MUTEX

OBIETTIVO

Gestire la concorrenza dal lato server per **directory** e **file**. Devo gestire:

- lettura
- scrittura
- listing delle directory

Mutex per la gestione delle directory

- **Mutex per ogni directory:** Ogni directory avrà un mutex per proteggere la sua struttura (evitare modifiche contemporanee mentre un thread esegue il listing).
- **Acquisizione e rilascio del mutex della directory:**
  - Quando un thread fa il listing di una directory, acquisisce il mutex della directory.
  - Una volta completato il listing, il mutex viene rilasciato.

Funzioni per le directory

- Acquisire il mutex per una directory:

```
void lock_directory_mutex(const char *directory_path);
```

- Rilasciare il mutex della directory:

```
void unlock_directory_mutex(const char *directory_path);
```

- Eseguire il listing della directory (acquisisce il mutex della directory):

```
void list_directory(const char *directory_path);
```

Mutex per la gestione dei file

- **Mutex per ogni file:** Ogni file avrà un mutex che viene acquisito prima di accedere al file per evitare conflitti quando più thread cercano di leggere o scrivere simultaneamente sullo stesso file.
- **Acquisizione e rilascio del mutex del file:**
  - Quando un thread deve leggere o scrivere su un file, acquisisce il mutex del file.
  - Una volta completata l'operazione, il mutex del file viene rilasciato.

Funzioni per i file:

- Acquisire il mutex per un file:

```
void lock_file_mutex(const char *file_path);
```

- Rilasciare il mutex del file:

```
void unlock_file_mutex(const char *file_path);
```

- Scrivere su un file (acquisisce il mutex del file):

```
void write_to_file(const char *file_path, const char *data);
```

- Leggere da un file (acquisisce il mutex del file):

```
void read_from_file(const char *file_path);
```

Gstione dinamica dei mutex

Poiché non conosci tutte le directory e i file che potrebbero essere utilizzati, è necessario gestire dinamicamente i mutex per ogni directory e file.

Funzioni di gestione dinamica dei mutex

- **Creare un mutex per una directory (se non esiste già):**

```
void create_directory_mutex(const char *directory_path);
```

- **Creare un mutex per un file (se non esiste già):**

```
void create_file_mutex(const char *file_path);
```

Inizializzazione e pulizia:

- I**nizializzare le strutture per i mutex:** Inizializza le strutture dati che memorizzano i mutex per directory e file.

```
void initialize_mutexes();
```

- **Pulire e deallocare le risorse:** Quando non sono più necessari, dealloca i mutex e le strutture dati.

```
void cleanup_mutexes();
```

Flusso di utilizzo delle funzioni

1. **Listing della directory:**
    - Acquisisci il mutex per la directory con `lock_directory_mutex()`.
    - Esegui il listing dei file con `list_directory()`.
    - Rilascia il mutex della directory con `unlock_directory_mutex()`.
2. **Scrittura su un file:**
    - Acquisisci il mutex per il file con `lock_file_mutex()`.
    - Esegui la scrittura con `write_to_file()`.
    - Rilascia il mutex del file con `unlock_file_mutex()`.
3. **Lettura da un file:**
    - Acquisisci il mutex per il file con `lock_file_mutex()`.
    - Esegui la lettura con `read_from_file()`.
    - Rilascia il mutex del file con `unlock_file_mutex()`.
4. **Gestione dinamica dei mutex:**
    - Usa `create_directory_mutex()` e `create_file_mutex()` per creare e memorizzare mutex per directory e file quando necessario.

Principio di funzionamento

- **Mutex della directory:** Protegge la struttura della directory per evitare che la directory venga modificata (aggiunta, rimozione o rinomina di file) durante il listing dei file. I thread possono **leggere** dalla directory (con `list_directory`) solo quando il mutex è acquisito.
- **Mutex del file:** Protegge l'accesso concorrente ai file, permettendo a un solo thread di **scrivere** o **leggere** un file alla volta.