

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

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

15 settembre 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 contigualmente 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.

Infine, un processo può distruggere una **shmem**. Se altri processi hanno la **shmem** nel loro spazio, l'operazione di distruzione deve prima attendere che tutti la rimuovano. Durante questo periodo di attesa la **shmem** non può essere aggiunta allo spazio di ulteriori processi. È anche possibile che, durante questo periodo, altri processi tentino di distruggere la stessa **shmem**: tutti questi processi attenderanno che la **shmem** venga liberata e distrutta.

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

```
struct des_shmem {
    natl npag;
    natl nusers;
    des_frame *first_frame;
    proc_elem *wait_detach;
};
```

Il campo **npag** contiene la dimensione (in pagine) della **shmem**. Il campo **nusers** conta i processi che hanno la **shmem** nel loro spazio di indirizzamento. Il campo **wait\_detach** è la coda dei processi in attesa per la distruzione della **shmem**. Tutti 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 al 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.

Aggiungiamo infine le seguenti primitive:

- `natl shmем_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. È un errore se `npag` è zero.
- `addr shmем_attach(natl id)` (tipo 0x5d, già realizzata): Permette ad un processo di aggiungere la `shmем id` al proprio spazio di indirizzamento e ne restituisce l'indirizzo di partenza.
- `void shmем_detach(natl id)` (tipo 0x5e, da realizzare): Permette ad un processo di eliminare la `shmем id` dal proprio spazio di indirizzamento. Abortisce il processo se la `shmем id` non è tra quelle a cui il processo è collegato.
- `void shmем_destroy(natl id)` (tipo 0x5f, da realizzare): Permette ad un processo di distruggere la `shmем` di identificatore `id`. È un errore se la `shmем` non esiste o se vi è attaccato il processo stesso che ha invocato la primitiva.

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

**SUGGERIMENTO:** molte funzioni di supporto si trovano già realizzate nel file `sistema.cpp` nella sezione marcata come `ESAME`. Consultare i commenti nel file per capire come utilizzarle.