Università degli Studi di Napoli Federico II
Corso di Laurea in Ingegneria Informatica
Esame di Sistemi Operativi
Proff. Cinque, Cotroneo, Natella

Prova pratica del 26/10/2023
Durata della prova: 75 minuti

ooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo

Lo studente completi il programma a corredo di questo documento, seguendo le seguenti indicazioni.

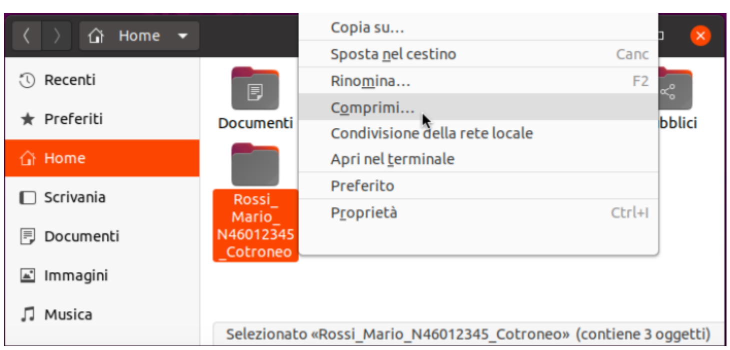
La prova sarà valutata come segue:
- **A**: Prova svolta correttamente.
- **B**: Il programma non esegue correttamente, con errori minori di programmazione o di concorrenza.
- **C**: Il programma non esegue correttamente, con errori significativi (voto max: 22).
- **INSUFFICIENTE**: Il programma non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

Istruzioni per la consegna dell'elaborato

L'elaborato dovrà essere svolto in una cartella dal nome: `Cognome_Nome_Matricola_Docente`

Esempio:
```
mkdir Rossi_Mario_N46012345_Cotroneo
cd Rossi_Mario_N46012345_Cotroneo
```
…. copiare nella cartella i file forniti per l'esame ….
…. sviluppare il programma ….
…. per la consegna, dalla shell (assumendo di essere ancora nella cartella di lavoro), creare un file compresso ("tar") con i seguenti comandi:
```
cd ..
tar -czvf ./Rossi_Mario_N46012345_Cotroneo.tar.gz  ./Rossi_Mario_N46012345_Cotroneo
```

In alternativa, è consentito creare il file compresso "tar" tramite l'interfaccia grafica.



All'interno della macchina virtuale, aprire il browser, collegarsi al Team dell'esame ed effettuare la consegna facendo l'upload del file compresso appena creato.

ooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo

Testo della prova

Si realizzi in linguaggio C/C++ un programma **multithread** che simuli il sottosistema di I/O di un sistema operativo. Il programma dovrà gestire un **vettore di buffer** in memoria, sui cui elementi vengono depositati temporaneamente i blocchi di dati prodotti dalle applicazioni. per essere trasferiti su disco in un secondo momento da parte di un "paging daemon" del sistema operativo. Per semplicità, si assuma che ogni buffer contenga un solo carattere. Gli accessi al vettore dovranno essere disciplinati tramite il costrutto Monitor, utilizzando lo schema del **produttore-consumatore con vettore di stato**.

Il programma dovrà avviare un gruppo di 3 thread che assumano il ruolo di **produttori**, rappresentando le applicazioni. Il monitor dovrà fornire un metodo `produzione()`, che inserisca un valore in un singolo buffer, e che sospenda il thread chiamante in caso che il vettore di buffer sia completamente pieno. Si simuli che la produzione abbia una durata di tempo variabile (scelta casualmente tra 1 e 3 secondi), utilizzando `sleep()` all'interno del metodo del monitor. Ognuno dei thread produttori dovrà effettuare 5 produzioni, attendendo 1 secondo prima di ogni produzione mediante la primitiva `sleep()`.

Il programma dovrà inoltre avviare un thread che assuma il ruolo di **consumatore**, rappresentando il "paging daemon" del sistema operativo. Il monitor dovrà fornire un metodo `consumazione()`, che prelevi un singolo elemento da uno dei buffer pieni, e che sospenda il chiamante se il vettore è completamente vuoto.

Inoltre, il metodo dovrà fornire un metodo `attesa()`, che non effettua alcuna consumazione. Il metodo si limita a verificare se vi sono almeno 3 buffer pieni nel vettore. In caso affermativo, il metodo esce immediatamente dal monitor, restituendo in uscita il numero di buffer pieni. In caso negativo, il metodo sospende il thread chiamante (su una condition variable aggiuntiva), fino a quando i produttori non abbiano inserito almeno 3 elementi nel vettore.

Il consumatore dovrà effettuare in totale 15 consumazioni. Ripetutamente, il consumatore dovrà chiamare 1 volta il metodo `attesa()`, ottenere il numero di buffer occupati, e poi chiamare più volte il metodo `consumazione()`, in base al numero di buffer occupati indicati dal metodo `attesa()`, senza utilizzare `sleep()` tra le operazioni.