

Zsh Hooks: Streamlined Guide to Enhancing Your Shell

After exploring “Let Zsh Display Proxy Settings Before Running Network Commands,” I delved deeper into Zsh hooks with ChatGPT. Here’s a concise overview for future reference.

In Zsh, **hooks** let you execute custom functions at specific points during shell operations. Beyond **preexec**, Zsh offers several hooks to enhance your environment:

1. precmd

- **When:** Before the prompt is displayed.
- **Use:** Update the prompt or perform cleanup.
- **Example:**

```
precmd() {  
    echo "Ready for the next command!"  
}
```

2. chpwd

- **When:** When the current directory changes.
- **Use:** Update environment variables or trigger actions based on the directory.
- **Example:**

```
chpwd() {  
    echo "Changed to: $PWD"  
}
```

3. preexec_functions & precmd_functions

- **When:** Similar to **preexec** and **precmd** but support multiple functions.
- **Use:** Attach multiple actions without overriding existing hooks.
- **Example:**

```
precmd_functions+=(additional_prcmd)  
  
additional_prcmd() {  
    echo "Additional precmd task."  
}
```

4. TRAPDEBUG

- **When:** After each command, before results display.
- **Use:** Debugging, logging commands.
- **Example:**

```
TRAPDEBUG() {  
    echo "Executed: $1"  
}
```

5. TRAPEXIT

- **When:** When the shell exits.
- **Use:** Cleanup tasks or display exit messages.
- **Example:**

```
TRAPEXIT() {  
    echo "Goodbye!"  
}
```

6. zle Hooks

- **When:** During line editing.
- **Use:** Customize command line behavior.
- **Example:**

```
zle-line-init() {  
    echo "Editing a new command."  
}  
  
zle -N zle-line-init
```

7. History Hooks (zshaddhistory, zshremovehistory)

- **When:** Adding or removing history entries.
- **Use:** Filter or manage history.
- **Example:**

```
zshaddhistory() {  
    [[ $1 == *"secret"* ]] && return 1  
}
```

```
    return 0
}
```

8. periodic

- **When:** At intervals set by `period`.
- **Use:** Routine checks or updates.
- **Example:**

```
periodic() {
    echo "Running periodic task..."
}
```

9. add-zsh-hook

- **Purpose:** Safely add functions to hooks.
- **Use:** Append multiple functions without overwriting.
- **Example:**

```
add-zsh-hook precmd another_precmd

another_precmd() {
    echo "Another precmd function."
}
```

Summary

Zsh's hook system is versatile, enabling automation and customization:

- **preexec:** Before command execution.
- **precmd:** Before prompt display.
- **chpwd:** On directory change.
- **TRAPDEBUG:** Post-command for debugging.
- **TRAPEXIT:** On shell exit.
- **zle Hooks:** During line editing.
- **History Hooks:** Managing command history.
- **periodic:** At set intervals.
- **add-zsh-hook:** Adding multiple hook functions.

Utilizing these hooks can greatly enhance your Zsh experience, making your shell more efficient and tailored to your workflow.