础操作、自动化脚本编写，后续可深入探索模块开发、高级运维场景（如云环境管理 ），推荐结合 [PowerShell 中文博客](https://www.pstips.net/powershell-online-tutorials) 拓展学习！

# 安装-Microsoft.PowerShell命令

winget search Microsoft.PowerShell

# 新建与重命名参数

--source：源文件名或文件夹名，在重命名操作时必须提供。

create\_new\_entity函数：该函数用于创建新的文件或文件夹。如果文件名包含扩展名，则创建文件；否则创建文件夹。

--prefix：文件名或文件夹名的前缀，用于生成新的名称。默认值为空字符串。

--lang：语言标识，如py、ps等，用于识别文件类型。默认值为空字符串。

-e, --ext：文件扩展名（可带点号），用于指定文件的类型。如果未提供，脚本会根据lang参数或源文件的扩展名自动确定。默认值为None。

--version：版本号，用于生成文件名时的版本标识。默认值为1.0。

--year：年份，用于生成文件名时的日期部分。默认值为空字符串。

--month：月份，用于生成文件名时的日期部分。默认值为空字符串。

--day：日期，用于生成文件名时的日期部分。默认值为空字符串。

操作相关

--overwrite：一个布尔型参数，如果指定该参数，脚本在重命名或创建文件 / 文件夹时会覆盖已存在的同名文件 / 文件夹。

--source：源文件名或文件夹名，在重命名操作时必须提供。

--new\_entity\_name：手动指定新的文件名或文件夹名。如果未提供，脚本会根据其他参数自动生成。

--create：一个布尔型参数，如果指定该参数，脚本会执行创建新文件或文件夹的操作。

# PowerShell多行语句的使用方法

1. 使用反引号（`）进行换行

在 PowerShell 中，反引号（`）是行继续符。当你在一行的结尾使用反引号时，它会告诉 PowerShell 这行命令还没结束，会继续到下一行。

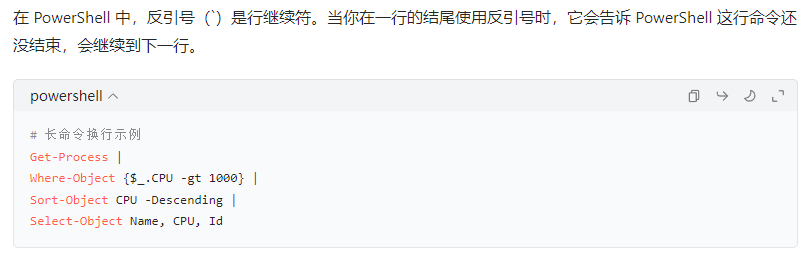
# 长命令换行示例

Get-Process |

Where-Object {$\_.CPU -gt 1000} |

Sort-Object CPU -Descending |

Select-Object Name, CPU, Id



你观察得很细致，在你提供的 PowerShell 代码示例里确实没有用到反引号（）\*\* 。这是因为 \*\*管道符（|）、逗号（,）、操作符（如 -gt 这类比较运算符 ）、括号（{}` 用于脚本块）等语法元素，本身就支持命令自然换行 ，不需要额外用反引号续行。

## 2. 利用语句块换行

某些命令本身就支持多行书写，像函数定义、循环结构等，无需使用反引号。

# 函数定义示例

function Get-Square {

param (

[int]$Number

)

return $Number \* $Number

}

# For循环示例

for ($i = 0; $i -lt 10; $i++) {

Write-Host "当前数字: $i"

}

## 3. 字符串中的多行处理

在 PowerShell 中，有两种字符串：单引号字符串和双引号字符串。它们都可以跨越多行。不过，要注意它们在处理换行符和变量扩展方面存在差异。

# 单引号字符串（不扩展变量）

$singleLine = '这是一个' +

'单行字符串'

# 双引号字符串（支持变量扩展）

$name = "PowerShell"

$multiLine = "这是一个

多行字符串，

欢迎使用 $name"

### 变量扩展

在 PowerShell 中，变量扩展是指在字符串中嵌入变量，当字符串被解析时，变量会被其实际值替换的过程。这是一个非常实用的功能，可以让你更方便地构建动态字符串。下面详细解释变量扩展的机制和用法：

1. 基本概念

变量扩展仅在双引号字符串（"）中生效，单引号字符串（'）会将变量名原样保留。

变量需以 $ 开头，PowerShell 会自动将其替换为变量的当前值。

## 4写多行字符串路径方法

方案 3：Here-String + 替换换行符

代码如下

# Here-String 多行书写，再替换换行符

$pythonScript = @"

E:\备份盘\带零文件夹\005\_计算机科学、程式、资料,硬件

\005\_400\_电脑编程!1\Scripts

\005\_490\_运行与调试\试验夹

\005\_430\_file\_rename\_creator\_py\_T1.py

"@ -replace "`r`n", "\" # 将换行符替换为\

Write-Output "拼接后的完整路径: $pythonScript"

代码如上

# 在 VS Code 中新建.ps1 文件

（PowerShell 脚本文件）

，可以通过以下几种方法：

使用快捷键：按下Ctrl + N（Windows/Linux）或Cmd + N（Mac），新建一个空白文件，然后点击 “文件” 菜单中的 “保存” 或使用快捷键Ctrl + S（Windows/Linux）或Cmd + S（Mac），在弹出的保存对话框中，输入文件名并加上.ps1扩展名，选择保存路径后点击 “保存” 即可。