DISASTROS_SEMAPHORES

-WHAT:

Il progetto consiste nell'implementazione dei semafori e delle funzioni che li regolano nel sistema operativo disastrOS.
Un semaforo è un tipo di dato astratto che permette ai processi di accedere ad una risorsa condivisa o alla sezione critica di un programma (nel nostro caso operazione di lettura/scrittura su un buffer circolare) in mutua esclusione e in modo sincronizzato.
Un semaforo consiste in una variabile intera e sfrutta due operazioni:
1) WAIT: quando un processo vuole accedere alla sezione critica o richiedere accesso alla risorsa viene eseguita una wait sul semaforo e il suo contatore viene decrementato;
2)POST: quando un processo abbandona la sezione critica o rilascia la risorsa viene eseguita una post sul semaforo e il suo contatore viene incrementato.

-HOW:

Sono state implementate all'interno del codice sorgente del sistema quattro system call che gestiscono il funzionamento dei semafori:

1)semOpen: crea un semaforo con un id e un contatore scelti dall'utente. Se il semaforo in questione è già presente nella lista globale dei semafori apre solamente un nuovo descrittore (che viene inserito anche come entry della lista dei descrittori del processo corrente), altrimenti lo alloca e lo aggiunge alla lista dei semafori.
La semOpen inoltre crea un puntatore al descrittore, che serve da entry nella lista dei descrittori attivi del semaforo.
Infine incrementa il contatore dei descrittori attivi del processo in esecuzione e se ha funzionato tutto correttamente imposta come valore di ritorno il file descriptor del semaforo appena aperto (altrimenti il valore di ritorno dipende dal tipo di errore).

2)semClose: dato in input il file descriptor di un semaforo lo rimuove e lo dealloca dalla lista dei descrittori del processo in esecuzione. Rimuove e dealloca anche il puntatore al descriptor nella lista dei descrittori attivi del semaforo, e se il semaforo non ha più descrittori attivi lo rimuove dalla lista globale e lo dealloca.
Infine decrementa il contatore dei descrittori attivi e se ha funzionato tutto correttamente imposta il valore di ritorno a zero (altrimenti il valore di ritorno dipende dal tipo di errore).

3)semWait: dato in input il file descriptor, decrementa il contatore del semaforo ad esso associato. Se il contatore diventa negativo inoltre sposta il descrittore e il processo nelle rispettive liste di attesa, imposta lo stato del processo corrente in Waiting e mette in esecuzione un nuovo processo, più precisamente il primo della lista dei processi nello stato di Ready (fondamentalmente esegue un context switch).
Infine se ha funzionato tutto correttamente imposta il valore di ritorno a zero (altrimenti il valore di ritorno dipende dal tipo di errore).

4)semPost: dato in input il file descriptor, incrementa il contatore del semaforo ad esso associato. Se dopo l'incremento il contatore è ancora negativo o nullo sposta il descrittore e il processo nelle rispettive ready queue e imposta lo stato del processo corrente in Ready.
Infine se ha funzionato tutto correttamente imposta il valore di ritorno a zero (altrimenti il valore di ritorno dipende dal tipo di errore).

È stato inoltre implementato nel programma di test un esempio del funzionamento dei semafori ispirato ad un modello con n produttori e n consumatori nel quale un buffer circolare di 50 elementi inizializzati a

0 subisce operazioni di lettura e scrittura da più processi in modo sincronizzato grazie all'utilizzo dei semafori e delle funzioni appena descritte.

-HOW TO RUN:
Per eseguire il test è sufficiente recarsi nella cartella 'sorgente', compilare tutti i file con 'make' ed eseguire il programma di test con './disastrOS_test'.