Prova pratica di Calcolatori Elettronici (nucleo v6.*)

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

30 gennaio 2013

1. Vogliamo modificare il nucleo in modo che un processo, alla fine della propria esecuzione, possa comunicare un risultato al processo che lo ha creato. Assumiamo che tale dato sia di tipo `natl`. A tal fine modifichiamo la primitiva `void terminate_p()` in `void terminate_p(natl result)` e aggiungiamo una primitiva `natl join()`. Un processo figlio passa il risultato da comunicare al processo padre come parametro della `terminate_p()`. La nuova primitiva `join()` deve essere usata da un processo padre quando vuole recuperare il risultato di uno dei suoi figli terminati. Se nessun figlio è ancora terminato, la primitiva sospende il processo padre in attesa che almeno un figlio termini. (Se il processo non ha figli viene abortito).

   Per realizzare tale meccanismo introduciamo i seguenti tipi:

   ```
   enum status_t { RUNNING, TERMINATED };
   struct child {
       status_t status;
       natl result;
       natl father;
       child* next;
   };
   ```

   La struttura `child` descrive lo stato di un processo figlio. Il campo `status` ci dice se il processo è già terminato oppure no. Il campo `result` contiene il risultato del processo, ed è significativo solo se il processo è già terminato. Il campo `father` contiene l'id del processo padre (se non ancora terminato, altrimenti contiene `0xFFFFFFFF`) e il campo `child` serve a costruire una lista dei processi figli di uno stesso processo.

   Aggiungiamo tre campi al descrittore di processo:

   ```
   child* me;
   child* children;
   bool waiting;
   ```

   Il campo `children` punta al primo elemento della lista (eventualmente vuota) dei figli di questo processo. Questo processo è a sua volta un figlio, e il campo `me` punta al proprio descrittore `child` nella lista dei figli del padre. Tale struttura viene allocata, inizializzata e inserita in lista alla creazione del processo figlio. Quando un processo vuole attendere che uno dei suoi figli termini pone a `true` il campo `waiting`. **Attenzione:** per motivi tecnici alcuni processi non hanno un padre (l'id del padre è 0xFFFFFFFF). In quel caso la struttura `child` non deve essere inserita in lista.

   La struttura `child` di un dato processo figlio deve essere deallocata quando il padre ha letto il risultato di quel processo. Per assicurarsi che le strutture vengano deallocate in tutti casi, adottiamo le seguenti regole seguite da ogni processo alla propria terminazione:

   1. il padre dealloca tutte le strutture `child` dei processi figli terminati e pone a `0xFFFFFFFF` il campo `father` delle altre. `father`;

2. il figlio dealloca la propria struttura `child` se il padre non esiste oppure è già terminato.

Modificare i file `sistema.cpp` e `sistema.s` in modo da realizzare le primitive e il codice mancante.