TOPICS Integrated Terminal ▼

综合终端

(https://github.com/Microsoft/vscode-docs/blob/master/docs/editor/integrated-terminal.md)

在Visual Studio Code中，您可以首先从工作区的根开始打开一个集成终端。这很方便，因为您不必切换窗口或更改现有终端的状态即可执行快速的命令行任务。

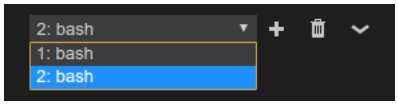
要打开终端：

- 将 Ctrl +` 键盘快捷键与反引号一起使用。
- 使用**查看** > **终端**菜单命令。
- 在**命令面板**（Ctrl + Shift + P）中，使用“**查看：切换集成终端**”命令。

> **注意：**如果您希望在VS Code之外工作，仍可以使用 Ctrl + Shift + C 键盘快捷键打开外壳。

管理多个终端

您可以创建多个打开到不同位置的终端，并在它们之间轻松导航。可以通过单击**TERMINAL**面板右上角的加号图标或触发 Ctrl + Shift +` 命令来添加终端实例。此操作将在下拉列表中创建另一个条目，可用于在它们之间进行切换。



按下垃圾桶按钮删除终端实例。

> **提示：**如果您使用多个终端的广泛应用，你可以添加键绑定的 `focusNext`， `focusPrevious` 并且 `kill` 在列出的命令键绑定部分 (/docs/editor/integrated-terminal#_terminal-keybindings)仅使用键盘，让他们之间的导航。

端子分割

您也可以通过触发 Ctrl + Shift + 5 命令或通过右键单击上下文菜单来拆分终端。

聚焦拆分的终端窗格时，可以使用以下命令之一移动焦点并调整大小：

键	命令
Alt +左	聚焦上一个窗格
Alt +右	聚焦下一个窗格
未分配	调整左窗格的大小
未分配	调整右窗格大小
未分配	调整窗格大小
未分配	调整窗格大小

组态

使用的外壳默认 `$SHELL` 在Linux和macOS上使用，在Windows 10上使用PowerShell，在Windows早期版本上使用cmd.exe。可以通过 `terminal.integrated.shell.*` 在用户设置中 (/docs/getstarted/settings)进行设置来手动覆盖这些设置 (/docs/getstarted/settings)。可以使用 `terminal.integrated.shellArgs.*` 用户设置将参数传递到终端外壳。

> **注意：**这些设置不会在工作空间范围内自动运行，您必须将*工作空间*列入白名单，以允许使用 `Terminal: Manage Workspace Shell Permissions` 命令设置外壳，外壳参数及其环境。

视窗

对于Windows，在终端下拉列表中有一个方便的shell选择器，可让您在几个检测到的shell之间进行选择，包括Command Prompt，PowerShell，PowerShell Core，Git Bash和WSL Bash。如果您希望在其中访问“ **终端：选择默认外壳程序**”命令，也可以通过“命令面板”使用。

就像在其他平台上一样，您可以微调设置文件中使用的确切可执行文件，例如：

```
// Command Prompt
"terminal.integrated.shell.windows": "C:\\Windows\\System32\\cmd.exe"
// PowerShell
"terminal.integrated.shell.windows": "C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe"
// Git Bash
"terminal.integrated.shell.windows": "C:\\Program Files\\Git\\bin\\bash.exe"
// Bash on Ubuntu (on Windows)
"terminal.integrated.shell.windows": "C:\\Windows\\System32\\bash.exe"
```

> **注意：**要用作集成终端，shell可执行文件必须是控制台应用程序，以便 `stdin/stdout/stderr` 可以重定向。

> **提示：**集成终端外壳在VS Code的许可下运行。如果您需要以提升的（管理员）权限或其他权限运行shell命令，则可以 `runas.exe` 在终端中使用平台实用程序。

壳参数

您可以在启动外壳程序时将参数传递给外壳程序。

例如，要启用运行bash作为登录shell（运行 `.bash_profile` ），请传入 `-l` 参数（带双引号）：

```
// Linux
"terminal.integrated.shellArgs.linux": ["-l"]
```

使用变量

在 `shell` ， `shellArgs` ， `env` ，和 `cwd` 终端设置的所有支持解决变量 (https://code.visualstudio.com/docs/editor/variables-reference)：

```
// Open the terminal in the currently opened file's directory
"terminal.integrated.cwd": "${fileDirname}"
```

终端显示设置

您可以使用以下设置来自定义终端的集成字体和行高：

- `terminal.integrated.fontFamily`
- `terminal.integrated.fontSize`
- `terminal.integrated.fontWeight`
- `terminal.integrated.fontWeightBold`
- `terminal.integrated.lineHeight`

终端键绑定

“ **视图：切换集成终端**”命令绑定到 Ctrl +`， 可以快速切换集成终端面板的视图和外观。

以下是可在集成终端中快速导航的键盘快捷键：

键	命令
Ctrl +`	显示集成终端
Ctrl + Shift +`	创建新终端
Ctrl + Alt + PageUp	向上滚动
Ctrl + Alt + PageDown	向下滚动
Shift + PageUp	向上滚动页面
Shift + PageDown	向下滚动页面
Ctrl + Home	滚动到顶部
Ctrl +结束	滚动到底部
未分配	清除终端

其他终端命令可用，并且可以绑定到您首选的键盘快捷键，例如：

- `workbench.action.terminal.focus` ： 聚焦终端。这类似于切换，但是如果可见，则将终端聚焦而不是隐藏终端。
- `workbench.action.terminal.focusNext` ： 聚焦下一个终端实例。
- `workbench.action.terminal.focusPrevious` ： 聚焦上一个终端实例。
- `workbench.action.terminal.focusAtIndexN` ： 将终端对准索引N（N = 1-9）
- `workbench.action.terminal.kill` ： 删除当前的终端实例。
- `workbench.action.terminal.runSelectedText` ： 在终端实例中运行选定的文本。
- `workbench.action.terminal.runActiveFile` ： 在终端实例中运行活动文件。

复制粘贴

复制和粘贴的键绑定遵循平台标准：

- Linux： Ctrl + Shift + C 和 Ctrl + Shift + V
- macOS： Cmd + C 和 Cmd + V
- Windows： Ctrl + C 和 Ctrl + V

右键点击行为

右键单击行为因平台而异：

- Linux：显示上下文菜单。
- macOS：选择光标下的单词并显示上下文菜单。
- Windows：如果有选择，则复制并拖放选择，否则粘贴。

可以使用 `terminal.integrated.rightClickBehavior` 设置进行配置。

强制键绑定通过终端

当焦点集中在集成终端中时，由于击键被传递到终端本身并由终端自身使用，因此许多键绑定将不起作用。有一个硬编码的命令列表，这些命令跳过了外壳程序的处理，而是发送到VS Code键盘绑定系统。您可以使用 `terminal.integrated.commandsToSkipShell` 设置来自定义此列表。通过将命令名称添加到列表中，可以将命令添加到此列表中，而通过将命令名称添加到前缀为的列表中，可以删除命令 `-` 。

```
{
  "terminal.integrated.commandsToSkipShell": [
    // Ensure the toggle sidebar visibility keybinding skips the shell
    "workbench.action.toggleSidebarVisibility",
    // Send quick open's keybinding to the shell
    "-workbench.action.quickOpen",
  ]
}
```

查看设置详细信息以查看默认命令的完整列表。

终端中的和弦键绑定

默认情况下，当和弦快捷键是最高优先级的快捷键时，它将始终跳过终端外壳（绕过 `terminal.integrated.commandsToSkipShell` ），并由VS Code而不是终端进行评估。除非您在Windows / Linux上并且希望您的外壳使用ctrl + k（对于bash而言，这会在光标后剪切行），否则通常这是所需的行为。可以使用以下设置禁用此设置：

```
{
  "terminal.integrated.allowChords": false
}
```

找

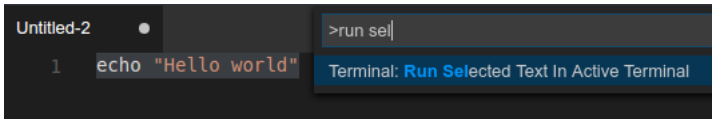
集成终端具有基本的查找功能，可以通过 unassigned 触发。

如果要让 Ctrl + F 转到外壳程序而不是在Linux和Windows上启动“查找”小部件，则需要删除键绑定，如下所示：

```
// Windows/Linux
{ "key": "ctrl+f", "command": "-workbench.action.terminal.focusFindWidget",
                     "when": "terminalFocus" },
// macOS
{ "key": "cmd+f",  "command": "-workbench.action.terminal.focusFindWidget",
                     "when": "terminalFocus" },
```

运行选定的文本

要使用该 `runSelectedText` 命令，请在编辑器中选择文本，然后运行命令**Terminal：通过命令面板**（ Ctrl + Shift + P ）**在Active Terminal中运行选定的文本**：



终端将尝试运行所选文本。

如果在活动编辑器中未选择任何文本，则光标所在的行将在终端中运行。

发送来自绑定的文本

该 `workbench.action.terminal.sendSequence` 命令可用于向终端发送特定的文本序列，包括转义序列。这使诸如发送箭头键，输入，光标移动等操作成为可能。下面的示例显示了使用此功能可以实现的功能，它会跳过光标左侧的单词（Ctrl +向左键）并按退格键：

```
{
  "key": "ctrl+u",
  "command": "workbench.action.terminal.sendSequence",
  "args": { "text": "\u001b[1;5D\u007f" }
}
```

此功能支持变量替换 (/docs/editor/variables-reference)。

请注意，该命令仅适用于 `\u0000` 通过字符代码使用字符的格式（不适用于 `\x00` ）。您可以阅读有关这些十六进制代码的更多信息，并且终端可以在以下资源上使用这些序列：

- XTerm控制序列 (http://invisible-island.net/xterm/ctlseqs/ctlseqs.html)
- C0和C1控制代码列表 (https://github.com/xtermjs/xterm.js/blob/0e45909c7e79c83452493d2cd46d99c0a0bb585f/src/common/data/EscapeSequences.ts)

重命名终端会话

现在可以使用**Terminal：Rename**（ `workbench.action.terminal.rename` ）命令**重命名** Integrated Terminal会话。新名称将显示在终端选择下拉列表中。

在特定文件夹中打开

默认情况下，终端将在资源管理器中打开的文件夹中打开。该 `terminal.integrated.cwd` 设置允许指定自定义路径打开：

```
{
  "terminal.integrated.cwd": "/home/user"
}
```

Windows上的拆分终端将在父终端开始的目录中启动。在macOS和Linux上，拆分终端将继承父终端的当前工作目录。可以使用以下 `terminal.integrated.splitCwd` 设置更改此行为：

```
{
  "terminal.integrated.splitCwd": "workspaceRoot"
}
```

还有一些扩展可以提供更多选项，例如Terminal Here (https://marketplace.visualstudio.com/items?itemName=Tyriar.vscode-terminal-here)。

更改任务和调试的外壳

您可以设置 `terminal.integrated.automationShell.<platform>` 覆盖任务和调试使用的shell和shell参数：

```
{
    "terminal.integrated.shell.osx": "/usr/local/bin/fish",
    // Use a fully POSIX-compatible shell and avoid running a complex ~/.fishrc
    // for tasks and debug
    "terminal.integrated.automationShell.osx": "/bin/sh"
}
```

更改终端的渲染方式

默认情况下，集成终端将使用多个 `<canvas>` 元素进行渲染，这些元素比DOM更好地进行了调整，以渲染经常更改的交互式文本。但是，Electron / Chromium在某些环境下渲染到画布上的速度较慢，因此VS Code还提供了后备DOM渲染器体验。VS Code会尝试检测性能下降，并为您提供通过通知进行更改的选项。您还可以通过 `terminal.integrated.rendererType` 在用户或工作空间设置中 (/docs/getstarted/settings)进行设置来直接更改渲染。

```
{
    "terminal.integrated.rendererType": "dom"
}
```

可能会提高性能的其他方法是，通过使用启动VS Code来忽略Chromium的GPU禁止列表 `code --ignore-gpu-blacklist` 。

有一个基于WebGL的实验性渲染器也可以启用：

```
{
    "terminal.integrated.rendererType": "experimentalWebgl"
}
```

下一步

终端的基础知识已包含在本文档中，请继续阅读以了解有关以下内容的更多信息：

- 任务 (/docs/editor/tasks) -任务使您可以与外部工具集成并充分利用终端。
- 精通VS Code的终端机 (https://www.growingwiththeweb.com/2017/03/mastering-vscodes-terminal.html) -一个外部博客，其中包含大量有关终端机的高级用户提示。
- 通过在VS Code中浏览您的keybindings.json文件，探索其余的终端命令。

常见问题

我在启动终端时遇到问题

对于此类问题，有专门的故障排除指南 (/docs/supporting/troubleshoot-terminal-launch)。

我可以将集成终端与Linux的Windows子系统一起使用吗？

是的，您可以选择Linux (https://docs.microsoft.com/windows/wsl/install-win10)的Windows子系统 (https://docs.microsoft.com/windows/wsl/install-win10)（WSL）bash shell作为您的终端默认值。如果启用了WSL（通过Windows功能），则可以从“ **选择默认外壳程序**”下拉菜单中选择“ **WSL Bash** ”。有关在WSL和Remote-WSL (https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.remote-wsl)扩展中工作的详细信息，请参见在WSL中开发 (/docs/remote/wsl)。 (/docs/remote/wsl) (https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.remote-wsl)

当终端获得焦点时，VS Code快捷方式X为什么不起作用？

当前，终端使用许多键绑定，从而阻止Visual Studio Code对它们做出反应。例如， `Ctrl + B` 即可在Linux和Windows上打开侧边栏。这是必要的，因为各种终端程序和/或外壳可能会自己响应这些键绑定。您可以使用该 `terminal.integrated.commandsToSkipShell` 设置来防止终端处理特定的按键绑定。

我可以在Windows的终端上使用Cmder的外壳吗？

是的，要在VS Code中使用Cmder (http://cmder.net/) shell，您需要在 `settings.json` 文件中添加以下设置：

```
"terminal.integrated.shell.windows": "C:\\WINDOWS\\System32\\cmd.exe",
"terminal.integrated.shellArgs.windows": ["/K", "C:\\cmder\\vendor\\init.bat"]
```

您可以参考Cmder的Wiki (https://github.com/cmderdev/cmder/wiki/Seamless-VS-Code-Integration)以获得更多信息。

macOS上的PowerShell抱怨" -l"参数，如何解决?

在将集成终端配置为在macOS上使用PowerShell时，您可能会因抱怨参数而遇到此错误 (https://github.com/Microsoft/vscode/issues/33022) `"-l"` 。要解决此问题，您将需要覆盖shell args设置，因为它默认 `["-l"]` 情况下默认运行登录shell（对于bash / zsh / etc）。

```
"terminal.integrated.shellArgs.osx": []
```

如何将默认的Windows终端改回PowerShell?

如果要将默认的Integrated Terminal外壳恢复为默认值（Windows上为PowerShell），则可以从用户设置 (/docs/getstarted/settings)（ Ctrl +， ）中删除该外壳替代。

例如，如果您已设置默认终端bash中，你会发现 `terminal.integrated.shell.windows` 在你的 `settings.json` 指向您的bash位置。

```
"terminal.integrated.shell.windows": "C:\\WINDOWS\\System32\\bash.exe",
```

删除该条目以使用内置的VS Code默认值，或将其设置为另一个Shell可执行文件路径。

为什么Cmd + k / Ctrl + k不清除终端?

通常 Cmd + k / Ctrl + k会 清除macOS / Windows上的终端，但是当用户或扩展添加了和弦键绑定时，这可能会停止工作。在 Cmd的+ K / CTRL + K 键绑定依靠VS代码键绑定的优先级系统，该系统定义哪些键绑定是活性在任何给定时间（用户>分机>默认值）。为了解决此问题，您需要重新定义将具有优先级的用户按键绑定，最好在用户 `keybindings.json` 文件的底部：

苹果系统：

```
{ "key": "cmd+k",                "command": "workbench.action.terminal.clear",
                                    "when": "terminalFocus" },
```

视窗：

```
{ "key": "ctrl+k",               "command": "workbench.action.terminal.clear",
                                    "when": "terminalFocus" },
```

为什么在启动集成终端时nvm抱怨前缀选项?

nvm（节点版本管理器）用户经常在VS Code的集成终端中首次看到此错误：

```
nvm is not compatible with the npm config "prefix" option: currently set to "/usr/local"
Run `npm config delete prefix` or `nvm use --delete-prefix v8.9.1 --silent` to unset it
```

这主要是macOS问题，在外部终端中不会发生。造成这种情况的典型原因如下：

- `npm` 已使用 `node` 您路径中某处的另一个实例进行了全局安装（例如 `/usr/local/bin/npm` ）。
- 为了在上使用开发工具 `$PATH` ，VS Code将在启动时启动bash登录shell。这意味着您 `~/.bash_profile` 已经运行，并且在集成终端启动时，它将运行**另一个**登录外壳，从而 `$PATH` 可能以意想不到的方式重新排序。

要解决此问题，您需要跟踪旧 `npm` 版本的安装位置，并删除旧版本及其过期的node_modules。您可以通过查找 `nvm` 初始化脚本并 `which npm` 在其运行之前运行来完成此操作，该脚本应在启动新终端时显示路径。

一旦找到npm的路径，就可以通过运行以下命令来解析符号链接，从而找到旧的node_modules：

```
ls -la /usr/local/bin | grep "np[mx]"
```

最后将为您提供解析路径：

```
... npm -> ../lib/node_modules/npm/bin/npm-cli.js
... npx -> ../lib/node_modules/npm/bin/npx-cli.js
```

从那里删除文件并重新启动VS Code应该可以解决此问题：

```
rm /usr/local/bin/npm /usr/local/lib/node_modules/npm/bin/npm-cli.js
rm /usr/local/bin/npx /usr/local/lib/node_modules/npm/bin/npx-cli.js
```

我可以在集成终端中使用电力线字体吗?

是的，您可以使用设置 (/docs/getstarted/settings)指定Powerline (https://powerline.readthedocs.io)字体。 `terminal.integrated.fontFamily` (/docs/getstarted/settings)

```
"terminal.integrated.fontFamily": "Meslo LG M DZ for Powerline"
```

请注意，您要指定字体系列，而不是像**Powers的Meslo LG M DZ Regular**这样的单个字体，其中**Regular**是特定的字体名称。

如何在macOS上配置zsh以使用Ctrl +向左/向右箭头跳转单词?

默认情况下， Ctrl +向左/向右 箭头将使bash中的单词跳转。您可以通过添加以下键绑定为zsh进行配置：

```
{
  "key": "ctrl+left",
  "command": "workbench.action.terminal.sendSequence",
  "args": { "text": "\u001bb" }
},
{
  "key": "ctrl+right",
  "command": "workbench.action.terminal.sendSequence",
  "args": { "text": "\u001bf" }
}
```

为什么我的终端机显示一个彩色三角形或一个完全黑色的矩形?

终端在某些环境中可能会出现渲染问题，例如，您可能会看到一个大的彩色三角形而不是文本。这通常是由驱动程序/ VM图形问题引起的，在Chromium中也是如此。您可以通过启动解决这些问题 `code` 用 `--disable-gpu` 标志或使用该设置 `"terminal.integrated.rendererType": "dom"` 避免使用画布在终端。

为什么终端的 `$PATH` 环境变量中存在重复的路径和/或为何将它们反向?

在macOS上可能会发生这种情况，因为终端是如何使用VS Code的环境启动的。当VS代码启动的第一次，为了你的源"的发展环境"，它启动您的外壳配置作为**登录shell**，它运行你的 `~/.profile` / `~/.bash_profile` / `~/.zprofile` 脚本。现在，当终端启动时，它也将作为登录外壳运行，它将标准路径放在前面（例如 `/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin` ）并重新初始化外壳环境。

为了更好地理解，您可以通过在操作系统的内置终端内启动内部登录外壳来模拟正在发生的事情：

```
# Add /test to the beginning of $PATH
export PATH=/test:$PATH
# Echo $PATH, /test should be at the beginning
echo $PATH
# Run bash as a login shell
bash -l
# Echo $PATH, the values should be jumbled
echo $PATH
```

不幸的是，与Linux不同，默认情况下，独立的macOS终端全都作为登录Shell运行，因为当用户登录系统时，macOS不会运行登录Shell。这会鼓励"不良行为"，例如在配置文件脚本中的别名应 `rc` 在非登录Shell上运行时应存在于脚本中时进行初始化。

有两个直接的解决方案。您可以设置 `"terminal.integrated.inheritEnv": false` ，这将剥夺大多数环境变量从终端的环境，除了一些重要的（如 `HOME` ， `SHELL` ， `TMPDIR` ，等）。

另一个解决方法是不再通过设置在终端中运行登录Shell `"terminal.integrated.shellArgs": []` 。如果您使用此修复程序，则将要确保将配置文件脚本中的所有别名都移到您的 `~/.bashrc` / `~/.zshrc` 文件中，因为别名仅适用于设置了它们的shell。

该文档对您有帮助吗?

是 **没有**

6/10/2020