Esami API Programming 2021 in Rust

15 febbraio 2021

La struttura generica **Exchanger<T>** permette a due thread di scambiarsi un valore di tipo **T**. Essa offre esclusivamente il metodo pubblico **fn exchange(&self, t: T) -> Option<T>**, che blocca il thread chiamante senza consumare CPU fino a che un altro thread non invoca lo stesso metodo sulla stessa istanza. Quando questo avviene, il metodo restituisce l'oggetto passato come parametro dal thread opposto.

Si implementi tale struttura, usando la libreria standard di Rust.

19 giugno 2021

Si implementi una struttura SingleThreadExecutor<F: FnOnce() + Send + 'static> che realizzi il concetto di ThreadPool basato su un singolo thread.

- **Coda dei compiti**: La struttura deve utilizzare una coda per accodare i compiti da eseguire. I compiti vengono rappresentati come funzioni o chiusure che soddisfano i tratti FnOnce() + Send.
- **Metodo submit(...)**: Attraverso questo metodo, è possibile affidare un compito all'istanza di SingleThreadExecutor. I compiti vengono accodati e saranno disponibili per l'elaborazione. Se l'esecutore è stato chiuso, eventuali tentativi di invocare submit(...) devono restituire un errore.
- **Metodo close()**: Questo metodo deve impedire l'ulteriore accodamento di compiti. Dopo la chiamata a close(), eventuali invocazioni di submit(...) devono fallire con un errore. Tuttavia, i compiti già accodati devono poter essere eseguiti.

Si implementi tale classe usando le funzionalità offerte dalla libreria standard di **Rust**, definendo tutte le parti eventualmente mancanti nella definizione della classe. Si faccia attenzione al fatto che il codice che può essere sottomesso all'esecutore è arbitrario e può contenere richieste di sottomissione di ulteriori compiti allo stesso esecutore.

```
struct SingleThreadExecutor<F: FnOnce()> {…}

impl <F: FnOnce()> SingleThreadExecutor<F> {
      pub fn new() -> (Self, Receiver<Box<F>>);
      pub fn submit(&self, task: F) -> Result<(), String>;
      pub fn close(&mut self);
}
```