

Programmazione di sistema

Iniziato	martedì, 6 giugno 2023, 13:48
Stato	Completato
Terminato	martedì, 6 giugno 2023, 13:50
Tempo impiegato	2 min. 16 secondi
Valutazione	0,00 su un massimo di 15,00 (0%)

Domanda 1

Risposta non data

Punteggio max.:
3,00

Definire il problema della dipendenze cicliche, includendo il motivo per cui si possa generare, ed indicare come in Rust sia possibile risolverle, fornendo un esempio pratico.



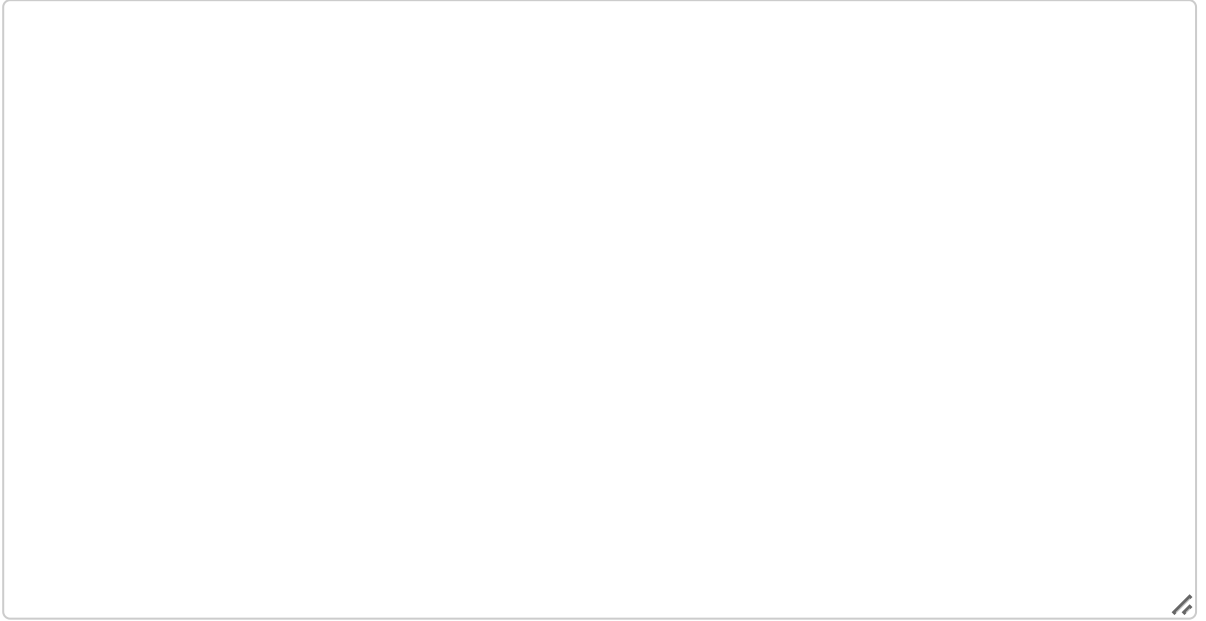
Domanda 2

Risposta non data

Punteggio max.:

3,00

Come si utilizza il sistema di gestione dei processi in Rust attraverso il pacchetto `std::process`? Quali sono le funzionalità offerte dalla libreria e come si gestiscono i processi figli nell'ambito del proprio programma?

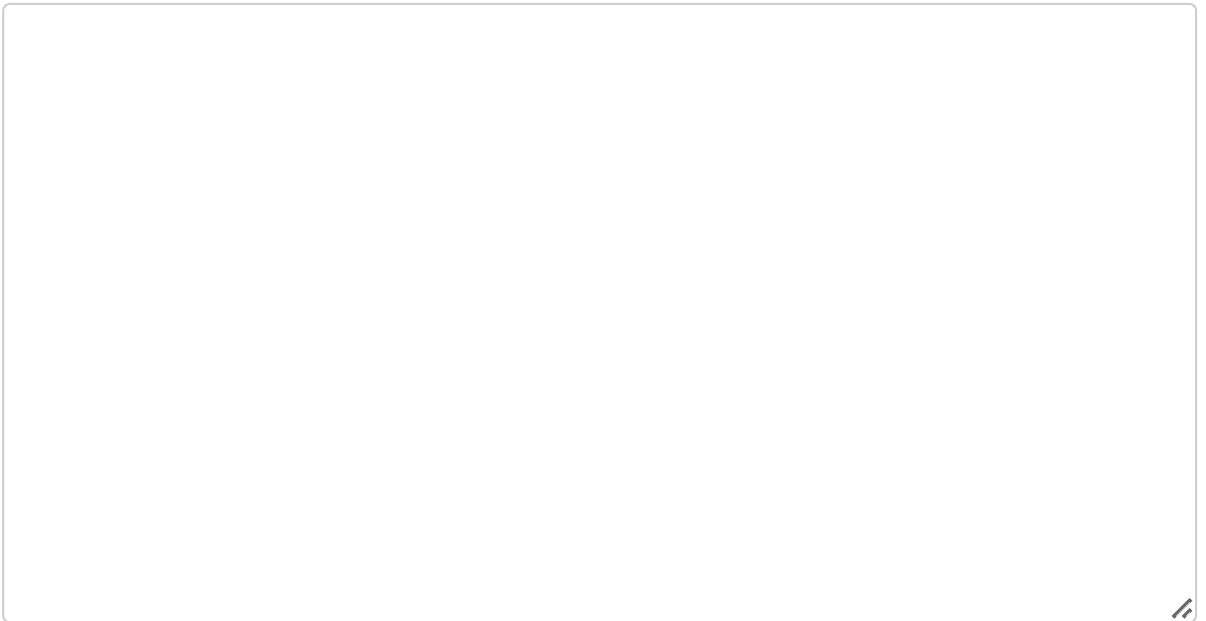
A large, empty rectangular box with a thin gray border, intended for the user's answer to the question. In the bottom right corner, there is a small, faint icon of a pencil.**Domanda 3**

Risposta non data

Punteggio max.:

3,00

Nel caso di programmazione concorrente in Rust e memoria condivisa quali sono i principali costrutti e come vengono utilizzati per evitare la condizione di deadlock?

A large, empty rectangular box with a thin gray border, intended for the user's answer to the question. In the bottom right corner, there is a small, faint icon of a pencil.

Domanda 4

Risposta non data

Punteggio max.:

6,00

Una barriera è un costrutto di sincronizzazione usato per regolare l'avanzamento relativo della computazione di più thread. All'atto della costruzione di questo oggetto, viene indicato il numero N di thread coinvolti. Non è lecito creare una barriera con meno di 2 thread.

La barriera offre un solo metodo, `wait()`, il cui scopo è bloccare temporaneamente l'esecuzione del thread che lo ha invocato, non ritornando fino a che non sono giunte altre N-1 invocazioni dello stesso metodo da parte di altri thread: quando ciò succede, la barriera si sblocca e tutti tornano. Successive invocazioni del metodo `wait()` hanno lo stesso comportamento: la barriera è ciclica. **Attenzione a non mescolare le fasi di ingresso e di uscita!**

Una `RankingBarrier` è una versione particolare della barriera in cui il metodo `wait()` restituisce un intero che rappresenta l'ordine di arrivo: il primo thread ad avere invocato `wait()` otterrà 1 come valore di ritorno, il secondo thread 2, e così via. All'inizio di un nuovo ciclo, il conteggio ripartirà da 1.

Si implementi la struttura dati `RankingBarrier` a scelta in linguaggio Rust o C++ '11 o successivi.

