# Prova pratica di Calcolatori Elettronici (nucleo v6.*)

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

30 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 aggiungiamo al nucleo le seguenti primitive

   ```
   void sem_add(natl sem)
   void sem_del(natl sem)
   void sem_multiwait()
   ```

   Con le prime due primitive, un processo può costruire una lista di semafori che vuole acquisire contemporaneamente. La primitiva `sem_multiwait()` provvede poi ad aquisire in modo indivisibile tutti i semafori che si trovano nella lista al momento della sua invocazioner: se tutti i semafori nella lista possono essere acquisiti senza bloccarsi, allora vengono acquisiti tutti. Altrimenti i contatori non vengono modificati e il processo viene sospeso in attesa che tutti diventino acquisibili.

   Si noti che, nel caso in cui non tutti i semafori siano acquisibili, dovremo comunque inserire il processo nella coda di uno solo di essi.

   Al fine di realizzare questo meccanismo, aggiungiamo i seguenti campi al descrittore di processo:

   ```
   des_sem *mysem[MAX_MULTISEM];
   natl nextsem;
   bool multiwait;
   ```

   I semafori della lista richiesta dal processo si troveranno nelle prime `nextsem` posizioni dell'array `mysem`. Il campo `multiwait` ci permette di sapere se il processo si è bloccato per via di una `sem_multiwait()` (`multiwait==true`) o di una comune `sem_wait()` (`multiwati==false`). Modifichiamo quindi la primtiva `sem_signal` in modo che, quando deve svegliare un processo, controlli il valore di questo campo nel suo descrittore. Se è `true`, 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 non tutti i semafori della lista siano acquisibili. (se `multiwait==false`, la `sem_signal()` deve comportarsi normalmente).

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

   Gestire correttamente eventuali *preemption*.