# Surveillance des modifications de répertoire avec l'API FSEvents

J'utilisais un blog Jekyll, et lorsque j'ai exécuté `bundle exec jekyll serve --incremental`, le terminal a affiché `fsevent_watch`. Par conséquent, j'ai posé la question à une IA.

---

Il semble que vous fassiez référence à `fsevent_watch`, qui n'est pas un outil ou une fonction autonome standard dans la plupart des contextes de programmation. Sur la base de votre question et des conventions courantes, je suppose que vous parlez de l'utilisation de l'API **FSEvents** sur macOS pour surveiller les événements du système de fichiers—peut-être vouliez-vous parler de quelque chose comme `fsevents` ou d'une utilité connexe. Si j'ai mal compris, veuillez clarifier !

L'API FSEvents est un framework spécifique à macOS qui permet aux applications de recevoir des notifications concernant les modifications dans un arbre de répertoires. Elle est couramment utilisée en C, Objective-C ou Swift pour surveiller les événements du système de fichiers comme la création, la suppression ou la modification de fichiers. Comme `fsevent_watch` n'est pas une commande ou une fonction reconnue (il pourrait s'agir d'une faute de frappe ou d'un outil tiers), je vais expliquer comment utiliser directement l'API FSEvents, qui est probablement ce que vous recherchez.

Voici un guide pratique pour utiliser FSEvents afin de surveiller un répertoire :

### Utilisation de FSEvents en C/Objective-C

L'API FSEvents fait partie du framework CoreServices sur macOS. Voici un exemple de base en C pour surveiller un répertoire pour les modifications :

1. **Inclure les en-têtes nécessaires** : Vous devrez inclure l'en-tête FSEvents du framework CoreServices.
2. **Configurer le flux d'événements** : Créez un flux d'événements pour surveiller un répertoire spécifique, définissez une fonction de rappel pour traiter les événements et planifiez-la avec une boucle d'exécution.
3. **Traiter les événements** : La fonction de rappel traite les événements (par exemple, fichier modifié, supprimé) et fournit des chemins et des drapeaux.

Voici un exemple minimal :

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

// Fonction de rappel pour traiter les événements du système de fichiers
```

```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("Changement détecté à : %s (Drapeaux : 0x%x)\n", paths[i], eventFlags[i]);
    }
}

int main() {
    // Chemin à surveiller (par exemple, votre répertoire personnel)
    CFStringRef myPath = CFStringCreateWithCString(NULL, "/Users/yourusername", kCFStringEncodingUTF8);
    CFArrayRef pathsToWatch = CFArrayCreate(NULL, (const void **)&myPath, 1, NULL);

    // Créer le flux d'événements
    FSEventStreamRef stream = FSEventStreamCreate(
        NULL,           // Allocateur
        &callback,      // Fonction de rappel
        NULL,           // Informations de rappel (contexte)
        pathsToWatch,   // Chemins à surveiller
        kFSEventStreamEventIdSinceNow, // Commencer avec les événements à partir de maintenant
        1.0,            // Latence en secondes (délai de regroupement)
        kFSEventStreamCreateFlagNone // Drapeaux (peut utiliser kFSEventStreamCreateFlagFileEvents pour les é
    );

    // Planifier le flux sur la boucle d'exécution actuelle
    FSEventStreamScheduleWithRunLoop(stream, CFRunLoopGetCurrent(), kCFRunLoopDefaultMode);

    // Démarrer le flux
    FSEventStreamStart(stream);

    // Exécuter la boucle pour écouter les événements
    printf("Surveillance de /Users/yourusername... Appuyez sur Ctrl+C pour arrêter.\n");
    CFRunLoopRun();
```

```
    // Nettoyage (inaccessible dans cet exemple sauf si arrêté)
    FSEventStreamStop(stream);
    FSEventStreamInvalidate(stream);
    FSEventStreamRelease(stream);
    CFRelease(pathsToWatch);
    CFRelease(myPath);

    return 0;
}
```

**Étapes expliquées :**

- **Chemins à surveiller** : Vous spécifiez un ou plusieurs répertoires (par exemple, `/Users/yourusername`) dans un `CFArray`.
- **Rappel** : La fonction `callback` est déclenchée lorsque des modifications surviennent. Elle reçoit un tableau de chemins affectés et de drapeaux (par exemple, `kFSEventStreamEventFlagItemCreated`).
- **Création du flux** : `FSEventStreamCreate` configure le surveillant. La latence (par exemple, `1.0` seconde) regroupe les événements pour éviter de submerger la fonction de rappel.
- **Boucle d'exécution** : Le flux s'intègre à une boucle d'exécution (comme `CFRunLoopRun`) pour garder le programme en vie et réactif.
- **Nettoyage** : Arrêtez et libérez le flux lorsque vous avez terminé (non montré dans cet exemple de boucle infinie).

**Compilation :** Compilez avec le framework CoreServices :

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

Exécutez-le :

```
./fsevents_example
```

**Points clés :**

- **Granularité** : Par défaut, FSEvents surveille les modifications au niveau des répertoires. Pour les événements au niveau des fichiers (par exemple, modifications spécifiques de fichiers), ajoutez le drapeau `kFSEventStreamCreateFlagFileEvents`.
- **Latence** : Le paramètre de latence (par exemple, `1.0`) contrôle la fréquence à laquelle la fonction de rappel est appelée. Des valeurs plus faibles donnent des mises à jour plus rapides mais plus d'appels.
- **Portée** : FSEvents surveille l'ensemble de l'arbre de répertoires de manière récursive—pas besoin de spécifier les sous-répertoires.

**Alternative : Utilisation d'un outil**

Si vous recherchez un outil en ligne de commande plutôt que de le coder vous-même, il n'y a pas de `fsevent_watch` dans l'arsenal standard de macOS. Cependant, des outils tiers comme `fswatch` utilisent FSEvents. Pour utiliser `fswatch` :

1. **Installer fswatch** (par exemple, via Homebrew) :

   ```
   brew install fswatch
   ```

2. **L'exécuter** :

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

   Cela affiche les chemins lorsque des modifications surviennent. Par exemple :

   ```
   fswatch ~/Documents
   ```

   Vous verrez des lignes comme `/Users/yourusername/Documents/file.txt` lorsque des fichiers changent.

3. **Avec un script** : Pipez-le vers un script pour des actions personnalisées :

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

**Si vous vouliez dire autre chose**

Si `fsevent_watch` fait référence à une bibliothèque spécifique, un outil ou une faute de frappe (par exemple, `fsevents-watch`), veuillez fournir plus de contexte—comme le langage de programmation ou l'environnement dans lequel vous travaillez—et je personnaliserai la réponse davantage.

Faites-moi savoir comment je peux affiner cela pour vous !