Progetto di sistemi operativi: Reazione a catena

Francesco Mauro, Riccardo Oro

A.A 2023/2024

Contents

1 Struttura del progetto

```
/
├── bin/
│   ├── atom
│   ├── activator
│   ├── inhibitor
│   ├── master
│   └── psu
├── src/
│   ├── atom.c
│   ├── activator.c
│   ├── inhibitor.c
│   ├── master.c
│   ├── psu.c
│   ├── header/
│   │   ├── atom.h
│   │   ├── activator.h
│   │   ├── inhibitor.h
│   │   ├── master.h
│   │   ├── psu.h
│   │   └── config/
│   └── config.txt
└── makefile
```

2 Processo master

- Si occupa di leggere da file attraverso la funzione scan_data() i valori di configurazione necessari per la simulazione che verranno associati alla struct config.
- Crea *N_ATOMI_INIT* atomi usando la funzione fork. Il processo figlio generato avrà associato un array contenente i parametri di configurazione e il PID del processo master, eseguirà *execvp* per azionare *./bin/atom*.
- Crea il processo *psu* usando la funzione fork. Il processo figlio generato avrà associato un array contenente i parametri di configurazione impostati nel file di configurazione, il PID del processo master e gli ID di alcuni IPC, eseguirà un *execvp* a *./bin/psu*.
- Crea il processo *inhibitor* sotto richiesta dell'utente attraverso la funzione fork. Il processo figlio generato avrà associato un array contenente i parametri di configurazione personalizzati e il PID del processo master, eseguirà un *execvp* a *./bin/inhibitor*.
- Si occupa di registrare i segnali che verranno gestiti dagli appositi handler, che permettono la gestione delle terminazioni della simulazione.

- Dopo aver inizializzato tutto il necessario, imposta un alarm con valore *SIM_ DURATION* che rappresenta la durata complessiva della simulazione. Il segnale *SIGALARM* è gestito attraverso un handler.
- Si occupa, in seguito allo scadere di un timer visibile a schermo, di far partire i processi atomi per la simulazione.
- Si occupa di gestire le terminazioni della simulazione.

2.0.1 Output a schermo

Ogni secondo mostra a schermo:

- Il numero di fissioni avvenute nell'ultimo secondo.
- Il numero di scorie raccolte nell'ultimo secondo.
- L'energia prodotta dei processi atomo nell'ultimo secondo.
- I bilanciamenti effettuati di processo inhibitor per limitare le fissioni dei processi atomo nell'ultimo secondo.

2.0.2 Semaforo

Il processo master usa un semaforo per sincronizzare l'inizializzazione degli atomi e il segnale di SIGCONT

3 Processo atom

Il processo atomo viene generato attraverso fork dal processo master. Dopo la sua nascita fa le seguenti operazioni:

- Inizializza gli *IPC* necessari.
- Registra il segnale *SIGCHLD* associandolo all'handler apposito.
- Registra il segnale *SIGUSR1* associandolo ad un handler che si occuperà, in caso di *MELTDOWN*, di inviare il segnale *SIGUSR1* al processo master che gestirà la terminazione della simulazione.
- Preleva e associa all'apposita struct i valori della configurazione attraverso *argv*.

Dopo queste operazioni il processo atomo si autoinvia un segnale *SIGSTOP* per indicare che ha inizializzato tutto il necessario, attendendo dal processo master il segnale *SIGCONT* che arriverà quando scadrà il timer a schermo e verrà impostato l'alarm con *SIM_ DURATION*. Successivamente, il processo atomo si mette all'opera e, dopo aver ricevuto e prelevato il comando di fissione dalla message queue che lo mette in comunicazione con il processo attivatore, la fissione dell'atomo è gestita dalla funzione atom_fission(). L'atomo saprà

se effettuare o meno la fissione dopo avere prelevato il valore all'interno della message queue. In caso di comando positivo:

- Nel caso in cui il suo numero atomico sarà inferiore a quello prestabilito all'interno della configurazione attraverso il parametro *MIN_A_ATOMICO*, esso non potrà effettuare fissione e aumenterà il numero delle scorie, effettuando una exit con parametro *EXIT_SUCCESS*.
- Altrimenti, se il comando prelevato in message queue è uguale a 1, l'atomo può effettuare fissione, aumentando il numero di fissioni e generando un processo figlio attraverso la funzione fork. Se essa andrà a buon fine, il processo figlio si occuperà di aumentare il numero di attivazioni, energia rilasciata e in caso di scorie nucleari.

3.0.1 Come calcolare il numero atomico?

Il numero atomico viene calcolato assegnado un valore randomico compreso tra *N_ATOM_MAX* e un valore randomico. Quest'ultimo viene randomizzato usando un seed che si basa sull'ora corrente, questo permette di rendere il valore non deterministico

3.0.2 Calcolo dell'energia rilasciata durante la fissione

Il processo *padre* divide il proprio numero atomico, i due valori vengono condivisi tramite pipe con i processi *figli* Quindi l'energia rilasciata viene calcolata seguendo la seguente formula $n1n2 - max(n1, n2)$ dove n1 e n2 sono i rispettivi numeri atomici. L'energia rilasciata in caso ci sia l'inibitore attivo nella configuarazione verrà condiviso in shared memory in modo tale che poi possa diminuire per l'azione dell'inibitore stesso

4 Processo psu

Il processo psu viene generato mediante funzione fork azionata dal processo master. Dopo che viene generato, il processo effettua le seguenti operazioni:

- Inizializza o preleva il suo id se già esistente ,una messagge queue necessaria per comunicare i parametri per l'aggiornamento delle statistiche.
- Un array di tipo pid_t la cui dimensione viene stabilità attraverso la funzione malloc ,che allochera una memoria paria a N_NUOVI_ATOMI * dimensione di un int ,all'interno di questo array andranno tutti i pid dei nuovi processi atomi generati dal processo psu tramite fork.
- Il valore per la quale il processo psu effettuera una nanosleep tra un ciclo di creazione di processi atomi e un altro ,esso sarà uguale al valore del parametro di configurazione STEP in formato nanosecondi.

Dopo aver effettuato l'inizializzazione di tutto il necessario il processo *psu* producen per ogni ciclo nuovi processi atomi per un totale corrispondente al parametro di configurazione N_NUOVI_ATOMI. Effettua una nanosleep con il valore prestabilito dal parametro di configurazione *STEP* in formato nanosecondi. La generazione dei processi atomo avviene tramite la funzione born_new_atom() che attraverso la funzione fork crea un processo figlio a cui verrà assegnato un array contentente tutti i parametri di configurazione necessari alla simulazione ,esso in fine effettuerà il cambio d'immagione attraverso la funzione execp per cosi trasformarsi in atomo. Ogni processo atomo creato viene subito spedito da parte del processo psu un segnale SIGCONT per farsi che il nuovo processo atomo dopo la sua Inizializzazione può procedere allo svolgimento dei propri compiti. In caso di fallimento durante la creazione del processo figlio esso invierà al processo master un segnale SIGUSR1 che avviserà del fallimento per cosi poi terminare la simulazione.

5 Processo activator

Processo attivatore viene generanto dal processo master attraverso la chimaata alla funzione fork che ,genererà un processo figlio a cui verrà assegnato un array contentente i parametri letti da file necessari alla simulazione ,successivamente chiamerà la funzione execvp per effettuare un cambio d'immagine per diventare effettivamente un processo attivatore . Dopo la sua nascia il proecsso attivatore effettuerà le seguenti operazioni:

- Inizializza o preleva la message queue necessaria per comunicare le fissioni al processo atomo.
- Genera in modo randomico un numero pari a 0 o 1 che rispettivamente hanno il seguente significato: 0 : Il processo atomo non effettua fissione. 1 : Il processo atomo ha il comando per effettuare una fissione. I valori vengono generato in modo randomico e inseriti in una apposita message queue su cui gli atomi senza conoscerne precedentemente il significato preleveranno il primo messaggio disponibile che ne determinerà l'operazione di fissione di quell'atomo.

6 Processo inhibitor

Il processo inibitore viene attivato impostando sul file di configurazione il valore 1 a fianco di *INHIBITOR*, in questo modo il processo master genererà attraverso la funzione fork un processo figlio con un array contente tutti i parametri di configurazione, in seguito la funzione *execvp* che permetterà l'effetiva creazione del processo in questione che effettuerà le seguenti operazioni:

- Registrare il segnale SIGINT che permetterà di gestire attraverso la combinazione ctrl+c da tastiera ,lo start and stop del processo a run-time.

- Preleva l'id della message queue su cui effettuera le limitazioni di attivazioni di processi atomo ,il numero di limitazioni è pari alla somma dei parametri di configurazione N_NUOVI_ATOMI + N_ATOMI_INIT.
- Preleva l'id della message queue su cui invierà i dati necessari per il calcolo delle statistiche della simulazione.
- Preleva una quantità di energià pari a ENERGY_DEMAND.

6.0.1 Meccanismo di assorbimento dell'energia e limitazione delle fissioni

Il prelievo di energià avviene attraverso l'utilizzo di una shared memeory su cui viene scritto il quantitativo di energia che il processo inibitore assorbe ogni secondo.
La limitazione delle fissioni di processi atomi avviene attraverso una meccanismo di scrittura sulla message queue che è in comunicazione tra processo attivatore e processo atomo, in modo tale da aumentare il numero di messaggi contenti il valore 0 ,che appunto non permette ad un processo atomo di effettuare l'operazione di fissione.

6.0.2 Possibilità di fermare e far riprendere il processo a discrezione dell'utente

Il meccanismo progettato per permettere al processo inibitore di bloccare e riprendere la proprio esecuzione da input da tastiera a run-time consiste nel mascheramente del segnale SIGINT attraverso un nuovo gestore dei segnale che se riceeve proprio quest'ultimo controlla attraverso un flag se il processo è in esecuzione o meno, in caso il processo inibitore fosse in esecuzione attraverso la combinazione ctrl+c da tastiera verrà inviato al processo inibitore un segnale SIGSTOP. Nel caso invece che il processo inibitore è in fase di stop e viene ricevuta la combinazione ctrl+c da input da tastiera viene inviato attraverso funzione kill un segnale SIGCONT modificando il flag che identifica lo stato del processo.