Prova pratica di Calcolatori Elettronici (nucleo v6.*)

C.d.L. in Ingegneria Informatica, Ordinamento DM 270

17 luglio 2010

1. Definiamo un *monitor* come un oggetto in cui può *entrare* un solo processo alla volta, e a cui è associata una *variabile di condizione*. Il processo che si trova all'interno del monitor è detto *possessore* del monitor. Il possessore del monitor può eseguire le operazioni di *attesa* e *notifica* sulla variabile di condizione associata. L'operazione di attesa rilascia il monitor e sospende il processo fino a quando un altro processo esegue l'operazione di notifica sulla stessa variabile di condizione. L'operazione di notifica risveglia uno dei processi in attesa sulla variabile di condizione, se ve ne sono, altrimenti non fa niente. Se un processo ne risveglia un altro tramite una operazione di notifica, gli cede anche il possesso del monitor, quindi si sospende su una coda di processi "urgenti" associata al monitor. I processi i "urgenti" hanno precedenza nel riottenere il possesso del monitor, non appena questo si libera.

   Per realizzare i monitor definiamo la seguente struttura (file `sistema.cpp`):

   ```
   struct des_monitor {
       natl owner;
       des_proc *w_enter;
       des_proc *w_cond;
       des_proc *urgent;
   };
   ```

   Il campo `owner` contiene l'id del processo che possiede il monitor (0 se nessun processo possiede attualmente il monitor). Il campo `w_cond` rappresenta la lista dei processi in attesa sulla variabile di condizione associata al monitor. Il campo `w_enter` rappresenta la lista dei processi in attesa di poter entrare nel monitor. Il campo `urgent` rappresenta la lista dei processi "urgenti".

   Le seguenti primitive, accessibili dal livello utente, operano sui monitor (nei casi di errore, abortiscono il processo chiamante):

   - `natl monitor_ini()` (già realizzata): inizializza un nuovo monitor, con tutti i campi a 0. Restituisce l'identificatore del nuovo monitor. Se non è possibile creare un nuovo monitor, restituisce `0xFFFFFFFF`.
   - `void monitor_enter(natl mon)` (già realizzata): tenta di entrare nel monitor di identificatore `mon`. Se il monitor appartiene già a qualche altro processo, sospende il processo in attesa di potervi entrare. È un errore tentare di entrare in un monitor che si possiede già.
   - `void monitor_leave(natl mon)`: esce dal monitor di identificatore `mon`. È un errore tentare di uscire da un monitor che non si possiede. La primitiva deve cedere il possesso del monitor ad uno degli eventuali processi "urgenti" o in attesa di entrare nel monitor, se ve ne sono.
   - `void monitor_wait(natl mon)`: si pone in attesa sulla variable di condizione associata al monitor di identificatore `mon`. È un errore tentare di porsi in attesa su una variable di condizione associata ad un monitor che non si possiede. Prima di sospendersi, il processo deve cedere il possesso del monitor ad uno dei processi "urgenti" o in attesa di entrare nel monitor, se ve ne sono.

- `void monitor_notify(natl mon)`(già realizzata): esegue una notifica sulla variabile di condizione associata al monitor di identificatore `mon`. È un errore tentare di eseguire una notifica su una variable di condizione associata ad un monitor che non si possiede.

Modificare i file `sistema.cpp` e `sistema.s` in modo da realizzare le primitive mancanti.