Überwachung von Verzeichnisänderungen mit der FSEvents-API

Ich habe einen Jekyll-Blog verwendet, und als ich `bundle exec jekyll serve --incremental` ausführte, zeigte das Terminal `fsevent_watch`. Daraufhin habe ich eine KI danach gefragt.

Es scheint, dass Sie möglicherweise `fsevent_watch` meinen, was kein Standard-Tool oder eine eigenständige Funktion in den meisten Programmierkontexten ist. Basierend auf Ihrer Frage und den gängigen Konventionen werde ich annehmen, dass Sie sich mit der **FSEvents API** auf macOS beschäftigen, um Dateisystemereignisse zu überwachen –vielleicht meinen Sie etwas wie `fsevents` oder ein verwandtes Hilfsprogramm. Wenn ich das falsch verstanden habe, bitte klären Sie das!

Die FSEvents API ist ein macOS-spezifischer Framework, der Anwendungen ermöglicht, Benachrichtigungen über Änderungen in einem Verzeichnisbaum zu erhalten. Es wird häufig in C, Objective-C oder Swift verwendet, um Dateisystemereignisse wie das Erstellen, Löschen oder Ändern von Dateien zu überwachen. Da `fsevent_watch` kein anerkannter Befehl oder eine Funktion ist (es könnte ein Tippfehler oder ein Drittanbieter-Tool sein), werde ich erklären, wie man die FSEvents API direkt verwendet, was wahrscheinlich das ist, was Sie suchen.

Hier ist eine praktische Anleitung zur Verwendung von FSEvents, um ein Verzeichnis zu überwachen:

Verwendung von FSEvents in C/Objective-C

Die FSEvents API ist Teil des CoreServices-Frameworks auf macOS. Hier ist ein grundlegendes Beispiel in C, um ein Verzeichnis auf Änderungen zu überwachen:

1. **Notwendige Header einbinden**: Sie müssen den FSEvents-Header aus dem CoreServices-Framework einbinden.
2. **Ereignisstrom einrichten**: Erstellen Sie einen Ereignisstrom, um ein bestimmtes Verzeichnis zu überwachen, definieren Sie eine Rückruf-Funktion, um Ereignisse zu verarbeiten, und planen Sie sie mit einer Laufschleife.
3. **Ereignisse verarbeiten**: Die Rückruf-Funktion verarbeitet die Ereignisse (z. B. Datei geändert, gelöscht) und liefert Pfade und Flags.

Hier ist ein minimales Beispiel:

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

// Rückruf-Funktion, um Dateisystemereignisse zu verarbeiten
```

```c
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("Änderung erkannt bei: %s (Flags: 0x%x)\n", paths[i], eventFlags[i]);
    }
}

int main() {
    // Zu überwachender Pfad (z. B. Ihr Home-Verzeichnis)
    CFStringRef myPath = CFStringCreateWithCString(NULL, "/Users/yourusername", kCFStringEncodingUTF8);
    CFArrayRef pathsToWatch = CFArrayCreate(NULL, (const void **)&myPath, 1, NULL);

    // Erstellen des Ereignisstroms
    FSEventStreamRef stream = FSEventStreamCreate(
        NULL,           // Allokator
        &callback,      // Rückruf-Funktion
        NULL,           // Rückruf-Info (Kontext)
        pathsToWatch,   // Zu überwachende Pfade
        kFSEventStreamEventIdSinceNow, // Starten Sie mit Ereignissen ab jetzt
        1.0,            // Latenz in Sekunden (Batching-Verzögerung)
        kFSEventStreamCreateFlagNone // Flags (kFSEventStreamCreateFlagFileEvents für ereignisbasierte Dateie
    );

    // Planen Sie den Strom in der aktuellen Laufschleife
    FSEventStreamScheduleWithRunLoop(stream, CFRunLoopGetCurrent(), kCFRunLoopDefaultMode);

    // Starten Sie den Strom
    FSEventStreamStart(stream);

    // Laufschleife ausführen, um auf Ereignisse zu hören
    printf("Überwache /Users/yourusername... Drücken Sie Strg+C, um zu stoppen.\n");
    CFRunLoopRun();
```

```
    // Aufräumen (in diesem Beispiel nicht erreichbar, außer wenn gestoppt)
    FSEventStreamStop(stream);
    FSEventStreamInvalidate(stream);
    FSEventStreamRelease(stream);
    CFRelease(pathsToWatch);
    CFRelease(myPath);

    return 0;
}
```

Schritte erklärt:

- **Zu überwachende Pfade**: Sie geben ein oder mehrere Verzeichnisse (z. B. `/Users/yourusername`) in einem `CFArray` an.
- **Rückruf**: Die `callback`-Funktion wird ausgelöst, wenn Änderungen auftreten. Sie erhält ein Array der betroffenen Pfade und Flags (z. B. `kFSEventStreamEventFlagItemCreated`).
- **Stromerstellung**: `FSEventStreamCreate` richtet den Überwacher ein. Die Latenz (z. B. `1.0` Sekunden) gruppiert Ereignisse, um die Rückruf-Funktion nicht zu überlasten.
- **Laufschleife**: Der Strom wird in eine Laufschleife (wie `CFRunLoopRun`) integriert, um das Programm am Leben und reaktionsfähig zu halten.
- **Aufräumen**: Stoppen und freigeben Sie den Strom, wenn Sie fertig sind (in diesem Beispiel nicht gezeigt).

Kompilierung: Kompilieren Sie mit dem CoreServices-Framework:

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

Führen Sie es aus:

```
./fsevents_example
```

Wichtige Punkte:

- **Granularität**: Standardmäßig überwacht FSEvents Änderungen auf Verzeichnisebene. Für ereignisbasierte Dateien (z. B. spezifische Dateiänderungen) fügen Sie das Flag `kFSEventStreamCreateFlagFileEvents` hinzu.
- **Latenz**: Der Latenzparameter (z. B. `1.0`) steuert, wie oft die Rückruf-Funktion aufgerufen wird. Niedrigere Werte geben schnellere Aktualisierungen, aber mehr Aufrufe.
- **Bereich**: FSEvents überwacht den gesamten Verzeichnisbaum rekursiv –es ist nicht notwendig, Unterverzeichnisse anzugeben.

Alternative: Verwendung eines Tools

Wenn Sie ein Befehlszeilen-Tool anstelle des eigenen Codierens suchen, gibt es kein `fsevent_watch` im Standard-Toolkit von macOS. Drittanbieter-Tools wie `fswatch` nutzen jedoch FSEvents. Um `fswatch` zu verwenden:

1. **Installieren Sie fswatch** (z. B. über Homebrew):

   ```
   brew install fswatch
   ```

2. **Führen Sie es aus**:

   ```
   fswatch /pfad/zur/überwachung
   ```

   Dies gibt Pfade aus, wenn Änderungen auftreten. Zum Beispiel:

   ```
   fswatch ~/Dokumente
   ```

   Sie werden Zeilen wie `/Users/yourusername/Dokumente/datei.txt` sehen, wenn Dateien geändert werden.

3. **Mit einem Skript**: Leiten Sie es an ein Skript für benutzerdefinierte Aktionen weiter:

   ```
   fswatch ~/Dokumente | while read -r datei; do echo "Geändert: $datei"; done
   ```

Wenn Sie etwas anderes meinten

Wenn `fsevent_watch` sich auf eine spezifische Bibliothek, ein Tool oder einen Tippfehler bezieht (z. B. `fsevents-watch`), geben Sie bitte mehr Kontext an –wie die Programmiersprache oder die Umgebung, in der Sie arbeiten –und ich werde die Antwort weiter anpassen.

Lassen Sie mich wissen, wie ich dies für Sie verfeinern kann!