Prova pratica di Calcolatori Elettronici (nucleo v6.*)

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

7 giugno 2012

1. Vogliamo aggiungere al nucleo un meccanismo di scambio di messaggi. In questo meccanismo un processo può inviare un messaggio `msg` ad un altro processo di cui conosce l'identificatore `id`, usando una primitiva `send(id, msg)`. Assumiamo che il messaggio sia un `natl`. Più processi possono inviare messaggi contemporaneamente ad uno stesso processo. Un processo può mettersi in ascolto di messaggi a lui indirizzati usando una primitiva `natl receive()`. La primitiva blocca il processo che la invoca fino a quando qualche altro processo non gli invia un messaggio; quindi la primitiva restituisce uno dei messaggi ricevuti. Supporremo inoltre che anche la primitiva `send` blocchi il processo che invia fino a quando il processo destinatario non è pronto a ricevere messaggio inviato.

   Puó accadere che il processo destinatario non esista, oppure termini prima di ricevere il messaggio. In questi casi la primitiva `send` deve restituire un errore.

   Per realizzare il precedente meccanismo aggiungiamo i seguenti campi al descrittore di processo:

   ```
   des_proc* senders;
   bool waiting;
   natl msg;
   ```

   Consideriamo il descrittore di processo di un dato processo P. Il campo `senders` è una coda di processi (ordinata per priorità) su cui si sospendono (se necessario) i processi che vogliono inviare un messaggio al processo P. Il campo `waiting` vale `true` quando il processo P stesso è sospeso in attesa della ricezione di un messaggio. Il campo `msg` è utilizzato quando P vuole inviare un messaggio ad un altro processo, ma si deve sospendere. Il suo scopo è di memorizzare il messaggio che P vuole inviare.

   Aggiungiamo dunque le seguenti primitive:

   - `bool send(natl id, natl msg)` (da realizzare): Invia il messaggio `msg` al processo di identificatore `id`. Ritorna quando il messaggio è stato ricevuto, restituendo `true`, oppure se non è stato possibile recapitare il messaggio, restituendo `false`.
   - `natl receive()` (da realizzare): Si pone in attesa di un messaggio. Ritorna quando un messaggio è stato ricevuto e lo restituisce.

   Modificare i file `sistema.cpp` e `sistema.s` in modo da realizzare le primitive e il codice mancante.