Prova pratica di Calcolatori Elettronici (nucleo v6.*)

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

5 giugno 2010

1. Un `mutex` è un tipo particolare di semaforo che può assumere soltanto due stati: libero oppure occupato. Inoltre, un mutex ricorda l'identità del processo che lo ha occupato. È un errore se lo stesso processo tenta di occupare nuovamente il mutex prima di averlo liberato. Inoltre, è un errore se il mutex viene liberato da un processo diverso da quello che lo aveva occupato.

   Per realizzare un mutex definiamo la seguente struttura (file `sistema.cpp`):

   ```
   struct des_mutex {
           natl owner;
           des_proc* waiting;
   };
   ```

   Se il mutex è occupato, il campo `owner` contiene l'id del processo che lo ha occupato, altrimenti `owner` vale 0. Il campo `waiting` serve a realizzare una lista di processi in attesa di acquisire il `mutex`.

   Le seguenti primitive, accessibili dal livello utente, operano sui `mutex`:

   - `natl mutex_ini()` (già realizzata): inizializza un nuovo mutex, con i campi `owner` e `waiting` entrambi a 0, e ne restituisce l'identificatore. Se non è possibile creare un nuovo mutex restituisce `0xFFFFFFFF`.
   - `void mutex_wait(natl mux)`: tenta di occupare il mutex di identificatore `mux`. Se il mutex è già occupato sospende il processo in attesa che il mutex venga prima liberato. Abortisce il processo in caso di errore.
   - `void mutex_signal(natl mux)`: libera il mutex di identificatore `mux`. Se qualche altro processo era in attesa, lo risveglia e gli cede il mutex (che in questo caso resta occupato). Gestisce una eventuale *preemption*. Abortisce il processo in caso di errore.

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