# Zsh フック

「Zsh にネットワークコマンド実行前にプロキシ設定を表示させる」を探求した後、ChatGPT と共に Zsh フックについてさらに深く掘り下げました。今後の参考のために、簡潔に概要をまとめます。

---

Zsh では、フックを使用することで、シェル操作の特定の時点でカスタム関数を実行できます。`preexec` 以外にも、Zsh は環境を強化するためのいくつかのフックを提供しています:

**1.** `precmd`

- タイミング: プロンプトが表示される前。
- 用途: プロンプトの更新やクリーンアップを行う。
- 例:

```
precmd() {
  echo " 次のコマンドの準備ができました！"
}
```

**2.** `chpwd`

- タイミング: カレントディレクトリが変更されたとき。
- 用途: 環境変数を更新したり、ディレクトリに基づいてアクションをトリガーする。
- 例:

```
chpwd() {
  echo " ディレクトリを変更しました: $PWD"
}
```

**3.** `preexec_functions` **&** `precmd_functions`

- タイミング: `preexec` や `precmd` と同様ですが、複数の関数をサポートします。
- 用途: 既存のフックを上書きせずに、複数のアクションを追加します。

- 例:

```
precmd_functions+=(additional_precmd)

additional_precmd() {
  echo " 追加の precmd タスク。"
}
```

#### 4. TRAPDEBUG

- タイミング: 各コマンドの実行後、結果が表示される前。
- 用途: デバッグ、コマンドのログ記録。
- 例:

```
TRAPDEBUG() {
  echo " 実行されたコマンド: $1"
}
```

#### 5. TRAPEXIT

- いつ: シェルが終了するとき。
- 用途: クリーンアップタスクや終了メッセージの表示。
- 例:

```
TRAPEXIT() {
  echo " さようなら！"
}
```

#### 6. zle フック

- タイミング: 行編集中。
- 用途: コマンドラインの動作をカスタマイズする。
- 例:

```
zle-line-init() {
  echo " 新しいコマンドを編集しています。"
}
zle -N zle-line-init
```

### 7. ヒストリーフック (zshaddhistory, zshremovehistory)

- タイミング: ヒストリーエントリを追加または削除するとき。
- 用途: ヒストリーをフィルタリングまたは管理する。
- 例:

```
zshaddhistory() {
  [[ $1 == *"secret"* ]] && return 1
  return 0
}
```

### 8. periodic

- タイミング: `period` で設定された間隔で実行されます。
- 用途: 定期的なチェックや更新に使用します。
- 例:

```
periodic() {
  echo " 定期タスクを実行中..."
}
```

### 9. add-zsh-hook

- 目的: フックに関数を安全に追加する。
- 使用法: 既存の関数を上書きせずに複数の関数を追加する。
- 例:

```
add-zsh-hook precmd another_precmd


another_precmd() {
```

```
  echo "Another precmd function."
}
```

### 概要

Zsh のフックシステムは多機能で、自動化とカスタマイズを可能にします：

- `preexec`: コマンド実行前に実行。
- `precmd`: プロンプト表示前に実行。
- `chpwd`: ディレクトリ変更時に実行。
- `TRAPDEBUG`: デバッグ用のコマンド後に実行。
- `TRAPEXIT`: シェル終了時に実行。
- `zle` フック: ライン編集中に実行。
- 履歴フック: コマンド履歴の管理時に実行。
- `periodic`: 設定された間隔で実行。
- `add-zsh-hook`: 複数のフック関数を追加。

これらのフックを活用することで、Zsh の使用体験を大幅に向上させ、シェルをより効率的に、そしてあなたのワークフローに合わせてカスタマイズすることができます。