

Prova pratica del 17/12/2024 - Traccia A – Codice Teams: tkapgf
Durata della prova: 100 minuti

Lo studente completi il programma a corredo di questo documento, in base alle indicazioni qui riportate. La prova sarà valutata come segue:

- **A:** Prova svolta correttamente.
- **B:** Il programma non esegue correttamente, con errori minori di programmazione o di concorrenza.
- **C:** Il programma non esegue correttamente, con errori significativi (voto max: 22).
- **INSUFFICIENTE:** Il programma non compila o non esegue, con errori gravi di sincronizzazione.

Testo della prova

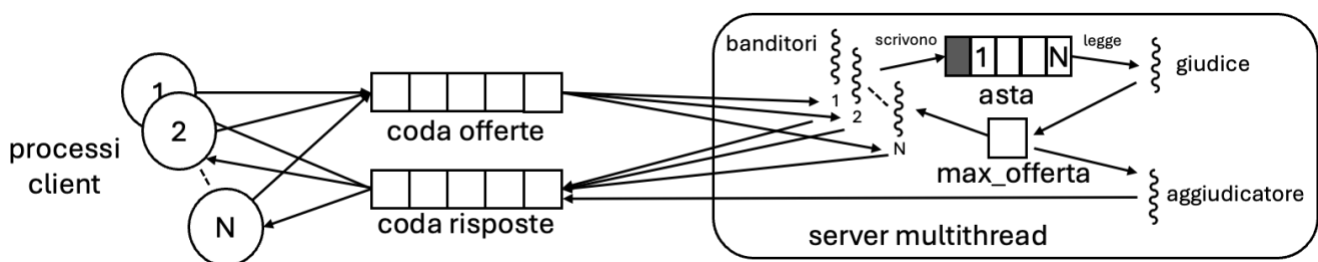
Si realizzi in linguaggio C un'applicazione client-server con **client multiprocesso** e **server multithread**, che simuli un'asta. La comunicazione tra client e server è realizzata con **code di messaggi**.

Gli N processi **client** inviano le loro offerte su una **coda di offerte**, inserendo un proprio id numerico (da 1 a N) e l'offerta generata casualmente, finché l'asta non è aggiudicata. Dopo ogni offerta il client attende la risposta del server su una **coda di risposte**. La risposta contiene lo stato dell'asta (se aggiudicata o meno e se si da quale client) e il massimo valore correntemente offerto, in base al quale il client aggiorna la sua base d'asta per poter rilanciare al ciclo successivo dopo un'attesa di 1 secondo. Quando l'asta risulta aggiudicata, il client termina riconoscendo il vincitore (se stesso o un altro client).

Come rappresentato in figura, il processo **server** è costituito da **3 tipologie di thread**:

- N thread **banditore**, uno per ciascun client; ogni banditore riceve le offerte del proprio client sulla coda di offerte, scrive su un **array asta** l'offerta ricevuta nella posizione corrispondente al proprio client (la posizione 0 non è utilizzata) e risponde sulla coda di risposte con la massima offerta corrente, memorizzata in una struttura **max_offerta**, finché l'asta non è terminata;
- 1 thread **giudice**, che periodicamente (ogni 2 secondi) accede in lettura all'array **asta** per determinare la massima offerta che poi memorizza in **max_offerta**. Dopo un numero predeterminato di turni, il thread giudice chiude l'asta stabilendo che la **max_offerta** corrente è quella aggiudicata e segnalando il thread aggiudicatore;
- 1 thread **aggiudicatore**, il quale è in attesa finché la **max_offerta** non è aggiudicata; quando viene segnalato dal thread giudice, invia un messaggio a tutti i client sulla coda di risposta con lo stato finale dell'asta (asta aggiudicata, con quale massima offerta e da quale client).

L'array **asta** è acceduto dai banditori e dal giudice attraverso lo schema **lettore-scrittore con starvation di entrambi**. La struttura **max_offerta** deve essere acceduta in **mutua esclusione**.



I processi client e il server multithread sono implementati in due eseguibili separati, **client_asta** e **server_asta**, che vengono avviati da un unico processo **main** attraverso **fork-exec**. Il processo **client_asta** genera 5 processi client (N=5); il processo **server_asta** genera 5 thread banditori, oltre che il giudice e l'aggiudicatore; il giudice effettua 5 turni prima di chiudere l'asta.