Corso di Sistemi Operativi, Facoltà di Informatica
Università di Torino

Relazione Progetto Sistemi Operativi

Anno Accademico 2023/2024

Corrao Mario
Grillo Giovanni
Olivero Alessandro

Novembre 2024

1 Descrizione del Progetto

Il progetto si compone di vari processi che cooperano per gestire la creazione, la gestione e la terminazione di "atomi". I processi principali sono:

- **Master**
- **Atomo**
- **Attivatore**
- **Alimentatore**
- **Inibitore**

2 Processo Master

Il processo master si occupa di:

1. **Creazione di IPCS**: Inizializza le strutture di comunicazione interprocessuale leggendo file di configurazione come `timeout.conf`, `explode.conf`, `meltdown.conf`, `blackout.conf`.
2. **Inizializzazione dei Semafori**: Imposta diversi semafori (`sem_atom`, `sem_inhibitor`, `sem_power_plant`, `sem_processes`, `sem_fission`) con valore iniziale pari a 1.
3. **Creazione di Memoria Condivisa**: Attiva segmenti di memoria condivisa per memorizzare variabili globali e stati del sistema.
4. **Gestione dei Segnali**: Configura la gestione dei segnali per la comunicazione tra processi (`SIGTERM`, `SIGUSR1`, `SIGUSR2`).
5. **Creazione di Processi Figli**: Genera processi figli (attivatore, alimentatore, inibitore e atomi) usando `fork()`.
6. **Terminazione**: Alla fine della simulazione, invia segnali di terminazione ai processi figli e rimuove gli IPCS.

In caso di interruzioni o errori di sistema, il processo master si occupa di gestire i segnali ricevuti (come `SIGTERM`) e di eseguire una chiusura ordinata, includendo l'aggiornamento finale del log con un riepilogo dell'ultima sessione di esecuzione.

3 Processo Attivatore

Il processo attivatore è responsabile dell'invio di messaggi agli atomi per avviare le operazioni di scissione. È strutturato come un ciclo continuo, che può essere interrotto tramite la ricezione di un segnale.
Ad ogni iterazione del ciclo, il processo dorme per un intervallo di tempo pari a `STEP_ATTIVATORE` utilizzando una funzione di sleep. In seguito, invia un messaggio agli atomi tramite una coda di messaggi inizializzata precedentemente, con un numero massimo di messaggi pari a `N_MSG`. Ogni messaggio rappresenta un comando di fissione per l'atomo destinatario.

4 Processo Alimentatore

Il processo alimentatore crea nuovi atomi periodicamente. È strutturato in modo simile al processo attivatore, eseguendo un ciclo continuo.
Ad ogni iterazione, viene creato un numero di nuovi atomi pari a `N_NUOVI_ATOMI`, che vengono inseriti nella centrale. L'esecuzione del ciclo è regolata dalla funzione `nanosleep()`, che sospende il processo per un intervallo di tempo pari a `STEP_ALIMENTAZIONE` nanosecondi.

5 Processo Atomo

I processi atomo ricevono messaggi dall'attivatore e gestiscono le operazioni di scissione. Ogni atomo, una volta ricevuto un comando, crea un processo figlio per simulare la fissione e aggiorna le strutture condivise.

5.1 Dettagli del Processo Atomo

Nel sistema, ogni processo atomo rappresenta un singolo atomo. La comunicazione con l'attivatore avviene tramite una coda di messaggi. Quando un messaggio viene ricevuto, l'atomo lo preleva dalla coda, individua il proprio puntatore nella memoria condivisa (utilizzando il suo PID) e crea un nuovo processo figlio per simulare la scissione. Dopo ogni scissione, il processo atomo aggiorna le strutture condivise e continua fino a ricevere un segnale di terminazione, momento in cui rilascia le risorse IPC e termina correttamente.
La funzione `do_fission()` è progettata per simulare la scissione di un atomo.
Prende come input l'atomo stesso e il PID del figlio generato tramite la chiamata `fork(atom_parent, child_pid)`. Durante la simulazione, viene creato un nuovo atomo figlio per rappresentare la scissione reale. Con probabilità del 50%, l'energia prodotta dalla scissione viene rilasciata e immagazzinata nella centrale. La funzione esegue controlli per garantire il corretto funzionamento del sistema, soprattutto quando l'inibitore è attivo, e invia segnali al processo master per gestire situazioni critiche, come il superamento del limite di energia immagazzinabile (`ENERGY_EXPLODE_THRESHOLD`).

6 Processo Inibitore

Il processo inibitore limita alcune operazioni di scissione e assorbe una parte di energia prodotta dalla scissione stessa:
Si fonda su un ciclo infinito che tiene il processo in ascolto su una coda di messaggi condivisa con i vari processi atomi. Un atomo quando sta per effettuare una fissione invia un messaggio all'inibitore che con probabilità 0,8 accetta la fissione (nello 0,2 avviene la negazione o la trasformazione dell'atomo in scoria). Se la fissione è stata accettata, l'atomo invia un messaggio indicando l' energia liberata a seguito della scissione, l'inibitore dopo averla ricevuta ne assorbe una parte e reinvia il resto all'atomo. Questo avviene se e solo se l'inibitore risulta "acceso". Il processo inibitore rimane attivo finché non riceve un segnale di terminazione.
È possibile accendere/spegnere l'inibitore a run-time tramite l'implementazione del `sig_handler`, che gestisce questa operazione quando riceve in input da tastiera (`CTRL + \`), ovvero `SIGQUIT`.

7 Stampe di Riepilogo

Il processo master si occupa anche di registrare le attività svolte dai vari processi e di produrre un riepilogo dei dati raccolti durante la simulazione. Per questo scopo, viene utilizzata la funzione `daily_log()`.

7.1 Funzione `daily_log()`

La funzione `daily_log()` è responsabile della creazione di un file di log in cui vengono riportati i principali eventi e statistiche del sistema. Tra le informazioni registrate troviamo:

- Il numero totale di processi atomo creati durante l'esecuzione.
- Il numero di operazioni di scissione effettuate e la quantità di energia prodotta.
- Eventuali segnalazioni di superamento della soglia di energia (`ENERGY_EXPLODE_THRESHOLD`).
- Stato attuale dell'inibitore (acceso o spento) e la sua influenza sulle operazioni di scissione.

La funzione `daily_log()` stampa queste informazioni sia sul terminale sia su un file di log dedicato. Questo file viene aggiornato periodicamente e fornisce una panoramica dello stato del sistema, facilitando il monitoraggio e la diagnosi di eventuali problemi.

8 Strutture IPC Utilizzate

Nel progetto sono state impiegate diverse strutture di comunicazione interprocessuale (IPC) per facilitare la sincronizzazione e lo scambio di dati tra i processi. Le principali strutture IPC utilizzate sono:

8.1 Memoria Condivisa (Shared Memory)

La memoria condivisa è utilizzata per memorizzare variabili globali e lo stato del sistema, tra cui:

- **PowerPlant**: Stato attuale della centrale, inclusi i livelli di energia immagazzinata.
- **Vars**: Variabili di sola lettura utilizzate per configurazioni e parametri globali.
- **Inhibitor**: Stato e configurazione del processo inibitore.
- **Atoms**: Dati relativi ai processi atomo, inclusi i puntatori ai processi attivi.

L'accesso alla memoria condivisa è sincronizzato tramite semafori per evitare condizioni di gara (race conditions) e garantire la consistenza dei dati.

8.2 Semafori (Semaphores)

I semafori sono utilizzati per controllare l'accesso alla memoria condivisa e per sincronizzare i processi. I semafori principali implementati nel sistema sono:

- `sem_atom`: Sincronizza l'accesso ai dati relativi agli atomi.
- `sem_inhibitor`: Controlla l'accesso alle variabili dell'inibitore.
- `sem_power_plant`: Regola l'accesso ai dati della centrale energetica.
- `sem_processes`: Coordina l'esecuzione dei processi principali.
- `sem_fission`: Sincronizza le operazioni di scissione per evitare conflitti.

L'uso dei semafori garantisce che solo un processo alla volta possa accedere alle sezioni critiche della memoria condivisa, prevenendo comportamenti non deterministici.

8.3 Code di Messaggi (Message Queues)

Le code di messaggi vengono utilizzate per la comunicazione asincrona tra processi, facilitando lo scambio di comandi e informazioni:

- `msg_stack`: Gestisce i comandi di scissione inviati dall'attivatore agli atomi.
- `inhibitor_stack`: Coordina la comunicazione tra l'inibitore e gli atomi per l'autorizzazione alla scissione e il monitoraggio dell'energia rilasciata.

Questa struttura permette una comunicazione non bloccante, migliorando l'efficienza e la reattività del sistema.