

# Contents

[Windows 终端](#)

[概述](#)

[入门](#)

[自定义设置](#)

[启动](#)

[交互](#)

[外观](#)

[配色方案](#)

[渲染](#)

[配置文件 - 常规](#)

[配置文件 - 外观](#)

[配置文件 - 高级](#)

[操作](#)

[命令行参数](#)

[命令面板](#)

[搜索](#)

[窗格](#)

[动态配置文件](#)

[Cascadia Code](#)

[提示和技巧](#)

[教程](#)

[Powerline 安装](#)

[SSH](#)

[设置选项卡标题](#)

[自定义终端库](#)

[自定义终端指南](#)

[PowerShell 中的 Powerline](#)

[Raspberry Ubuntu](#)

[毛玻璃主题](#)

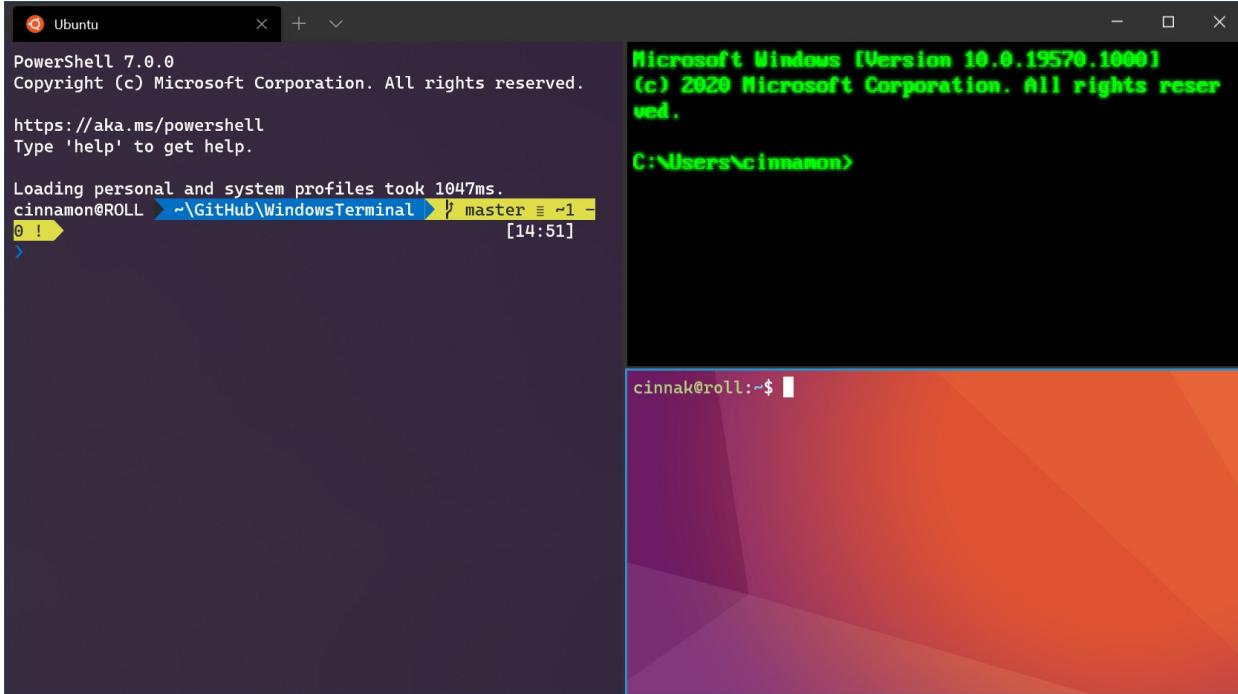
怀旧式命令提示符

疑难解答

# 什么是 Windows 终端？

2021/2/1 ·

Windows 终端是一个面向命令行工具和 shell(如命令提示符、PowerShell 和适用于 Linux 的 Windows 子系统 (WSL))用户的新式终端应用程序。它的主要功能包括多个选项卡、窗格、Unicode 和 UTF-8 字符支持、GPU 加速文本呈现引擎，你还可用它来创建你自己的主题并自定义文本、颜色、背景和快捷方式。



## NOTE

控制台、终端和 shell 之间有何区别？阅读 Scott Hanselman 的解释。

## 支持多种命令行应用程序的多个配置文件

任何具有命令行接口的应用程序都可以在 Windows 终端中运行。这包括从 PowerShell 和命令提示符到 Azure Cloud Shell 和任何 WSL 分发(如 Ubuntu 或 Oh-My-Zsh)的所有应用程序。

## 自定义方案和配置

可以将 Windows 终端配置为具有多种配色方案和设置。若要了解如何创建自己的配色方案，请访问[配色方案页面](#)。还可以在[自定义终端库](#)中查找自定义终端配置。

## 自定义操作

可在 Windows 终端中使用多种自定义命令，获得更加自然的体验。如果不喜欢特定的键盘快捷方式，可以将其更改为你喜欢的方式。

例如，若要复制命令行中的文本，默认的快捷方式为 `ctrl+shift+c`。你可以将其更改为 `ctrl+1` 或你喜欢的其他方式。要打开新的选项卡，默认快捷方式是 `ctrl+t`，但你可能想要将其更改为 `ctrl+2`。用于在打开的选项卡之间进行切换的默认快捷方式是 `ctrl+tab`，可以将这个快捷方式更改为 `ctrl+-` 并改为用于创建新选项卡。

可在[操作页面](#)上了解如何自定义快捷方式。

## Unicode 和 UTF-8 字符支持

Windows 终端可以显示 Unicode 和 UTF-8 字符，如各种语言的表情符号和字符。

## GPU 加速文本呈现

Windows 终端使用 GPU 来呈现其文本，从而提供比默认 Windows 命令行体验更好的性能。

## 背景图像支持

可以在 Windows 终端窗口中显示背景图像和 gif。有关如何向配置文件添加背景图像的信息，请参阅[“配置文件 - 外观”页](#)。

## 命令行参数

可以使用命令行参数将 Windows 终端设置为在特定配置中启动。可以指定要在新选项卡中打开哪个配置文件、应选择哪个文件夹目录，指定使用拆分窗口窗格打开终端，并指定选择应专注于哪个选项卡。

例如，若要使用三个窗格从 PowerShell 打开 Windows 终端（左窗格运行命令提示符配置文件，右窗格拆分为两个，一个用于 PowerShell，另一个用于运行 WSL 的默认配置文件），请输入：

```
wt -p "Command Prompt" `; split-pane -p "Windows PowerShell" `; split-pane -H wsl.exe
```

了解如何在[命令行参数页](#)上设置命令行参数。

# 安装和设置 Windows 终端

2021/2/1 ·

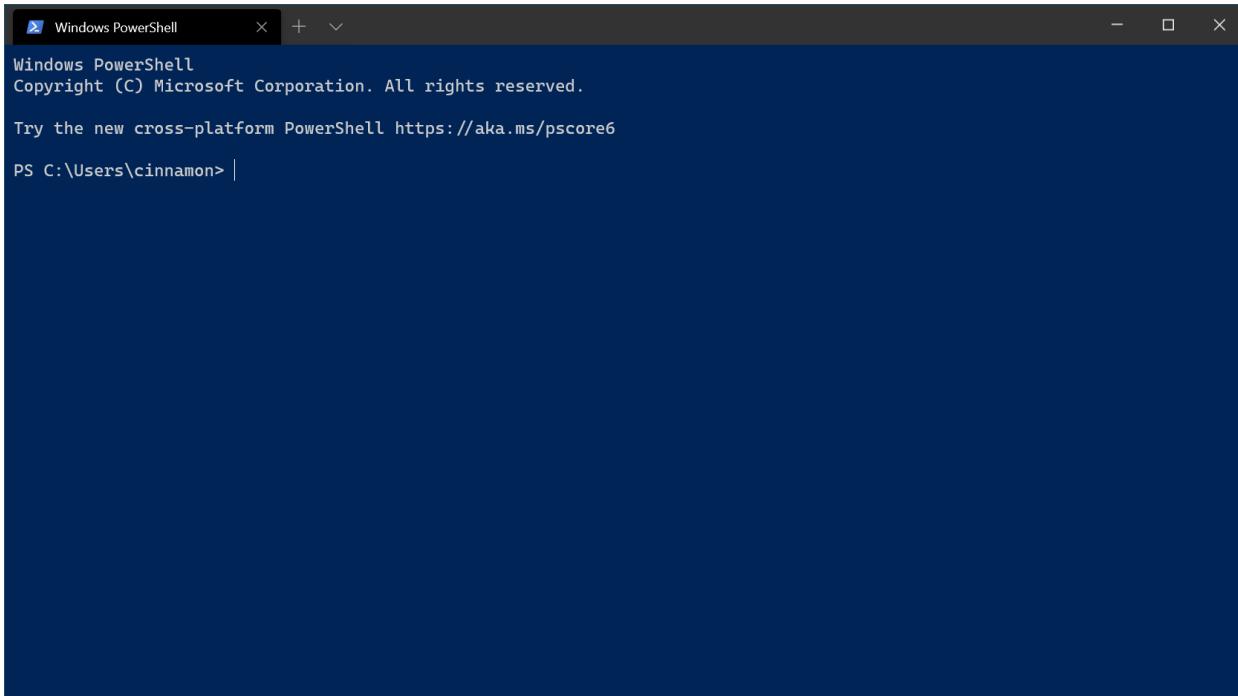
## 安装

可以从 [Microsoft Store](#) 安装 Windows 终端。

如果你无法访问 Microsoft Store, [GitHub 发布页](#) 上发布有内部版本。如果从 GitHub 安装, 终端将不会自动更新为新版本。

## 首次运行

安装后打开终端时, 它会在打开的选项卡中通过 PowerShell 作为默认配置文件启动。



### 动态配置文件

如果已安装 WSL 发行版或多个版本的 PowerShell, 终端将自动为你创建配置文件。详细了解[动态配置文件页](#)上的动态配置文件。

## 打开新选项卡

可以按 **ctrl+shift+t** 或选择 + (加号) 按钮, 打开默认配置文件的新选项卡。若要打开其他配置文件, 请选择 + 按钮旁的 ▾ (箭头) 打开下拉菜单。然后可以从中选择要打开的配置文件。

## 打开新窗格

可以使用窗格并行运行多个 shell。若要打开窗格, 可以使用 **alt+shift+d**。此键绑定将打开焦点配置文件的重複窗格。详细了解[窗格页](#)上的窗格。

## 配置

若要自定义 Windows 终端的设置, 请在下拉菜单中选择“设置”。这会在默认文本编辑器中打开 `settings.json` 文件。(默认文本编辑器在 [Windows 设置](#) 中定义。)

终端支持自定义影响整个应用程序的全局属性、影响每个配置文件的设置的配置文件属性以及允许你使用键盘与终端交互的键绑定。

## 命令行参数

可以使用命令行参数在特定配置中启动终端。这些参数允许通过自定义配置文件设置打开具有特定选项卡和窗格的终端。详细了解[命令行参数页](#)上的命令行参数。

## 疑难解答

如果使用终端时遇到任何问题，请参考[故障排除页](#)。如果发现任何 bug 或要提出功能请求，可以选择终端的“关于”菜单中的反馈链接转到[GitHub 页](#)，可以在其中提出新问题。

# Windows 终端中的配色方案

2021/2/1 ·

## 创建自己的配色方案

可以在 `settings.json` 文件的 `schemes` 数组中定义配色方案。它们是使用以下格式写入的：

```
{  
  "name" : "Campbell",  
  
  "cursorColor": "#FFFFFF",  
  "selectionBackground": "#FFFFFF",  
  
  "background" : "#0C0C0C",  
  "foreground" : "#CCCCCC",  
  
  "black" : "#0C0C0C",  
  "blue" : "#0037DA",  
  "cyan" : "#3A96DD",  
  "green" : "#13A10E",  
  "purple" : "#881798",  
  "red" : "#C50F1F",  
  "white" : "#CCCCCC",  
  "yellow" : "#C19C08",  
  "brightBlack" : "#767676",  
  "brightBlue" : "#3B78FF",  
  "brightCyan" : "#61D6D6",  
  "brightGreen" : "#16C60C",  
  "brightPurple" : "#B4009E",  
  "brightRed" : "#E74856",  
  "brightWhite" : "#F2F2F2",  
  "brightYellow" : "#F9F1A5"  
},
```

除 `name` 以外，每个设置都接受十六进制格式（`"#rgb"` 或 `"#rrggbb"`）的字符串形式的颜色。`cursorColor` 和 `selectionBackground` 设置是可选的。

## 包含的配色方案

Windows 终端将这些配色方案包含在 `defaults.json` 文件中，可按住 `alt` 并选择设置按钮来访问该文件。如果要在命令行配置文件中设置配色方案，请添加 `colorScheme` 属性，并将配色方案的 `name` 作为值。

```
"colorScheme": "COLOR SCHEME NAME"
```

**Campbell**

Ubuntu

Background | Foreground colors

```
ESC[40m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[40m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[41m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[41m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[42m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[42m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[43m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[43m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[44m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[44m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[45m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[45m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[46m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[46m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[47m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[47m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m
```

cinnak@roll:~\$ █

## Campbell Powershell

Ubuntu

Background | Foreground colors

```
ESC[40m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[40m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[41m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[41m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[42m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[42m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[43m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[43m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[44m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[44m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[45m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[45m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[46m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[46m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[47m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[47m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m
```

cinnak@roll:~\$ █

## Vintage

```
Ubuntu      + - ×
Background | Foreground colors
-----
ESC[40m | [31m [32m [33m [34m [35m [36m [37m
ESC[40m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[41m | [30m [32m [33m [34m [35m [36m [37m
ESC[41m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[42m | [30m [31m [33m [34m [35m [36m [37m
ESC[42m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[43m | [30m [31m [32m [34m [35m [36m [37m
ESC[43m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[44m | [30m [31m [32m [33m [35m [36m [37m
ESC[44m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[45m | [30m [31m [32m [33m [34m [36m [37m
ESC[45m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[46m | [30m [31m [32m [33m [34m [35m [37m
ESC[46m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[47m | [30m [31m [32m [33m [34m [35m [36m
ESC[47m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

cinnak@roll:~$ █
```

## One Half Dark

```
Ubuntu      + - ×
Background | Foreground colors
-----
ESC[40m | [31m [32m [33m [34m [35m [36m [37m
ESC[40m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[41m | [30m [32m [33m [34m [35m [36m [37m
ESC[41m | [1;30m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[42m | [30m [31m [33m [34m [35m [36m [37m
ESC[42m | [1;30m [1;31m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[43m | [30m [31m [32m [34m [35m [36m [37m
ESC[43m | [1;30m [1;31m [1;32m [1;34m [1;35m [1;36m [1;37m

ESC[44m | [30m [31m [32m [33m [35m [36m [37m
ESC[44m | [1;30m [1;31m [1;32m [1;33m [1;35m [1;36m [1;37m

ESC[45m | [30m [31m [32m [33m [34m [36m [37m
ESC[45m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;36m [1;37m

ESC[46m | [30m [31m [32m [33m [34m [35m [37m
ESC[46m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[47m | [30m [31m [32m [33m [34m [35m [36m
ESC[47m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

cinnak@roll:~$ █
```

## One Half Light

Ubuntu

Background | Foreground colors

```
ESC[40m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[40m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m

ESC[41m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[41m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[42m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[42m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[43m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[43m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[44m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[44m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[45m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[45m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[46m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[46m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[47m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[47m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m
```

cinnak@roll:~\$ █

## Solarized Dark

Ubuntu

Background | Foreground colors

```
ESC[40m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[40m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[41m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[41m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[42m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[42m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[43m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[43m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[44m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[44m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[45m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[45m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[46m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[46m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[47m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[47m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m
```

cinnak@roll:~\$ █

## Solarized Light

Ubuntu

Background | Foreground colors

```
ESC[40m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[40m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m

ESC[41m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[41m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[42m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[42m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[43m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[43m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[44m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[44m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[45m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[45m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[46m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[46m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[47m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[47m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m
```

cinnak@roll:~\$ █

## Tango Dark

Ubuntu

Background | Foreground colors

```
ESC[40m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[40m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[41m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[41m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[42m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[42m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[43m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[43m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[44m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[44m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[45m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[45m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[46m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[46m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[47m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[47m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m
```

cinnak@roll:/mnt/c/Users/cinnamon\$ █

## Tango Light

Ubuntu

Background | Foreground colors

---

ESC[40m	[30m	[31m	[32m	[33m	[34m	[35m	[36m	[37m
ESC[40m	[1;30m	[1;31m	[1;32m	[1;33m	[1;34m	[1;35m	[1;36m	[1;37m
ESC[41m	[30m	[32m	[33m	[34m	[35m	[36m	[37m	
ESC[41m	[1;30m	[1;31m	[1;32m	[1;33m	[1;34m	[1;35m	[1;36m	[1;37m
ESC[42m	[30m	[31m	[33m	[34m	[35m	[36m	[37m	
ESC[42m	[1;30m	[1;31m	[1;32m	[1;33m	[1;34m	[1;35m	[1;36m	[1;37m
ESC[43m	[30m	[31m	[32m	[34m	[35m	[36m	[37m	
ESC[43m	[1;30m	[1;31m	[1;32m	[1;33m	[1;34m	[1;35m	[1;36m	[1;37m
ESC[44m	[30m	[31m	[32m	[33m	[35m	[36m	[37m	
ESC[44m	[1;30m	[1;31m	[1;32m	[1;33m	[1;34m	[1;35m	[1;36m	[1;37m
ESC[45m	[30m	[31m	[32m	[33m	[34m	[36m	[37m	
ESC[45m	[1;30m	[1;31m	[1;32m	[1;33m	[1;34m	[1;35m	[1;36m	[1;37m
ESC[46m	[30m	[31m	[32m	[33m	[34m	[35m	[37m	
ESC[46m	[1;30m	[1;31m	[1;32m	[1;33m	[1;34m	[1;35m	[1;36m	[1;37m
ESC[47m	[30m	[31m	[32m	[33m	[34m	[35m	[36m	[37m
ESC[47m	[1;30m	[1;31m	[1;32m	[1;33m	[1;34m	[1;35m	[1;36m	[1;37m

---

cinnak@roll:/mnt/c/Users/cinnamon\$ █

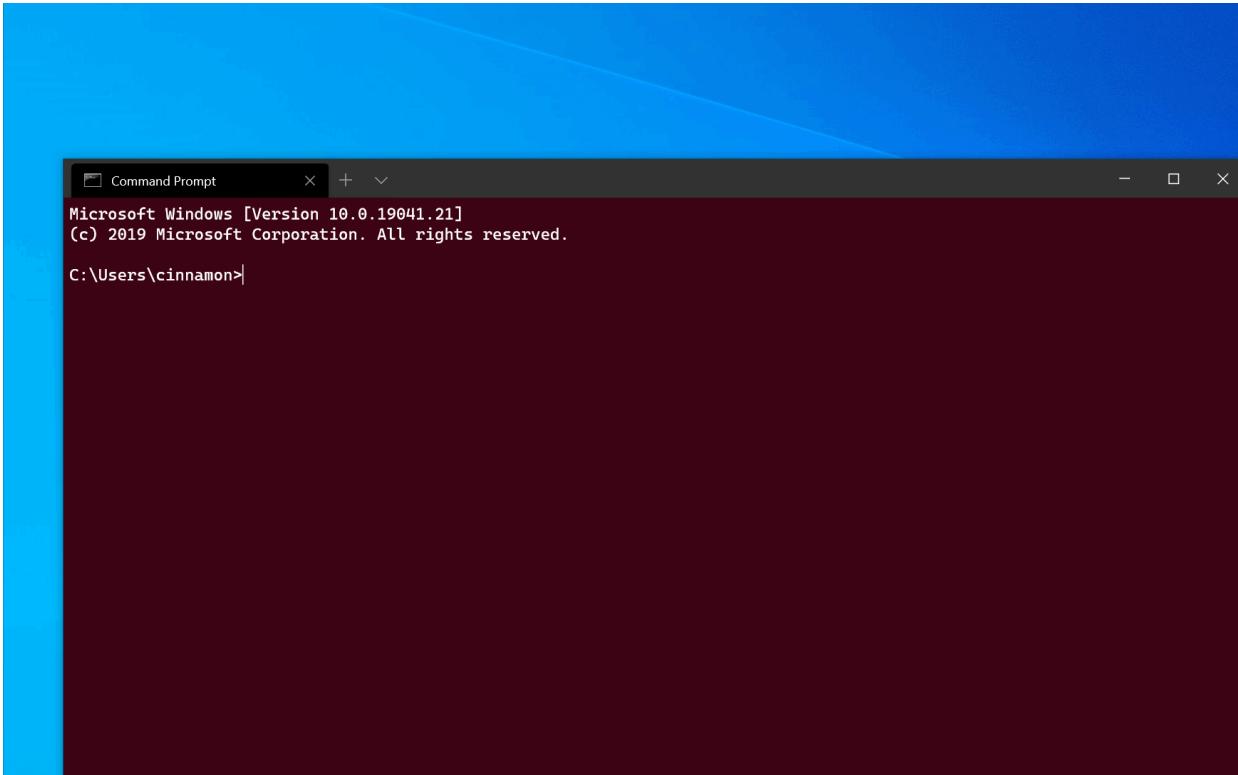
# 使用 Windows 终端的命令行参数

2021/2/1 •

可以使用 `wt.exe` 从命令行打开 Windows 终端的新实例。还可以改为使用执行别名 `wt`。

## NOTE

如果从 [GitHub](#) 上的源代码生成了 Windows 终端，则可以使用 `wtd.exe` 或 `wtd` 打开该生成。



## 命令行语法

`wt` 命令行接受两种类型的值：“选项”和“命令”。“选项”是一系列标志和其他参数，可以将 `wt` 命令行的行为作为一个整体来控制。“命令”提供应该实现的操作或操作列表（以分号分隔）。如果未指定命令，则默认情况下会将命令指定为 `new-tab`。

```
wt [options] [command ; ]
```

若要显示列出可用命令行参数的帮助消息，请输入：`wt -h`、`wt --help`、`wt -?` 或 `wt /?`。

## 选项和命令

下面是 `wt` 命令行支持的命令和选项的完整列表。

命令	描述
<code>--help</code> 、 <code>-h</code> 、 <code>-?</code> 、 <code>/?</code>	显示帮助消息。

--maximized、-M	以最大化形式启动终端。
--fullscreen、-F	以全屏形式启动终端。

### IMPORTANT

--maximized、-M 和 --fullscreen、-F 仅在 Windows 终端预览中可用。

new-tab	--profile, -p profile-name、 --startingDirectory, -d starting-directory 、commandline、--title	创建新选项卡。
split-pane	-H, --horizontal、 -V, --vertical、 --profile, -p profile-name、 --startingDirectory, -d starting-directory 、commandline、--title	拆分新窗格。
focus-tab	--target, -t tab-index	聚焦于特定选项卡。

### IMPORTANT

--title 仅在 Windows 终端预览中可用。

## 命令行参数示例

命令可能会略有不同，具体取决于所使用的命令行。

### 打开新的配置文件实例

若要打开新的终端实例(在此示例中，该命令将打开名为“Ubuntu-18.04”的配置文件)，请输入：

- 命令提示符
- PowerShell
- Linux

```
wt -p "Ubuntu-18.04"
```

-p 标志用于指定应打开的 Windows 终端配置文件。将“Ubuntu-18.04”替换为已安装的任何终端配置文件的名称。这将始终打开一个新窗口。Windows 终端尚不能在现有实例中打开新选项卡或窗格。

### 以一个目录为目标

若要指定应该用作控制台起始目录的文件夹(在本例中为 d:\目录)，请输入：

- 命令提示符
- PowerShell
- Linux

```
wt -d d:\
```

## 多个选项卡

若要打开具有多个选项卡的新终端实例，请输入：

- 命令提示符
- PowerShell
- Linux

```
wt ; ;
```

若要打开具有多个选项卡的新终端实例（在本例中为命令提示符配置文件和 PowerShell 配置文件），请输入：

- 命令提示符
- PowerShell
- Linux

```
wt -p "Command Prompt" ; new-tab -p "Windows PowerShell"
```

## 多个窗格

若要使用一个选项卡打开一个包含三个窗格（分别运行命令提示符配置文件、PowerShell 配置文件以及运行 WSL 命令行的默认配置文件）的新终端实例，请输入：

- 命令提示符
- PowerShell
- Linux

```
wt -p "Command Prompt" ; split-pane -p "Windows PowerShell" ; split-pane -H wsl.exe
```

`-H` 标志（或 `--horizontal`）指示你希望水平拆分窗格。`-V` 标志（或 `--vertical`）指示你希望垂直拆分窗格。

## 选项卡标题（预览）

若要打开带有自定义选项卡标题的新终端实例，可使用 `--title` 参数。若要在打开两个选项卡时设置每个选项卡的标题，请输入：

- 命令提示符
- PowerShell
- Linux

```
wt --title tabname1 ; new-tab -p "Ubuntu-18.04" --title tabname2
```

### IMPORTANT

此功能仅在 Windows 终端预览中可用。

## 选项卡焦点

若要打开带有特定焦点选项卡的新终端实例，请使用 `-t` 标志（或 `--target`）以及选项卡-索引号。若要在第一个选项卡中打开默认配置文件，并在第二个选项卡（`-t 1`）中打开焦点“Ubuntu-18.04”配置文件，请输入：

- 命令提示符
- PowerShell
- Linux

```
wt ; new-tab -p "Ubuntu-18.04" ; focus-tab -t 1
```

## PowerShell 中多个命令的示例

Windows 终端使用分号字符 `;` 作为分隔符来分隔 `wt` 命令行中的命令。遗憾的是，PowerShell 也使用 `;` 作为命令分隔符。若要解决此问题，可以使用以下技巧从 PowerShell 运行多个 `wt` 命令。在下面的所有示例中，将创建一个新的终端窗口，其中包含三个窗格：一个运行命令提示符，一个运行 PowerShell，最后一个运行 WSL。

下面的示例使用 `Start-Process` 命令来运行 `wt`。如需了解终端为什么使用 `Start-Process`，请参阅下面的[使用 start](#)。

### 单引号括起来的参数

在此示例中，`wt` 参数以单引号 (`'`) 括起来。如果不计算任何内容，则此语法非常有用。

```
start wt 'new-tab "cmd" ; split-pane -p "Windows PowerShell" ; split-pane -H wsl.exe'
```

### 转义的引号

将变量中包含的值传递到 `wt` 命令行时，请使用以下语法：

```
$ThirdPane = "wsl.exe"  
start wt "new-tab cmd ; split-pane -p `"$Windows PowerShell`" ; split-pane -H $ThirdPane"
```

请注意，使用 `\`` 将 `-p` 参数中的“Windows PowerShell”的双引号 (`"`) 转义给 `split-pane` 参数。

### 使用 `start`

上述所有示例显式使用 `start` 启动终端。

下面的示例不使用 `start` 运行命令行。但是，可以通过另外两种方法来对命令行转义：

- 仅对分号进行转义，使 `PowerShell` 忽略它们，并将它们直接传递到 `wt`。
- 使用 `--%`，因此 PowerShell 会将命令行的其余部分视为应用程序的参数。

```
wt new-tab "cmd" `; split-pane -p "Windows PowerShell" `; split-pane -H wsl.exe
```

```
wt --% new-tab cmd ; split-pane -p "Windows PowerShell" ; split-pane -H wsl.exe
```

在这两个示例中，新创建的 Windows 终端窗口将通过正确分析所有提供的命令行参数来创建窗口。

但是目前不建议使用这些方法，因为 PowerShell 会等待新创建的终端窗口关闭，然后再将控制权返回给 PowerShell。默认情况下，在返回到提示符之前，PowerShell 将始终等待 Windows 应用商店应用程序（如 Windows 终端）关闭。请注意，这与命令提示符的行为不同，后者会立即返回到提示符。

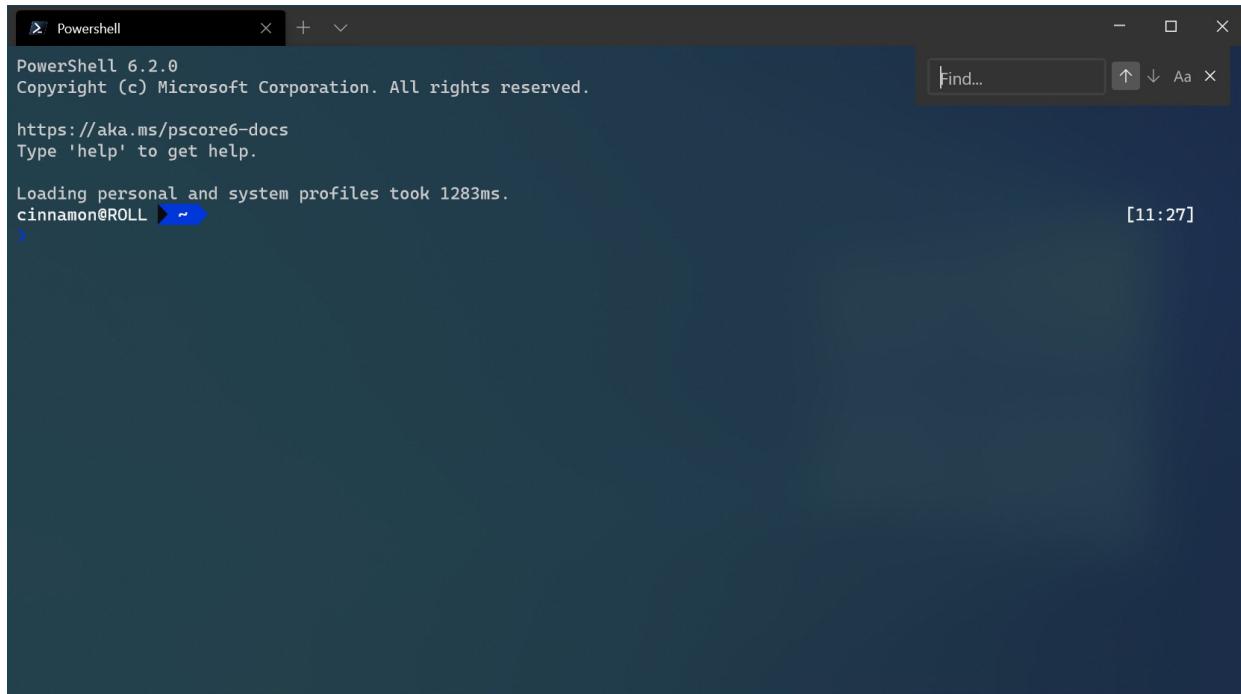
# 如何在 Windows 终端中进行搜索

2021/2/1 ·

Windows 终端附带一项搜索功能，可用于在文本缓冲区中查找特定关键字。当尝试查找之前运行的命令或特定文件名时，这非常有用。

## 使用搜索

默认情况下，可以通过键入 `ctrl+shift+f` 打开搜索对话框。打开后，可以在文本框中键入要查找的关键字，并点击进行搜索。



配置: [PowerShell 中的 Powerline](#)

## 方向搜索

终端会默认从文本缓冲区的底部到顶部进行搜索。可以在搜索对话框中选择一个箭头来更改搜索方向。

A screenshot of a Windows terminal window titled "Powershell". The search bar at the top contains the text "authoring". The search results show a list of files and folders in the directory "C:\Users\cinnamon\GitHub\terminal-docs\TerminalDocs". The results are identical to the ones shown in the previous screenshot, listing files like "panes.md", "search.md", and "TOC.yml".

```
-a---- 3/18/2020 10:01 AM      303 panes.md
-a---- 3/18/2020 11:30 AM     1947 search.md
-a---- 3/18/2020 10:10 AM    1138 TOC.yml
-a---- 3/18/2020 10:08 AM     358 troubleshooting.md

cinnamon@ROLL ~\GitHub\terminal-docs\TerminalDocs > authoring +1 ~1 -0 !
[11:34]

> ls

Directory: C:\Users\cinnamon\GitHub\terminal-docs\TerminalDocs

Mode           LastWriteTime       Length Name
----           -----          ---- 
d----          3/12/2020  2:07 PM          breadcrumb
d----          3/12/2020  2:13 PM          customize-settings
d----          3/18/2020  11:28 AM          images
d----          3/18/2020  10:10 AM          tutorials
-a---          3/18/2020  10:02 AM        346 background-images.md
-a---          3/12/2020  2:07 PM        1806 command-line-arguments.md
-a---          3/12/2020  2:07 PM        1091 docfx.json
-a---          3/18/2020  10:01 AM        567 get-started.md
-a---          3/12/2020  2:07 PM        2828 index.md
-a---          3/18/2020  10:01 AM        303 panes.md
-a---          3/18/2020  11:30 AM        1947 search.md
-a---          3/18/2020  10:10 AM        1138 TOC.yml
-a---          3/18/2020  10:08 AM        358 troubleshooting.md

cinnamon@ROLL ~\GitHub\terminal-docs\TerminalDocs > authoring +1 ~1 -0 !
[11:34]
```

## 大小写匹配搜索

如果要缩小搜索结果范围，可以在搜索中添加大小写匹配选项。可以选择大小写匹配按钮来切换大小写匹配，而显示的结果将只匹配使用特定字母大小写输入的关键字。

A screenshot of a Windows terminal window titled "Powershell". The search bar at the top contains the text "authoring". The search results show a list of files and folders in the directory "C:\Users\cinnamon\GitHub\terminal-docs\TerminalDocs". The results are identical to the ones shown in the previous screenshots, listing files like "panes.md", "search.md", and "TOC.yml".

```
-a---- 3/18/2020 10:01 AM      303 panes.md
-a---- 3/18/2020 11:36 AM     1983 search.md
-a---- 3/18/2020 10:10 AM    1138 TOC.yml
-a---- 3/18/2020 10:08 AM     358 troubleshooting.md

cinnamon@ROLL ~\GitHub\terminal-docs\TerminalDocs > authoring +2 ~1 -0 !
[11:43]

> ls

Directory: C:\Users\cinnamon\GitHub\terminal-docs\TerminalDocs

Mode           LastWriteTime       Length Name
----           -----          ---- 
d----          3/12/2020  2:07 PM          breadcrumb
d----          3/12/2020  2:13 PM          customize-settings
d----          3/18/2020  11:36 AM          images
d----          3/18/2020  10:10 AM          tutorials
-a---          3/18/2020  10:02 AM        346 background-images.md
-a---          3/12/2020  2:07 PM        1806 command-line-arguments.md
-a---          3/12/2020  2:07 PM        1091 docfx.json
-a---          3/18/2020  10:01 AM        567 get-started.md
-a---          3/12/2020  2:07 PM        2828 index.md
-a---          3/18/2020  10:01 AM        303 panes.md
-a---          3/18/2020  11:36 AM        1983 search.md
-a---          3/18/2020  10:10 AM        1138 TOC.yml
-a---          3/18/2020  10:08 AM        358 troubleshooting.md

cinnamon@ROLL ~\GitHub\terminal-docs\TerminalDocs > authoring +2 ~1 -0 !
[11:43]
```

## 在窗格中搜索

搜索对话框也适用于窗格。当聚焦于窗格时，可以打开搜索对话框，而且对话框将显示在该窗格的右上角。然后输入任何关键字后，将只显示该窗格中找到的结果。

```

Directory: C:\Users\cinnamon\GitHub\terminal-docs\TerminalDocs
Mode LastWriteTime Length Name
---- 
d---- 3/12/2020 2:07 PM breadcrumb
d---- 3/12/2020 2:13 PM customize-settings
d---- 3/18/2020 11:45 AM images
d---- 3/18/2020 10:18 AM tutorials
-a--- 3/18/2020 10:02 AM 346 background-images.md
-a--- 3/12/2020 2:07 PM 1886 command-line-arguments.md
-a--- 3/12/2020 2:07 PM 1091 docfx.json
-a--- 3/18/2020 10:01 AM 567 get-started.md
-a--- 3/12/2020 2:07 PM 2828 index.md
-a--- 3/18/2020 10:01 AM 303 panes.md
-a--- 3/18/2020 11:46 AM 2042 search.md
-a--- 3/18/2020 10:18 AM 1138 TOC.yml
-a--- 3/18/2020 10:08 AM 358 troubleshooting.md

cinnamon@ROLL ~\GitHub\terminal-docs\TerminalDocs > /> authoring ± +3 -1 -0 [11:56]
> ls

Directory: C:\Users\cinnamon\GitHub\terminal-docs\TerminalDocs
Mode LastWriteTime Length Name
---- 
d---- 3/12/2020 2:07 PM breadcrumb
d---- 3/12/2020 2:13 PM customize-settings
d---- 3/18/2020 11:45 AM images
d---- 3/18/2020 10:18 AM tutorials
-a--- 3/18/2020 10:02 AM 346 background-images.md
-a--- 3/12/2020 2:07 PM 1886 command-line-arguments.md
-a--- 3/12/2020 2:07 PM 1091 docfx.json
-a--- 3/18/2020 10:01 AM 567 get-started.md
-a--- 3/12/2020 2:07 PM 2828 index.md
-a--- 3/18/2020 10:01 AM 303 panes.md
-a--- 3/18/2020 11:46 AM 2042 search.md
-a--- 3/18/2020 10:18 AM 1138 TOC.yml
-a--- 3/18/2020 10:08 AM 358 troubleshooting.md

cinnamon@ROLL ~\GitHub\terminal-docs\TerminalDocs > /> authoring ± +3 -1 -0 [11:56]
> ls

Directory: C:\Users\cinnamon\GitHub\terminal-docs\TerminalDocs
Mode LastWriteTime Length Name
---- 
d---- 3/12/2020 2:07 PM breadcrumb
d---- 3/12/2020 2:13 PM customize-settings
d---- 3/18/2020 11:45 AM images
d---- 3/18/2020 10:18 AM tutorials
-a--- 3/18/2020 10:02 AM 346 background-images.md
-a--- 3/12/2020 2:07 PM 1886 command-line-arguments.md
-a--- 3/12/2020 2:07 PM 1091 docfx.json
-a--- 3/18/2020 10:01 AM 567 get-started.md
-a--- 3/12/2020 2:07 PM 2828 index.md
-a--- 3/18/2020 10:01 AM 303 panes.md
-a--- 3/18/2020 11:46 AM 2042 search.md
-a--- 3/18/2020 10:18 AM 1138 TOC.yml
-a--- 3/18/2020 10:08 AM 358 troubleshooting.md

cinnamon@ROLL ~\GitHub\terminal-docs\TerminalDocs > /> authoring ± +3 -1 -0 [11:56]
>

```

## 自定义搜索键绑定

可以使用你喜欢的任何键绑定(快捷键组合)打开搜索对话框。若要更改搜索键绑定, 请打开 `settings.json` 文件, 然后搜索 `find` 命令。默认情况下, 此命令设置为 `ctrl+shift+f`。

```
// Press ctrl+shift+f to open the search box
{ "command": "find", "keys": "ctrl+shift+f" },
```

例如, 可以将“`ctrl+shift+f`”改为“`ctrl+f`”, 因此在键入 `ctrl+f` 时, 将打开搜索对话框。

若要详细了解键绑定, 请访问[键绑定页](#)。

# Windows 终端中的窗格

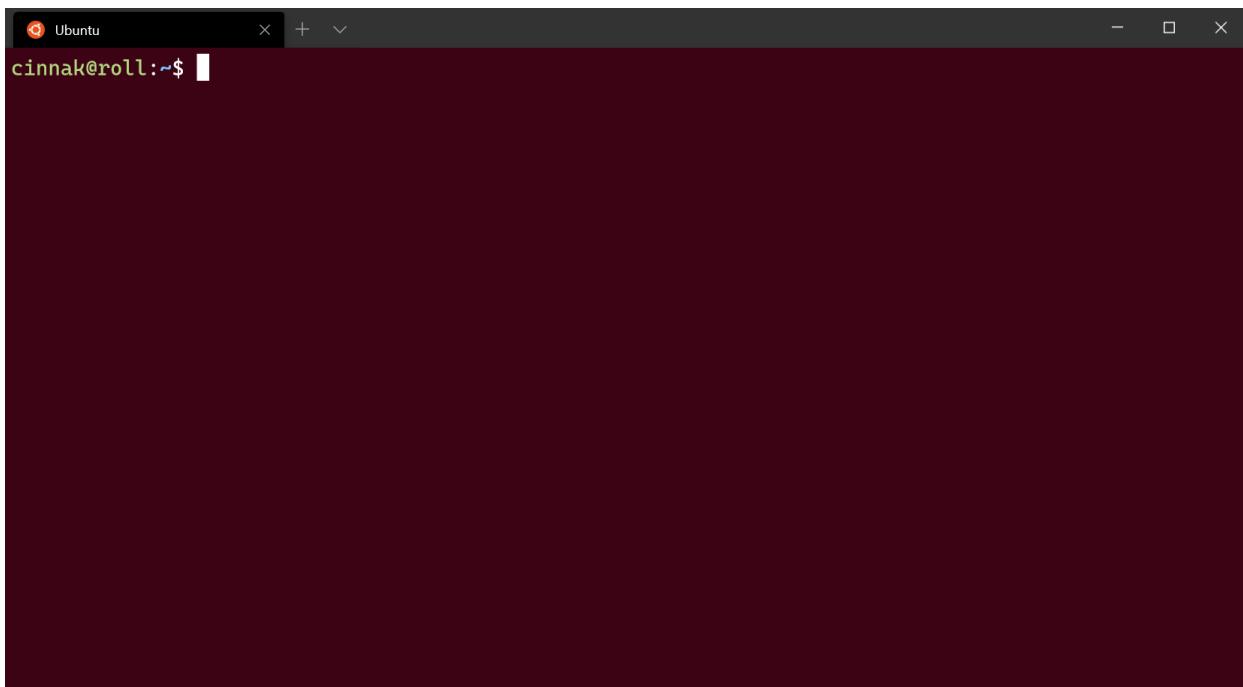
2021/2/1 •

通过窗格，你可以在同一个选项卡中并行运行多个命令行应用程序。这可以最大程度地减少在选项卡之间切换的需求，以便你一次查看多个提示符。

## 创建新窗格

### 使用键盘

可以在 Windows 终端中创建新的垂直或水平窗格。垂直拆分将在焦点窗格的右侧打开一个新窗格，而水平拆分将在焦点窗格下方打开一个新窗格。若要创建默认配置文件的新垂直窗格，可以键入 `alt+shift+plus`。若要创建默认配置文件的新水平窗格，可以键入 `alt+shift+-`。



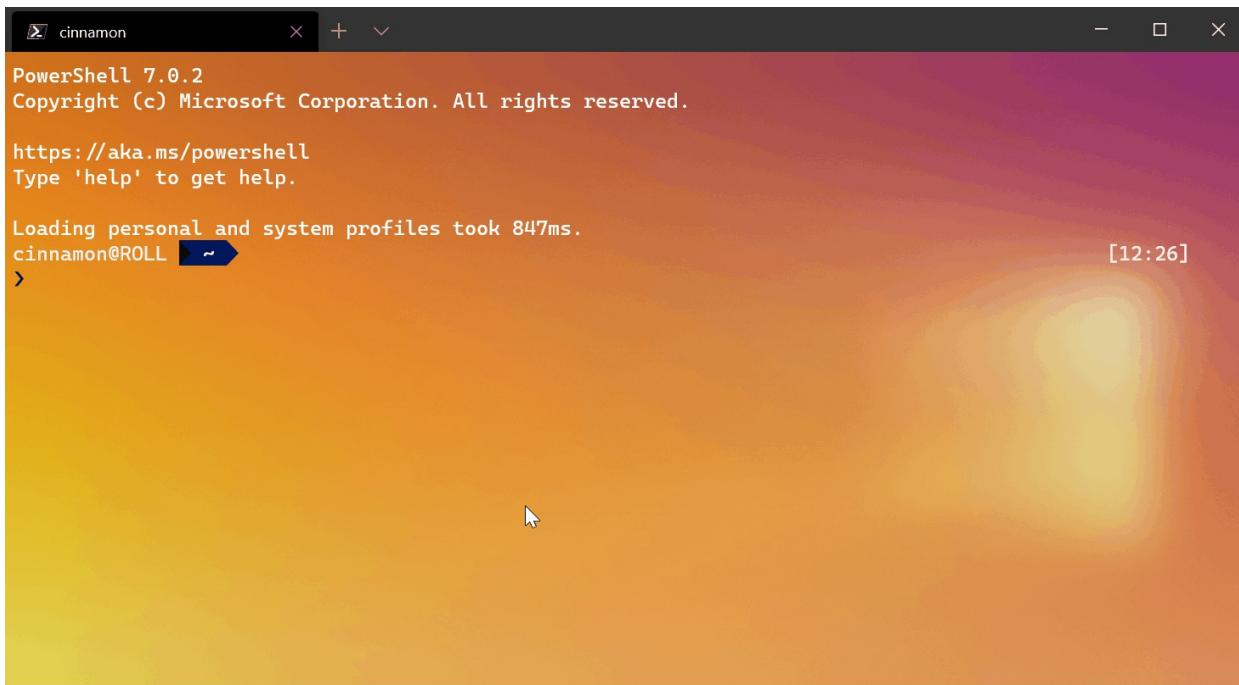
配置: [Raspberry Ubuntu](#)

如果要更改这些键绑定，可以使用 `splitPane` 操作以及 `profiles.json` 文件中 `split` 属性的 `vertical` 或 `horizontal` 值来创建新的绑定。如果你只想使用具有最大表面积的窗格，则可以将 `split` 设置为 `auto`。若要详细了解键绑定，请访问 [键绑定页](#)。

```
{ "command": { "action": "splitPane", "split": "vertical" }, "keys": "alt+shift+plus" },
{ "command": { "action": "splitPane", "split": "horizontal" }, "keys": "alt+shift+-" },
{ "command": { "action": "splitPane", "split": "auto" }, "keys": "alt+shift+|" }
```

### 使用下拉菜单(预览)

如果希望通过下拉菜单打开新窗格，可以按住 `alt` 并单击所需的配置文件。这会 `auto` 将活动窗口或窗格拆分为所选配置文件的新窗格。`auto` 拆分模式按具有最长边缘(可用于创建窗格)的方向进行拆分。

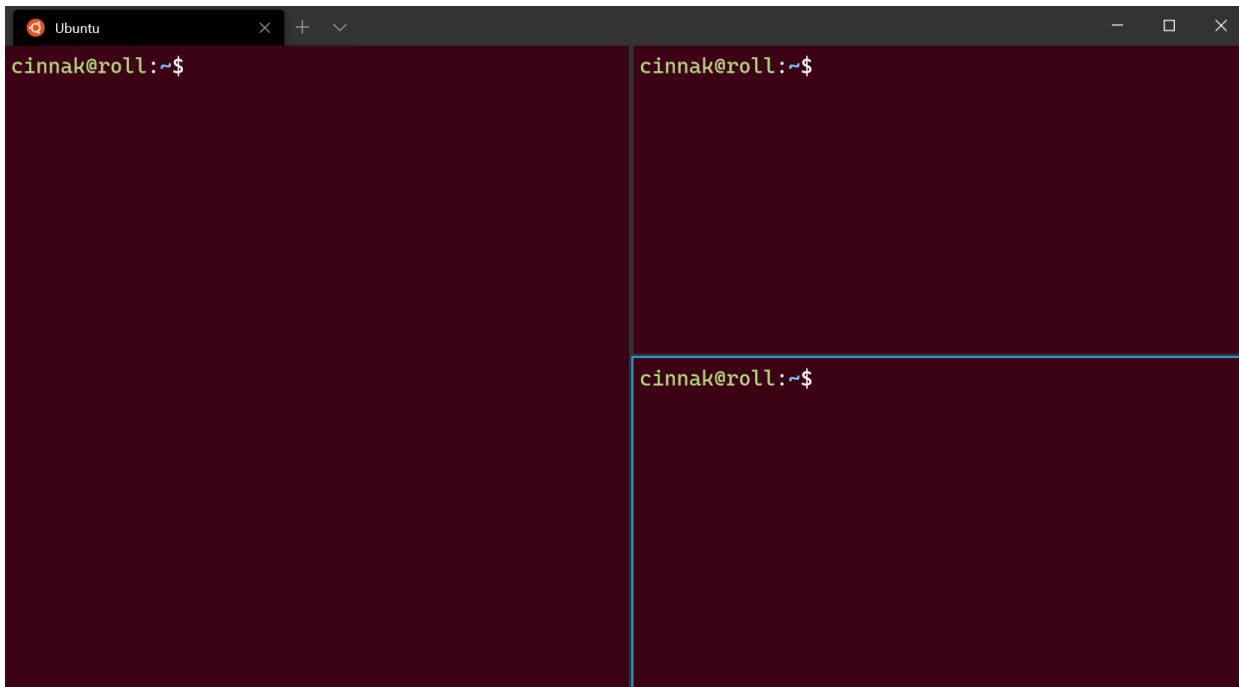


#### IMPORTANT

此功能仅在 [Windows 终端预览](#) 中可用。

## 在窗格间进行切换

终端允许使用键盘在窗格之间导航。如果按住 alt 键，则可以使用箭头键在窗格之间移动焦点。可以通过窗格周围的主题色边框来识别哪个窗格是焦点。请注意，此主题色是在 Windows 颜色设置中设置的。

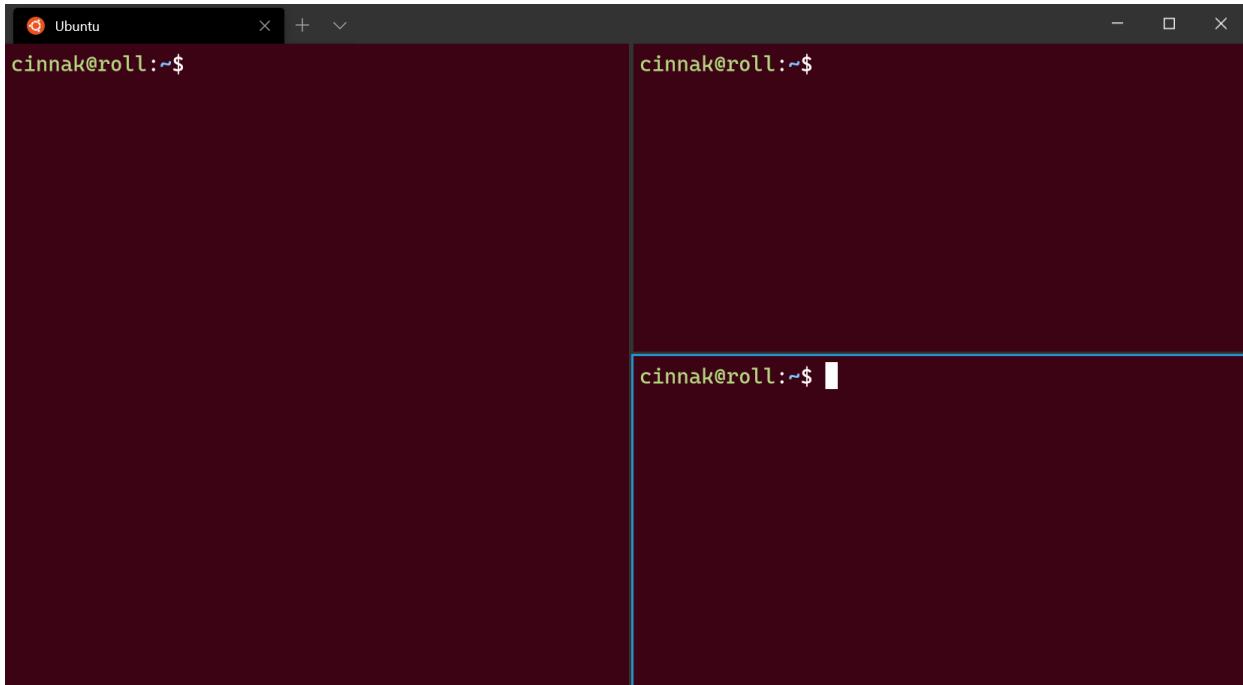


可以为 `moveFocus` 命令添加键绑定并将 `direction` 设置为 `down`、`left`、`right` 或 `up` 来自定义此项。

```
{ "command": { "action": "moveFocus", "direction": "down" }, "keys": "alt+down" },
{ "command": { "action": "moveFocus", "direction": "left" }, "keys": "alt+left" },
{ "command": { "action": "moveFocus", "direction": "right" }, "keys": "alt+right" },
{ "command": { "action": "moveFocus", "direction": "up" }, "keys": "alt+up" }
```

## 调整窗格大小

可以按住 alt+shift 并使用箭头键调整焦点窗格的大小，从而调整窗格的大小。

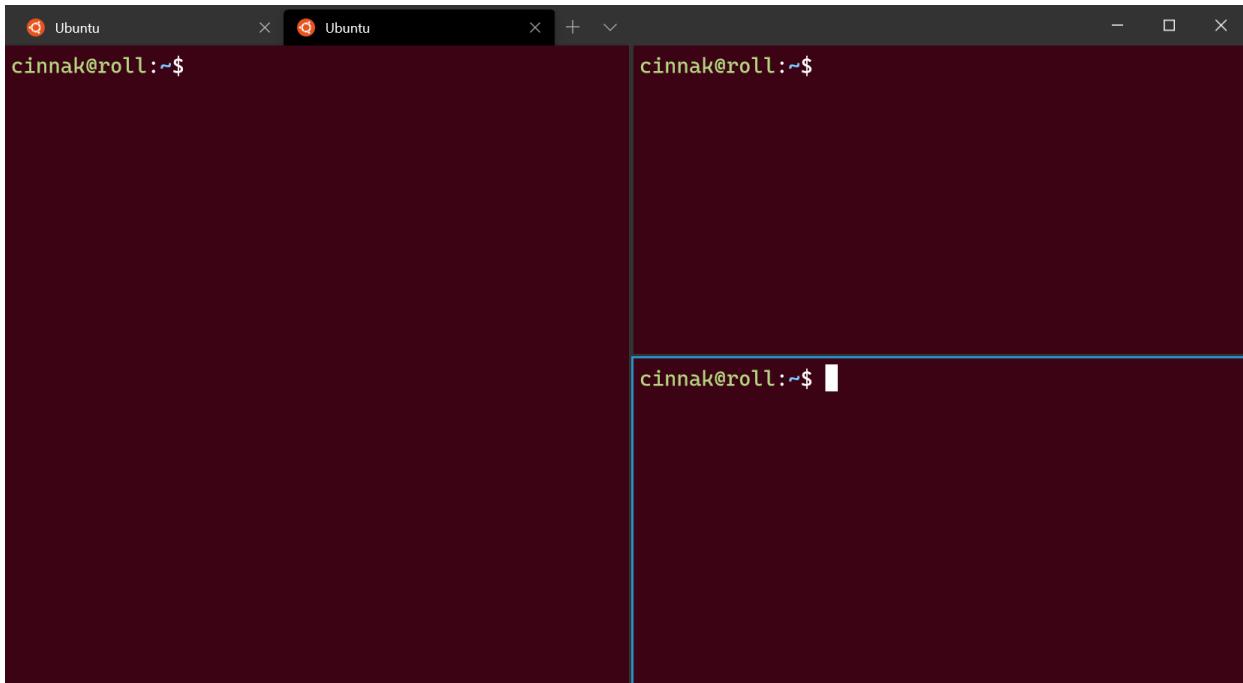


若要自定义此键绑定，可以使用 `resizePane` 操作并将 `direction` 设置为 `down`、`left`、`right` 或 `up` 来添加新的绑定。

```
{ "command": { "action": "resizePane", "direction": "down" }, "keys": "alt+shift+down" },
{ "command": { "action": "resizePane", "direction": "left" }, "keys": "alt+shift+left" },
{ "command": { "action": "resizePane", "direction": "right" }, "keys": "alt+shift+right" },
{ "command": { "action": "resizePane", "direction": "up" }, "keys": "alt+shift+up" }
```

## 关闭窗格

可以键入 `ctrl+shift+w` 来关闭焦点窗格。如果只有一个窗格，`ctrl+shift+w` 将关闭该选项卡。与往常一样，关闭最后一个选项卡将关闭该窗口。



可以添加使用 `closePane` 命令的键绑定来更改关闭窗格的键。

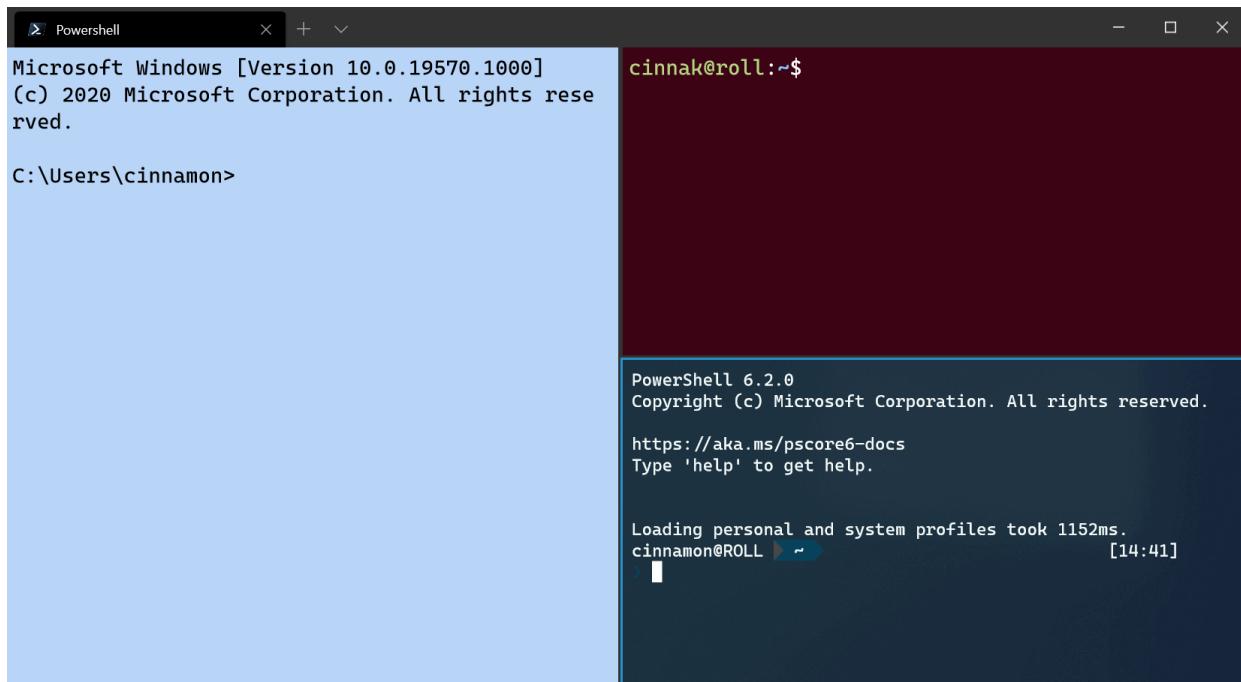
```
{ "command": "closePane", "keys": "ctrl+shift+w" }
```

## 使用键绑定自定义窗格

可以自定义在新窗格内打开的内容，具体取决于自定义键绑定。

### 复制窗格

终端允许将焦点窗格的配置文件复制到另一个窗格中。



可以通过 `duplicate` 将 `splitMode` 属性作为值添加到 `splitPane` 键绑定来完成此操作。

```
{ "command": { "action": "splitPane", "split": "auto", "splitMode": "duplicate" }, "keys": "alt+shift+d" }
```

### 新终端参数

## Windows 终端中的新终端参数

使用键绑定打开新的窗格或选项卡时，可以通过包括配置文件的名称、guid 或索引来指定要使用的配置文件。

如果未指定配置文件，则使用默认配置文件。可以将 `profile` 或 `index` 作为参数添加到 `splitPane` 或 `newTab` 键绑定来完成此操作。请注意，索引从 0 开始。

```
{ "command": { "action": "splitPane", "split": "vertical", "profile": "profile1" }, "keys": "ctrl+a" },
{ "command": { "action": "splitPane", "split": "vertical", "profile": "{00000000-0000-0000-0000-000000000000}" }, "keys": "ctrl+b" },
{ "command": { "action": "newTab", "index": 0 }, "keys": "ctrl+c" }
```

此外，你还可以替换配置文件的某些方面，例如配置文件的命令行可执行文件、起始目录或选项卡标题。这可以通过将 `commandline`、`startingDirectory` 和/或 `tabTitle` 添加到 `splitPane` 或 `newTab` 键绑定来实现。

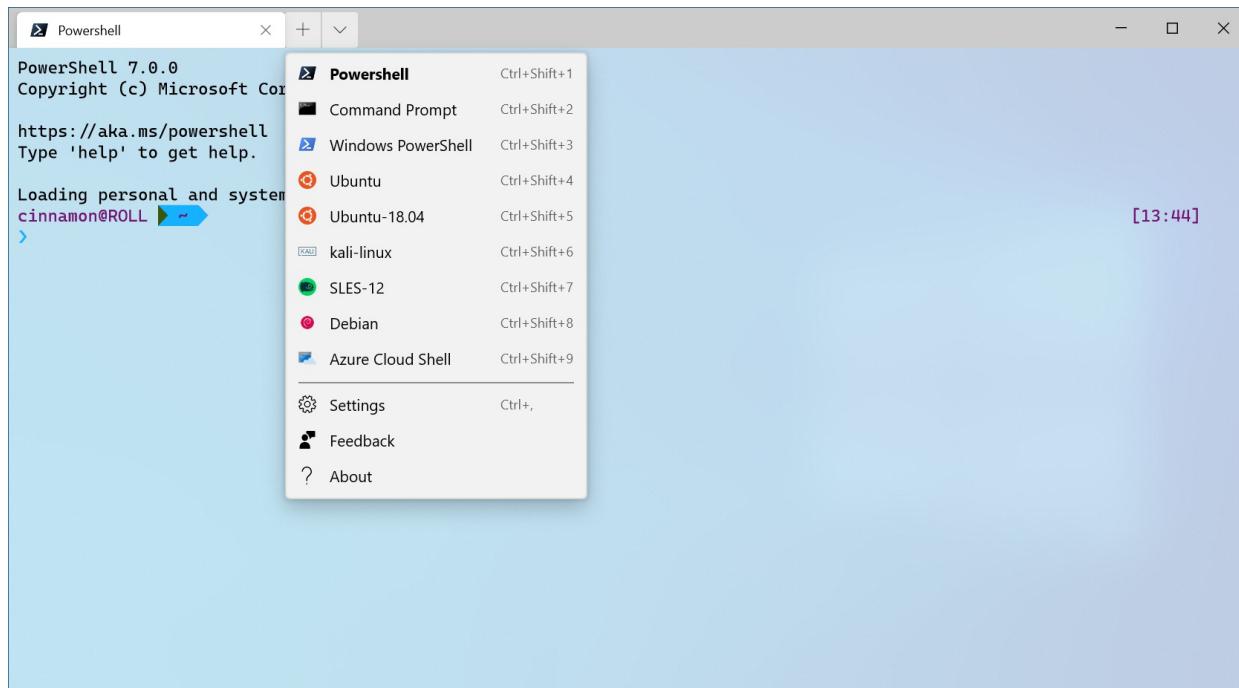
```
{ "command": { "action": "splitPane", "split": "auto", "profile": "profile1", "commandline": "foo.exe" },  
  "keys": "ctrl+a" },  
{ "command": { "action": "newTab", "profile": "{00000000-0000-0000-0000-000000000000}", "startingDirectory":  
  "C:\\\\foo" }, "keys": "ctrl+b" },  
{ "command": { "action": "newTab", "index": 0, "tabTitle": "bar", "startingDirectory": "C:\\\\foo",  
  "commandline": "foo.exe" }, "keys": "ctrl+c" }
```

# Windows 终端中的动态配置文件。

2021/2/1 •

如果在计算机上安装了这些 shell, Windows 终端将自动为你创建适用于 Linux 的 Windows 子系统 (WSL) 和 PowerShell 配置文件。这样, 你就可以更轻松地将所有 shell 包含在终端中, 无需查找其可执行文件。这些生成的配置文件具有 `source` 属性, 用于指示终端在何处查找正确的可执行文件。

安装终端后，它会将 PowerShell 设置为默认配置文件。若要了解如何更改默认配置文件，请访问[全局设置页](#)。



## 配置: 浅色主题

## 安装 Windows 终端后安装新 shell

无论是在终端安装之前还是之后安装新 shell，终端都将为新安装的 shell 创建新的配置文件。

## 隐藏配置文件

若要通过终端下拉菜单隐藏配置文件，请将 `hidden` 属性添加到 `settings.json` 文件中的配置文件对象，并将其设置为 `true`。

"hidden": true

如果删除动态创建的配置文件，终端将自动重新生成配置文件，并在 `settings.json` 文件中将其替换。

## 防止生成配置文件

若要防止生成动态配置文件，可以将配置文件生成器添加到全局设置中的 `disabledProfileSources` 数组。有关此设置的详细信息，请参阅[全局设置页](#)。

```
"disabledProfileSources": ["Windows.Terminal.Wsl", "Windows.Terminal.Azure",  
"Windows.Terminal.PowershellCore"]
```

# Cascadia Code

2021/2/1 ·

Cascadia Code 是 Microsoft 提供的一种新的等宽字体，可为命令行应用程序和文本编辑器提供全新的体验。Cascadia Code 是与 Windows 终端一起开发的。建议将此字体与终端应用程序和文本编辑器（如 Visual Studio 和 Visual Studio Code）一起使用。

## Cascadia Code 版本

有多个版本的 Cascadia Code 可供使用，其中包括连字和字形。所有版本的 Cascadia Code 都可以从 [Cascadia Code GitHub 发布页](#) 下载。Windows 终端在其包中提供 Cascadia Code 和 Cascadia Mono，并默认使用 Cascadia Mono。

字体	连字	Powerline
Cascadia Code	是	否
Cascadia Mono	否	否
Cascadia Code PL	是	是
Cascadia Mono PL	否	是

## Powerline 和编程连字

Powerline 是一个常用的命令行插件，用于在提示中显示附加信息。它使用一些附加的字形来正确显示此信息。若要详细了解如何在命令提示符中设置 Powerline，请访问 [Windows 终端中的 Powerline 页](#)。

编程连字是通过组合字符创建的字形。它们在编写代码时最有用。“Code”变体包含连字，而“Mono”变体不包含连字。

## 参与 Cascadia Code

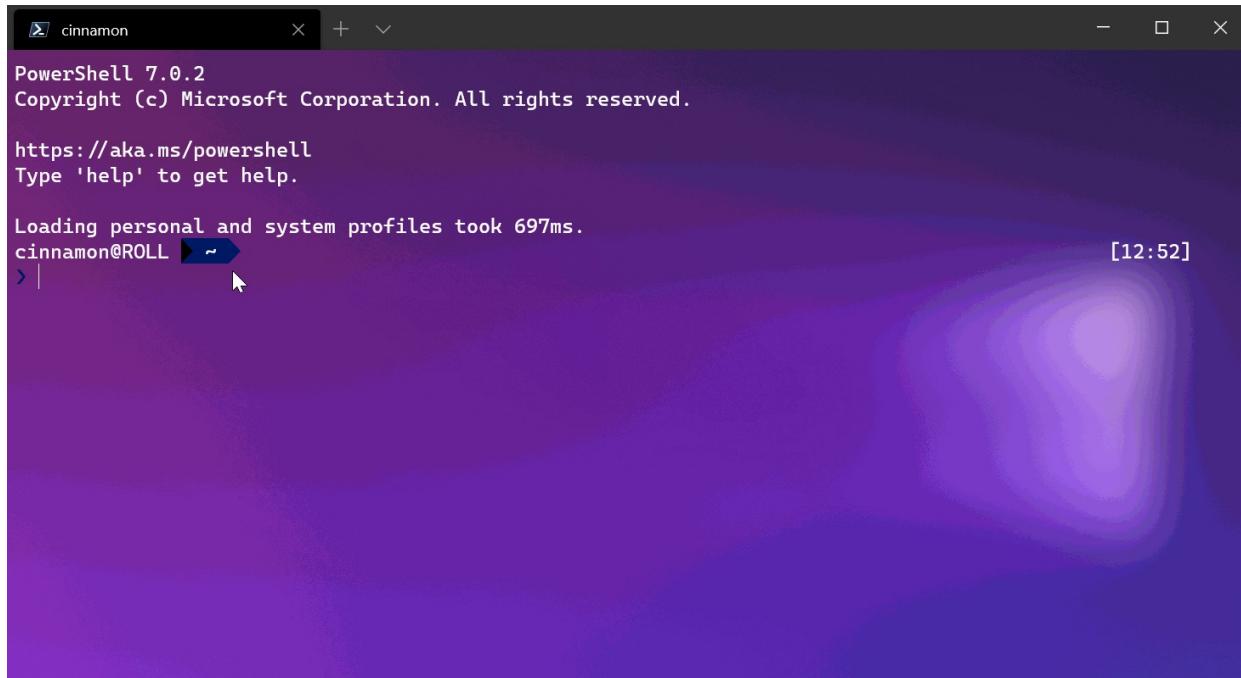
Cascadia Code 已获得 [GitHub 上的 SIL 开源字体授权](#) 的许可。

# Windows 终端提示与技巧

2021/2/1 ·

## 重命名选项卡(预览)

可以右键单击某个选项卡，然后选择“重命名选项卡”，对该终端会话的一个选项卡进行重命名。在上下文菜单中单击此选项后，选项卡标题会变为一个文本字段，可以在其中编辑标题。如果要为每个终端实例设置该配置文件的选项卡标题，可参阅[选项卡标题教程](#)，了解详细信息。



### IMPORTANT

此功能仅在 [Windows 终端预览](#) 中可用。

## 为选项卡配色(预览)

可以右键单击某个选项卡，然后选择“颜色”，为该终端会话的选项卡配色。可以从预定义的颜色列表中进行选择，也可以单击“自定义”，使用颜色选取器或 RGB/HSV 或 hex 字段来选取任何颜色。

A screenshot showing three separate windows, each titled "cinnamon", running the Windows Terminal application. Each window displays a PowerShell session with the following output:

```
PowerShell 7.0.2
Copyright (c) Microsoft Corporation. All rights reserved.

https://aka.ms/powershell
Type 'help' to get help.

Loading personal and system profiles took 697ms.
cinnamon@ROLL ~ [12:52]
```

The windows are arranged horizontally, with the central one being slightly larger.

#### TIP

使用 hex 字段将选项卡设置为与背景颜色相同的颜色，让外观颜色在整体上显得更一致。

#### IMPORTANT

此功能仅在 [Windows 终端预览](#) 中可用。

## 用鼠标进行缩放

可以通过按住 `ctrl` 和滚动来缩放 Windows 终端的文本窗口。缩放后，终端会话将保持新的缩放效果。如果要更改字体大小，可参阅[配置文件设置页面](#)，详细了解字体大小功能。

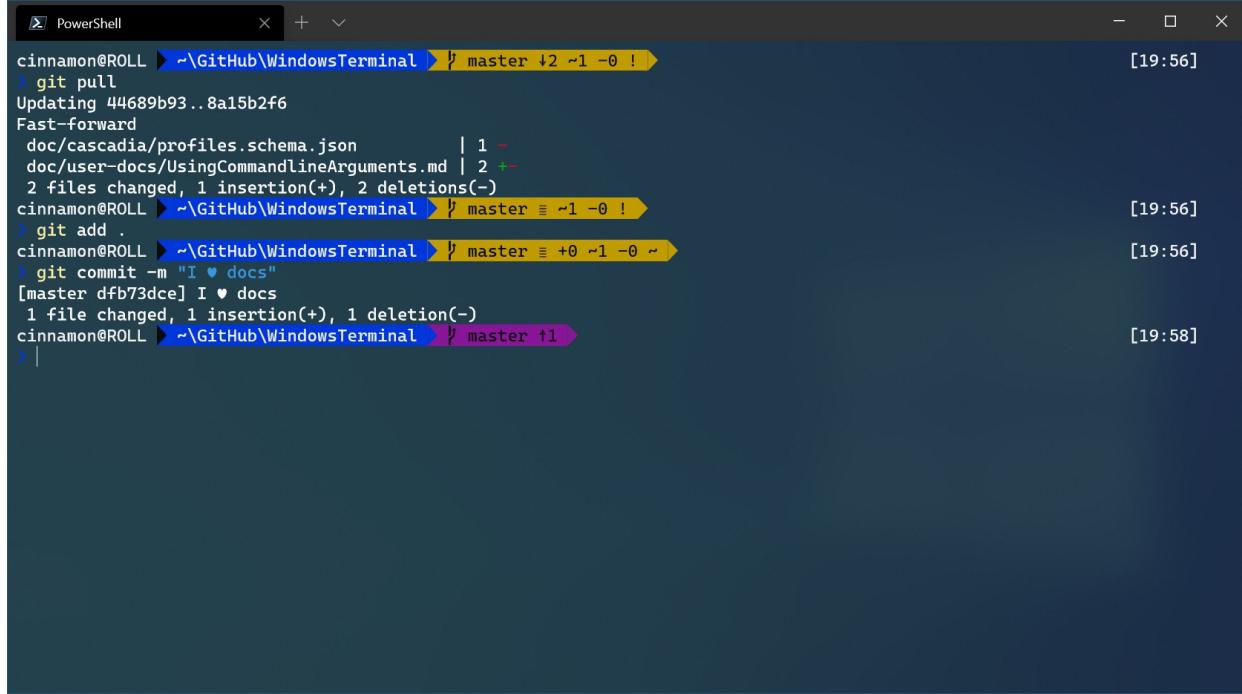
## 用鼠标调整背景的不透明度

可以通过按住 `ctrl+shift` 和滚动来调整背景的不透明度。调整后，终端会话将保持新的不透明度。如果要更改配置文件的 `acrylic` 不透明度，可参阅[配置文件设置页面](#)，详细了解 `acrylic` 背景效果

# 教程：在 Windows 终端中设置 Powerline

2021/2/1 •

Powerline 提供自定义的命令提示符体验，提供 Git 状态颜色编码和提示符。



A screenshot of a PowerShell terminal window titled "PowerShell". The window shows a sequence of git commands being run:

```
cinnamon@ROLL ~\GitHub\WindowsTerminal > git pull
Updating 44689b93..8a15b2f6
Fast-forward
 doc/cascadia/profiles.schema.json      | 1 -
 doc/user-docs/UsingCommandLineArguments.md | 2 ++
 2 files changed, 1 insertion(+), 2 deletions(-)
cinnamon@ROLL ~\GitHub\WindowsTerminal > git add .
cinnamon@ROLL ~\GitHub\WindowsTerminal > git commit -m "I ❤ docs"
[master dfb73dce] I ❤ docs
 1 file changed, 1 insertion(+), 1 deletion(-)
cinnamon@ROLL ~\GitHub\WindowsTerminal > master ↑1
```

The terminal uses a dark theme with Powerline status bars at the bottom of each line, indicating the current branch, status (e.g., modified files), and other git information.

在本教程中，你将了解如何执行以下操作：

- 在 PowerShell 中设置 Powerline
- 在 Ubuntu/WSL 中设置 Powerline
- 添加缺少的 Powerline 字形

## 必备条件

### 安装 Powerline 字体

Powerline 使用字形来设置提示符样式。如果你的字体不包含 Powerline 字形，则在整个提示符中，你可能会看到若干 Unicode 替换字符“&#x25AF”。尽管 [Cascadia Mono](#) 不包括 Powerline 字形，但你可以安装 Cascadia Code PL 或 Cascadia Mono PL，这两者包含 Powerline 字形。可以从 [Cascadia Code GitHub 发布页](#) 安装这些字体。

## 在 PowerShell 中设置 Powerline

### PowerShell 必备条件

如果尚未安装，请 [安装适用于 Windows 的 Git](#)。

使用 PowerShell，安装 Posh-Git 和 Oh-My-Posh：

```
Install-Module posh-git -Scope CurrentUser
Install-Module oh-my-posh -Scope CurrentUser
```

## TIP

如果尚未安装 NuGet，可能需要安装它。如果是这种情况，PowerShell 命令行会询问是否要安装 NuGet。选择 [Y]“是”。你可能还需要批准从不受信任的存储库 [PSGallery](#) 中安装模块。选择 [Y]“是”。

[Posh-Git](#) 将 Git 状态信息添加到提示，并为 Git 命令、参数、远程和分支名称添加 tab 自动补全。[Oh-My-Posh](#) 为 PowerShell 提示符提供主题功能。

如果使用的是 PowerShell Core，请安装 PSReadline：

```
Install-Module -Name PSReadLine -Scope CurrentUser -Force -SkipPublisherCheck
```

[PSReadline](#) 允许在 PowerShell 中自定义命令行编辑环境。

### 自定义 PowerShell 提示符

使用 `notepad $PROFILE` 或所选的文本编辑器打开 PowerShell 配置文件。这不是你的 Windows 终端配置文件。你的 PowerShell 配置文件是一个脚本，该脚本在每次启动 PowerShell 时运行。[详细了解 PowerShell 配置文件](#)。

在 PowerShell 配置文件中，将以下内容添加到文件的末尾：

```
Import-Module posh-git
Import-Module oh-my-posh
Set-Theme Paradox
```

现在，每个新实例启动时都会导入 Posh-Git 和 Oh-My-Posh，然后从 Oh-My-Posh 设置 Paradox 主题。Oh-My-Posh 附带了若干[内置主题](#)。

### 在设置中将 Cascadia Code PL 设置为 fontFace

若要设置 Cascadia Code PL 以便与 PowerLine 一起使用（在系统中下载、解压缩和安装之后），需要通过从“Windows 终端”下拉菜单中选择“设置”（[Ctrl+,](#)）来打开 settings.json 文件中的[配置文件设置](#)。

settings.json 文件打开后，找到 Windows PowerShell 配置文件，并添加 `"fontFace": "Cascadia Code PL"`，以便将 Cascadia Code PL 指定为字体。这样就会显示很好看的 Cascadia Code Powerline 字形。在编辑器中选择“保存”后，终端应会即刻显示出变化。

Windows PowerShell 配置文件 settings.json 文件现在应如下所示：

```
{
    // Make changes here to the powershell.exe profile.
    "guid": "{61c54bbd-c2c6-5271-96e7-009a87ff44bf}",
    "name": "Windows PowerShell",
    "commandline": "powershell.exe",
    "fontFace": "Cascadia Code PL",
    "hidden": false
},
```

## 在 WSL Ubuntu 中设置 Powerline

### WSL Ubuntu 必备条件

Ubuntu 有几个 Powerline 安装方式选项。本教程将使用 Go 和 Powerline-Go：

```
sudo apt install golang-go
go get -u github.com/justjanne/powerline-go
```

## 自定义 Ubuntu 提示符

使用 `nano ~/.bashrc` 或所选的文本编辑器打开 `~/.bashrc` 文件。这是一个 bash 脚本，该脚本在每次启动 bash 时运行。添加以下内容，但请注意 GOPATH 可能已经存在：

```
GOPATH=$HOME/go
function _update_ps1() {
    PS1=$(($GOPATH/bin/powerline-go -error $?))
}
if [ "$TERM" != "linux" ] && [ -f "$GOPATH/bin/powerline-go" ]; then
    PROMPT_COMMAND=_update_ps1; $PROMPT_COMMAND"
fi
```

## 其他资源

- [Scott Hanselman 的“如何在 Windows 终端中设置不错的提示符”](#)
- [如何在 Linux 中更改/设置 bash 自定义提示符 \(PS1\)](#)

# 教程 : Windows 终端中的 SSH

2021/2/1 ·

Windows 10 提供可在 Windows 终端中使用的内置的 SSH 客户端。

在本教程中, 你将了解如何在使用 SSH 的 Windows 终端中设置配置文件。

## 创建档案

你可以通过执行 `ssh user@machine` 在命令提示符下启动 SSH 会话, 系统将提示你输入密码。可以将 `commandline` 设置添加到 `settings.json` 文件中的配置文件来创建在启动时执行此项的 Windows 终端配置文件。

```
"commandline": "ssh cinnamon@roll"
```

## 指定起始目录

若要指定 Windows 终端调用的 ssh 会话的起始目录, 可以使用以下命令:

```
"commandline": "ssh -t bob@foo \"cd /data/bob && exec bash -l\""
```

`-t` 标志强制执行伪终端分配。这可用于在远程计算机上执行任意基于屏幕的程序, 例如实现菜单服务。将需要使用转义双引号, 因为 bourne 外壳派生物不会为单引号中的字符串执行任何额外分析。

有关更多信息, 请参阅:

- [GH 问题: 如何指定 ssh 会话的起始目录?](#)
- [StackOverflow: 如何直接通过 ssh 连接到特定目录?](#)

## 资源

- [如何启用和使用 Windows 10 的新内置 SSH 命令](#)

# 教程：在 Windows 终端中配置选项卡标题

2020/5/15 •

默认情况下，选项卡标题设置为 shell 的标题。如果选项卡由多个窗格组成，则会将该选项卡的标题设置为当前焦点窗格的标题。如果要对设置为选项卡标题的内容进行自定义，请按照本教程的说明操作。

在本教程中，你将了解如何执行以下操作：

- 使用 `tabTitle` 设置
- 设置 shell 的标题
- 使用 `suppressApplicationTitle` 设置

## 使用 `tabTitle` 设置

`tabTitle` 设置允许定义 shell 的新实例的起始标题。如果未设置，则改为使用配置文件 `name`。每个 shell 以不同的方式响应此设置。

SHELL	示例
PowerShell	标题已设置。
命令提示符	标题已设置。如果命令正在运行，它将暂时追加到标题的末尾。
Ubuntu	标题会被忽略，但设置为 <code>user@machine:path</code>
Debian	标题已设置。

### NOTE

尽管 Ubuntu 和 Debian 都运行 bash，但它们具有不同的行为。这是为了表明不同的分发可能具有不同的行为。

## 设置 shell 的标题

Shell 可以完全控制自己的标题。但是，每个 shell 以不同的方式设置其标题。

SHELL	示例
PowerShell	<code>\$Host.UI.RawUI.WindowTitle = "New Title"</code>
命令提示符	<code>TITLE "New Title"</code>
bash*	<code>echo -ne "\033]0;New Title\a"</code>

请注意，在与 shell 交互时，某些 Linux 分发版（即 Ubuntu）会自动设置其标题。如果上面的命令不起作用，请运行以下命令：

```
PS1=$  
PROMPT_COMMAND=  
echo -ne "\033]0;New Title\a"
```

这会将标题更改为“新标题”，同时将提示符设置为“\$”。

## 使用 `suppressApplicationTitle` 设置

由于 shell 可以控制其标题，因此它可以随时选择覆盖选项卡标题。例如，PowerShell 的 `posh-git` 模块将有关 Git 存储库的信息添加到标题中。

Windows 终端允许在配置文件中将 `suppressApplicationTitle` 设置为 `true` 来禁止对标题进行更改。这会使配置文件的新实例将可见标题设置为 `tabTitle`。如果未设置 `tabTitle`，则会将可见标题设置为配置文件的 `name`。

请注意，这会将 shell 的标题与选项卡上显示的可见标题分离开来。如果在设置了标题的位置读取 shell 的变量，则它可能与选项卡的标题不同。

## 资源

- [将控制台标题设置为当前工作目录](#)
- [更改 Ubuntu 16.04 上终端的标题](#)

# 自定义终端指南

2021/2/1 ·

下面是一些颜色方案，你可以根据自己的设计尝试或使用这些方案。

## 安装方案

将 JSON 从 "图式" 部分复制到 settings.json 上的正确部分，例如：

早于：

```
"schemes": [],
```

晚于：

```
"schemes": [
  {
    "name": "Retro",
    "background": "#000000",
    "black": "#00ff00",
    "blue": "#00ff00",
    "brightBlack": "#00ff00",
    "brightBlue": "#00ff00",
    "brightCyan": "#00ff00",
    "brightGreen": "#00ff00",
    "brightPurple": "#00ff00",
    "brightRed": "#00ff00",
    "brightWhite": "#00ff00",
    "brightYellow": "#00ff00",
    "cyan": "#00ff00",
    "foreground": "#00ff00",
    "green": "#00ff00",
    "purple": "#00ff00",
    "red": "#00ff00",
    "white": "#00ff00",
    "yellow": "#00ff00"
  }
]
```

然后添加配置文件的特定部分，例如：

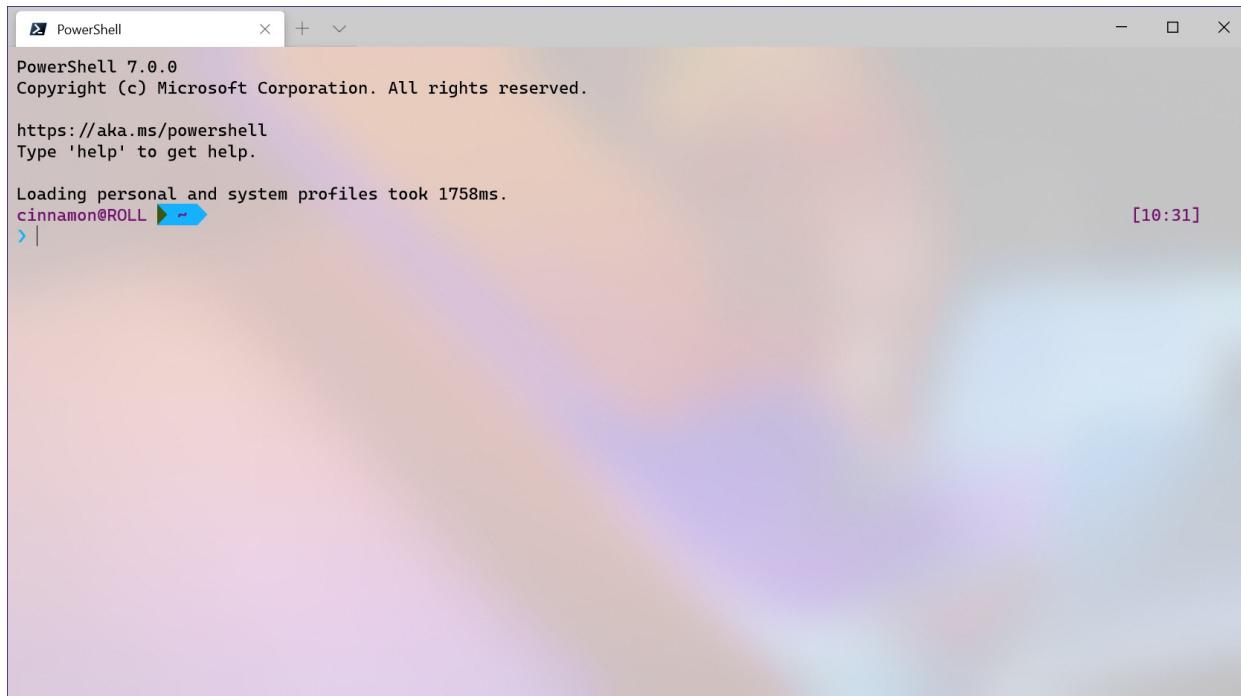
早于：

```
{
  "guid": "{234ab24f-34dd-ff3-ade434aad345}",
  "name": "Command Prompt",
  "commandline": "cmd.exe",
  "hidden": false
}
```

晚于：

```
{  
    "guid": "{234ab24f-34dd-ff3-ade434aad345}",  
    "name": "Command Prompt",  
    "commandline": "cmd.exe",  
    "hidden": false,  
    "colorScheme" : "Retro",  
    "cursorColor" : "#FFFFFF",  
    "cursorShape": "filledBox",  
    "fontSize" : 16,  
    "padding" : "5, 5, 5, 5",  
    "tabTitle" : "Command Prompt",  
    "fontFace": "PxPlus IBM VGA8",  
    "experimental.retroTerminalEffect": true  
}
```

## 毛玻璃玻璃



[详细信息](#)

## Powerline

```
cinnamon@ROLL ~\GitHub\WindowsTerminal > ! master ↵2 ↵1 -0 !
> git pull
Updating 44689b93..8a15b2f6
Fast-forward
 doc/cascadia/profiles.schema.json | 1 -
 doc/user-docs/UsingCommandLineArguments.md | 2 +-
 2 files changed, 1 insertion(+), 2 deletions(-)
cinnamon@ROLL ~\GitHub\WindowsTerminal > ! master ↵ +0 ↵1 -0 !
> git add .
cinnamon@ROLL ~\GitHub\WindowsTerminal > ! master ↵ +0 ↵1 -0 !
> git commit -m "I ❤ docs"
[master dfb73dce] I ❤ docs
 1 file changed, 1 insertion(+), 1 deletion(-)
cinnamon@ROLL ~\GitHub\WindowsTerminal > ! master ↵1
> |
```

[详细信息](#)

## Raspberry Ubuntu

```
Background | Foreground colors
-----
ESC[40m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[40m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[41m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[41m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[42m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[42m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[43m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[43m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[44m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[44m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[45m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[45m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

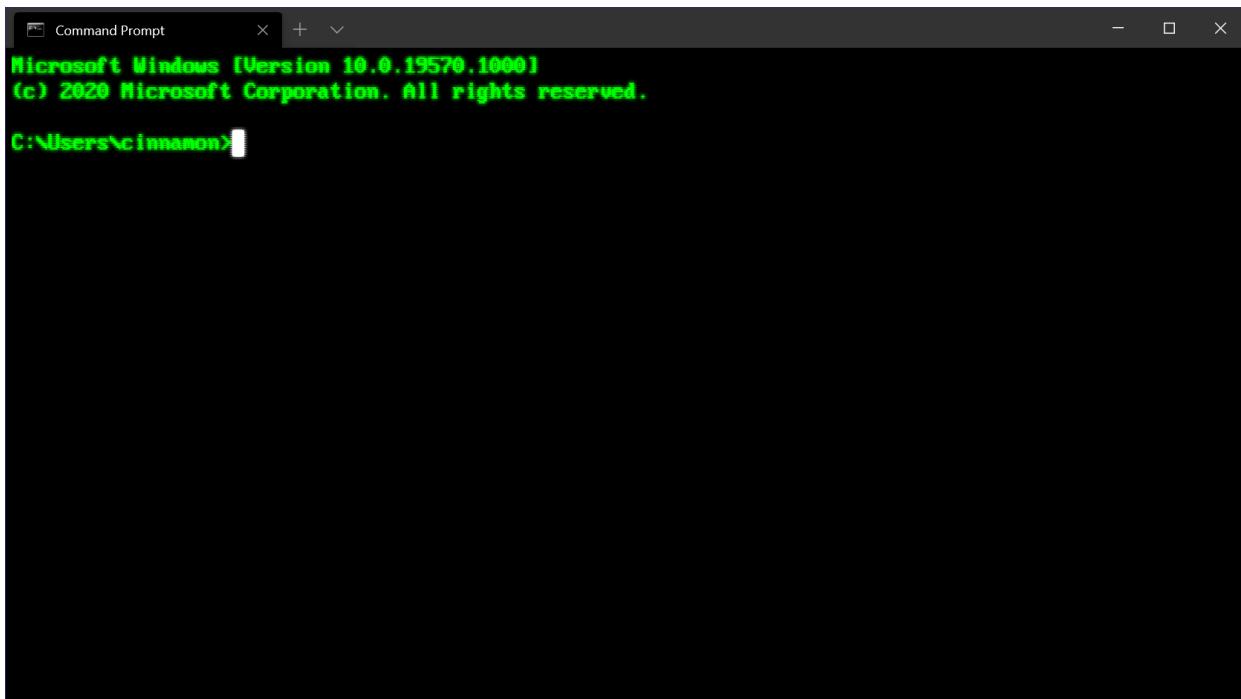
ESC[46m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[46m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[47m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[47m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m
```

cinnak@roll:/mnt/c/Users/cinnamon\$ █

[详细信息](#)

## 怀旧命令



[详细信息](#)

## 共享！

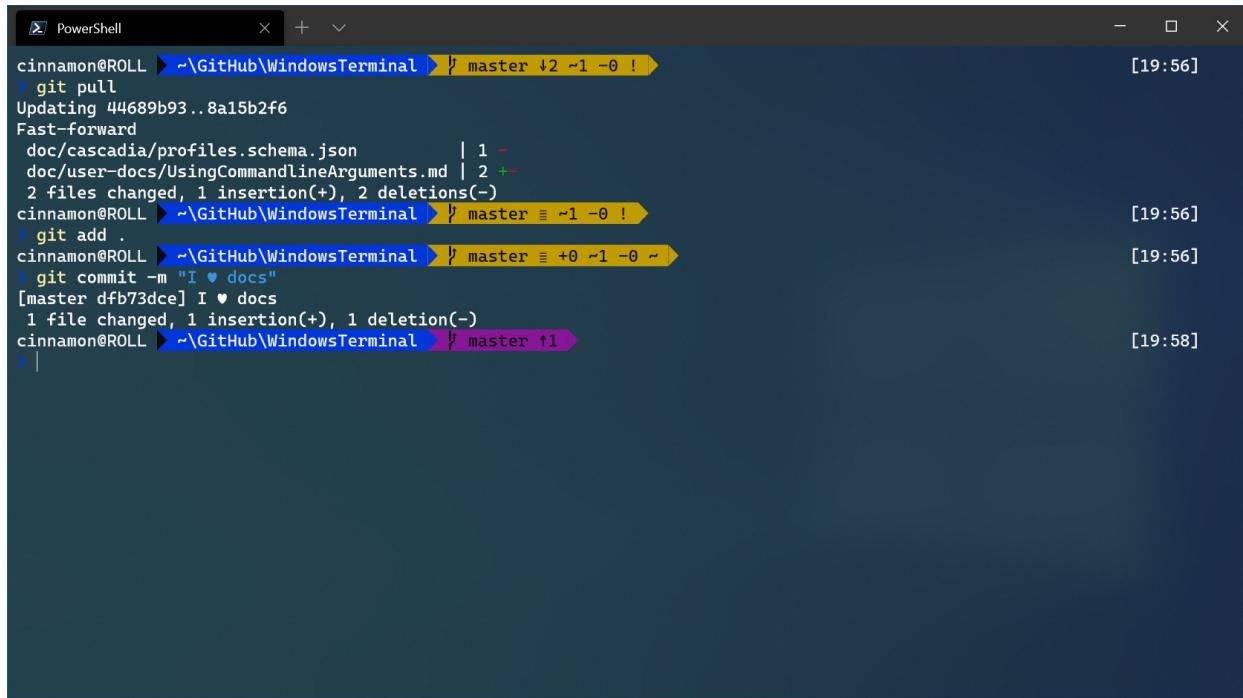
你是否有想要共享的 Windows 终端方案？在 [Twitter](#) 上显示我们！

# Windows 终端的 PowerShell 主题中的 Powerline

2021/2/1 •

该提示使用 Powerline 进行样式化，并使用 `Cascadia Code PL` 可从 [Cascadia 代码 GitHub 版本页](#) 下载的字体。

## 了解如何设置 Powerline



```
{
    "theme": "dark",
    "profiles": [
        {
            "name" : "Powershell",
            "source" : "Windows.Terminal.PowershellCore",
            "acrylicOpacity" : 0.7,
            "colorScheme" : "Campbell",
            "cursorColor" : "#FFFFFFD",
            "fontFace" : "Cascadia Code PL",
            "useAcrylic" : true
        }
    ]
}
```

# Windows 终端中的 Raspberry Ubuntu

2021/2/1 •

The screenshot shows a Windows terminal window titled "Ubuntu". The title bar includes standard window controls (minimize, maximize, close) and the title text. The main area displays a color palette test. The text output consists of several lines of ESC sequences followed by color swatches. The colors are arranged in a grid and include various shades of black, white, red, green, blue, cyan, magenta, and yellow. The background of the terminal window is dark.

```
Background | Foreground colors
-----
ESC[40m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[40m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[41m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[41m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[42m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[42m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[43m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[43m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[44m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[44m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[45m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[45m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[46m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[46m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m

ESC[47m | [30m [31m [32m [33m [34m [35m [36m [37m
ESC[47m | [1;30m [1;31m [1;32m [1;33m [1;34m [1;35m [1;36m [1;37m
```

cinnak@roll:/mnt/c/Users/cinnamon\$ █

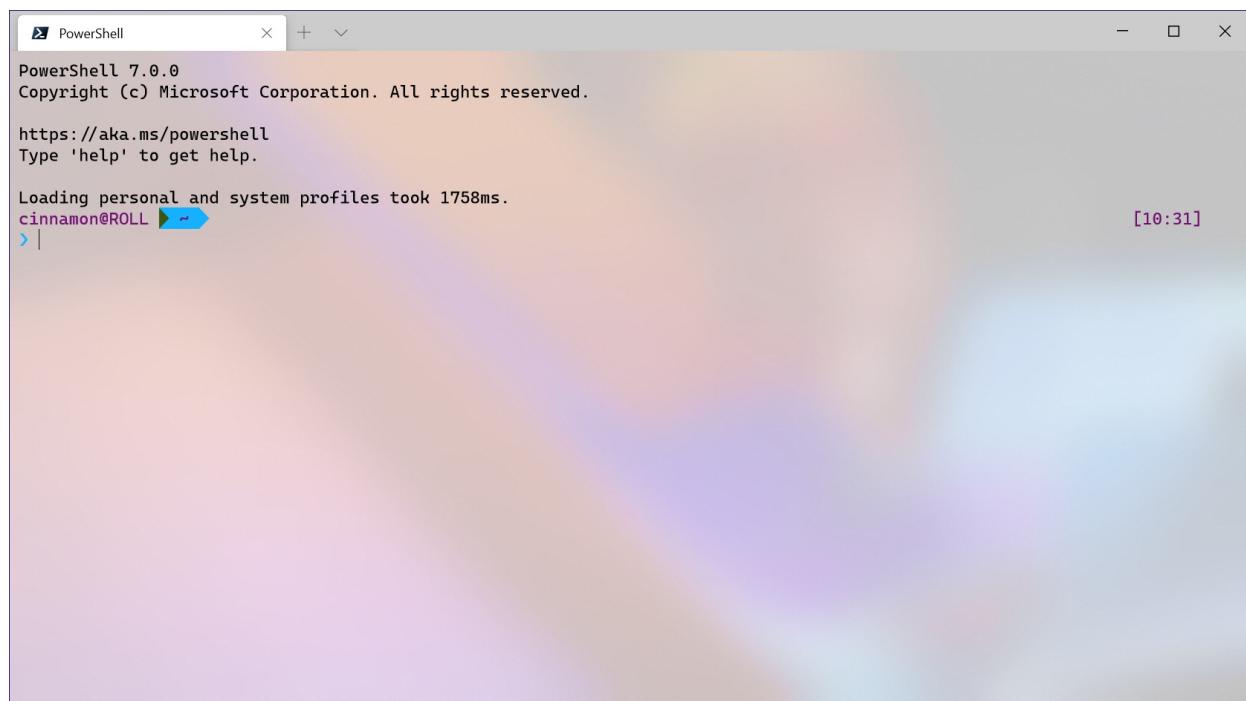
```
{
  "theme": "dark",
  "profiles": [
    {
      "name" : "Ubuntu",
      "source" : "Windows.Terminal.Wsl",
      "colorScheme" : "Raspberry",
      "cursorColor" : "#FFFFFF",
      "fontFace" : "Cascadia Code",
      "padding" : "5, 5, 5, 5",
      "suppressApplicationTitle": true,
      "tabTitle": "Ubuntu"
    }
  ],
  "schemes": [
    {
      "name" : "Raspberry",
      "background" : "#3C0315",
      "black" : "#282A2E",
      "blue" : "#0170C5",
      "brightBlack" : "#676E7A",
      "brightBlue" : "#80c8ff",
      "brightCyan" : "#8ABEB7",
      "brightGreen" : "#B5D680",
      "brightPurple" : "#AC79BB",
      "brightRed" : "#BD6D85",
      "brightWhite" : "#FFFFFD",
      "brightYellow" : "#FFFD76",
      "cyan" : "#3F8D83",
      "foreground" : "#FFFFFD",
      "green" : "#76AB23",
      "purple" : "#7D498F",
      "red" : "#BD0940",
      "white" : "#FFFFFD",
      "yellow" : "#E0DE48"
    }
  ]
}
```

# Windows 终端中的毛玻璃玻璃主题

2021/2/1 •

该提示使用 Powerline 进行样式化，并使用 [Cascadia Code PL](#) 可从 Cascadia 代码 GitHub 版本页下载的字体。

[了解如何设置 Powerline](#)

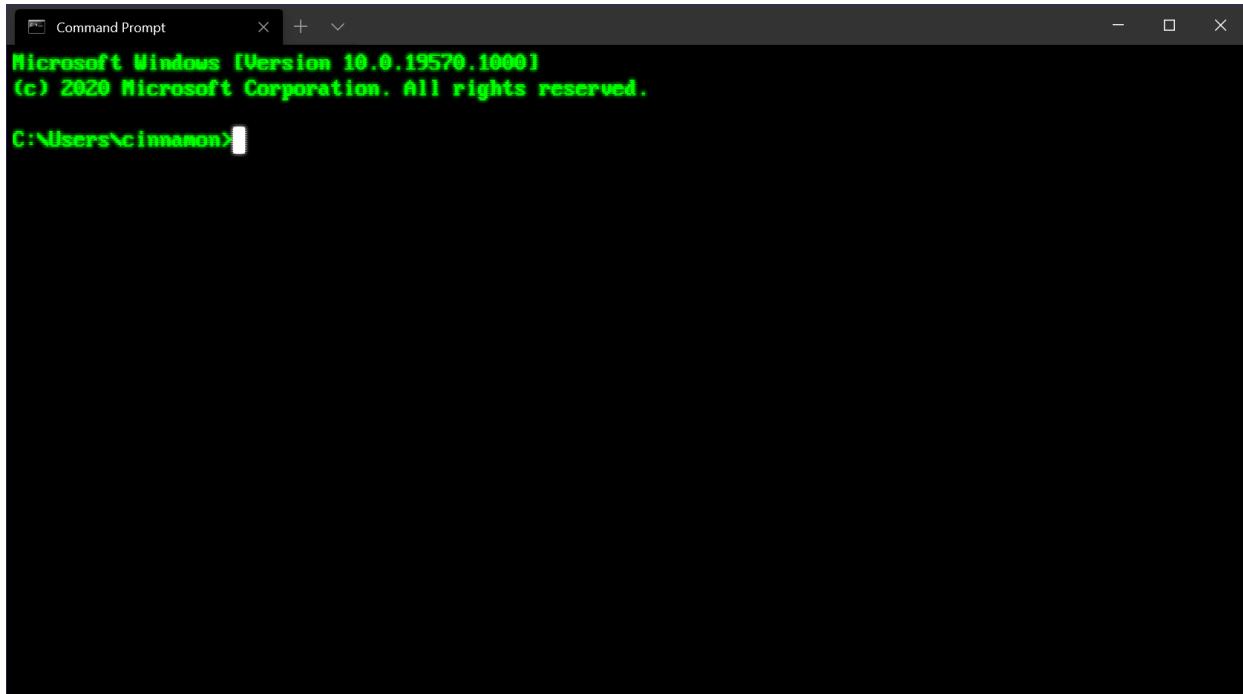


```
{  
    "theme": "light",  
    "profiles": [  
        {  
            "name" : "PowerShell",  
            "source" : "Windows.Terminal.PowershellCore",  
            "acrylicOpacity": 0.7,  
            "colorScheme" : "Frost",  
            "cursorColor" : "#000000",  
            "fontFace" : "Cascadia Code PL",  
            "useAcrylic": true  
        }  
    ],  
    "schemes": [  
        {  
            "name" : "Frost",  
            "background" : "#FFFFFF",  
            "black" : "#3C5712",  
            "blue" : "#17b2ff",  
            "brightBlack" : "#749B36",  
            "brightBlue" : "#27B2F6",  
            "brightCyan" : "#13A8C0",  
            "brightGreen" : "#89AF50",  
            "brightPurple" : "#F2A20A",  
            "brightRed" : "#F49B36",  
            "brightWhite" : "#741274",  
            "brightYellow" : "#991070",  
            "cyan" : "#3C96A6",  
            "foreground" : "#000000",  
            "green" : "#6AAE08",  
            "purple" : "#991070",  
            "red" : "#8D0C0C",  
            "white" : "#6E386E",  
            "yellow" : "#991070"  
        }  
    ]  
}
```

# Windows 终端中的怀旧式命令提示符

2021/2/1 ·

提示使用的是 `PxPlus IBM VGA8` 不包含在 Windows 终端中的字体。



```
{  
    "theme": "dark",  
    "profiles": [  
        {  
            "name": "Command Prompt",  
            "commandline": "cmd.exe",  
            "closeOnExit" : true,  
            "colorScheme" : "Retro",  
            "cursorColor" : "#FFFFFF",  
            "cursorShape": "filledBox",  
            "fontSize" : 16,  
            "padding" : "5, 5, 5, 5",  
            "tabTitle" : "Command Prompt",  
            "fontFace": "PxPlus IBM VGA8",  
            "experimental.retroTerminalEffect": true  
        }  
    ],  
    "schemes": [  
        {  
            "name": "Retro",  
            "background": "#000000",  
            "black": "#00ff00",  
            "blue": "#00ff00",  
            "brightBlack": "#00ff00",  
            "brightBlue": "#00ff00",  
            "brightCyan": "#00ff00",  
            "brightGreen": "#00ff00",  
            "brightPurple": "#00ff00",  
            "brightRed": "#00ff00",  
            "brightWhite": "#00ff00",  
            "brightYellow": "#00ff00",  
            "cyan": "#00ff00",  
            "foreground": "#00ff00",  
            "green": "#00ff00",  
            "purple": "#00ff00",  
            "red": "#00ff00",  
            "white": "#00ff00",  
            "yellow": "#00ff00"  
        }  
    ]  
}
```

# 排查 Windows 终端中的问题

2021/2/1 ·

本指南解决了使用 Windows 终端时可能会遇到的一些常见错误和障碍。

## 将 WSL 分发设置为启动时在主 ~ 目录中开始

默认情况下，配置文件的 `startingDirectory` 为 `%USERPROFILE%` (`C:\Users\<YourUsername>`)。这是一个 Windows 路径。但对于 WSL，可能需要改用 WSL 主路径。`startingDirectory` 仅接受 Windows 样式路径，因此将其设置为在 WSL 分发中启动需要使用前缀。

从 Windows 10 版本 1903 开始，可以使用 `\wsl$\  
`` 前缀来寻址 WSL 分发的文件系统。对于名称为 `DistroName` 的任何 WSL 分发，请使用 `\wsl$\DistroName` 作为 Windows 路径，该路径指向该分发文件系统的根目录。

例如，以下设置会在其主文件路径中启动“Ubuntu-18.04”分发：

```
{  
    "name": "Ubuntu-18.04",  
    "commandline": "wsl -d Ubuntu-18.04",  
    "startingDirectory": "//wsl$/Ubuntu-18.04/home/<Your Ubuntu Username>",  
}
```

## 设置选项卡标题

若要使 shell 自动设置选项卡标题，请访问[设置选项卡标题教程](#)。如果要设置自己的选项卡标题，请打开 `settings.json` 文件，然后执行以下步骤：

1. 在所选命令行的配置文件中，添加 `"suppressApplicationTitle": true` 以禁止 shell 发出的任何标题更改事件。仅将此设置添加到配置文件会将选项卡标题设置为配置文件的名称。
2. 如果需要不是配置文件名称的自定义选项卡标题，请添加 `"tabTitle": "TITLE"`。用首选选项卡标题替换“TITLE”。

## PowerShell 中的命令行参数

请访问[命令行参数页](#)，了解命令行参数在 PowerShell 中的运作方式。

## WSL 中的命令行参数

请访问[命令行参数页](#)，了解命令行参数在 WSL 中的运作方式。

## 设置 `startingDirectory` 时出现的问题

如果配置文件中忽略 `startingDirectory`，请先检查以确保你的 `settings.json` 的语法是正确的。为了帮助检查此语法，`"$schema": "https://aka.ms/terminal-profiles-schema"` 会自动注入。某些应用程序（如 [Visual Studio Code](#)）在你进行编辑时，可以使用这个注入的架构来验证 json 文件。

如果设置正确，则可能正在运行一个启动脚本，该脚本单独设置终端的起始目录。例如，PowerShell 具有自己单独的配置文件概念。如果在此处更改起始目录，它将优先于在 Windows 终端中定义的设置。

或者, 如果正在使用 `commandline` 配置文件设置运行脚本, 则可能是在此处设置位置。类似于 PowerShell 配置文件, 此处命令的优先级高于 `startingDirectory` 配置文件设置。

`startingDirectory` 的目的是在给定目录中启动新的 Windows 终端实例。如果终端运行更改其目录的任何代码, 那么这里可能是进行查看的不错位置。

## Ctrl+= 不会增加字体大小

如果使用德语键盘布局, 则可能会遇到此问题。如果将主键盘布局设置为德语, 则 `ctrl+=` 将反序列化为 `ctrl+shift+0`。这是德语键盘的正确映射。

更重要的是, 应用永远不会收到 `ctrl+shift+0` 击键。这是因为如果有多个活动的键盘布局, 则 Windows 将保留 `ctrl+shift+0`。

若要禁用此功能以便 `ctrl+=` 正常运行, 请按照此[博客文章](#)中的“更改 Windows 10 中切换键盘布局的热键”说明进行操作。

将“切换键盘布局”选项更改为“未分配”(或关闭 `ctrl+shift`), 然后依次选择“确定”、“应用”。`ctrl+shift+0` 现在应作为键绑定工作, 并传递给终端。

另一方面, 如果对多种输入语言使用此热键功能, 则可以在 `settings.json` 文件中[配置自己的自定义键绑定](#)。

## 文本模糊

如果不对上一帧中的数据进行模糊处理, 某些显示驱动程序和硬件组合就无法正常处理滚动和/或脏区域。若要缓解此问题, 可以添加[这些全局呈现设置](#)的组合, 以降低由于终端文本呈现器导致的硬件显示拉伸。