Prova pratica di Calcolatori Elettronici (nucleo v6.*)

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

13 gennaio 2014

1. Un modo per evitare il problema del *blocco critico* nell'utilizzo dei semafori (di mutua esclusione) è di fare in modo che ogni processo acquisisca in una sola operazione indivisibile tutti i semafori di cui ha bisogno. Se qualche semaforo non può essere acquisito, allora non deve esserne acquisito nessuno: il processo si deve bloccare fino a quando tutti diventano disponibili.

   Per supportare questo meccanismo, anche se in forma limitata, aggiungiamo al nucleo la primitiva

   ```
   void sem_multiwait(int sem1, int sem2)
   ```

   che acquisisce i semafori `sem1` e `sem2` in modo indivisibile: se `sem1` e `sem2` possono essere acquisiti senza bloccarsi, allora vengono acquisiti entrambi. Altrimenti i contatori non vengono modificati e il processo viene sospeso in attesa che entrambi diventino acquisibili.

   Si noti che, se sia `sem1` che `sem2` non sono acquisibili, dovremmo inserire il processo che ha invocato la primitiva in due code. Poichè non possiamo farlo, adottiamo la seguente tecnica. Blocchiamo il processo in ogni caso su uno solo dei due semafori (quello non acquisibile se ve ne è uno solo, oppure uno qualunque se entrambi non sono acquisibili) e aggiungiamo il campo

   ```
   des_sem *other_sem
   ```

   al descrittore di processo. Utilizziamo questo campo per memorizzare il puntatore all'altro semaforo (quello su cui non stiamo bloccando il processo). Modifichiamo quindi la primtiva `sem_signal` in modo che, quando deve svegliare un processo, controlli il valore di questo campo nel suo descrittore. Se non è nullo, la `sem_signal` si deve preoccupare di completare le operazioni della `sem_multiwait` per conto del processo svegliato. Questo può anche comportare che il processo debba essere sospeso nuovamente, se accade che `other_sem` non è acquisibile. Si noti che in questo caso il processo dovrà essere sospeso su `other_sem`, in quanto il semaforo corrente è diventato acquisibile per effetto della `sem_signal`.

   Modificare i file `sistema.cpp` e `sistema.s` completando le parti mancanti.

   Gestire correttamente eventuali *preemption*.