Corso di Sistemi Operativi, facoltà di Informatica - Università di Torino

Relazione progetto Sistemi Operativi 2022/2023

Autori:
Bergesio Simone
Grillo Giovanni
Del Ponte Lucrezio

Gen 2023

INPUT/OUTPUT

Il file "**Progetto.conf**" contiene

- le variabili di tipo SO_... che vengono inizializzate Run Time
- i flag per attivare / disattivare alcune funzioni del programma (SetOn..) Impostati di defaulta TRUE

Il file "**Progetto.out**" contiene l'output del programma

- **STAMPA INIZIALE**:
  contiene i porti, le navi e i tipi di merce creati
- **STAMPA GIORNALIERA**:
  -**merci** : in porto, in nave, scaduta in porto e in nave, rimasta ferma in porto

  -**navi**: in mare con carico, in mare senza carico, in porto, rallentate e distrutte

  -**porti**: l'offerta, la domanda, la merce spedita e ricevuta dal porto stesso, banchine libere / occupate
- **STAMPA FINALE:**
  -**merci** : in porto, in nave, scaduta in porto e in nave, rimasta ferma in porto Caricata da una nave, merce totale generata

  -**navi** : in mare con carico, in mare senza carico, in porto, rallentate e distrutte

  -**porti** : l'offerta, la domanda, la merce spedita e ricevuta dal porto stesso, banchine libere / occupate, porto che ha offerto e ricevuto più merce

  Nello stdout viene stampato lo stato dell'esecuzione del programma, con un piccolo riassunto delle operazioni giornaliere

  Nello stderr vengono stampati i messaggi di errore del programma (file che genera errore, n° riga, pid processo, codice errore, stringa associata)

IL PROGETTO

I porti, le navi e le merci sono definiti attraverso delle "struct".

- Per quanto riguarda le merci, il primo tipo di merce ha di default size = 1, per evitare problemi nella creazione qualora dovesse restare un'unità di merce da generare e le size dei tipi merce fossero tutte maggiori di 1

- Nei porti sono utilizzati dei vettori (offerta, domanda, spedita, ricevuta, scaduta) che contengono le informazioni necessarie per ogni "t_merce". La variabile "scadenza" è una matrice di tipo SO_DAYS SO_MERCI che contiene il giorno di scadenza e la quantità
- Per quanto riguarda le navi, vengono utilizzati dei vettori per la stiva e per LaTeXmerce scaduta. La variabile "scadenza" è una maniera gestita in maniera analoga ai porti

  I porti, le merci e le navi, oltre che le variabili globali, sono tutte in memoria condivisa, in modo che siano accessibili a tutti i processi

Nella "lib_header.h" sono contenute le funzioni per

- le librerie di C e alcune define (in particolare la PERMESSI, usata per associare la maschera di permessi agli IPCs creati, e la ERROR usata come macro per la segnalazione degli errori durante l'esecuzione e la fase di debuging)
- la creazione dei semafori e delle memorie condivise
- la release e la reserve per incrementare e decrementare il valore del semaforo
- le funzioni di montaggio e smontaggio delle memorie condivise (AttShm e DetShm che non solo permettono l'utilizio delle memorie ma ne gestiscono anche l'accesso escusivo attraverso il semavoro semShm)
- la deallocazione di tutti gli IPC utilizzati (l'unica memoria che necessima di essere smontata è la shmVar, dal momento che le altre sono gestite dal sistema AttShm/DetShm)

PROCESSO MASTER

Il Master è il processo che viene eseguito per dare il via all'intero programma.
Si occupa dell'allocazione degli IPC e della creazione iniziale di porti, navi e tipo merci
Una volta che la creazione è andata a buon fine, attraverso la funzione **execl()** vengono avvati i processi "Movimento.c", "Maelstrom.c" e "Giornaliera.c": i primi due verranno eseguiti in background in un loop invinito fino al momento in cui vengono terminati dal Master stesso, il processo Giornaliera invece viene lanciato e durante la sua esecuzione il processo Master viene messo in wait fino al suo completamento. Dopo la terminazione dei processi, il processi secondari, il processo insiste sul semaforo semProcessi (che tiene il conto dei processi figli generati dai sottoprocessi) fin quanto non si azzera (ovvero fino alla

terminazione di tutti i processi figlio), in seguito esegue la stampa finale, richiamare la deallocazione delle risorse e terminare.

Per terminare i processi sono state usate contemporaneamente 2 strade: ai processi Movimento e Maelstrom viene inviato il segnale di terminazione SIGTERM attraverso la funzione kill, in secondo luogo viene "attivata" la variabile globale condivisa var->flagTerminazione, controllata regolarmente dagli altri processi in modo da avviare l'operazione di terminazione quando opportuno.

Infine nel caso dall'analisi del valore del semaforo semProcessi risulti impossible termianre correttamente il programma (con un marigine di tolleranza arbitrariametne posto a 5 cicli di controllo in 5 secondi), viene forzata la terminazione del gruppo di processi associato al processo principale attraverso l'utilizzo della funzione killpg e il segnale SIGKILL.

In oltre per costruzione l'attShm necessita del var->flagTerminazione a 0, quini viene esso viene deincrementato nuovamente per poter eseguire la stampa finale.

PROCESSO GIORNALIERA

Il processo Giornaliera si occupa di gestire il trascorrere dei giorni. "Ogni giorno" che passa (sleep 1 sec) si occupa di creare l'offerta e la domanda, di gestire il decremento, le tempeste e le mareggiate

- **Il DECREMENTO**
  Ogni giorno si occupa di decrementare il tempo di vita delle merci di 1.
  Verifica inoltre se qualche merce presente in porto è scaduta e per quale quantità, e la "sposta" eventualmente dall'offerta alla merce scaduta; ripete poi lo stesso procedimento per le navi
- **L'OFFERTA**

  Viene generata giornalmente una quantità SO_FILL / SO_DAYS di merce offerta dai porti. Una volta scelto un porto casuale tra 0 e n_porti/2 e un tipo merce idoneo, viene creata una quantità random di merce che rispetti i parametri richiesti (non superi la quantità giornaliera da creare e sia multiplo della sua size unitaria)
- **LA DOMANDA**

  Viene generata giornalmente una quantità SO_FILL / SO_DAYS di merce richiesta dai porti. Una volta scelto un porto casuale tra n_porti/2 e n_porti e un tipo merce idoneo, viene creata una quantità random di merce che rispetti i

parametri richiesti (non superi la quantità giornaliera da creare e sia multiplo della sua size unitaria)

La scelta di dividere i porti in domandanti e offerenti è fatta per evitare che, con grandi quantità di merci da generare e piccole size unitarie, i tipi_merce siano già presenti in domanda/offerta, impedendo quindi di completare la creazione offerta/domanda

(Dovuto dal fatto che un tipo merce non può trovarsi sia in domanda che in offerta nello stesso porto)

Viene richiamata la funzione tempesta(), che ogni giorno sceglie in modo randomico una nave e la rallenta di SO_STORM_DURATION ore, e la funzione mareggiata(), che sceglie un porto e ne blocca le operazioni per SO_SWELL_DURATION ore.

PROCESSO MAELSTROM

Il maelstrom è gestito da un processo apposito che, ogni SO_MAELSTROM ore, finchè presenti, affonda una nave. Viene inviato un segnale di kill alla nave interessata e il PID della nave stessa viene messo a -1, in modo che non rientri più tra quelle attive. Il contenuto della stiva della nave viene perduto.

PROCESSO MOVIMENTO

Per gestire le navi è stato scelto di utilizzare una pila, in cui le navi vengono inizialemnte inserite in ordine di creazione e poi man mano vengono estratte dalla cima della pile e poi rimesse (sempre in cima) al termine della loro operazione. Per+ implementare questo concetto vengono usati 3 IPCs:

- una shmPila, formata da un vettore di dimensione (n_navi + 1): nelle prime n_navi posizioni vengono inseriti gli identificativi numerici associati alle navi, mentre l'ultima cella viene utilizzata per tenere il conto delle navi presenti nella pila.
- Una semaforo semPila diviso in 2 sottosemafori:

1. = Di dimensione n_navi, tiene anch'esso il conto delle navi presenti nella pila, in modo da bloccare il programma nel caso in cui la pila sia vuota, finquando non viene aggiunta nuovamente una nave.
2. = viene usato per regolare gli accessi alla shmPila.

La funzione di movimento si ramifica in due processi principali, uno che si occupa di generare le navi, mentre l'altro ne

gestisce le terminazioni. Il primo processo inizia scegliendo un porto casuale e estraendo una nave dalla pila (variabile che verranno ereditate dal primo processo figlio generato), poi aggiorna il pid della nave attraverso la funzione aggiornaPidNave (ogni nave conviene una valore pidNavi che può essere >0 nel caso in cui la nave sia attualmente impiegata in un operazione, =0 se la nave e ferma e =-1 se la nave è stata distrutta). Il secondo processo è invece un ciclo infinito che riceve e processa i messaggi dalla coda di messaggi msgPila: un messaggio contiene l'identificativo della nave che ha terminato la sua esecuzione, mentre il tipo del messaggio specifica che tipo di terminazione ha ricevuto (una normale terminazione ha il tipo 1 e prevede che venga rimesso a 0 il pid della nave in questione, mentre il tipo 2 identifica una terminazione di tipo maelstrom, per cui il pid della nave viene segnato come -1 e la nave non viene rimessa nella pila)

Ogni processo nave generato esegue un viaggio (e la sua nanosleep), un tentativo di attracco al porto e, se quest'ultimo è andato a buon fine, un'operazione di carico-scarico (e relativa nanosleep), infine viene liberata la banchine e viene inviato il messaggio al secondo processo in modo da renderla nuovamente disponibile prima di terminare.

Dal momento che la maggior parte delle operazioni sono da considerarsi "operazioni critiche" sulle memorie, è stato scelto di salvare i segnali ricevuti attraverso dei flag, in modo da controllarli solo nei momenti non-critici del processo. Per far ciò è stata creata una funzione controlloSegnali che prende come argomenti una maschera binaria contenente i flag (in or) da controllare in quella specifica situazione:

- o La tempesta viene controllata solo dopo il viaggio e prima dell'aggiornamento della posizione a quella del porto. Dal momento che una tempesta si limita a rallentate il viaggio della nave di un tempo fisso, sarà sufficiente aggiungere una nanosleep per quella durata.
- o La maelstrom viene controllata in ogni punto dell'esecuzione e, oltre a terminare il processo, si occupa anche dell'invio del messaggio di distruzione
- o La mareggiata infine viene controllata all'attracco e dopo le operazioni di carico/scarico della nave (la di caricamento dei valori nelle memorie condivise è

considerata una sezione critica). Per controllare la durata della mareggiata, viene usato un ciclo infinito che blocca l'esecuzione finché il valore del semaforo del porto del gruppo semMareggiata non è pari a 0 (ovvero finché non termina la mareggiata).

- o (inoltre questa funzione controlla anche il flag var->flagTerminazione per l'eventuale terminazione del processo per fine esecuzione

CARICO/SCARICO

Durante l'esecuzione della funzione Scarico, tutta la merce presente sulla nave che è anche richiesta dal porto viene trasferita ad esso. Il ritorno della funzione è il totale di merce scaricata.
Per ogni tipo merce viene controllato se rispetta i requisiti di scaricamento, in questo caso la merce viene spostata e la variabile scarico viene incrementata dalla quantità massima scaricabile (pari al minore tra la quantità richiesta e la quantità presente in nave).
Durante l'esecuzione della funzione Carico, viene spostata la merce presente in porto sulla nave fino a rimpirla (oppure finche non ci sono più merci di dimensione abbastanza piccola da essere trasferite sulla nave)