Zsh Hooks：简化指南以增强您的 Shell

在探索了“让 Zsh 在运行网络命令之前显示代理设置”之后，我与 ChatGPT 深入研究了 Zsh hooks。以下是供将来参考的简明概述。

在 Zsh 中，**hooks** 允许您在 Shell 操作的特定点执行自定义函数。除了 `preexec` 之外，Zsh 还提供了多个 hooks 来增强您的环境：

1. `precmd`

- **何时运行**：在提示符显示之前。
- **用途**：更新提示符或执行清理任务。
- **示例**：

```
precmd() {
  echo " 准备执行下一条命令！ "
}
```

2. `chpwd`

- **何时运行**：当前目录发生变化时。
- **用途**：更新环境变量或根据目录触发操作。
- **示例**：

```
chpwd() {
  echo " 已切换到：$PWD"
}
```

3. `preexec_functions` & `precmd_functions`

- **何时运行**：类似于 `preexec` 和 `precmd`，但支持多个函数。
- **用途**：附加多个操作而不覆盖现有 hooks。
- **示例**：

```
precmd_functions+=(additional_precmd)

additional_precmd() {
```

```
  echo " 执行额外的 precmd 任务。"
}
```

4. **TRAPDEBUG**

- **何时运行**：每条命令执行后，结果显示之前。
- **用途**：调试、记录命令。
- **示例**：

```
TRAPDEBUG() {
  echo " 已执行：$1"
}
```

5. **TRAPEXIT**

- **何时运行**： Shell 退出时。
- **用途**：执行清理任务或显示退出消息。
- **示例**：

```
TRAPEXIT() {
  echo " 再见！"
}
```

6. **zle Hooks**

- **何时运行**：在命令行编辑过程中。
- **用途**：自定义命令行行为。
- **示例**：

```
zle-line-init() {
  echo " 正在编辑新命令。"
}
zle -N zle-line-init
```

7. **历史 Hooks (zshaddhistory, zshremovehistory)**

- **何时运行**：添加或移除历史记录条目时。
- **用途**：过滤或管理历史记录。
- **示例**：

```
zshaddhistory() {
  [[ $1 == *"secret"* ]] && return 1  # 防止添加包含 "secret" 的命令
  return 0
}
```

8. periodic

- **何时运行**：根据 period 参数设定的间隔时间。
- **用途**：执行常规检查或更新。
- **示例**：

```
periodic() {
  echo " 正在运行定期任务..."
}
```

9. add-zsh-hook

- **目的**： Zsh 提供的用于安全添加函数到 hooks 列表的工具。
- **用途**：在不覆盖现有函数的情况下，附加多个函数到 hook。
- **示例**：

```
add-zsh-hook precmd another_precmd

another_precmd() {
  echo " 另一个 precmd 函数。"
}
```

总结

Zsh 的 hook 系统功能强大且灵活，能够实现自动化和自定义：

- **preexec**：在命令执行之前。
- **precmd**：在提示符显示之前。
- **chpwd**：在目录更改时。
- **TRAPDEBUG**：命令执行后用于调试。
- **TRAPEXIT**：Shell 退出时。
- **zle Hooks**：在命令行编辑期间。
- **历史 Hooks**：管理命令历史记录。
- **periodic**：按设定的间隔时间执行。

- **`add-zsh-hook`**：添加多个 hook 函数。

利用这些 hooks 可以大大增强您的 Zsh 使用体验，使您的 Shell 更加高效并符合您的工作流程。