UNIVERSITÀ
DI TORINO

Relazione Progetto Sistemi Operativi 2022/2023

Lorenzo Pace - Leonardo Venturini - Davide Vitturini

Con la seguente relazione si vogliono descrivere le scelte implementative più importanti, le problematiche riscontrate e la loro risoluzione.
Una prima attenzione si pone alla lettura runtime dei parametri di configurazione da file, elencati come variabili intere o double.
Per assegnare il valore a queste variabili si utilizza una funzione, presente in ogni processo, che legge un file di testo. Questo file viene letto riga per riga e di ogni riga viene prelevato il nome del parametro e il valore del parametro, entrambi sotto forma di stringa. Successivamente il nome del parametro è confrontato all'interno di un if per identificare la variabile corretta. Il valore del parametro prima di essere assegnato viene convertito in intero o double. Ogni processo controlla e aggiorna solo i parametri a lui necessari.
Ora descriviamo e analizziamo tutti i processi utilizzati.

1. *Processo Master*

La simulazione viene avviata dal processo Master.
Esso si occupa: della generazione dei processi nave, porto e meteo, della stampa del dump al termine di ogni giorno simulato e della terminazione della simulazione.
Prima della generazione dei processi, crea e inizializza un semaforo al valore SO_PORTI+SO_NAVI+2, utilizzato per la sincronizzazione dei processi necessaria all'avvio della simulazione. I processi nave, porto e meteo una volta creati eseguono un'operazione di -1 sul semaforo ed una wait for zero. Il master, dopo la generazione dei processi figli, esegue anch'esso le medesime operazioni sul semaforo. Quando anche l'ultimo processo ha eseguito l'operazione -1, la simulazione può iniziare e tutti i processi sono sincronizzati.
Come ulteriore operazione preliminare genera anche tre memorie condivise. La prima è un array di struct di dimensione SO_PORTI, aventi come campi le coordinate e il pid di ogni porto. Questi vengono inseriti dalla funzione di generazione dei porti. Lo scopo è associare la posizione dei porti nella mappa con il corrispettivo pid. Non è protetta da un semaforo perché i processi vi accedono in sola lettura.

La seconda memoria condivisa è un array di struct di dimensione SO_NAVI, usata per sapere se una nave è in mare oppure in quale porti stia operando. I campi della struct sono due interi: port e ship. Il primo ha valore maggiore di zero se la nave è presso un porto di pid indicato dall'intero stesso, 0 se la nave è in mare e -1 se la nave è stata uccisa dal maelstrom mentre il secondo è il pid della nave. Inoltre è protetta da un semaforo mutex, per evitare inconsistenze poiché è accessibile in lettura e scrittura.
La terza memoria condivisa implementa il dump, alla quale hanno accesso i porti, le navi e il meteo. Questi modificano i valori del dump nel corso della simulazione, quindi è protetta da un semaforo mutex per evitare inconsistenze. Quando il master stampa il dump, richiede l'accesso alla memoria condivisa facendo una -1 sul semaforo mutex, fermando le eventuali modifiche che potrebbero arrivare dai processi figli. Una volta terminate le stampe rilascia il semaforo.
La durata del giorno è implementata attraverso una nanosleep di un secondo, al termine della quale il processo master esegue tre azioni:

1. invio del segnale SIGUSR1 al processo meteo per indicargli l'inizio di un nuovo giorno;
2. richiama la funzione di aggiornamento delle date di scadenza delle merci. Questa è implementata attraverso l'invio del segnale SIGUSR1 verso tutti i porti e verso tutte le navi in mare. Alla ricezione del segnale i porti aggiornano la data di scadenza delle merci di cui hanno offerta. Le navi in mare aggiornano le date di scadenza del proprio carico di merci;
3. richiama la funzione di stampa del dump.

Prima della nanosleep di un secondo, ossia all'inizio di ogni giorno simulato, viene richiamata la funzione di generazione giornaliera di merci, la quale inserisce in una coda di messaggi un numero casuale di messaggi. Questi sono implementati attraverso una struct contenente il tipo e la quantità di merce da inviare, e come type del messaggio il pid del porto a cui è inviato. Il master invia il segnale SIGALRM ai porti che devono generare la merce.
Al termine della simulazione il master invia un segnale di SIGTERM ai processi figli e si mette in attesa della loro terminazione con una wait.
Superata la wait esegue la stampa del dump finale e la deallocazione delle risorse IPC che ha creato.

2. *Processo Porto*

Il processo porto inizialmente preleva l'id di tutti i semafori e delle memorie condivise che gli serviranno nel corso della simulazione. In secondo luogo genera un semaforo inizializzato ad un numero di banchine estratto casualmente tra 1 e SO_BANCHINE. In seguito invoca le funzioni di generazione di offerta e domanda. Una volta generate si mette in attesa di un segnale attraverso una sigsuspend, quindi durante la simulazione i porti sono passivi e reagiscono solo all'arrivo di segnali.
I segnali che vengono gestiti dall'handler sono:

- SIGUSR2: gestione delle mareggiata;
- SIGUSR1: gestione aggiornamento scadenza merci;
- SIGALRM: generazione giornaliera merci;
- SIGTERM: terminazione processo.

Per gestire la mareggiata il porto blocca tutte le banchine libere, prelevandone il numero con una semctl, e decrementando il semaforo delle banchine con quel valore. Così facendo blocca l'accesso alle eventuali navi che lo richiederebbero durante la mareggiata. In seguito invia il segnale SIGQUIT alle navi che sono attraccate presso il porto, fermando così le operazioni di carico e scarico.
Ogni porto ha la propria coda di messaggi per gestire la domanda. Ogni messaggio è una struct con type identificato dal tipo della merce, e come contenuto un intero che rappresenta la quantità di lotti richiesta. La chiave è il pid del processo porto.
Inoltre ogni porto crea una memoria condivisa per la gestione delle offerte, avente come chiave il pid del porto stesso. Sono protette da un semaforo per impedire inconsistenze che verrebbero causate da letture e scritture contemporanee da parte di navi differenti. La memoria condivisa consiste in un array di struct di dimensione SO_MERCI. Le struct hanno 3 campi: due interi ed una struct good. Il primo intero indica se c'è domanda di quel tipo presso il porto, il secondo se c'è offerta. La struct good determina le informazioni sull'eventuale offerta di un determinato tipo di merce. Ha tre interi come campi. Il primo specifica il tipo della merce, il secondo la quantità di lotti e il terzo la data di scadenza.
Quando il porto riceve il segnale SIGTERM, dealloca le risorse IPC create ed esegue una exit(EXIT_SUCCESS).

3. Processo Nave

Il processo nave alla sua creazione si occupa di prelevare l'id di tutti i semafori e delle memorie condivise dei quali avrà bisogno durante la sua esecuzione.
I segnali che vengono gestiti dall'handler sono:

- SIGUSR2: gestione tempesta;
- SIGUSR1: gestione aggiornamento scadenza merci;
- SIGQUIT: gestione mareggiata;
- SIGALRM: gestione maelstrom;
- SIGTERM: terminazione del processo.

Il flag della struct sigaction è SA_NODEFER, necessario all'esecuzione annidata dei segnali.
Alla ricezione di un segnale le operazioni semop e nanosleep non vengono rieseguite quindi abbiamo aggiunto un ciclo while che controlli, qualora l'operazione ritorni -1, se l'errore è stato causato dall'invio del segnale, in tal caso viene eseguita nuovamente l'operazione.
Una volta completate le operazioni preliminari ed effettuata la sincronizzazione con gli altri processi, una nave cerca il porto più vicino e vi si dirige, utilizzando una nanosleep.
Quando una nave senza carico arriva presso un porto effettua l'attracco (docking) eseguendo le seguenti azioni: esegue un'operazione di -1 sul semaforo banchine, aggiorna la memoria condivisa delle ship indicando il porto presso il quale ha effettuato l'attracco e aggiorna di conseguenza il dump. Una volta in porto la nave compie le operazioni di ricerca ed eventuale caricamento della merce.
La ricerca avviene individuando la merce con la data di scadenza minore all'interno della memoria condivisa supply del porto. Se trovata, utilizzando la data di scadenza di quella merce, cerca un porto che la richieda e la cui distanza ne impedisca lo scadere durante il viaggio.
Il disormeggio viene implementato effettuando un'operazione di +1 sul semaforo banchine. Si modifica la memoria condivisa delle ship portando l'intero port a zero (nave in mare) e aggiornando il dump a seconda del carico o meno della nave.
Qualora non si trovi della merce verrà individuato il porto più vicino presso il quale inviare la nave. Questa operazione tiene conto del porto precedente, corrente e successivo così da evitare il più possibile il rimbalzo delle navi presso i porti.
L'aggiornamento delle date di scadenza delle merci viene gestito tramite il segnale SIGUSR1. Una volta ricevuto il segnale si effettua, tramite due semctl, un controllo per verificare se il semaforo del dump era già riservato al processo. Una volta effettuato il controllo si aggiorna il dump, prelevando o non prelevando il semaforo.
Quando la nave riceve il segnale SIGTERM o SIGALRM, dealloca le risorse IPC create ed esegue una exit(EXIT_SUCCESS). Nel caso del SIGALRM la nave, se trasporta un carico, effettua una msgsnd sulla coda di messaggi del porto a cui stava portando la merce, in modo tale da ricreare domanda di quel tipo.

4. Processo Meteo

Il processo meteo inizialmente preleva l'id di tutti i semafori e delle memorie condivise che gli serviranno nel corso della simulazione.
I segnali che vengono gestiti dall'handler sono:

- SIGUSR1: invio segnali di tempesta e mareggiata a navi e porti;
- SIGTERM: terminazione del processo.

Il flag della struct sigaction è SA_NODEFER, necessario all'esecuzione annidata dei segnali.
La gestione di tempesta e mareggiata avviene grazie alla ricezione del segnale SIGUSR1, inviato dal master, che indica l'inizio di un nuovo giorno.
Per la mareggiata viene scelto un porto causale, dalla memoria condivisa contenente le informazioni sui porti, e si invia il segnale SIGUSR1.
Per la tempesta viene prima creato un array contenente tutte le navi in mare e successivamente, se non vuoto, si sceglie in maniera randomica una nave e le si invia il segnale SIGUSR2.
Il processo meteo si occupa della gestione del maelstrom, andando a richiamare la corrispondente funzione ogni SO_MAELSTROM ore utilizzando una nanosleep.
Nello specifico la funzione inizialmente crea un array contenente tutte le navi in mare e successivamente, se non vuoto, sceglie in maniera randomica una nave e le invia il segnale SIGALRM.
Quando il meteo riceve il segnale SIGTERM, dealloca le risorse IPC create ed esegue una exit(EXIT_SUCCESS).