目录

[### \*\*1. 安装与启动\*\* 1](#_Toc200919601)

PowerShell ISE 教程：入门与进阶

PowerShell ISE（Integrated Scripting Environment）是微软为 PowerShell 提供的官方集成开发环境，专为脚本编写和调试设计。本教程将带你从基础到进阶，全面掌握 PowerShell ISE 的使用。

# 1. 安装与启动

#### \*\*安装\*\*

- \*\*Windows 10/11\*\*：默认已安装（PowerShell 5.1 及以下版本）。

- \*\*PowerShell 7+\*\*：需从 [Microsoft Store](https://www.microsoft.com/store/productId/9MZ1SNWT0N5D) 或官网下载安装。

#### \*\*启动方式\*\*

- 按下 `Win + R`，输入 `powershell\_ise` 并回车。

- 在开始菜单中搜索 \*\*PowerShell ISE\*\*。

### \*\*2. 界面介绍\*\*

PowerShell ISE 的界面主要由三部分组成：

1. \*\*脚本编辑器\*\*（顶部）：编写和编辑 PowerShell 脚本。

2. \*\*命令提示符\*\*（底部）：实时执行 PowerShell 命令。

3. \*\*输出窗口\*\*（底部）：显示命令执行结果。

![PowerShell ISE 界面](https://i.imgur.com/5JZv4cK.png)

### \*\*3. 基础操作\*\*

#### \*\*3.1 编写第一个脚本\*\*

1. 在脚本编辑器中输入以下代码：

```powershell

# 输出问候语

Write-Host "Hello, PowerShell ISE!"

# 获取当前日期

$date = Get-Date

Write-Host "当前日期: $date"

```

2. 保存脚本（`Ctrl + S`），文件后缀为 `.ps1`（如 `hello.ps1`）。

3. 执行脚本：

- 点击工具栏的 \*\*运行脚本\*\* 按钮（绿色三角形）。

- 按 `F5` 键。

- 在命令提示符中输入 `.\hello.ps1` 并回车。

#### \*\*3.2 命令提示符使用\*\*

- 在命令提示符中输入单行命令，按 `Enter` 执行。

```powershell

Get-Process # 获取当前运行的进程

Get-Service # 获取系统服务

```

- 使用 \*\*向上箭头\*\* 键快速重复之前的命令。

#### \*\*3.3 自动完成功能\*\*

- 输入命令或参数的前几个字符，按 `Tab` 键自动补全。

```powershell

Get-Pro # 按 Tab 自动补全为 Get-Process

```

### \*\*4. 调试脚本\*\*

PowerShell ISE 提供强大的调试功能：

#### \*\*4.1 设置断点\*\*

- 在代码行号旁边点击，或按 `F9`，设置/取消断点。

```powershell

# 在这行设置断点

$result = 1 + 2

Write-Host "结果: $result"

```

#### \*\*4.2 调试命令\*\*

- `F5`：开始调试。

- `F10`：逐过程执行（跳过函数内部）。

- `F11`：逐语句执行（进入函数内部）。

- `Shift + F11`：跳出当前函数。

- `Ctrl + Shift + F5`：重启调试。

### \*\*5. 高级功能\*\*

#### \*\*5.1 脚本分析工具\*\*

- 点击工具栏的 \*\*分析脚本\*\* 按钮（放大镜图标），检查代码中的潜在问题。

#### \*\*5.2 多文件编辑\*\*

- 使用 \*\*新建选项卡\*\*（`Ctrl + T`）同时编辑多个脚本文件。

#### \*\*5.3 自定义布局\*\*

- 通过 \*\*视图\*\* 菜单调整界面布局，例如显示/隐藏侧边栏。

### \*\*6. 实用技巧\*\*

#### \*\*6.1 快速注释/取消注释\*\*

- 选中代码，按 `Ctrl + K + C` 注释。

- 选中代码，按 `Ctrl + K + U` 取消注释。

#### \*\*6.2 从命令历史复制\*\*

- 右键点击命令提示符中的历史命令，选择 \*\*复制\*\*。

#### \*\*6.3 导入模块\*\*

- 在脚本中使用 `Import-Module` 加载额外功能：

```powershell

Import-Module ActiveDirectory # 导入 Active Directory 模块

```

### \*\*7. 常见问题\*\*

#### \*\*7.1 执行策略限制\*\*

- 错误信息：`无法加载文件，因为在此系统上禁止运行脚本`。

- 解决方案：以管理员身份运行 PowerShell ISE，执行：

```powershell

Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser

```

#### \*\*7.2 中文乱码\*\*

- 解决方案：在脚本开头添加编码设置：

```powershell

[Console]::OutputEncoding = [System.Text.Encoding]::UTF8

```

### \*\*8. 进阶学习资源\*\*

- \*\*官方文档\*\*：[PowerShell 官方文档](https://docs.microsoft.com/zh-cn/powershell/)

- \*\*在线教程\*\*：[PowerShell 入门教程](https://www.learnpwsh.com/)

- \*\*社区\*\*：[PowerShell.org](https://powershell.org/)

### \*\*总结\*\*

PowerShell ISE 是学习和使用 PowerShell 的理想工具，通过本教程，你已掌握：

- 基本操作、脚本编写与执行。

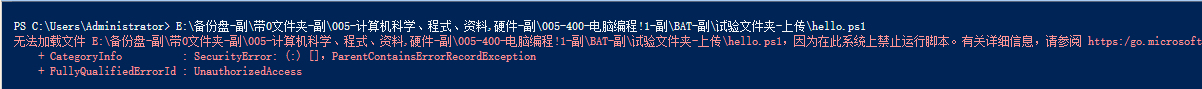
- 调试技巧与断点设置。

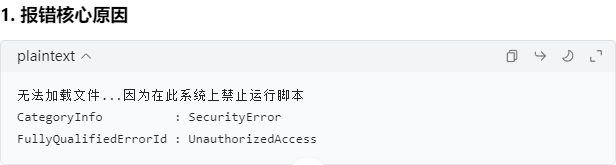
- 常用快捷键和高级功能。

继续实践，你将能够高效管理系统、自动化任务，并充分发挥 PowerShell 的强大功能！

# 2解除PowerShell 执行策略限制

## 1报错





2. 解决步骤（以管理员身份运行 PowerShell）

方式 1：临时设置宽松策略（适合测试）

# 以管理员身份运行 PowerShell，执行以下命令

Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser

3. 验证方法（可选）

重启后，打开 PowerShell 执行：

Get-ExecutionPolicy -Scope CurrentUser

输出 RemoteSigned 就说明策略还在，无需重复设置。

## 2完整版

从报错信息来看，这是 \*\*PowerShell 执行策略限制\*\* 导致的脚本无法运行问题，具体分析和解决方法如下：

### 1. 报错核心原因

```

无法加载文件...因为在此系统上禁止运行脚本

CategoryInfo : SecurityError

FullyQualifiedErrorId : UnauthorizedAccess

```

Windows 默认的 PowerShell \*\*执行策略（Execution Policy）\*\* 会限制脚本运行，目的是防止恶意脚本自动执行。当前策略可能是 `Restricted`（默认值），禁止运行本地/远程脚本。

### 2. 解决步骤（以管理员身份运行 PowerShell）

#### 方式 1：临时设置宽松策略（适合测试）

```powershell

# 以管理员身份运行 PowerShell，执行以下命令

Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser

```

- \*\*参数说明\*\*：

- `RemoteSigned`：允许运行本地创建的脚本（无需数字签名），但远程下载的脚本需要签名。

- `Scope CurrentUser`：仅对当前用户生效，不影响系统全局。

执行后会提示确认，输入 `Y` 回车即可。

#### 方式 2：更严格的策略（适合生产环境）

如果想保持严格安全，可仅针对当前脚本 bypass 策略：

```powershell

# 临时允许运行当前脚本，无需修改全局策略

powershell -ExecutionPolicy Bypass -File "E:\备份盘...\hello.ps1"

```

- 每次运行脚本时用此命令，避免全局放宽策略。

### 3. 验证是否生效

```powershell

# 查看当前执行策略

Get-ExecutionPolicy

# 预期输出（方式 1 生效后）：RemoteSigned

```

### 4. 额外注意事项

- \*\*脚本路径别带中文/空格\*\*：虽然 PowerShell 支持，但某些场景可能出问题，尽量用纯英文路径（如 `E:\scripts\hello.ps1` ）。

- \*\*PowerShell 版本\*\*：如果是 PowerShell 7+，需确保以管理员身份启动对应的终端（PowerShell 7 或 Windows Terminal 运行 `pwsh`）。

按上述步骤调整执行策略后，脚本就能正常运行啦，本质是 Windows 的安全机制在起作用，调整后即可解除限制~

# 豆包入门教程

### PowerShell ISE 教程：入门与进阶

PowerShell ISE（Integrated Scripting Environment）是微软为 PowerShell 提供的官方集成开发环境，专为脚本编写和调试设计。本教程将带你从基础到进阶，全面掌握 PowerShell ISE 的使用。

### \*\*1. 安装与启动\*\*

#### \*\*安装\*\*

- \*\*Windows 10/11\*\*：默认已安装（PowerShell 5.1 及以下版本）。

- \*\*PowerShell 7+\*\*：需从 [Microsoft Store](https://www.microsoft.com/store/productId/9MZ1SNWT0N5D) 或官网下载安装。

#### \*\*启动方式\*\*

- 按下 `Win + R`，输入 `powershell\_ise` 并回车。

- 在开始菜单中搜索 \*\*PowerShell ISE\*\*。

### \*\*2. 界面介绍\*\*

PowerShell ISE 的界面主要由三部分组成：

1. \*\*脚本编辑器\*\*（顶部）：编写和编辑 PowerShell 脚本。

2. \*\*命令提示符\*\*（底部）：实时执行 PowerShell 命令。

3. \*\*输出窗口\*\*（底部）：显示命令执行结果。

![PowerShell ISE 界面](https://i.imgur.com/5JZv4cK.png)

### \*\*3. 基础操作\*\*

#### \*\*3.1 编写第一个脚本\*\*

1. 在脚本编辑器中输入以下代码：

```powershell

# 输出问候语

Write-Host "Hello, PowerShell ISE!"

# 获取当前日期

$date = Get-Date

Write-Host "当前日期: $date"

```

2. 保存脚本（`Ctrl + S`），文件后缀为 `.ps1`（如 `hello.ps1`）。

3. 执行脚本：

- 点击工具栏的 \*\*运行脚本\*\* 按钮（绿色三角形）。

- 按 `F5` 键。

- 在命令提示符中输入 `.\hello.ps1` 并回车。

#### \*\*3.2 命令提示符使用\*\*

- 在命令提示符中输入单行命令，按 `Enter` 执行。

```powershell

Get-Process # 获取当前运行的进程

Get-Service # 获取系统服务

```

- 使用 \*\*向上箭头\*\* 键快速重复之前的命令。

#### \*\*3.3 自动完成功能\*\*

- 输入命令或参数的前几个字符，按 `Tab` 键自动补全。

```powershell

Get-Pro # 按 Tab 自动补全为 Get-Process

```

### \*\*4. 调试脚本\*\*

PowerShell ISE 提供强大的调试功能：

#### \*\*4.1 设置断点\*\*

- 在代码行号旁边点击，或按 `F9`，设置/取消断点。

```powershell

# 在这行设置断点

$result = 1 + 2

Write-Host "结果: $result"

```

#### \*\*4.2 调试命令\*\*

- `F5`：开始调试。

- `F10`：逐过程执行（跳过函数内部）。

- `F11`：逐语句执行（进入函数内部）。

- `Shift + F11`：跳出当前函数。

- `Ctrl + Shift + F5`：重启调试。

### \*\*5. 高级功能\*\*

#### \*\*5.1 脚本分析工具\*\*

- 点击工具栏的 \*\*分析脚本\*\* 按钮（放大镜图标），检查代码中的潜在问题。

#### \*\*5.2 多文件编辑\*\*

- 使用 \*\*新建选项卡\*\*（`Ctrl + T`）同时编辑多个脚本文件。

#### \*\*5.3 自定义布局\*\*

- 通过 \*\*视图\*\* 菜单调整界面布局，例如显示/隐藏侧边栏。

### \*\*6. 实用技巧\*\*

#### \*\*6.1 快速注释/取消注释\*\*

- 选中代码，按 `Ctrl + K + C` 注释。

- 选中代码，按 `Ctrl + K + U` 取消注释。

#### \*\*6.2 从命令历史复制\*\*

- 右键点击命令提示符中的历史命令，选择 \*\*复制\*\*。

#### \*\*6.3 导入模块\*\*

- 在脚本中使用 `Import-Module` 加载额外功能：

```powershell

Import-Module ActiveDirectory # 导入 Active Directory 模块

```

### \*\*7. 常见问题\*\*

#### \*\*7.1 执行策略限制\*\*

- 错误信息：`无法加载文件，因为在此系统上禁止运行脚本`。

- 解决方案：以管理员身份运行 PowerShell ISE，执行：

```powershell

Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser

```

#### \*\*7.2 中文乱码\*\*

- 解决方案：在脚本开头添加编码设置：

```powershell

[Console]::OutputEncoding = [System.Text.Encoding]::UTF8

```

### \*\*8. 进阶学习资源\*\*

- \*\*官方文档\*\*：[PowerShell 官方文档](https://docs.microsoft.com/zh-cn/powershell/)

- \*\*在线教程\*\*：[PowerShell 入门教程](https://www.learnpwsh.com/)

- \*\*社区\*\*：[PowerShell.org](https://powershell.org/)

### \*\*总结\*\*

PowerShell ISE 是学习和使用 PowerShell 的理想工具，通过本教程，你已掌握：

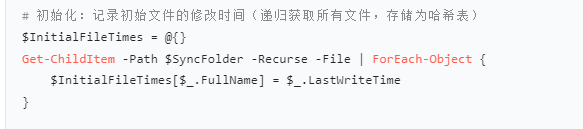
- 基本操作、脚本编写与执行。

- 调试技巧与断点设置。

- 常用快捷键和高级功能。

继续实践，你将能够高效管理系统、自动化任务，并充分发挥 PowerShell 的强大功能！

# 检测云盘同步脚本解释



这段 PowerShell 代码的作用是\*\*递归遍历指定目录（`$SyncFolder` ）下的所有文件\*\*，并记录每个文件的\*\*完整路径\*\*和对应的\*\*最后修改时间\*\*，存储到哈希表 `$InitialFileTimes` 中，用于后续检测文件是否被修改（比如云盘同步后的变化）。以下逐行详细解释：

---

### 1. 整体功能

```powershell

# 初始化：记录初始文件的修改时间（递归获取所有文件，存储为哈希表）

$InitialFileTimes = @{}

Get-ChildItem -Path $SyncFolder -Recurse -File | ForEach-Object {

$InitialFileTimes[$\_.FullName] = $\_.LastWriteTime

}

```

- 核心目标：\*\*为后续对比文件是否被修改，保存“初始状态”\*\*（记录同步目录中所有文件的路径和最后修改时间）。

- 适用场景：常配合“云盘同步检测”“文件变化监控”等需求，判断同步完成后文件是否有更新。

---

### 2. 逐行拆解

#### （1）`$InitialFileTimes = @{}`

- \*\*作用\*\*：创建一个\*\*空的哈希表\*\*（类似字典、键值对集合），用于存储文件路径（键）和最后修改时间（值）。

- 哈希表优势：通过 `键（文件路径）` 快速查找 `值（修改时间）`，比数组遍历更高效。

#### （2）`Get-ChildItem -Path $SyncFolder -Recurse -File`

- \*\*`Get-ChildItem`\*\*：PowerShell 中用于\*\*获取目录/文件列表\*\*的核心命令，类似 `dir`（CMD）或 `ls`（Linux）。

- \*\*`-Path $SyncFolder`\*\*：指定要遍历的\*\*目标目录\*\*（`$SyncFolder` 是变量，需提前定义，比如 `$SyncFolder = "D:\CloudSync"` ）。

- \*\*`-Recurse`\*\*：开启\*\*递归遍历\*\*，会深入所有子目录，获取嵌套文件夹中的文件。

- \*\*`-File`\*\*：只筛选\*\*文件\*\*，排除文件夹（若不加 `-File` ，会同时返回文件和文件夹）。

#### （3）`| ForEach-Object { ... }`

- \*\*`|`\*\*：管道符，将前面命令（`Get-ChildItem` ）的输出，传递给后面的 `ForEach-Object` 处理。

- \*\*`ForEach-Object`\*\*：遍历管道中的每个对象（这里是每个文件），执行大括号内的逻辑。

#### （4）`$InitialFileTimes[$\_.FullName] = $\_.LastWriteTime`

- \*\*`$\_`\*\*：代表管道中当前遍历的\*\*文件对象\*\*（每次循环对应一个文件）。

- \*\*`$\_.FullName`\*\*：文件的\*\*完整路径\*\*（如 `D:\CloudSync\Folder\File.txt` ），作为哈希表的\*\*键\*\*。

- \*\*`$\_.LastWriteTime`\*\*：文件的\*\*最后修改时间\*\*（类型是 `DateTime` ），作为哈希表的\*\*值\*\*。

- 效果：将每个文件的路径和修改时间，一一存入 `$InitialFileTimes` 哈希表，后续可通过 `$InitialFileTimes["文件路径"]` 快速获取初始修改时间。

---

### 3. 完整逻辑示例

结合“检测云盘同步后文件变化”的完整场景，代码可能如下：

```powershell

# 配置：云盘本地同步目录

$SyncFolder = "D:\CloudSync"

# 要触发的批处理文件

$BatchFile = "C:\Scripts\YourTask.bat"

# 初始化：记录初始文件的修改时间（递归获取所有文件，存储为哈希表）

$InitialFileTimes = @{}

Get-ChildItem -Path $SyncFolder -Recurse -File | ForEach-Object {

$InitialFileTimes[$\_.FullName] = $\_.LastWriteTime

}

# 监控循环：持续检测文件修改时间变化

while ($true) {

$hasChange = $false # 标记是否检测到变化

# 遍历当前文件，对比修改时间

Get-ChildItem -Path $SyncFolder -Recurse -File | ForEach-Object {

$currentTime = $\_.LastWriteTime

$initialTime = $InitialFileTimes[$\_.FullName]

# 如果当前修改时间 ≠ 初始修改时间，说明文件被修改

if ($currentTime -ne $initialTime) {

$hasChange = $true

return # 找到变化，跳出循环

}

}

# 若检测到变化，触发批处理并退出

if ($hasChange) {

Write-Host "云盘同步完成（文件修改时间变化），触发批处理..."

Start-Process -FilePath $BatchFile -WindowStyle Hidden

break

}

# 每隔 10 秒检查一次（可调整频率）

Start-Sleep -Seconds 10

}

```

---

### 4. 常见扩展与问题

#### （1）扩展：支持“首次同步前目录为空”

若云盘同步目录初始可能为空（`$InitialFileTimes` 为空），需加判断避免报错：

```powershell

if ($InitialFileTimes.Count -gt 0) {

# 执行后续对比逻辑

} else {

Write-Host "初始目录为空，等待同步..."

}

```

#### （2）问题：权限不足导致无法访问文件

若脚本运行时权限不足（比如云盘目录在系统保护路径），`Get-ChildItem` 会报错。解决方法：

- 右键 PowerShell 或脚本，选\*\*以管理员身份运行\*\*。

- 或修改目录权限，赋予当前用户访问权限。

#### （3）优化：减少重复代码

可将“获取文件修改时间”逻辑封装为函数：

```powershell

function Get-FileTimes {

param($Path)

$fileTimes = @{}

Get-ChildItem -Path $Path -Recurse -File | ForEach-Object {

$fileTimes[$\_.FullName] = $\_.LastWriteTime

}

return $fileTimes

}

# 调用函数

$InitialFileTimes = Get-FileTimes -Path $SyncFolder

```

E:\备份盘\带0文件夹\005-计算机科学、程式、资料,硬件\005-400-电脑编程!1\BAT\云端-本地配置-互传-BAT\xcopy文件-云端覆盖本地配置-T0.bat

---

### 总结

这段代码是\*\*文件监控/同步检测\*\*的“初始化环节”，通过哈希表记录文件的初始状态。后续结合循环对比，就能实现“检测文件是否被修改（如同步完成）”的功能，常用于自动化脚本（如云盘同步后触发任务、文件夹变化监控等）。

若要实现更复杂的逻辑（如忽略某些文件、支持增量更新），可在此基础上扩展哈希表的操作和条件判断~

# 复制目录

在PowerShell中，复制目录的命令可以简化为一行，适用于快速操作或脚本中的简洁写法。以下是几种常用的简单复制命令：

### \*\*1. 基本复制命令（递归复制目录及内容）\*\*

```powershell

Copy-Item -Path "源目录" -Destination "目标目录" -Recurse -Force

```

- \*\*说明\*\*：

- `-Recurse`：递归复制子目录。

- `-Force`：覆盖已存在的文件（可选）。

- \*\*示例\*\*：

```powershell

Copy-Item -Path "C:\Source\Folder" -Destination "D:\Backup\" -Recurse -Force

```

### \*\*2. 复制多个目录（使用通配符）\*\*

```powershell

Copy-Item -Path "C:\Source\Folder\*" -Destination "D:\Backup\" -Recurse -Force

```

- \*\*说明\*\*：复制所有以 `Folder` 开头的目录。

### \*\*3. 复制目录内容（不包含源目录本身）\*\*

```powershell

Copy-Item -Path "C:\Source\Folder\\*" -Destination "D:\Backup\" -Recurse -Force

```

- \*\*说明\*\*：仅复制 `Folder` 内的文件和子目录，而非 `Folder` 目录本身。

### \*\*4. 复制多个不连续的目录（用数组）\*\*

```powershell

@("C:\Dir1", "C:\Dir2", "C:\Dir3") | ForEach-Object { Copy-Item -Path $\_ -Destination "D:\Backup\" -Recurse -Force }

```

- \*\*说明\*\*：通过管道批量处理多个目录。

### \*\*5. 简化别名（适用于命令行快速操作）\*\*

```powershell

cp -r "源目录" "目标目录"

```

- \*\*说明\*\*：`cp` 是 `Copy-Item` 的别名，`-r` 等价于 `-Recurse`。

### \*\*注意事项\*\*

1. \*\*路径格式\*\*：

- 使用双引号包裹路径，避免空格或特殊字符导致的问题。

- 目标目录需提前存在，否则会将目标路径视为文件名（除非使用 `-Container` 参数）。

2. \*\*覆盖策略\*\*：

- `-Force` 参数会静默覆盖已存在的文件，谨慎使用。

- 若需交互式确认，可省略 `-Force` 并手动处理冲突。

3. \*\*错误处理\*\*：

简单命令未包含异常捕获，建议在脚本中添加 `try-catch` 或 `-ErrorAction` 参数。

### \*\*完整示例（带错误处理）\*\*

```powershell

try {

Copy-Item -Path "C:\重要文件" -Destination "D:\备份\" -Recurse -Force -ErrorAction Stop

Write-Host "复制完成！" -ForegroundColor Green

} catch {

Write-Error "复制失败：$\_"

}

```

根据你的需求选择合适的命令，建议在脚本中使用完整的 `Copy-Item` 语法以便更好地控制行为。