Relazione dello sviluppo del progetto di Sistemi Operativi

Sviluppato da:

- Polizzi Giovanni Gabriel
- Riccardi Andrea
- Spinolo Emanuele

1. Introduzione

La consegna del progetto ha richiesto la simulazione di una reazione atomica a catena. Il processo principale, il Master, gestisce lo svolgimento della simulazione. Un processo detto Alimentatore ogni n secondi genera un numero di processi Atomo, mentre il processo Attivatore comunica agli atomi di realizzare la scissione.

Seguendo, un processo Inibitore, che può essere attivato o meno all'inizio della simulazione, si occuperà di inibire la reazione (attraverso due meccanismi spiegati più avanti). Infine, il processo Atomo attende il segnale di scissione, per poi dividersi in due atomi alla ricezione di esso.

Per poter personalizzare l'esecuzione del programma si possono modificare alcuni parametri della simulazione senza dover ricompilare. I parametri dovranno essere inseriti in un file il quale deve essere passato come parametro al master, questo verificherà l'esistenza del file di configurazione, se non presente procederà a creare un file "opt.conf" contenente conterrà tutti i parametri di configurazione con assegnati i rispettivi valori di default.

Per semplificare lo sviluppo della comunicazione tra i processi sono stati implementati dei metodi di utilità (nel file "ipc_manager.h") allo scopo di permettere una comprensione del codice più semplificata.

Sono state implementate delle stampe di Debug, le quali possono servire a visualizzare l'evoluzione della simulazione. Per attivare la modalità **DEBUG** si dovrà ricompilare il progetto digitando il comando "***make debug***"; questa implementazione realizza anche le stampe degli atomi. Data l'elevata quantità di atomi, il terminale si satura velocemente di stampe; per evitare che ciò accada si può compilare la versione di debug senza le stampe degli atomi digitando il comando "***make d_no_atom***".

2. Lettura impostazioni

Per il corretto funzionamento della simulazione è necessario leggere le informazioni iniziali da un file di configurazione. I dati di configurazione permettono di personalizzare la simulazione, questi valori vengono letti all'avvio della simulazione da un programma che si incarica di leggere un file e riconoscere certi pattern nei quali devono essere formattate le impostazioni, di seguito si entrerà più nel dettaglio sulle impostazioni e sul funzionamento di questo lettore delle informazioni.

2.1 Le impostazioni

Di seguito verranno descritte le impostazioni ed il loro scopo all'interno della simulazione:

- **ENERGY_DEMAND**: Energia che sarà prelevata ogni secondo dal master.
- **N_ATOMI_INIT**: Numero di atomi che il master genererà per l'avvio della simulazione.
- **N_ATOMI_MAX**: Massimo numero atomico che può avere un atomo.
- **MIN_N_ATOMICO**: Numero atomico minimo necessario a un atomo per scindersi, se inferiore a questa soglia l'atomo diventerà una scoria.
- **SIM_DURATION**: Durata della simulazione in secondi.
- **STEP**: Tempo di attesa in nanosecondi prima che l'alimentatore possa produrre nuovi atomi.
- **N_NUOVI_ATOMI**: Numero di nuovi atomi che l'alimentatore dovrà generare.
- **ENERGY_EXPLODE_THRESHOLD**: Massimo valore di energia liberata che si puo avere, al di sopra la simulazione terminerà per "*explode*".

2.2 Il Lettore

Nei file "***read_setting.c***" e "***read_setting.h***" si trovano le definizioni e l'implementazione del lettore delle impostazioni.

Il lettore è stato progettato per leggere le impostazioni da un file chiamato "***opt.conf***" o da un file specificato all'avvio del master, inserendo il nome di questo tramite il terminale (deve essere nella stessa cartella del programma chiamante).

- Esempio: " **./master test.conf** "

Se questo file non viene trovato allora si procede a generarne uno che avrà dei valori di default scelti per portare la simulazione a terminare per "timeout".

Se nel file viene identificato un commento, il resto della riga verrà ignorato (l'inizio di un commento viene indicato col simbolo "#").

Le impostazioni possono essere inserite in un qualsiasi ordine ma devono essere strutturate nel seguente modo:

<param_name> = <param_value>;

Le variabili <param_name> e <param_value> possono assumere i seguenti valori:

<param_name>:	ENERGY_DEMAND, N_ATOMI_INIT, N_ATOMI_MAX, MIN_N_ATOMICO, SIM_DURATION, STEP, N_NUOVI_ATOMI, ENERGY_EXPLODE_THRESHOLD.
<param_value>:	Assume un valore numerico qualsiasi, tranne per step dove il valore può essere al massimo di 9 cifre.

Per implementare l'introduzione dei parametri in un ordine qualsiasi è stata creata una piccola versione di un lexer e di un un parser.

3. Utilizzo degli oggetti IPC

Di seguito verrà spiegata l'idea dietro l'utilizzo degli oggetti IPC nella simulazione.

3.1 Memoria condivisa

La memoria condivisa viene utilizzata per registrare le statistiche che il master stampa ogni secondo, per controllare l'accesso in scrittura e lettura di questa porzione di memoria si usa il semaforo SEM_STATS il cui decide che processo avrà accesso alla memoria condivisa. La memoria condivisa tiene traccia dei seguenti dati:

• Numero di attivazioni totali e relative all'ultimo secondo.

• Numero di scissioni totali e relative all'ultimo secondo.

• Quantità di energia prodotta totale e relativa all'ultimo secondo.

• Quantità di energia consumata totale e relativa all'ultimo secondo.

• Quantità di scorie prodotte totali e relative all'ultimo secondo.

3.2 Semafori

Nella simulazione si utilizza un set di 3 semafori che gestiscono funzioni differenti.

Di seguito verrà spiegato il motivo dell'utilizzo di questi semafori e quali problemi risolvono:

- **SEM_READY**: il semaforo viene inizializzato con il numero di processi iniziali, che sono **master**, **attivatore**, **alimentatore**, *n_atom_init* processi **atomo** e, se presente, l'**inibitore**. Questo semaforo gestisce la sincronizzazione iniziale necessaria per l'avvio della simulazione.

- **SEM_ACTIVATOR**: il semaforo viene inizializzato a 0. Il suo compito è di gestire l'attivazione degli atomi da parte dell'attivatore. Quando questo semaforo è posto a 0, l'alimentatore dovrà rilasciare *N* risorse, le quali rappresentano il numero di atomi da scindere. Gli atomi tenteranno di prelevare una risorsa da questo semaforo. Fin quando gli atomi non otterranno una risorsa non potranno scindersi, seguendo dunque uno schema di concorrenza.

- **SEM_STATS**: il semaforo viene inizializzato a 1. Questo garantisce l'accesso alla memoria condivisa di un processo alla volta per mantenere la consistenza della risorsa.

3.3 Coda di messaggi

L'utilizzo della coda di messaggi nella simulazione dipende dalla presenza del processo

Inibitore, questo oggetto gestisce la comunicazione tra gli atomi e il processo **inibitore**. Gli atomi

invieranno un messaggio di tipo "**1"**, questo sarà gestito dall' inibitore il quale invierà un messaggio

con una risposta di tipo "**pid_atomo_richiedente**".

I messaggi contengono le seguenti informazioni:

- Energia liberata/Da assorbire.
- Flag_scoria.
- Pid_processo_mittente.

4. Interazioni tra i processi

La simulazione si divide in tre fasi: avvio, simulazione e arresto. Di seguito verranno spiegate le interazioni tra i vari processi e gli oggetti IPC.

4.1 Avvio

Per iniziare la simulazione si dovrà avviare il master, verrà chiesto all'utente se si vuole avviare l'inibitore o no; dopodiché, verranno lette le impostazioni usando le modalità descritte nel punto (*2*). Il master procederà a creare il set di 3 semafori [*SEM_READY*, *SEM_ACTIVATOR*, *SEM_STATS*], per poi inizializzarli secondo lo schema: semafori (*SEM_READY* = **numero_processi_tot**, *SEM_ACTIVATOR* = ***0***, *SEM_STATS* = **1**). In seguito, il master creerà la memoria condivisa, la quale dovrà tenere traccia delle statistiche, e, nel caso in cui l'inibitore venga generato all'inizio, si creerà anche la coda di messaggi.

Dopo la creazione degli oggetti IPC, il master si occuperà della creazione dei processi alimentatore, attivatore e, nel caso, inibitore, generando poi *n_ATOM_INIT* processi atomo.

Una volta finito di creare tutti i processi, il master riserverà una risorsa dal semaforo SEM_READY per comunicare che lui pronto, e si metterà in attesa che il semaforo abbia *zero* risorse.

I processi alimentatore, attivatore, inibitore e i processi atomo salveranno i parametri in

delle variabili, imposteranno il proprio signal_handler; Continuando, riserveranno una risorsa ciascuno al semaforo SEM_READY, per comunicare che sono pronti, e si metteranno in attesa che questo semaforo abbia *zero* risorse.

Una volta che il semaforo SEM_READY avrà *zero* risorse la simulazione partirà:

****Nota:**

Gli atomi ricevono un parametro che gli permette di capire se sono o meno i primi atomi (*creati dal master all'inizio*) per capire se devono prelevare o meno una risorsa da SEM_READY.

4.2 Simulazione

Sotto verranno descritti i compiti dei singoli processi e la loro interazione:

4.2.1 Master

Imposta un timer con il tempo totale della simulazione, dopodiché ogni secondo il master svolge le seguenti azioni:

1. Resta in attesa per un secondo, dopodiché richiede una risorsa a SEM_STATS.
2. Calcola l'energia disponibile (*energia_prodotta_totale* - *energia_consumata_totale*).
3. Verifica che l'energia disponibile sia maggiore dell'energia da prelevare (se così non fosse il programma terminerebbe con *blackout*).
4. Aggiorna le statistiche, sia totali che al secondo, dell'energia consumata.
5. Stampa le statistiche.
6. Resetta le statistiche al secondo.
7. Rilascia una risorsa da SEM_STATS.

4.2.2 Alimentatore

Ogni STEP nano-secondi crea *N_NUOVI_ATOMI* processi atomo. Questo processo può generare l'errore di **meltdown**.

4.2.3 Attivatore

In un ciclo verifica che il semaforo SEM_ACTIVATOR sia a *zero*, in seguito:

1. Calcola un numero randomico (*num_casuale*) che rappresenta il numero di processi d'attivare.
2. Richiede una risorsa al semaforo SEM_STATS.
3. Aggiorna il numero di attivazioni totali e al secondo.
4. Rilascia una risorsa al semaforo SEM_STATS.
5. Rilascia *num_casuale* risorse in SEM_ACTIVATOR e ritorna al inizio del ciclo.

4.2.4 Atomo

Richiede una risorsa a SEM_ACTIVATOR, verifica se l'atomo è una scoria: nel caso sia scoria allora viene realizzata la procedura per le scorie descritta nelle note, altrimenti vengono preparati tutti i parametri dell'atomo che verrà creato con la scissione.

In seguito, si calcola l'energia liberata e il numero atomico di entrambi i processi, per poi procedere con la scissione.

Se l'inibitore è presente nella simulazione viene inviato un messaggio alla coda di messaggi, nel cui si richiedono due messaggi: se l'atomo deve diventare scoria o meno, e l'energia assorbita da esso: l'atomo attende la risposta al messaggio, dopo la quale aggiorna l'energia liberata attraverso la seguente formula:

nuova_energia_liberata = vecchia_energia_liberata - energia_assorbita

Se l'atomo deve diventare scoria allora si realizza la procedura per le scorie, aggiornando anche l'energia liberata totale e al secondo insieme al numero di scissioni.

Se non è attivo l'inibitore, viene riservata una risorsa nel semaforo SEM_STATS, si aggiorna l'energia liberata totale e al secondo, e il numero di scissioni si rilascia una risorsa in SEM_STATS, e si ritorna al inizio della procedura. ****Note atomo:** -Di seguito verrà descritto la procedura per il caso in cui un atomo sia una scoria, si richiede una risorsa al semaforo SEM_STATS, aggiorna il contatore delle scoria totali e al secondo, rilascia la risorsa e termina la sua esecuzione -Durante la scissione la simulazione può terminare per meltdown.

4.2.5 Inibitore

La presenza del processo inibitore si può decidere all'avvio della simulazione, durante la quale può essere attivato/disattivato premendo i tasti ***Ctrl+C***.

L'inibitore durante la sua esecuzione svolgerà in loop le seguenti operazioni:

1. Riceverà dei messaggi di tipo **1** dalla coda di messaggi; questo tipo di messaggi saranno inviati dagli atomi e l'inibitore deciderà se l'atomo che ha mandato il messaggio dovrà diventare scoria o meno, inoltre deciderà la quantità di energia che verrà inibita nell'operazione.

2. Terminati i calcoli l'inibitore invierà un messaggio di tipo **pid_atomo**.

3. Scriverà nel file "*logs.txt*" le operazioni svolte.

4. Riserverà una risorsa nel semaforo SEM_STATS.

5. Aggiornerà il numero di energia assorbita.

6. Rilascerà una risorsa nel semaforo SEM_STATS e ritornerà all'inizio del ciclo. Se l'inibitore non è attivo, il messaggio da mandare avrà le variabili al suo interno con valore **0,** dopodiché si tornerà all'inizio del ciclo.

4.3 Arresto

Durante la simulazione possono avvenire situazioni particolari che ne determinano la terminazione, i quali sono "**timeout**" (si verifica nel master), "**blackout**" (si verifica nel master), "**explode**" (si verifica negli atomi dopo una scissione) e "**meltdown**". Durante la simulazione si devono terminare tutti i processi della simulazione ed eliminare gli oggetti IPC creati, per farlo abbiamo pensato al seguente meccanismo realizzato dal master:

Per eliminare i processi *atomo* e il processo *attivatore* si eliminano i semafori, se uno di questi processi proverà utilizzare un semaforo, la funzione richiamata restituirà un errore attraverso il quale si terminerà l'esecuzione del processo in questione. Per eliminare, se presente, l'inibitore e l'alimentatore si invia un segnale a questi processi ed essi termineranno. Una volta finito, il master stamperà le statistiche alla fine della simulazione, eliminerà la memoria condivisa, se presente, la coda di messaggi e stamperà la causa di terminazione, dopodiché anche il master terminerà la sua esecuzione.