Progetto di laboratorio - Sistemi Operativi

Daniele Vavalà

7/06/2024

1. Struttura

1.1. Organizzazione

Il progetto è composto da 5 tipi di processi diversi che comunicano tramite coda di messagi, memorie condivise, semafori e pipe. Il processo master si occupa di creare le strutture di IPC, gestire la stampa periodica e terminare la simulazione. Il processo attivatore comunica ai processi atomi di scindersi mandando un messaggio con mtype uguale a 1 e mtext vuoto all'interno di una coda di messaggi, raggiungibile tramite una key comune e a cui sono collegati tutti i processi atomi. Durante la scissione i processi atomi estraggono un numero casuale che va da uno fino al loro numero atomico meno uno, a quel punto, dopo aver comunicato il numero atomico scelto all'atomo figlio via pipe, lo sottragono al proprio numero atomico calcolando l'energia liberata. Il processo alimentazione immette periodicamente nuovi atomi all'interno della simulazione usando la stessa funzione utilizzata dal master per inizializzare i primi atomi. Ogni processo, ad operazioni avvenute, comunica al master del successo tramite una memoria condivisa, contenente una struct che raggruppa ogni variabile da registrare e incrementare nel corso della simulazione. Allo scattare di ogni secondo il processo master blocca l'accesso alla memoria condivisa, ne stampa i risultati, li somma a una propria struct locale che tiene il conto delle statistiche complessive del programma e prima di liberarne di nuovo l'accesso ne setta tutti i valori a zero.

1.2. Sincronizzazione

La sincronizzazione viene gestita tramite un array di semafori. In particolare il primo viene inizializzato al numero di processi che vengono creati all'inizio della simulazione e il suo scopo è quello di far partire ogni processo solo ed esclusivamente quando tutti hanno concluso le operazioni di inizializzazione di variabili e strutture IPC. Il secondo invece è un semaforo mutex che si occupa di bloccare e sbloccare determinate sezioni critiche del codice per evitare problemi di desincronizzazione. Infine l'ultimo è un semaforo che regola le comunicazioni tra il processo master e il processo inibitore.

1.3. Gestione della concorrenza

La concorrenza derivante dalla memoria condivisa, al cui interno sono contenuiti i valori della simulazione al secondo, è gestita tramite un semaforo di mutua esclusione. La concorrenza derivante invece dalle attivazioni comunicate dal processo attivatore viene gestita in automatico, infatti la system call msgrcv viene eseguita in modo atomico dai processi, non necessitando (al contrario della memoria condivisa) di alcun meccanismo di sincronizzazione.

1.4. Conclusione

La fine della simulazione può avvenire in vari casi e grazie all'arrivo di diversi segnali, tutti gestiti tramite uno stesso handler. La funzione associata distingue tra i diversi segnali tramite uno switch che, in base al parametro signum in input, decide quale messaggio stampare. In caso di Timeout il segnale corrispondente è SIGALARM, mentre in caso di Meltdown sarebbe SIGUSR2. Una volta stampato il messaggio il processo master si occupa di comunicare a tutti i processi appartenti al suo stesso gruppo di terminare, infine cancella tutte le strutture di IPC e termina la simulazione.

2. Processo Inibitore

2.1. Attivazione

Il processo inibitore può essere attivato all'inizio della simulazione o durante l'esecuzione premendo il tasto invio. Quest'ultimo meccanismo viene gestito grazie all'attivazione della flag O_ASYNC sullo standard input, che permette così l'arrivo di un segnale al processo master ogni qualvuolta ci sia un cambiamento riguardante l'I/O in un determinato file descriptor, permettendo di conseguenza una comunicazione asincrona tra l'utente e il processo.

2.2. Gestione

Il meccanismo viene gestito tramite un handler che cattura il signale "SIGIO" generato dalla flag O_ASYNC e lo propaga al processo attivatore. Quest'ultimo, all'interno del proprio handler, cambia il valore di mtype da 1 al pid del processo inibitore (viceversa per disattivarlo), variando così il destinatario dai processi atomo al processo inibitore. Il processo inibitore, bloccato in attesa dalla system call msgrcv, una volta che vede arrivare un messaggio con il suo pid come mtype estrae un numero casuale per decidere se reinoltrare il messaggio per comunicare di fare una scissione o se non fare nulla.