**RELAZIONE – Progetto di laboratorio di sistemi operativi**

Abbiamo coordinato l'avanzamento dei compiti mediante l'uso di Github, lavorando in modo coordinato su componenti diverse del progetto. Inizialmente, abbiamo esaminato i requisiti e sviluppato una tabella di marcia per strutturare le attività, successivamente procedendo con l'implementazione in modo individuale. Per testare i risultati ottenuti, utilizzato macchine virtuali equipaggiate con sistemi operativi Ubuntu e Debian eseguite tramite Oracle VirtualBox. Inoltre, abbiamo effettuato test su una macchina in dual boot con sistema operativo Ubuntu.

**Struttura**

La struttura dei file è suddivisa in:

* config.txt: Configurazione testuale per i parametri recuperati a runtime;
* log.txt: File testuale generato contenente lo storico delle operazioni eseguite dall’inibitore;
* makefile: File utilizzato per semplificare la fase di compilazione;
* src: Insieme di file sorgente;

**Comunicazione fra processi**

Il nostro obiettivo primario consisteva nel conseguire un livello ottimale di concorrenza tra i processi, preservando simultaneamente l'integrità delle sezioni critiche. A tal fine, abbiamo impiegato tutti gli strumenti di seguito elencati:

**Utilizzo dei segnali**

Abbiamo scelto di enfatizzare l'uso dei segnali come principale mezzo di sincronizzazione, superando la semplice utilizzo di kill <signal> <pid>. Abbiamo implementato un apposito gruppo di processi per gli atomi, consentendoci di inviare il segnale a tutto il gruppo. Inoltre, abbiamo adottato anche la funzione sigqueue per inviare dati in concomitanza al segnale.

Nel master sono stati usati:

* SIGINT, per l’attivazione e disattivazione dell’inibitore da parte dell’utente;
* SIGTSTP, per interrompere prematuramente la simulazione da parte dell’utente oppure nel caso di errore generato all’interno di uno dei sottoprocessi;
* SIGKILL, per uccidere tutti i processi al termine della simulazione;

Nella comunicazione tra attivatore e atomi è stato usato:

* SIGUSR1, per avviare la fissione;

**Utilizzo della memoria condivisa**

La memoria condivisa ha rivestito un ruolo di rilievo nel contesto del nostro progetto, poiché abbiamo deliberatamente adottato questa come la principale infrastruttura di trasmissione dei dati tra i processi. L'abbiamo principalmente impiegata per la comunicazione di informazioni connesse alla fissione degli atomi, quali l'energia generata, il numero di fissioni e le scorie prodotte. Questa scelta si è dimostrata pressoché inevitabile, considerando l'elevato numero di processi da coordinare; nessun altro sistema di comunicazione avrebbe consentito un'implementazione altrettanto agevole per l'accesso ai dati.

**Utilizzo dei semafori**

Tuttavia, l'ampio ricorso alla memoria condivisa ci ha imposto di adottare un insieme di semafori al fine di regolare l'accesso alla sezione critica. Non si è trattato dell'unico insieme di semafori implementato, poiché ne abbiamo utilizzato un secondo per avviare la simulazione una volta che tutti i processi avevano completato l'inizializzazione dei propri dati.

**Utilizzo di pipe**

Abbiamo inoltre optato per l'impiego di una pipe al fine di facilitare la comunicazione tra il master e l'attivatore del numero di attivazioni. Benché questa scelta avrebbe potuto essere elusa mediante il ricorso esclusivo alla memoria condivisa, abbiamo deliberatamente deciso di implementarla per apportare una variazione al contesto.

**Utilizzo di pile di messaggi**

In conclusione, abbiamo scelto di implementare una pila di messaggi per agevolare la comunicazione tra il master e l'inibitore. Questa selezione potrebbe apparire superflua dato che l'inibitore ha accesso diretto alla memoria condivisa contenente tutti i dati necessari. Tuttavia, ciò non è del tutto accurato, poiché il processo master, ad ogni iterazione, provvede a ripristinare i dati presenti nella memoria condivisa. Pertanto, diventa necessario un ulteriore mezzo di comunicazione.

**Team**

Luca B, Luis B e Alessio C.