DisastrOS Semaphore Project

WHAT

L'obiettivo del progetto era quello di implementare i semafori nel sistema operativo didattico disastrOS.

Il funzionamento di un semaforo è del tutto analogo a quello utilizzato nelle città: regolare il traffico affinché solo un preciso numero di processi possano al massimo accedere ad una risorsa condivisa o alla sezione critica di un programma, ossia una parte di codice che, se eseguita simultaneamente da più processi, potrebbe portare inconsistenza nei risultati finali. In questo progetto si è andato ad implementare il classico problema produttore-consumatore, incontrato in passato in corsi come Sistemi di Calcolo parte II.

L'implementazione di un semaforo si concretizza nella realizzazione di quattro system call:

- semOpen, che permette di creare il semaforo
- semClose, utilizzato per la sua chiusura
- semWait, che viene chiamata da un processo che vuole mettersi in attesa del suo turno per operare
- semPost, che viene chiamata da un processo non appena ha terminato di operare

La logica di funzionamento del semaforo resta quella "classica". Quando un processo lancia una semWait una variabile interna al semaforo viene decrementata, se il suo valore è minore di zero il processo viene messo in pausa. Quando un processo lancia una semPost il contatore viene incrementato di uno e, se il contatore è minore o uguale a 0, un nuovo processo fra quelli in attesa passa nello stato Running.

HOW

int SemOpen (int id, int count)

Passati in input un numero identificativo del semaforo (id) ed il numero dei processi che simultaneamente devono essere "lasciati passare" (count), deve essere allocato un semaforo purché questo non sia stato già creato in precedenza (e quindi se presente nella lista globale dei semafori, chiamata *semaphore_list*). Si deve poi procedere ad apire un nuovo descrittore che possa essere usato dal processo corrente (e che quindi sia inserito nella sua lista dei descrittori) ed un puntatore al descrittore che sia inserito nella lista dei descrittori attivi del semaforo. Output: in caso di successo il file descriptor del semaforo che è stato aperto, diversamente un errore

int SemClose (int fd)

Passato in input il file descriptor di un semaforo lo rimuove dalla lista dei descrittori del processo corrente (e lo dealloca); rimuove il puntatore al descrittore nella lista dei descrittori attivi del semaforo (e lo dealloca) e, se il semaforo non ha più descrittori attivi o descrittori dei processi in attesa, rimuove il puntatore al descrittore dall lista globale dei descrittori e lo dealloca. Output: in caso di successo viene restituito il valore zero, diversamente un errore

int SemWait (int fd)

Passato in input il file descriptor, esegue le seguenti operazioni: - decrementa il contatore del semaforo - se il contatore ha un valore negativo: - il processo viene messo in stato Waiting - il descrittore viene messo nella lista dei descrittori in attesa - il processo viene messo fra quelli in attesa - viene prelevato il primo processo nella lista Ready e viene messo in esecuzione

Output: in caso di successo viene restituito il valore zero, diversamente un errore

int SemPost (int fd)

Passato in input il file descriptor, esegue le seguenti operazioni: - incrementa il contatore del semaforo - se il contatore ha valore minore o uguale a zero prepara il processo ed il suo descrittore prendendolo dalla lista di quelli in attesa e lo mette in stato Ready

Output: in caso di successo viene restituito il valore zero, diversamente un errore

disastrOS_test.c

Per testare il corretto funzionamento delle funzioni realizzate è stato implementato il problema del produttore-consumatore. I processi sono suddivisi equamente in produttori (che scrivono dati su un buffer) e consumatori (che leggono i dati sullo stesso buffer). Non solo deve essere consentita la lettura e la scrittura ad un solo processo in un dato momento, ma deve essere anche regolata la produzione di dati affinché non si vadano a sovrascrivere dati non ancora letti o si vadano a leggere dati non ancora prodotti.

HOW TO RUN

make -> ./disastrOS_test -> "Oh yeah!" :)