Prova pratica di Calcolatori Elettronici (nucleo v6.*)

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

17 gennaio 2018

1. Vogliamo aggiungere al nucleo un meccanismo tramite il quale un processo può terminare forzatamente l'esecuzione di un altro processo. Aggiungiamo però la seguente limitazione: un processo può terminare solo sé stesso, oppure uno dei suoi *discendenti* (cioè i processi creati da esso stesso, oppure creati da questi, e così via). È un errore se un processo tenta di terminare un altro processo che non appartiene alla sua discendenza.

   Per realizzare il meccanismo aggiungiamo la seguente primitiva:

   ```
   bool kill(natl id);
   ```

   La primitiva deve terminare forzatamente il processo di identificatore `id`, qualunque cosa esso stesse facendo (quindi anche se era bloccato su un semaforo o sospeso sulla coda del timer). Se non esiste alcun processo di identificatore `id`, la primitiva restituisce `false`, altrimenti `true`. Abortisce il processo in caso di errore.

   Per tenere traccia della discendenza aggiungiamo il seguente campo al descrittore di processo:

   ```
   struct des_proc *parent;
   ```

   Il campo deve puntare al più giovane antenato vivente del processo. Alla creazione, punta al processo padre (quello che ha invocato la `activate_p()`); se il processo padre termina prima del figlio, il campo `parent` dovrà essere opportunamente aggiornato.

   Modificare il file `sistema.cpp` per aggiungere le parti mancanti nella realizzazione di questo meccanismo.

   ATTENZIONE: quando si fa terminare forzatamente un processo, tutte le sue risorse devono essere rilasciate come il processo se avesse chiamato `terminate_p()`. Fare anche attenzione a lasciare sempre semafori in uno stato consistente.