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

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

27 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 il primo dei messaggi ricevuti.

   Ogni processo possiede una coda di `MAX_MSG` messaggi pendenti. La primitiva `send` si limita ad accodare un nuovo messaggio nella coda del processo destinatario e ritorna senza attendere che il destinatario prelevi il messaggio. Se la coda è piena il messaggio viene scartato e la primitiva restituisce un errore.

   Può 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:

   ```
   bool waiting;
   natl msg[MAX_MSG];
   natl first_free;
   natl first_unread;
   natl n_msg;
   ```

   Consideriamo il descrittore di un processo $P$. Il campo `waiting` vale `true` se il processo $P$ stesso è sospeso in attesa di un messaggio. I campi `msg`, `first_free`, `first_unread` e `n_msg` servono a realizzare una coda circolare di messaggi in attesa di essere ricevuti.

   Aggiungiamo infine le seguenti primitive:

   - `natl send(natl id, natl msg)` (da realizzare): . Invia il messaggio `msg` al processo di identificatore `id`. Restituisce 0 se il messaggio è stato correttamente accodato, 1 il processo non esiste e 2 se la coda era piena.
   - `natl receive()` (da realizzare): Si pone in attesa di un messaggio. Ritorna quando almeno un messaggio è stato ricevuto restituisce quello accodato da più tempo.

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

   **N.B.** Gestire correttamente eventuali *preemption*.