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

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

21 luglio 2021

1. Definiamo una "msg-area" come una zona della memoria fisica accessibile tramite indirizzi della parte utente/privata di un processo. Ogni msg-area, in ogni istante, è accessibile da un solo processo, e ciascun processo può accedere al più a `MAX_MEM_AREA` msg-area per volta (dunque anche nessuna). Tutte le msg-area di un processo sono accessibili nelle prime `MSG_AREA_PAGES` pagine della parte utente/privata della memoria virtuale del processo.

   Se un processo può accedere ad una msg-area $m$ diciamo che è l'*owner* (temporaneo) di $m$. Un processo può creare una nuova msg-area, diventandone contestualmente l'owner, tramite la primitiva `macreate(natl npag)`, che ne richiede la dimensione in pagine e ne restituisce un puntatore alla base. Un processo $P_1$ che è owner di una msg-area $m$ può *spedirla* a un processo diverso $P_2$, purché questo abbia lo spazio per accoglierla. Dopo la spedizione, $P_2$ diventa il nuovo owner di $m$ e vi può accedere liberamente in lettura e scrittura, mentre $P_1$ non è più l'owner e non può più accedervi.

   Una msg-area è descritta dalla seguente struttura:

   ```
   struct des_ma {
       void* base;
       natq  npag;
   };
   ```

   dove `base` punta alla base della msg-area (nella memoria virtuale del processo owner) e `npag` è la dimensione della msg-area in pagine. Un `des_ma` in cui `base` è un `nullptr` e `npag` è zero è detto *non valido*.

   Per realizzare il meccanismo precedente aggiungiamo almeno i seguenti campi ai descrittori di processo:

   ```
   des_ma ma[MAX_MEM_AREA];
   vaddr next_ma;
   ```

   I descrittori validi dell'array `ma` sono relativi alle msg-area di cui il processo è owner. Il campo `next_ma` contiene l'indirizzo della base della prossima msg-area creata o ricevuta dal processo.

   Introduciamo inoltre le seguenti primitive (abortiscono il processo in caso di errore):

   - `void* macreate(natl npag)` (già realizzata): crea una nuova msg-area, grande `npag` pagine, e restitutisce un puntatore alla base, o `nullptr` se non è stato possibile crearla. È un errore se `npag` è zero.
   - `bool masend(void *m, natl pid)` (da realizzare): attende, se necessario, che il processo `pid` invochi `marecv()`, quindi gli trasferisce la msg-area di indirizzo base `m`. Restituisce `false` se il processo `pid` non esiste, oppure se non è stato possibile eseguire il trasferimento, per esempio per esaurimento della memoria. Nei casi in cui restituisce `false` la primitiva deve lasciare la situazione corrente inalterata. In particolare, se il processo destinatario era bloccato nella `marecv()`, non deve essere sbloccato. È un errore se `m` non è l'indirizzo base di nessuna delle msg-area del processo che invoca la primitiva, oppure se un processo tenta di inviare la msg-area a se stesso o a un processo di livello sistema.

- `des_ma marecv()` (da realizzare): blocca, se necessario, il processo in attesa che un altro processo trasferisca con sucesso una msg-area, quindi restituisce il descrittore della msg-area ricevuta. Restituisce un descrittore non valido se non è possibile ricevere nuove msg-area. Gli indirizzi virtuali che rendono la msg-area accessible non devono sovrapporsi con gli indirizzi appartenuti a tutte le msg-area di cui il processo è stato, o è, owner.

Modificare il file `sistema.cpp` in modo da realizzare le primitive mancanti.

**SUGGERIMENTO**: per rispettare il vincolo sugli indirizzi delle msg-area ricevute è sufficiente avanzare sempre `next_ma`. Quando `next_ma` esce dalla regione destinata alle msg-area, il processo non potrà più creare o ricevere altre msg-area.