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?

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?

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

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 che coinvolga 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 nei linguaggi Rust o C++ '11 o successivi.