

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

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

24 luglio 2019

1. Vogliamo fornire ai processi la possibilità di bloccare l'esecuzione di un processo a scelta, quando questo passa da una certa istruzione. Per far questo forniamo alcune primitive. Con la primitiva `bpattach(natl id, vaddr rip)` un processo *master* installa un breakpoint (istruzione `int3`, codice operativo `0xCC`) all'indirizzo `rip` per il processo `id`, che diventa *slave*. Da quel momento in poi il processo *slave* si blocca se passa da `rip`. Nel frattempo, usando la primitiva `bpwait()`, il processo *master* può sospendersi in attesa che lo *slave* passi dal breakpoint. Infine, con la primitiva `bpdetach()`, il processo *master* rimuove il breakpoint e risveglia eventualmente lo *slave*, il quale prosegue la sua esecuzione come se non fosse mai stato intercettato.

Si noti che processi che non sono *slave* non devono essere intercettati. Inoltre, se un processo esegue `int3` senza che ciò sia stato richiesto da un *master*, il processo deve essere abortito.

Aggiungiamo i seguenti campi ai descrittori di processo:

```
des_proc *bp_master;
des_proc *bp_slave;
vaddr bp_addr;
natb bp_orig;
natl bp_slave_id;
bool bp_waiting;
```

dove: `bp_master` punta al processo *master* di questo processo (nullo se il processo non ha un *master*); `bp_slave` punta al processo *slave* di questo processo (nullo se il processo non ha uno *slave*); `bp_addr`, `bp_orig` e `bp_slave_id` sono significativi solo per i processi *master* e contengono, rispettivamente, l'indirizzo a cui è installato breakpoint; il byte originariamente contenuto a quell'indirizzo e l'id dello *slave*; `bp_waiting` vale `true` nel descrittore di uno *slave* se il *master* è bloccato, e viceversa.

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

- `bool bpattach(natl id, vaddr rip)`: (realizzata in parte): se il processo chiamato non è *slave* e il processo `id` esiste e non è già uno *slave* o un *master*, installa il breakpoint all'indirizzo `rip` e restituisce `true`, altrimenti restituisce `false`; è un errore se `rip` non appartiene all'intervallo `[ini_utn_c, fin_utn_c)` (zona utente/condivisa), il processo è già *master* cerca di diventare *master* di se stesso.
- `natl bpwait()`: (già realizzata): attende che il processo *slave* passi dal breakpoint; è un errore invocare questa primitiva se il processo non è *master*;
- `void bpdetach()` (da realizzare): disfa tutto ciò che ha fatto la `bpattach()` e risveglia eventualmente il processo *slave*; è un errore invocare questa primitiva se il processo non è *master*;

**Attenzione:** bisogna fare in modo che solo i processi *slave* vengano intercettati, ma la parte utente/condivisa è appunto condivisa tra tutti i processi.