Relazione progetto SO a.a 2023/24 - Reazione a catena

Niccolò Faraca, Francesca Giaretto, Davide Giordano

1. **Master**
   Il processo master alloca le strutture dati necessarie, crea i processi alimentazione, attivatore e inibitore, e `N_ATOMI_INIT` atomi. Infine il master chiede se si desidera attivare l'inibitore fin da subito. Una volta eseguite queste operazioni, il semaforo `STARTSEM`, condiviso da tutti i processi, dà il via alla simulazione. A questo punto, il master si occupa solo delle seguenti operazioni:
   - stampare le statistiche ogni secondo;
   - azzerare le statistiche relative dopo averle stampate;
   - prelevare la quantità `ENERGY_DEMAND` di energia;
   - gestire la ricezione dei semafori;
   - gestire le terminazioni.

2. **Atomo**
   I processi atomo si collegano alle strutture dati create dal master, verificano di avere numero atomico maggiore di `MIN_N_ATOMICO`, mandano il proprio PID all'attivatore e chiamano la funzione pause(), dopodiché rimangono in attesa del segnale di scissione da parte dell'attivatore.

3. **Alimentazione**
   Il processo alimentazione crea per tutta la durata della simulazione un numero `N_NUOVI_ATOMI` di atomi ogni `STEP_ALIMENTAZIONE` secondi.

4. **Attivatore**
   Quando un atomo viene creato con successo, manda il suo PID all'attivatore tramite una coda di messaggi ed entra nella funzione pause(). Ogni `STEP_ATTIVATORE` nanosecondi, l'attivatore preleva il primo PID nella coda di messaggi e gli manda `SIGUSR2`, che sblocca l'atomo da pause() e gli permette di effettuare la scissione.

5. **Inibitore**
   Abbiamo deciso di far assorbire all'inibitore ogni `STEP_INIBITORE` nanosecondi un decimo dell'energia relativa all'ultimo secondo contenuta in shared memory.
   Per rendere probabilistica la scissione degli atomi, abbiamo deciso che quando l'inibitore è acceso, alla lettura del PID, verrà generato un numero randomico tra 0 e 10; se il numero è minore o uguale a 2, l'atomo effettuerà una scissione, sennò diventerà una scoria.

6. **Gestione terminazioni**
   La simulazione è governata dalla variabile *`termination`*, contenuta nella shared memory comune a tutti i processi. A inizio simulazione il suo valore è 0, ma diventa 1 nel momento in cui avviene una delle condizioni che provocano una terminazione (tranne per il caso di meltdown, che abbiamo gestito separatamente). In base alla

condizione verrà modificata `message`, anch'essa variabile in memoria condivisa, che indica il motivo della terminazione e viene stampata a schermo.

3.1 Timeout
Gestito come nell'introduzione del punto 3.

3.2 Meltdown
Alla prima occorrenza di meltdown viene inviato `SIGUSR2` al processo master che gestisce la terminazione nel signal handler in modo analogo agli altri casi. Per evitare che il messaggio di terminazione venga sovrascritto da altri meltdown abbiamo implementato un semaforo apposito che permette che questo segnale venga inviato una sola volta.

3.3 Explode
Gestito come nell'introduzione del punto 3.

3.4 Blackout
Gestito come nell'introduzione del punto 3.

3.5 Altre terminazioni
Per praticità di verifica abbiamo implementato due ulteriori terminazioni: "segmentation fault" e "by user" (quando l'utente invia ctrl + C).