Università degli Studi di Napoli Federico II
Corso di Laurea in Ingegneria Informatica
Esame di Sistemi Operativi

Prova pratica 28/09/2023
Durata della prova: 75 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/C++ un processo servente **multithread** basato su **produttore-consumatore**. Il processo servente, denominato **Server**, riceve richieste di aggiornamento di un contatore Count, rappresentante il numero totale di prenotazioni (ad es. di posti in un ristorante o simili) provenienti da un gruppo di 3 processi **Client**. Ogni client invia 10 richieste, costituite da un numero di prenotazioni generato casualmente tra 1 e 5. Ogni richiesta viene depositata nella coda, gestita come **Coda** circolare condivisa di interi e protetta da opportuni **semafori**. La coda è costituita da un buffer di 5 numeri interi. Ogni Client si comporta dunque come un **Produttore** delle richieste di prenotazione nella Coda. Una volta inviate le 10 richieste, i processi client inviano una richiesta con valore -1 e terminano.

Il processo Servente è costituito da 3 thread **Worker** i quali sono in attesa di richieste di prenotazione sulla Coda. Ogni Worker si comporta come un **Consumatore** di richieste di prenotazione. Una volta prelevato un elemento dalla Coda, il thread Worker aggiorna il contatore condiviso Count (in mutua esclusione) e torna ad aspettare una nuova richiesta finchè non riceve una richiesta con valore -1, la quale provoca la terminazione del thread stesso e di tutti gli altri thread. A tal fine, i thread condividono una variabile intera **end**, inizializzata a 0, e posta a 1 dal primo Worker che preleva il valore -1 dalla Coda. Il valore di end posto a 1 provoca la terminazione di tutti i Worker.

Il codice dei Client e del Server deve risiedere in **due eseguibili distinti**, che sono invocati da un **programma principale** attraverso una delle varianti della primitiva **exec**. Quando tutti i Client e il Server terminano, il programma principale rimuove la coda e i semafori e termina a sua volta.