Prova pratica di Calcolatori Elettronici (nucleo v6.*)

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

11 gennaio 2022

1. Definiamo una "memory-area" come una zona della memoria privata di un processo, accessibile in lettura/scrittura a partire dal primo indirizzo della parte utente/privata. I processi non hanno inizialmente una memory-area, ma possono crearne una tramite la primitiva `macreate(dim)`, dove `dim` è la dimensione in pagine.

   Un processo sorgente può inviare una *copia* della propria memory-area ad un altro processo destinario, con identificatore `pid`, usando la primitiva `macopy(pid)`, purché il destinatario non ne possegga già un'altra. Più processi possono ricevere una copia della stessa memory-area, sia direttamente dal processo che l'aveva creata, sia indirettamente da un processo che l'aveva a sua volta ricevuta.

   Per ottimizzare la copia utilizziamo il meccanismo del *copy-on-write*: la `macopy()` fa inizialmente accedere entrambi i processi (sorgente e destinario) agli stessi frame della memoria fisica, ma in sola lettura. La copia verrà eseguita solo quando, successivamente, uno dei processi tenta di scrivervi. Questo permette di evitare le copie se i processi si limitano a leggere.

   Visto che la stessa memory-area può essere copiata più volte, ogni frame di una memory-area può essere condiviso tra più processi. Aggiungiamo dunque ai descrittori dei frame (`des_frame`) un campo `nma` destinato a contare il numero di processi che condividono quel frame. Quando la memory-area viene creata `nma` viene posto a 1 per tutti i frame che la compongono; quando la memory-area viene copiata tutti i suoi `nma` vengono incrementati.

   Come detto, ciascun frame verrà effettivamente copiato solo quando, e se, uno dei processi che lo condividono tenterà di accedervi in scrittura. Più precisamente, consideriamo un frame f condiviso tra i processi P_1, P_2, ..., P_n (quindi con `nma` $= n$). Se un qualsiasi processo P_i, con $1 \leq i \leq n$, tenta di accedere in scrittura a un indirizzo v mappato su f, la MMU causerà un page fault. La routine di gestione, riconosciuta la causa del page fault, allocherà un nuovo frame f' e vi copierà il contenuto di f, decrementandone `nma` e cambiando la traduzione di v in modo che ora punti a f' e la scrittura sia abililitata. A questo punto il processo P_i potrà ripetere l'accesso.

   Per realizzare il meccanismo appena descritto aggiungiamo il campo `masize` ai descrittori di processo. Il campo contiene la dimensione in pagine della memory-area del processo; vale 0 se il processo non ha ancora creato o ricevuto una memory-area.

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

   - `void* macreate(natq size)` (realizzata in parte): crea una memory-area di `size` pagine. È un errore se `size` è zero o maggiore di `MAX_MA_PAGES`, oppure se il processo possiede già una memory-area. Restituisce l'indirizzo della memory-area, o `nullptr` se non è stato possibile crearla.
   - `bool macopy(natq pid)` (realizzata in parte): invia al processo `pid` una copia della memory-area del processo chiamante, usando il meccanismo del copy-on-write. È un errore se il processo chiamante non possiede una memory-area o se `pid` supera `MAX_PROC_ID`. Restituisce `false` se il processo destinatario non esiste o possiede già una memory area, oppure se non è stato possibile completare l'operazione (memoria esaurita); restituisce `true` altrimenti.

Modificare il file `sistema.cpp` in modo da specificare la parti mancanti. Attenzione a deallocare le memory-area correttamente quando un processo termina.