Prova pratica di Calcolatori Elettronici (nucleo v6.*)

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

12 giugno 2013

1. Un mutex con *priority inheritance* (*pim*) è un semaforo di mutua esclusione che tiene conto delle priorità dei processi per evitare il fenomeno dell'*inversione di priorità*: se un processo ad altra priorità P_1 è bloccato in attesa di acquisire la mutua esclusione su una risorsa posseduta da un processo a bassa priorità P_2, non vogliamo che processi a priorità intermedia tra P_1 e P_2 interrompano P_2, perché questo allungherebbe ingiustamente il tempo di attesa di P_1. I pim risolvono questo problema facendo in modo che il processo che possiede la mutua esclusione innalzi la propria priorità, ponendola uguale alla maggiore tra le priorità dei processi in attesa sulla stessa risorsa e la sua. L'innalzamento di priorità è temporaneo: quando il processo libera la risorsa, ritorna alla sua priorità originaria.

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

```
struct des_pim {
       natl orig_prio;
       des_proc *owner;
       des_proc *waiting;
};
```

   Il campo `owner` punta al processo che possiede la mutua esclusione (0 se la risorsa è libera). Il campo `orig_prio` memorizza la priorità originaria del processo che possiede la mutua esclusione. Il campo `waiting` è una lista di processi in attesa di acquisire la mutua esclusione.

   Le seguenti primitive, accessibili dal livello utente, operano sui pim. In caso di errore abortiscono il processo.

   - `natl pim_init()` (già realizzata): inizializza un nuovo pim e ne restituisce l'identificatore. Se non è possibile creare un nuovo pim restituisce `0xFFFFFFFF`.
   - `void pim_wait(natl pim)` (da realizzare): tenta di acquisire la mutua esclusione sul pim di identificatore `pim`.
   - `void pim_signal(natl pim)` (da realizzare): rilascia la mutua esclusione sul pim di identificatore `pim`.

   Nota: Si assuma che i processi non acquisiscano mai più di una mutua esclusione per volta.

   Attenzione: poichè i pim cambiano dinamicamente la priorità di processi mentre questi possono trovarsi già in qualche coda, succede che le code dei processi non sono più ordinate per priorità. È dunque necessario modificare la funzione `rimozione_lista()` in modo che non si limiti ad estrarre dalla testa, ma cerchi il processo a maggiore priorità tra tutti quelli in lista.

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