

Tracce di esercizi in C per Sistemi Operativi

Processi

1. Creare un programma che legge un file e conta le occorrenze di una parola nel seguente modo:
 - Il primo processo legge dall'inizio alla metà e conta le occorrenze della parola.
 - Il secondo processo legge dalla metà fino alla fine e conta le occorrenze della parola.
 - Inviano poi il numero di occorrenze al processo padre, il quale le somma e le stampa a video.
2. Generare due processi figli che comunicano con il padre.
 - Uno dei processi genera numeri casuali [0-100] ed invia al padre solo i numeri pari.
 - L'altro processo genera numeri casuali [0-100] ed invia al padre solo i numeri dispari.
 - Il padre fa la loro somma e quando la somma > 190, termina l'esecuzione dei figli.
3. Un processo padre genera due processi figli.
 - Il primo processo figlio invia al padre un numero casuale da 0 a 100.
 - Il padre legge questo numero, lo moltiplica per un k casuale e lo manda al secondo figlio.
 - Il secondo figlio legge il numero inviato dal padre e lo stampa a video.
4. Due processi leggono dallo stesso file che si trova all'interno di una directory, (es. `/data/file.txt`).

Alternativa - Usare `dirent` e `readdir` per leggere all'interno della directory e controllare se il file esiste o meno. Controllare che il file sia in modalità lettura, altrimenti restituire errore.

Alternativa - invece di restituire un errore, cambiare i permessi al file con `chmod`.

 - Il primo processo legge dall'inizio del file fino a metà,
 - Il secondo legge dalla metà in poi. I figli mandano il contenuto al padre
 - Il padre lo stampa nel seguente formato: `[PID_FIGLIO] -> TESTO`
5. Scrivere un programma che esegue la moltiplicazione tra matrici 3x3 usando la programmazione parallela.
 - Il primo processo figlio computa la prima colonna.
 - Il secondo processo figlio computa la seconda colonna.
 - Il processo padre computa la terza colonna e riceve dai figli i due vettori colonna computati e compone la matrice finale e la stampa.
6. Generare due processi figli che comunicano con il padre.
 - Uno dei processi genera numeri casuali [0-50] ed invia al padre solo i numeri multipli di 3.
 - L'altro processo genera numeri casuali [51-100] ed invia al padre solo i numeri multipli di 2.
 - Il padre stampa i numeri ricevuti ed esegue la loro somma quando la somma > 130.
7. Scrivere un programma C che crea un processo figlio.
 - Il padre legge un numero casuale e lo invia al figlio attraverso una pipe.
 - Il figlio fa il quadrato del numero e lo invia nuovamente al padre solo se il quadrato è pari, attraverso la pipe.
 - Il padre legge il numero e lo stampa.
8. Scrivere un programma C che svolge le seguenti richieste:
 - Un processo padre genera due processi figli. Ciascun processo inizializza un proprio array di N interi.
 - Il primo processo invia al processo padre solo i numeri in posizioni pari, e il secondo processo solo i numeri in posizioni dispari.
 - Il padre riceve questi numeri e li scrive in un array di N interi, mettendo in posizioni pari i numeri ricevuti dal primo figlio, e in posizioni dispari i numeri ricevuti dal secondo figlio.
 - Il padre stampa l'array e calcola il max e il min.

Thread

Mutex

1. Scrivere un programma C che segue le seguenti specifiche. Il processo eseguito, inizialmente crea un buffer come array di 11 numeri interi, inizializzati a zero. In seguito genera tre thread utilizzando le librerie POSIX secondo le seguenti specifiche:
 - Il primo thread sceglie casualmente una cella del buffer e vi scrive il numero +1, qualsiasi sia il valore presente nella cella.
 - Il secondo thread sceglie casualmente una cella del buffer e vi scrive il numero -1, qualsiasi sia il valore presente nella cella.
 - Il terzo thread controlla se tutte le celle del buffer sono state inizializzate. In caso positivo, determina se il numero di celle contenenti un valore pari a +1 è maggiore di quelle con -1 e termina tutti e tre i thread. Mentre un thread ha accesso al buffer, nessun altro thread deve accedervi. Una volta che un thread ha acceduto in lettura o scrittura al buffer, deve attendere un numero di secondi random tra 0 e 3
2. Si scriva un programma con tre thread che risolvono il seguente problema: Un buffer di n elementi inizializzato con a -1 viene riempito nel seguente modo:
 - Il primo thread aggiunge nelle posizioni pari del buffer un numero casuale da 0 a 100.
 - Il secondo thread aggiunge nelle posizioni dispari del buffer un casuale da 100 a 200.
 - Il terzo thread somma gli elementi e modifica il buffer nel seguente modo:
`buff[0] = buff[0]; buff[1] = buff[1] + buff[0]; buff[2] = buff[1] + buff[2]`. Si proponga una soluzione di mutua esclusione.
3. Scrivere un programma in C con tre thread che operano su due array di dimensione N inizialmente a 0.
 - Il primo thread scrive in un array A numeri casuali tra 1 e 150, scrivendo un numero per volta in posizioni randomiche.
 - Il secondo thread scrive in un array B numeri casuali tra 150 e 300, scrivendo un numero per volta in posizioni randomiche.
 - Il terzo thread controlla se entrambi gli array sono stati inizializzati, in caso affermativo calcola il massimo in A e in B, calcola il minimo in A e in B. Infine determina il $\max\{\max(A), \max(B)\}$ e il $\min\{\min(A), \min(B)\}$.

Semafori

1. Due thread, il produttore inserisce numeri pari da 0 a 100 in posizioni pari, e numeri dispari da 100 a 200 in posizioni dispari all'interno di un buffer di N elementi, inizializzato a -1, il consumatore legge dal buffer un numero pari e un numero dispari, li somma e stampa la loro somma.
2. Scrivere un programma in C che data una stringa di N caratteri crea N/2 threads che stampano ciascuno un carattere della stringa in maiuscolo. (Semafori binari o mutex)
3. Sei thread, un scrittore e 5 lettori. Lo scrittore scrive su un buffer numeri dispari da 0 a 50 nelle posizioni pari e numeri pari da 50 a 100 nelle posizioni dispari. I lettori leggono coppie di numeri (pari, dispari), li somma e li stampa.