Progetto SO 2023/24
Reazione a catena

Versione definitiva

Bini, Radicioni, Schifanella C.

Indice

1 Composizione gruppo di studenti

Il progetto sarà sviluppato da un gruppo, composto da 1 a **massimo 3 componenti**. Il gruppo dovrebbe essere composto da studenti dello **stesso turno**, i quali discuteranno con il docente del proprio turno. È tuttavia consentita anche la discussione del progetto di laboratorio da parte di un gruppo di studenti di turni diversi. In questo caso, **tutti** gli studenti del gruppo discuteranno con **lo stesso docente**. Esempio: Tizio (turno T1) e Caio (turno T2) decidono di fare il progetto insieme. Lo consegnano e vengono convocati dal prof. Sempronio il giorno X. Tale giorno Tizio e Caio si presentano e ricevono entrambi una valutazione dal Prof. Sempronio (docente del T1, anche se Caio fa parte del turno T2, il cui docente di riferimento è il prof. Calpurnio).

2 Consegna

Il progetto è costituito da:

1. il codice sorgente

2. una breve relazione che sintetizzi le scelte progettuali compiute

Il progetto si consegna compilando la seguente Google Form, cui si accede con credenziali istituzionali,

- `https://forms.gle/SH9Ny25JsweSjy359`

la quale richiederà il caricamento di:

- progetto stesso (un unico file in formato .tgz o .zip NON .rar). Il nome del file deve essere composto dall'unione dei cognomi dei componenti del gruppo (es.: TizioCaio.zip)
- cognome, nome, matricola, email di ciascun componente del gruppo.

Dopo il caricamento del progetto, verrete convocati dal docente con cui discuterete (si veda Sezione 1 in caso di gruppo composto da studenti di turni diversi). Attenzione: **compilare una sola form per progetto** (e **non una per ogni componente del gruppo**). Una eventuale ulteriore consegna prima dell'appuntamento **annullerà la data dell'appuntamento**.

La consegna deve avvenire almeno **10 giorni prima** degli appelli scritti per dare modo al docente di pianificare i colloqui:

- se consegnate con almeno 10 giorni di anticipo rispetto alla data di un appello, il docente propone una data per la discussione entro l'appello seguente
- altrimenti, la data sarà successiva all'appello seguente.

Esempio: per avere la certezza di un appuntamento per la discussione di progetto entro l'appello del 19/01/2023, si deve consegnare entro le ore 24:00 di **Martedì 09/01/2023**.

3 Valutazione e validità

Il progetto descritto nel presente documento potrà essere discusso se **consegnato entro il 30 Novembre 2024**. Dal Dicembre 2024 sarà discusso il progetto assegnato durante l'anno accademico 2024/25.

Tutti i membri del gruppo devono partecipare alla discussione. La valutazione del progetto è **individuale** ed espressa in 30-esimi. Durante la discussione

- verrà chiesto di illustrare il progetto
- verrà chiesto di commentare il codice e eventualmente di apportare piccole modifiche al progetto
- verranno proposti quesiti sul programma "Unix" del corso anche non necessariamente legati allo sviluppo del progetto.

È necessario ottenere una votazione di almeno **18** su 30 per poter essere ammessi allo scritto. In caso di superamento della discussione del progetto, la votazione conseguita consentirà di partecipare allo scritto per i **cinque appelli successivi** alla data di superamento. Non sono ammesse eccezioni o deroghe a tale periodo di validità.

In caso di mancato superamento, lo studente si potrà ripresentare soltanto dopo almeno **un mese** dalla data del mancato superamento

Si ricorda che il voto del progetto ha un peso di $\frac{1}{4}$ sulla votazione finale di Sistemi Operativi.

4 Pubblicazione del proprio progetto, plagio, sanzioni

Copiare altri progetti o parte di essi impedisce una corretta valutazione. Per questo motivo, gli studenti che consegnano il progetto sono consapevoli che:

- la condivisione con altri gruppi attraverso canali pubblici o privati (a titolo di esempio: google drive, canali telegram, github, etc.) di tutto o parte del progetto non è consentita fino a tutto Novembre 2024;

- la copiatura di tutto o parte del progetto non è consentita;
- eventuali frammenti di codice estratti da parti di codice visto a lezione o da altri repository pubblici devono essere opportunamente dichiarati.

Nel momento in cui uno studente non rispettasse le sopra citate norme di comportamento, dopo essere stato convocato ed aver avuto modo di illustrare la propria posizione, potrà essere oggetto delle seguenti sanzioni:

- se lo studente avrà nel frattempo superato l'esame di Sistemi Operativi anche successivamente alla data di discussione del progetto, la verbalizzazione del voto verrà annullata;
- se lo studente avrà soltanto superato la discussione del progetto ma non l'esame, la valutazione del progetto verrà annullata e lo studente non potrà accedere ad ulteriore discussione di progetto prima dei due appelli successivi alla data di evidenza di copiatura.

5 Descrizione del progetto: versione minima (voto max 24 su 30)

Si intende simulare una reazione a catena. A tal fine sono presenti i seguenti processi:

- un processo *master* che gestisce la simulazione e mantiene delle *statistiche*
- processi *atomo* che si scindono in altri processi atomo, generando energia
- un processo *attivatore*
- un processo *alimentazione* che aggiunge nuovi atomi

5.1 Processo master

Il processo master

- gestisce la simulazione, inizializza le strutture, crea processi figlio
- ogni secondo:
  - stampa lo stato corrente della simulazione e le statistiche
  - preleva una quantità `ENERGY_DEMAND` di energia.

All'inizio della simulazione, il processo *master*

- crea `N_ATOMI_INIT` processi *atomo*
- crea il processo *attivatore*
- crea il processo *alimentazione*
- avvia la simulazione. La simulazione dovrà partire solamente quando tutti i processi sopra citati sono stati creati e hanno eseguito le rispettive istruzioni di inizializzazione

5.2 Processo atomo

Ogni processo *atomo* è dotato di un numero atomico casuale compreso tra `1` e `N_ATOM_MAX`. Il numero atomico è un'informazione privata del processo che viene assegnata dal processo padre dopo la sua creazione (la funzione di scelta del numero atomico è demandata al programmatore. Può essere estratto un semplice numero casuale o utilizzata qualche altra distribuzione di probabilità).

La scissione dell'atomo è simulata con una fork.

- Un processo atomo padre effettua una fork di un nuovo atomo.

- La somma dei numeri atomici di padre e figlio dopo la scissione è uguale a quello del padre prima della scissione.
- Quando avviene la scissione, viene incrementata l'*energia* liberata nelle statistiche mantenute dal master. La quantità di energia liberata dipende dai numeri atomici dei due atomi dopo la scissione, secondo la seguente funzione

$$\mathsf{energy}(n_1, n_2) = n_1 n_2 - \max(n_1, n_2),$$

  con n_1 e n_2 uguali ai numeri atomici dei due atomi dopo la fissione. Si osservi che l'energia liberata è massima quando avviene una scissione in atomi con ugual numero atomico, mentre vale 0 quando n_1 o n_2 vale 1.
- La scissione è comunicata dal processo *attivatore*. Tale meccanismo di comunicazione è scelto a discrezione dello sviluppatore (si possono pensare a soluzioni alternative con pipe, code di messaggi, semafori, o segnali).
- Quando un atomo con numero atomico minore o uguale a `MIN_N_ATOMICO` riceve il comando di scissione, esso termina e viene conteggiato nelle statistiche fra le *scorie*.

5.3 Processo attivatore

Ogni `STEP_ATTIVATORE`, sulla base di proprie politiche (a discrezione degli sviluppatori), il processo attivatore comunica a uno o più atomi la necessità di effettuare una scissione.

Nota bene: l'attivatore NON crea nuovi atomi, ma ordina esclusivamente ad atomi presenti di scindersi. Saranno gli atomi che a loro volta ne creeranno di nuovi.

5.4 Processo alimentazione

Ogni `STEP_ALIMENTAZIONE` nanosecondi, il processo *alimentazione* immette nuovo combustibile, ovvero crea `N_NUOVI_ATOMI` atomi.

5.5 Terminazione

La simulazione termina in una delle seguenti circorstanze:

timeout raggiungimento della durata impostata `SIM_DURATION`

explode energia liberata (al netto di quella consumata da master) maggiore del valore `ENERGY_EXPLODE_THRESHOLD`

blackout prelievo di una quantità di energia maggiore di quella disponibile. La quantità di energia disponibile è definita come la quantità di energia prodotta dalla scissione degli atomi meno la quantità di energia prelevata dal processo master.

meltdown fallimento delle fork di uno qualunque dei processi

Il gruppo di studenti deve produrre configurazioni che siano in grado di generare la terminazione in ognuno dei casi sopra descritti.

Al termine della simulazione, l'output del programma deve riportare anche la causa di terminazione.

6 Descrizione del progetto: versione "normal" (max 30)

Nella versione completa del progetto, è presente anche un processo *inibitore* che controlla reazione attraverso i seguenti due meccanismi:

- assorbe parte della quantità di energia prodotta dalla scissione dell'atomo diminuendo la quantità di energia che viene liberata
- limita il numero di scissioni agendo sull'operazione di scissione rendendola probabilistica (ad esempio decidendo se la scissione debba avvenire o meno oppure trasformando in scoria uno degli atomi prodotti dopo la scissione)

Il meccanismo di assorbimento e quello di limitazione delle scissioni sono scelti dal programmatore e devono essere basati su qualche criterio adattivo.

La presenza o meno del processo inibitore deve poter essere scelta a run-time, all'inizio della simulazione. Nel caso in cui il processo inibitore sia attivo, ci si aspetta che la terminazione per "explode" e "meltdown" non avvenga.

Inoltre, l'utente deve poter fermare (e far ripartire) il processo inibitore più volte da terminale attraverso un meccanismo a scelta del programmatore.

7 Configurazione

Tutti i parametri di configurazione sono letti a **tempo di esecuzione**, da file o da variabili di ambiente. Quindi, un cambiamento dei parametri non deve determinare una nuova compilazione dei sorgenti (non è consentito inserire i parametri uno alla volta da terminale una volta avviata la simulazione).

8 Linee guida per la valutazione

Ogni secondo il sistema deve produrre una stampa in cui sono elencati:

- numero di attivazioni occorse ad opera del processo *attivatore* (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 (totale e relativa all'ultimo secondo);
- (per la versione "normal") quantità di energia assorbita dal processo *inibitore*
- (per la versione "normal") log delle operazioni di bilanciamento condotte dal processo *inibitore*.

9 Requisiti implementativi

Il progetto (sia in versione "minimal" che "normal") deve

- evitare l'attesa attiva
- utilizzare almeno memoria condivisa, semafori e un meccanismo di comunicazione fra processi a scelta fra code di messaggi o pipe,
- essere realizzato sfruttando le tecniche di divisione in moduli del codice (per esempio, i vari processi devono essere lanciati da eseguibili diversi con `execve(...)`),
- essere compilato mediante l'utilizzo dell'utility `make`
- massimizzare il grado di concorrenza fra processi
- deallocare le risorse IPC che sono state allocate dai processi al termine del gioco
- essere compilato con almeno le seguenti opzioni di compilazione:

  ```
  gcc -Wvla -Wextra -Werror
  ```

- poter eseguire correttamente su una macchina (virtuale o fisica) che presenta parallelismo (due o più processori).

Per i motivi introdotti a lezione, ricordarsi di definire la macro `_GNU_SOURCE` o compilare il progetto con il flag `-D_GNU_SOURCE`.