

ARTICLE TYPE

Relazione Progetto di Sistemi Operativi

Oskar Heise

Matricola: 976322, email: oskar.heise@edu.unito.it, Turno: T2

Consegna

Si intende simulare il traffico di navi cargo per il trasporto di merci di vario tipo, attraverso dei porti. Questo viene realizzato tramite i processi *master*, *mave* e *porto*.

Nella simulazione esistono diversi tipi diversi di merce. Queste vengono generate presso i porti. Le navi nascono da posizioni casuali e senza merce e il loro spostamento viene realizzato tramite una *nanosleep*. I porti hanno una certa domanda e offerta e le proprie banchine vengono gestite tramite *semafori*.

Master.c

Il processo Master è quello incaricato di creare gli altri processi e gestire la simulazione. Viene inoltre creato e inizializzato il semaforo nominato *semaforo_master*, indispensabile per la corretta sincronizzazione dei processi *nave* e *porto*. Prima della creazione di questi ultimi, viene aperta una memoria condivisa, necessaria per il passaggio delle informazioni da porti a navi. La creazione dei processi figli avviene tramite due cicli *for* distinti, ecco come funziona:

```

43   for(i = 0; i < SO_PORTI; i++){
44       sem_wait();
45       switch (){
46           case -1:
47               /*errore*/
48           case 0:
49               /*execvp*/
50           default:
51               /*inserimento array shm*/
52       }
53   }
```

Subito dopo l'ingresso nel ciclo, il semaforo, inizializzato a 1, viene diminuito di valore, attraverso un'operazione di *sem_wait*. I processi vengono creati attraverso l'operazione di *fork()* e uno *switch* si occupa di compiere le varie operazioni in base al valore ritornato:

- case -1: si è verificato un errore;
- case 0: accedo al processo porto attraverso una *execvp*;
- default: inserisco l'array di strutture contenente la merce nella memoria condivisa.

Per la creazione dei processi *nave* viene fatto lo stesso identico procedimento, con l'aggiunta di un'altra memoria condivisa necessaria per il conteggio delle statistiche delle navi. Al termine del processo *master*, mi preoccupo di deallocare le memorie condivise e di rimuovere i semafori nominati.

Porti.c

I porti vengono localizzati in una certa posizione nella mappa e gestiscono un numero casuale di banchine. I primi quattro porti vengono sempre generati agli angoli della mappa. Questo viene fatto attraverso *generatore_posizione_iniziale_porto()*, presente nella libreria personalizzata *header.h* di cui parlerò in seguito.

All'inizio del codice dei porti ricevo l'array della memoria condivisa, e procedo a generare tutte le informazioni dei porti. tutte queste informazioni vengono inserite in un array di strutture e inserite nella memoria condivisa. Nel processo viene inoltre gestito *semaforo_master* e incrementato di 1 attraverso l'operazione di *sem_post*.

L'operazione più particolare presente nel codice è la seguente:

```

19   /*setto indici*/
20   pid_di_stampa = (getppid()) / (getpid()
    - SO_PORTI + 1);
```

Questa operazione sarà anche presente nel processo *nave*, e mi è risultata utile per gestire l'operazione di stampa del risultato finale. Il calcolo che viene fatto dà come risultato 0 ad un solo processo e 1 a tutti gli altri. In questo modo, solamente un processo si occuperà della pubblicazione dei report giornalieri e finale, e si eviteranno così ripetizioni indesiderate. All'interno dei processi porto, *pid_di_stampa* viene utilizzato per la creazione della coda di messaggi.

L'altra operazione dubbia presente nel codice è la seguente:

```

56   shared_memory_porto[(getpid() - getppid())
    - 1] = porto;
```

In questa linea di codice, utilizzo nuovamente i *pid* dei processi per utilizzarli come indici di un array, ed inserire in maniera corretta le informazioni di ciascun porto all'interno della memoria condivisa che utilizzerò successivamente nei processi *nave*.

Navi.c

Le navi sono i processi più importanti dell'intero programma, ed è qui che gestisco la maggior parte delle operazioni del Progetto, ovvero:

- Movimento delle navi;
- Scambio delle merci;
- Stampa dei report.

Una buona parte del codice è dedicata alla memorizzazione delle variabili utili per le statistiche intermedie e finali, e la prima cosa che faccio nel codice è proprio quella di resettarle. Successivamente, mi occupo di ricevere i diversi array dalla memoria condivisa, in particolare le informazioni dei porti. Dopo, genero tutte le informazioni della nave, in particolare la posizione iniziale della nave, attraverso la funzione `generatore_posizione_iniziale_nave()`.

Una parte fondamentale del processo *nave* è quella della gestione del tempo. Ho deciso di gestire il timer attraverso la creazione di un *thread*:

```
138 pthread_create(&tid, NULL, &threadproc,
                NULL);
```

Ecco cosa significano i parametri della funzione:

- *&tid*: puntatore ed identificare di thread;
- *NULL*: indica che si vuole un comportamento di default;
- *&threadproc*: nome della procedura da far eseguire al thread;
- *NULL*: puntatore che viene passato come argomento a *start_routine*, in questo caso niente.

Questa funzione è presente prima del *main* e si occupa interamente della stampa dei report e, al termine del tempo simulato, si occupa anche di terminare tutti i processi. La stampa viene effettuata solamente da un processo, scelto attraverso l'operazione spiegata in nel capitolo precedente e contenuta nella variabile *pid_di_stampa*. L'unità di misura del secondo viene gestita attraverso una *sleep()*, che "addormenta" il thread per un secondo prima di ricominciare a stampare il report. Quando i giorni sono terminati, oppure quando sia la richiesta che l'offerta sono a 0, viene stampato il report finale. Tornando nel *main*, attraverso un ciclo *while*, comincia la simulazione vera e propria. Ho diviso la simulazione in diversi step:

1. Controllo della scadenza della merce nelle navi e nei porti;
2. Scelta del porto in cui sbarcare;
3. Spostamento della nave;
4. Memorizzazione dei valori per le statistiche parziali e finali;
5. Caricamento e scaricamento delle navi;

Dopo aver controllato la scadenza della merce, attraverso dei semplici cicli *for*, mi occupo di scegliere il porto in cui far sbarcare la nave:

```
180 if((informazioni_porto[i].
    merce_richiesta_id == merce_nella_nave[
    indice_nave].id_merce || tappe_nei_porti
    == 0 || merce_nella_nave[indice_nave].
```

```
tempo_vita_merce < 0) &&
distanza_minima_temporanea >
distanza_nave_porto(nave.posizione_nave,
    informazioni_porto[i].posizione_porto_X
    , informazioni_porto[i].
    posizione_porto_Y) && informazioni_porto
[i].numero_banchine_libere != 0 &&
informazioni_porto[i].numero_lotti_merce
> 0)
```

Questa condizione, sceglie semplicemente il porto più vicino, qualora la nave fosse priva di carico, ma, in caso contrario, controllando anche se ci sono delle banchine libere e se la richiesta è compatibile con il carico dell'imbarcazione. Successivamente la nave viene "teletrasportata" al porto dopo un certo numero di tempo, grazie alla funzione *nanosleep* e calcolato attraverso la formula:

$$\frac{\text{distanza tra posizione di partenza e arrivo}}{\text{velocità di navigazione}}$$

Una volta arrivata al porto, l'accesso è controllato attraverso un semaforo apposito dichiarato ad inizio codice e qualora lo spazio nelle banchine fosse presente, la nave è pronta a scaricare e caricare la nuova merce.

Una volta fatto ciò, la nave è pronta a ripartire.

header.h

La libreria *header.h* contiene tutte le librerie utilizzate nel codice, alcune costanti, strutture e variabili utilizzate da molti dei processi utilizzati e, ultime ma non meno importanti, tutte le funzioni impiegate nello sviluppo del codice.

Le funzioni più interessanti sono probabilmente quelle dedicate alla gestione delle risorse di *Intern Process Communication*, in particolare la memoria condivisa e le code di messaggi. In tutte queste funzioni ho inserito un controllo, che avvisa l'utente attraverso *errno*. In caso di errore, le memorie condivise, per esempio, vengono eliminate attraverso la funzione *shmctl* e il comando *IPC_RMID*.

Le altre due funzioni interessanti presenti nella libreria personalizzata, sono ovviamente quelle incaricate della stampa dei report giornalieri e finali:

```
void print_report_finale(struct
    struct_conteggio_nave *conteggio_nave,
    struct_struct_merce *merce_nella_nave,
    int numero_giorno, struct_struct_porto *
    informazioni_porto, int *
    somma_merci_disponibili, int *
    conteggio_merce_consegnata, int *
    totale_merce_generata_inizialmente, int
    *merce_scaduta_in_nave, int *
    merce_scaduta_in_porto, int
    tappe_nei_porti)
```

Queste due funzioni ricevono un gran numero di dati, e si occupano di stamparli, oltre a fare su alcuni di essi una serie di ulteriori calcoli.

makefile

Make è un'*utility* che automatizza il processo di compilazione ed esecuzione dei programmi. Come da consegna ho aggiunto le seguenti opzioni di compilazione:

```

1 compexec: compile execute
2
3 compile:
4     gcc -std=c89 -Wpedantic Master.c -o
      Master
5     gcc -std=c89 -Wpedantic Porti.c -o
      Porti
6     gcc -std=c89 -Wpedantic Navi.c -o Navi
7
8 execute:
9     ./Master
10
11 gdb:
12     gcc -std=c89 -Wpedantic -g -O0 Master.c
      -o Master
13     gcc -std=c89 -Wpedantic -g -O0 Porti.c
      -o Porti
14     gcc -std=c89 -Wpedantic -g -O0 Navi.c -
      o Navi
15     gdb Master

```

Funzionamento e Risultati

Una volta che ci si reca nella cartella dedicata al progetto, si compila il tutto attraverso il comando *make compexec*, che compila ed esegue il codice. Ecco un esempio del giorno 1 (screenshot al fondo del documento):

```

1 REPORT GIORNO 1
2 Merci:
3     Tipologia: 0 -> Disponibile: 56532
      tonnellate & Consegna: 0 tonnellate
4     Tipologia: 1 -> Disponibile: 0 tonnellate
      & Consegna: 0 tonnellate
5     Tipologia: 2 -> Disponibile: 12084
      tonnellate & Consegna: 0 tonnellate
6 Navi:
7     Con un carico a bordo: 0 - Senza un carico
      a bordo: 5 - Nel porto: 0
8 Porti:
9     Numero porto: 1 - Merce disponibile: 12084
      - Merce totale spedita e ricevuta in
      tonnellate: 0, 0 - Numero di banchine
      libere: 4 - Lotti rimanenti: 2
10    Numero porto: 2 - Merce disponibile: 10044
      - Merce totale spedita e ricevuta in
      tonnellate: 0, 0 - Numero di banchine
      libere: 2 - Lotti rimanenti: 3
11    Numero porto: 3 - Merce disponibile: 46488
      - Merce totale spedita e ricevuta in
      tonnellate: 0, 0 - Numero di banchine
      libere: 3 - Lotti rimanenti: 6
12    Numero porto: 4 - Merce disponibile: 0 -
      Merce totale spedita e ricevuta in
      tonnellate: 0, 0 - Numero di banchine
      libere: 0 - Lotti rimanenti: 0

```

Una volta terminati i giorni o le merci disponibili dei porti, viene stampato il report finale (screenshot al fondo del documento):

```

1 REPORT FINALE
2 Navi:
3     Numero di navi in mare con un carico a
      bordo: 5
4     Numero di navi in mare senza un carico: 0
5     Numero di navi che occupano una banchina:
      0
6 Merci:
7     Tipologia: 0 ->
8     Disponibile: 6104 tonnellate & Consegna:
      1239 tonnellate
9     Tonnellate di merce iniziale: 56532
10    Tonnellate rimaste nel porto: 6104 &
      Scadute nel porto: 0 & Scadute nella
      nave: 0 & Consegna da qualche nave:
      1239
11    Tipologia: 1 ->
12    Disponibile: 0 tonnellate & Consegna: 0
      tonnellate
13    Tonnellate di merce iniziale: 0
14    Tonnellate rimaste nel porto: 0 & Scadute
      nel porto: 0 & Scadute nella nave: 0
      & Consegna da qualche nave: 0
15    Tipologia: 2 ->
16    Disponibile: 0 tonnellate & Consegna: 0
      tonnellate
17    Tonnellate di merce iniziale: 12084
18    Tonnellate rimaste nel porto: 0 & Scadute
      nel porto: 0 & Scadute nella nave: 0
      & Consegna da qualche nave: 0
19 Porti:
20    Numero porto: 1 - Merce totale spedita e
      ricevuta in tonnellate: 3021, 0 -
      Numero di banchine libere: 5 - Lotti
      rimanenti: 1
21    Numero porto: 2 - Merce totale spedita e
      ricevuta in tonnellate: 1116, 0 -
      Numero di banchine libere: 3 - Lotti
      rimanenti: 2
22    Numero porto: 3 - Merce totale spedita e
      ricevuta in tonnellate: 4997, 1239 -
      Numero di banchine libere: 6 - Lotti
      rimanenti: 2
23    Numero porto: 4 - Merce totale spedita e
      ricevuta in tonnellate: 0, 0 - Numero
      di banchine libere: 0 - Lotti rimanenti
      : 0
24
25 Il porto che ha offerto la quantita'
      maggiore di merce e': 3
26 Il porto che ha richiesto la quantita'
      maggiore di merce e': 3
27
28 FINE SIMULAZIONE
29

```

Difetti e Conclusione

Per onestà intellettuale, è doveroso elencare i difetti e i malfunzionamenti del codice da me prodotto. Questi problemi si concentrano tutti sulla stampa dei report giornalieri e finale. Quello che a volte succede, difatti, è che l'esecuzione termini in punti casuali senza alcuna spiegazione apparente, oppure che il report finale non venga prodotto. L'unica spiegazione che mi è venuta in mente è che i processi vengano terminati a causa di qualche problema dettato dai dati inseriti e dal codice, e che quindi il processo incaricato di gestire la stampa si trovi impossibilitato a farlo. Purtroppo questo problema è invadente e rende difficoltoso analizzare con costanza i risultati dell'esecuzione.

Altri problemi minori sono piccoli errori nei valori delle statistiche. Talvolta i porti non vengono popolati di merci. Qui di seguito, gli screenshot dei risultati

```
REPORT GIORNO 1
Merci:
  Tipologia: 0 -> Disponibile: 56532 tonnellate & Consegnata: 0 tonnellate
  Tipologia: 1 -> Disponibile: 0 tonnellate & Consegnata: 0 tonnellate
  Tipologia: 2 -> Disponibile: 12084 tonnellate & Consegnata: 0 tonnellate
Navi:
  Con un carico a bordo: 0 - Senza un carico a bordo: 5 - Nel porto: 0
Porti:
  Numero porto: 1 - Merce disponibile: 12084 - Merce totale spedita e ricevuta in tonnellate: 0, 0 - Numero di banchine libere: 4 - Lotti rimanenti: 2
  Numero porto: 2 - Merce disponibile: 10044 - Merce totale spedita e ricevuta in tonnellate: 0, 0 - Numero di banchine libere: 2 - Lotti rimanenti: 3
  Numero porto: 3 - Merce disponibile: 46488 - Merce totale spedita e ricevuta in tonnellate: 0, 0 - Numero di banchine libere: 3 - Lotti rimanenti: 6
  Numero porto: 4 - Merce disponibile: 0 - Merce totale spedita e ricevuta in tonnellate: 0, 0 - Numero di banchine libere: 0 - Lotti rimanenti: 0
```

```
REPORT FINALE
Navi:
  Numero di navi in mare con un carico a bordo: 5
  Numero di navi in mare senza un carico: 0
  Numero di navi che occupano una banchina: 0
Merci:
  Tipologia: 0 ->
    Disponibile: 6104 tonnellate & Consegnata: 1239 tonnellate
    Tonnellate di merce iniziale: 56532
    Tonnellate rimaste nel porto: 6104 & Scadute nel porto: 0 & Scadute nella nave: 0 & Consegnata da qualche nave: 1239
  Tipologia: 1 ->
    Disponibile: 0 tonnellate & Consegnata: 0 tonnellate
    Tonnellate di merce iniziale: 0
    Tonnellate rimaste nel porto: 0 & Scadute nel porto: 0 & Scadute nella nave: 0 & Consegnata da qualche nave: 0
  Tipologia: 2 ->
    Disponibile: 0 tonnellate & Consegnata: 0 tonnellate
    Tonnellate di merce iniziale: 12084
    Tonnellate rimaste nel porto: 0 & Scadute nel porto: 0 & Scadute nella nave: 0 & Consegnata da qualche nave: 0
Porti:
  Numero porto: 1 - Merce totale spedita e ricevuta in tonnellate: 3021, 0 - Numero di banchine libere: 5 - Lotti rimanenti: 1
  Numero porto: 2 - Merce totale spedita e ricevuta in tonnellate: 1116, 0 - Numero di banchine libere: 3 - Lotti rimanenti: 2
  Numero porto: 3 - Merce totale spedita e ricevuta in tonnellate: 4997, 1239 - Numero di banchine libere: 6 - Lotti rimanenti: 2
  Numero porto: 4 - Merce totale spedita e ricevuta in tonnellate: 0, 0 - Numero di banchine libere: 0 - Lotti rimanenti: 0
Il porto che ha offerto la quantita' maggiore di merce e': 3
Il porto che ha richiesto la quantita' maggiore di merce e': 3
FINE SIMULAZIONE
```