

# Prova pratica di Calcolatori Elettronici (nucleo v6.\*)

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

7 febbraio 2020

1. Vogliamo fornire ai processi la possibilità di bloccarsi in attesa che un altro processo riceva una eccezione o termini. Un processo  $P$  deve prima registrarsi, tramite la primitiva `proc_attach(natl id)`, con il processo di identificatore `id`, chiamiamolo  $Q$ , di cui vuole controllare la terminazione. Da questo momento in poi, se  $Q$  riceve una eccezione o invoca `terminate_p()`, deve essere messo in *pausa*. Diremo che  $P$  è il *master* di  $Q$  e che  $Q$  è lo *slave* di  $P$ . Un processo master può registrarsi con un numero qualunque di slave. Una volta registratosi, il processo master può invocare la primitiva `proc_wait()` per bloccarsi in attesa che almeno uno dei suoi slave vada in pausa (l'attesa può essere nulla se qualche slave era già andato in pausa nel frattempo). La primitiva `proc_wait()` restituisce al processo  $P$  il numero dell'eccezione ricevuta da uno dei suoi slave in pausa, o il valore 32 se lo slave aveva invocato `terminate_p()`. In caso di più slave in pausa, la primitiva restituisce il valore relativo allo slave con priorità maggiore. A questo punto lo slave in questione termina la pausa e completa la gestione dell'eccezione o della `terminate_p()` (in entrambi i casi viene di fatto distrutto). Una successiva invocazione della `proc_wait()` restituirà il valore relativo al successivo slave in pausa, in ordine di priorità, e così via fino all'esaurimento della coda.

Per realizzare questo meccanismo aggiungiamo i seguenti campi al descrittore di processo:

```
des_proc *slaves;
bool is_waiting;
des_proc *paused_slaves;

des_proc *master;
des_proc *next_slave;
natl last_exception;
```

I primi tre campi sono relativi ai master, con il seguente significato: `slaves` è una lista di tutti gli slave del master; `is_waiting` vale `true` se il master è in attesa nella `proc_wait()`; `paused_slaves` è una coda che contiene tutti gli slave attualmente in pausa. I secondi tre campi sono relativi agli slave, con il seguente significato: `master` punta al master dello slave; `next_slave` è usato per creare la lista di tutti gli slave dello stesso master (lista la cui testa è il puntatore `slaves` nel master); `last_exception` contiene il numero dell'ultima eccezione ricevuta dallo slave (o 32 se lo slave aveva invocato `terminate_p()`).

Si modifichino i file `sistema/sistema.s` e `sistema/sistema.cpp` per implementare il meccanismo e le seguenti primitive (abortiscono il processo in caso di errore):

- `bool proc_attach(natl id)`: (tipo 0x59, già realizzata) La primitiva restituisce `false` se il processo che la invoca è uno slave, oppure se il processo `id` non esiste oppure è già un master. È un errore se il processo  $P$  è già master o cerca di diventare master di se stesso. Altrimenti fa in modo che  $P$  diventi il master di `id` e restituisce `true`.
- `natl proc_wait()`: (tipo 0x5a, da realizzare): attende che un processo slave vada in pausa per la ricezione di una eccezione (nota: si trascurino i page fault, tipo 14, e le interruzioni non mascherabili, tipo 2) o termini normalmente e restituisce il numero dell'eccezione, o 32 nel caso di terminazione normale. È un errore invocare questa primitiva se il processo non è master;