監控目錄變更的 FSEvents API

我使用的是一個 Jekyll 博客，當我運行 `bundle exec jekyll serve --incremental` 時，終端顯示了 `fsevent_watch`。因此，我向一個 AI 查詢了這個問題。

看來你可能是指 `fsevent_watch`，這並不是大多數編程環境中標準的獨立工具或函數。根據你的問題和常見的慣例，我會假設你在使用 **FSEvents API** 在 macOS 上監控文件系統事件——也許你指的是像 `fsevents` 或相關的工具。如果我誤解了，請澄清！

FSEvents API 是一個專屬於 macOS 的框架，允許應用程序接收有關目錄樹變化的通知。它通常用於 C、Objective-C 或 Swift 程式設計中，用於監控文件系統事件，如文件創建、刪除或修改。由於 `fsevent_watch` 不是一個被認可的命令或函數（可能是拼寫錯誤或第三方工具），我將解釋如何直接使用 FSEvents API，這可能是你需要的。

以下是使用 FSEvents 監控目錄的實用指南：

在 C/Objective-C 中使用 FSEvents

FSEvents API 是 macOS 上 CoreServices 框架的一部分。以下是一個基本的 C 範例，用於監控目錄變化：

1. **包含必要的標頭**：你需要從 CoreServices 框架中包含 FSEvents 標頭。
2. **設置事件流**：創建一個事件流來監控特定目錄，定義一個回調函數來處理事件，並將其與運行循環進行調度。
3. **處理事件**：回調函數處理事件（例如，文件變更、刪除）並提供路徑和標誌。

以下是一個最小範例：

```
#include <CoreServices/CoreServices.h>
#include <stdio.h>

// 回調函數來處理文件系統事件
void callback(
    ConstFSEventStreamRef streamRef,
    void *clientCallBackInfo,
    size_t numEvents,
    void *eventPaths,
    const FSEventStreamEventFlags eventFlags[],
    const FSEventStreamEventId eventIds[])
{
```

```
    char **paths = (char **)eventPaths;
    for (size_t i = 0; i < numEvents; i++) {
        printf(" 變更檢測到: %s (標誌: 0x%x)\n", paths[i], eventFlags[i]);
    }
}

int main() {
    // 要監控的路徑（例如，你的家目錄）
    CFStringRef myPath = CFStringCreateWithCString(NULL, "/Users/yourusername", kCFStringEncodingUTF8);
    CFArrayRef pathsToWatch = CFArrayCreate(NULL, (const void **)&myPath, 1, NULL);

    // 創建事件流
    FSEventStreamRef stream = FSEventStreamCreate(
        NULL,           // 分配器
        &callback,      // 回調函數
        NULL,           // 回調信息（上下文）
        pathsToWatch,   // 要監控的路徑
        kFSEventStreamEventIdSinceNow, // 從現在開始的事件
        1.0,            // 延遲（秒數，批處理延遲）
        kFSEventStreamCreateFlagNone // 標誌（可以使用 kFSEventStreamCreateFlagFileEvents 以獲取文件級事件）
    );

    // 在當前運行循環中調度流
    FSEventStreamScheduleWithRunLoop(stream, CFRunLoopGetCurrent(), kCFRunLoopDefaultMode);

    // 開始流
    FSEventStreamStart(stream);

    // 運行循環以監聽事件
    printf(" 監控 /Users/yourusername... 按 Ctrl+C 結束。\n");
    CFRunLoopRun();

    // 清理（在這個例子中不可達，除非停止）
    FSEventStreamStop(stream);
    FSEventStreamInvalidate(stream);
    FSEventStreamRelease(stream);
    CFRelease(pathsToWatch);
    CFRelease(myPath);

    return 0;
```

```
}
```

步驟說明：

- **要監控的路徑**：你在 `CFArray` 中指定一個或多個目錄（例如，`/Users/yourusername`）。
- **回調**：當變更發生時，`callback` 函數被觸發。它接收受影響的路徑數組和標誌(例如，`kFSEventStreamEventFlagItem`
- **流創建**：`FSEventStreamCreate` 設置監視器。延遲（例如，`1.0` 秒）批處理事件以避免淹沒回調。
- **運行循環**：流與運行循環（例如，`CFRunLoopRun`）集成，以保持程序活躍和響應。
- **清理**：完成後停止並釋放流（在這個無限循環例子中未顯示）。

編譯： 使用 CoreServices 框架編譯：

```
gcc -o fsevents_example fsevents_example.c -framework CoreServices
```

運行它：

```
./fsevents_example
```

關鍵點：

- **細粒度**：默認情況下，FSEvents 監控目錄級變更。要獲取文件級事件（例如，特定文件修改），添加 `kFSEventStreamCreateFlagFileEvents` 標誌。
- **延遲**：延遲參數（例如，`1.0`）控制回調的觸發頻率。較低的值提供更快的更新，但更多的調用。
- **範圍**：FSEvents 遞歸監控整個目錄樹——無需指定子目錄。

替代方案：使用工具

如果你想要一個命令行工具而不是自己編寫，標準 macOS 工具包中沒有 `fsevent_watch`。然而，第三方工具如 `fswatch` 利用 FSEvents。要使用 `fswatch`：

1. **安裝 fswatch**（例如，通過 Homebrew）：

   ```
   brew install fswatch
   ```

2. **運行它**：

   ```
   fswatch /path/to/watch
   ```

   當變更發生時，這將輸出路徑。例如：

   ```
   fswatch ~/Documents
   ```

   你將看到類似 `/Users/yourusername/Documents/file.txt` 的行，當文件變更時。

3. **與腳本一起使用**：將其管道到腳本以進行自定義操作：

   ```
   fswatch ~/Documents | while read -r file; do echo " 變更: $file"; done
   ```

如果你指的是其他東西

如果 `fsevent_watch` 參考的是特定的庫、工具或拼寫錯誤（例如，`fsevents-watch`），請提供更多上下文——例如你正在使用的編程語言或環境，我將進一步調整回答。

讓我知道我如何可以為你改進這個！