18 ottobre 2021

La struttura generica Processor<T> consente a un insieme di thread produttori di inviare oggetti istanza del tipo T (che si assume copiabile) a un thread consumatore, il cui comportamento è definito tramite una funzione fornita come parametro del costruttore.

Il costruttore di tale struttura riceve, come parametro, una funzione (o una closure) che accetta un argomento di tipo T e restituisce (). La struttura fornisce una coda per gestire gli oggetti inviati dai produttori e offre i seguenti metodi:

1. **Metodo send(&self, item: T)**: Permette ai produttori di sottomettere un oggetto da elaborare. L'oggetto viene inserito in una coda in attesa che il thread consumatore esterno lo elabori. Se il metodo close(...) è stato invocato, eventuali chiamate a send(...) devono generare un errore o produrre un comportamento indefinito, a scelta dell'implementazione.
2. **Metodo close(&self)**: Segnala la fine dell'accettazione di nuovi elementi. Dopo l'invocazione di questo metodo:
    - Non sarà più possibile inviare nuovi dati tramite send(...).
    - Il metodo non ritorna fino a quando la coda non è vuota e tutte le operazioni di elaborazione sono state completate.

La struttura Processor<T> deve garantire la sincronizzazione tra i produttori e il consumatore esterno utilizzando primitive di sincronizzazione di Rust. La logica di elaborazione e il ciclo di vita del thread consumatore devono essere gestiti esternamente.

Viene fornita la seguente dichiarazione di struct Rust come punto di partenza:

```
impl<T: Send + 'static> Processor<T> {
    fn new<F>(f: F) -> Self where F: Fn(T) + Send + 'static {}
    fn send(&self, item: T) {}
    fn close(&self) {}
}
```