Prova pratica di Calcolatori Elettronici (nucleo v6.*)

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

23 settembre 2020

1. Vogliamo permettere ai processi di modificare il permesso di scrittura per alcuni indirizzi del proprio spazio di indirizzamento. Per farlo definiamo una nuova primitiva (tipo 0x5a):

   ```
   bool mprotect(vaddr v, natq n, bool w);
   ```

   La primitiva riceve un undirizzo virtuale di partenza, `v`, un numero di byte `n` e un flag `w`. La primitiva deve impostare i bit R/W nel percorso di traduzione di tutti gli indirizzi dell'intervallo $[\texttt{v}, \texttt{v}+\texttt{n})$ in modo che le scritture da parte del processo siano permesse (se `w` vale `true`) o vietate (se `w` vale `false`).

   Per semplicità, la primitiva permette di eseguire questa operazione solo per intervalli che cadono interamente all'interno della parte utente condivisa, che va da `ini_unt_c`, incluso, a `fin_utn_c`, escluso. Se l'intervallo richiesto non è interamente contenuto tra questi indirizzi, il processo che ha invocato la primitiva deve essere abortito. Si ricordi che, per come abbiamo realizzato il sistema, questa parte della memoria virtuale di ogni processo contiene solo pagine residenti (e dunque anche le relative tabelle sono residenti). Inoltre, tutte le tabelle dal livello 3 a scendere sono condivise tra tutti i processi.

   Non ci sono altri vincoli su `v` e `n`. In particolare, sia `v` che `v+n` potrebbero non essere allineati alla pagina. Resta però inteso che la primitiva aggiusterà i permessi del minimo insieme di pagine che contengono l'intervallo richiesto.

   Attenzione: la primitiva deve cambiare i permessi solo per il processo che la invoca, anche se gli indirizzi si trovano nella parte condivisa tra tutti i processi. Questo vuol dire che la primitiva deve prima creare un percorso di traduzione privato per il processo (ma solo il percorso di traduzione deve diventare privato: le pagine vere e proprie devono restare condivise).

   La primitiva potrebbe fallire perché deve duplicare delle tabelle e non ci sono più frame liberi. In tal caso restituisce `false`. Altrimenti restituisce `true`.

   Poiché la memoria è limitata, è necessario non duplicare tabelle più del necessario. Inoltre, alla terminazione del processo, è anche necessario rendere di nuovo liberi i frame eventualmente usati per contenere le tabelle duplicate.

   Modificare i file `sistema.s` e `sistema.cpp` in modo da realizzare la primitiva `mprotect()`, scrivendo esclusiamente negli spazi contrassegnati con `SOLUZIONE` (non è necessario però usarli tutti).

   Il file `utente/prog/pmprotect.in` contiene i test, con dei commenti che spiegano cosa ci aspetta da ciascuno di essi.

   Suggerimento 1: per duplicare un percorso di traduzione ci si può ispirare alla funzione `duplica()` dell'appello del 23 Settembre 2019.

   Suggerimento 2: non preoccuparsi di ottimizzare troppo il tempo di esecuzione: è accettabile visitare il percorso di traduzione daccapo, anche più volte, se questo semplifica la soluzione. In particolare, nel caso in cui `mprotect()` debba lavorare su più pagine, è del tutto accettabile visitare il percorso di traduzione daccapo per ogni pagina, anche se questo comporta di visitare più volte le stesse tabelle di livello superiore.