SEMAFORI IN DISASTROS

I semafori

I semafori sono dei costrutti che, grazie ad una variabile intera memorizzata al loro interno, garantiscono la mutua esclusione tra processi, sincronizzando così l'accesso a una o più risorse condivise.

Un semaforo viene inizializzato con valore non negativo, dopodiché sono possibili due operazioni: decremento e incremento. Se un processo decrementa il valore del semaforo, e questo assume valore negativo, allora il processo verrà messo in stato di attesa (per il momento non può accedere alla risorsa condivisa), fino a quando un altro processo non lo sbloccherà; questo può avvenire con l'operazione di incremento, poiché se dopo l'incremento il semaforo è non positivo, allora uno dei processi in attesa sarà messo in stato di *ready*.

Problema del produttore-consumatore

Supponendo di avere un buffer condiviso che può contenere dati, è possibile che alcuni processi (produttori) generino questi dati e li inseriscano nel buffer, mentre gli altri (consumatori) accedano all'area condivisa per prendere questi dati e servirsene in qualche modo. È chiaro che l'accesso al buffer deve essere regolato, in particolare:

- solo un processo alla volta può accedere all'area condivisa (mutua esclusione);
- un produttore non può aggiungere dati se il buffer è pieno;
- un consumatore non può prelevare dati se il buffer è vuoto.

Tutto ciò è possibile usando tre semafori: uno il cui valore indica quanti "spazi" liberi sono presenti nel buffer (e perciò bloccherà o sbloccherà i produttori); uno che indica quanti "spazi" sono invece occupati (blocca o sblocca i consumatori); infine un semaforo binario (assume solo valore 0 o 1), che garantisce la mutua esclusione nella sezione critica. Il modello produttore-consumatore sarà usato per testare i semafori.

Semafori e DisastrOS

In DisastrOS, i semafori sono definiti con delle struct, e per il loro funzionamento è stato necessario implementare quattro chiamate a sistema:

- `disastrOS_semOpen(int semaphore_id, int count)` prende in input l'id del semaforo e il valore contenuto al suo interno `count`, il quale deve essere non negativo (in caso contrario ritorna un errore). Se il semaforo ancora non esiste, esso viene allocato con gli `id` e `count` specificati; dopodiché, il processo corrente salva nella sua lista di descrittori dei semafori quello del semaforo appena aperto, mentre il semaforo memorizza un puntatore a tale descrittore nella sua lista di descrittori. La funzione ritorna il nuovo *file descriptor* del semaforo (intero non negativo), o un intero negativo in caso di errore.

- `disastrOS_semClose(int fd)` prende come parametro il *file descriptor* del semaforo da chiudere, e "dissocia" il processo corrente dal semaforo. Se la funzione trova con

successo fd tra i descrittori del processo, allora questo viene rimosso, e lo stesso accade anche al suo puntatore salvato nel semaforo; dopodiché, entrambe le risorse vengono liberate. Infine, se il semaforo non è utilizzato da altri processi, viene rimosso dalla lista di semafori globale e liberato. Viene restituito 0 in caso di successo, un intero negativo in caso di errore.

- disastrOS_semWait(int fd) è la funzione che decrementa count. Dopo aver cercato, come per semClose, il descrittore del semaforo opportuno grazie a fd, il count del semaforo viene diminuito di una unità. Se count è minore di 0 il processo deve aspettare: il puntatore al descrittore salvato nel semaforo viene spostato nella lista di waiting_descriptors, viene cercato nella lista globale di processi *ready* il prossimo processo da eseguire, il processo corrente viene messo nella lista globale di processi in attesa (e quindi il suo stato diventa Waiting), ed infine può essere eseguito il prossimo processo. La funzione ritorna 0 se ha successo, un intero negativo in caso contrario.

- disastrOS_semPost(int fd) serve per incrementare il contatore nel semaforo. Prende in input il *file descriptor* del semaforo e lo cerca tra i descrittori associati al processo, poi incrementa di 1 il count. Se il suo valore è non positivo, allora uno dei processi in attesa può essere sbloccato: il primo descrittore in waiting_descriptors viene spostato nei descrittori non in attesa, il relativo processo assume stato Ready e viene spostato dalla waiting_list nella ready_list (entrambe globali). La funzione restituisce 0 in caso di successo o un intero negativo se c'è un errore.

Il test

Possiamo testare il funzionamento dei semafori con il modello produttore-consumatore, implementato in DisastrOS_test.c.

Abbiamo un vettore buf di dimensione SIZE contenente degli interi. Il buffer è accessibile da più processi da regolare con i semafori, e viene inizializzato con tutti zeri al suo interno. Inizialmente, ogni processo apre i tre semafori necessari (come già spiegato) per la mutua esclusione: uno con count pari a SIZE, uno con count 0, e uno con count 1. Viene anche aperto un quarto semaforo con valore negativo, ma serve solo per testare il funzionamento degli errori. Ogni processo sarà un produttore o un consumatore a seconda del suo *process ID*, e tenterà di utilizzare il buffer condiviso secondo la seguente logica:

- uno 0 all'interno del vettore simula uno "spazio" del buffer vuoto, cioè deve ancora essere occupato da un produttore o l'elemento che conteneva è stato prelevato da un consumatore;
- quando un produttore accede in sezione critica al buffer, inserisce al suo interno un elemento; al posto di uno degli zeri scriveremo il *PID* del processo;
- quando un consumatore accede in sezione critica al buffer preleva da esso un elemento: il processo sostituirà quindi un intero diverso da 0 con uno 0 (lo spazio ora è libero).

Alla fine della loro esecuzione i processi chiudono i semafori.

Per provare il programma è necessario compilarlo con make ed eseguirlo con ./disastrOS_test.

Stefano Bonetti, 1764624