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

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

23 luglio 2015

1. Prevediamo che un processo possa creare delle zone di memoria, dette `shmem`, ciascuna con un identificatore unico. I processi che vogliono accedere ad una `shmem` devono aggiungerla al proprio spazio di indirizzamento, specificandone l'identificatore. Una volta aggiunta, la `shmem` sarà disponibile contiguamente all'interno della parte utente/condivisa dello spazio di indirizzamento del processo. Un processo può aggiungere più `shmem` al proprio spazio e le diverse `shmem` non devono sovrapporsi. Non è importante che i processi che condividono una stessa `shmem` la vedano tutti allo stesso indirizzo. In qualunque momento, un processo può eliminare dal proprio spazio di indirizzamento una `shmem` precedentemente aggiunta.

   Per descrivere una `shmem` aggiungiamo al nucleo la seguente struttura dati:

   ```
   struct des_shmem {
       natl npag;
       des_frame *first_frame;
   };
   ```

   Il campo `npag` contiene la dimensione (in pagine) della `shmem`. Tutte i frame che contengono la `shmem`, nell'ordine in cui devono comparire nella memoria di tutti i processi che la condividono, sono mantenuti in una lista la cui testa è puntata dal campo `first_frame`. Ogni frame punta alla successivo tramite un nuovo campo `des_frame *next_shmem` che abbiamo aggiunto ai descrittori di frame.

   Inoltre, aggiungiamo i seguenti campi ai descrittori di processo:

   ```
   addr avail_addr;
   des_attached *att;
   ```

   Il campo `avail_addr` contiene il primo indirizzo libero nella parte utente/condivisa del processo. Tutti gli indirizzi da `avail_addr` fino a `fin_utn_c` (escluso) sono disponibili per contenere zone di memoria condivisa. Il campo `att` è la testa di una lista di elementi di tipo `des_attached`, il cui scopo è di tener traccia di tutte le `shmem` a cui il processo è collegato. Il tipo `des_attached` è così definito:

   ```
   struct des_attached {
       natl id;
       addr start;
       des_attached *next;
   }
   ```

   Il campo `id` è l'identificatore di una shmem `shmem`; il campo `start` è l'indirizzo virtuale (nello spazio di indirizzamento del processo) a partire dal quale questa `shmem` è visibile. Il campo `next` punta al prossimo elemento della lista.

   Aggiungiamo infine le seguenti primitive:

- `natl shmem_create(natl npag)` (tipo 0x5c, già realizzata): Crea una nuova zona di memoria condivisibile tra più processi, grande `npag` pagine, e ne restituisce l'identificatore.
- `addr shmem_attach(natl id)` (tipo 0x5d, già realizzata): Permette ad un processo di aggiungere la `shmem id` al proprio spazio di indirizzamento e ne restituisce l'indirizzo di partenza.
- `void shmem_detach(natl id)` (tipo 0x5e, da realizzare): Permette ad un processo di eliminare la `shmem id` dal proprio spazio di indirizzamento. Abortisce il processo se la `shmem id` non è tra quelle a cui il processo è collegato.

Modificare i file `sistema.cpp` e `sistema.S` in modo da realizzare le primitive appena descritte.

**ATTENZIONE**:

- nella `shmem_detach()` è necessario deallocare tutto ciò che non serve più, incluse eventuali tabelle delle pagine ormai vuote. Per sapere velocemente se una tabella delle pagine è vuota aggiungiamo un campo `num_present` ai descrittori di frame. Se il corrispondente frame contiene una tabella questo campo deve contare il numero di entrate con P=1 che questa contiene.
- le pagine e le tabelle delle `shmem` non hanno blocchi in memoria di massa. Assumere che il loro indirizzo in memoria di massa sia 0.
- nella `shmem_detach()`, tralasciare la gestione di `avail_addr`.