Prova pratica di Calcolatori Elettronici (nucleo v6.*)

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

15 settembre 2021

1. Definiamo una "msg-area" come una zona della memoria fisica composta da un certo numero di frame. I processi possono accedere alle msg-area tramite delle *view*, cioè intervalli di indirizzi della parte utente/privata della loro memoria virtuale. Se un processo p possiede una view con indirizzo base `vv` per una msg-area `ma`, diciamo che `ma` è visibile a partire da `vv` in p. In ogni processo, le view occupano sempre intervalli di indirizzi non sovrapposti. Un processo può creare una nuova msg-area, ottenendone contestualmente una view. Tutti i processi che hanno almeno una view su una data msg-area sono detti *owner* della msg-area. Tutti gli owner di una msg-area possono *condividerla* con altri processi, che acquisicono così una propria view e diventano anch'essi owner della msg-area (se non lo erano già). Le varie view di una stessa msg-area possono partire da indirizzi virtuali diversi, e uno stesso processo può avere più view della stessa msg-area. Un processo che sia owner di una msg-area può anche eliminare tutte le view di un processo (incluso se stesso) tramite una operazione di *revoca*. Una msg-area viene deallocata (i suoi frame rientrano nella lista dei frame liberi) quando non esistono più view corrispondenti. Quando un processo termina vengono automaticamente distrutte tutte le sue view.

   Una msg-area è descritta dalla seguente struttura:

   ```
   struct des_ma {
       natq  npag;
       natq  views;
   };
   ```

   dove `npag` è la dimensione della msg-area in pagine e `views` è il numero di view esistenti per questa msg-area. Un `des_ma` in cui `npag` è zero è detto *non valido*.

   Una view è descritta dalla seguente struttura:

   ```
   struct des_view {
       des_ma* ma;
       vaddr base;
   };
   ```

   Dove `ma` punta alla descrittore della corrispondente msg-area e `base` è l'indirizzo virtuale della view nello spazio di indirizzamento del processo a cui la view appartiene. Una view in cui `ma` è `nullptr` è detta *non valida*.

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

   ```
   des_view view[MAX_MSG_AREA_VIEW];
   vaddr next_view;
   ```

   I descrittori validi dell'array `view` sono relativi alle view del processo. Il campo `next_view` contiene l'indirizzo della base della prossima view 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 una corrispondente view nel processo corrente; restituisce un puntatore alla base della view, o `nullptr` se non è stato possibile creare la msg-area o la view. È un errore se `npag` è zero.
- `void* mashare(void *vv, natl pid)` (da realizzare): crea nel processo `pid` una nuova view per la msg-area che è visibile, nel processo corrente, a partire dall'indirizzo `vv` (`pid` può anche coincidere con il processo corrente). Restituisce l'indirizzo (valido nello spazio di indirizzamento del processo `pid`) della nuova view, oppure `nullptr` se non è stato possibile completare l'operazione (perché la view non esiste, o il processo destinatario non esiste o non può creare altre view). È un errore tentare di condividere una msg-area con un processo di livello sistema.
- `bool marevoke(void *vv, natl pid)` (da realizzare): elimina dal processo `pid` (che può essere anche il processo corrente) tutte le view corrispondenti alla msg-area visibile a partire dall'indirizzo `vv` nel processo corrente. Il processo `pid` non può dunque più accedere alla msg-area (a meno che qualche owner non la condivida nuovamente con lui). Restituisce `true` in caso di successo (anche nel caso particolare in cui `pid` non aveva alcuna view della msg-area) e `false` se l'operazione non è possibile (perché il processo corrente non ha una view che parte da `vv`, oppure perché il processo destinatario non esiste).

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

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