

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

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

26 gennaio 2022

1. Definiamo una “memory-area” come una zona della memoria privata di un processo  $P_1$  gestita da un secondo processo  $P_2$ , detto *monitor* di  $P_1$ . Un processo  $P_2$  può diventare il monitor di  $P_1$  tramite una primitiva `ma_attach()`, che definisce anche la dimensione (in pagine) della memory-area di  $P_1$ . Da quel momento in poi, tutti i page-fault generati da  $P_1$  su indirizzi che cadono all’interno della sua memory-area verranno intercettati da  $P_2$ . Il processo  $P_2$  può sospendersi in attesa che  $P_1$  causi un page fault tramite una primitiva `ma_wait()`. La primitiva restituisce a  $P_2$  l’indirizzo  $v$  che ha causato il fault in  $P_1$ . Il processo  $P_1$  non viene distrutto e resta in attesa che  $P_2$  risolva il page fault, creando una traduzione per  $v$ . Il processo  $P_2$  può creare una traduzione usando una primitiva `ma_map(src)`, dove `src` deve essere un indirizzo dello spazio utente/condiviso, accessibile in scrittura. La primitiva `ma_map` deve creare il mapping tra la pagina che contiene  $v$  (nello spazio di  $P_1$ ) e il frame che corrisponde a `src`, quindi deve risvegliare  $P_1$ , in modo che questi riesegua l’accesso a  $v$  e prosegua con la sua normale esecuzione.

Casi particolari: se  $P_1$  termina mentre è monitorato da  $P_2$ , la `ma_wait()` deve comunque risvegliare  $P_2$ , ma restituire 0. Se invece è  $P_2$  a terminare mentre sta monitorando  $P_1$ , la memory-area di  $P_1$  resta nello stato in cui  $P_2$  l’ha lasciata, ma eventuali nuovi page-fault si devono comportare normalmente (abortendo  $P_1$ ).

Per realizzare il meccanismo appena descritto aggiungiamo i seguenti campi al descrittore di processo:

```
// significativi per i processi monitor
des_proc *monitored; // processo monitorato
bool pending_event;

// significativi per i processi monitorati
des_proc *monitor;    // processo monitor
vaddr last_cr2;       // indirizzo che ha causato il fault
natq ma_npag;         // dimensione in pagine della memory area

// significativo per entrambi
bool waiting;
```

Il booleano `pending_event` vale `true` se il processo monitorato ha causato un “evento” (ha generato un page fault nella memory-area, oppure è terminato): serve alla primitiva `ma_wait()` per capire se sospendere o meno il processo (monitor) che la invoca. Il significato del booleano `waiting` dipende dal tipo di processo: per i processi monitor indica che il processo è sospeso nella `ma_wait()`; per i processi monitorati indica che il processo ha causato un page-fault nella memory area e sta aspettando una `ma_map()` che lo risvegli.

Aggiungiamo inoltre le seguenti primitive (abortiscono il processo in caso di errore)

- `bool ma_attach(natw pid, natq ma_npag)` (da realizzare): il processo che invoca la primitiva diventa monitor del processo di identificatore `pid`. Il processo `pid` acquisisce una memory-area grande `ma_npag` pagine a partire dall’indirizzo `ini_utn_p`. La primitiva restituisce `false` se il

processo `pid` o il processo chiamante sono già monitorati, oppure se `pid` non esiste. È un errore se `pid` non è un identificatore valido, oppure se `npag` non è compreso tra 1 e `MAX_MA_PAGES` (inclusi), oppure se il processo chiamante è già monitor.

- `vaddr ma_wait()` (da realizzare): Attende che il processo monitorato generi un page fault nella memory area, o termini. Nel primo caso restituisce l'indirizzo che ha causato il fault, nel secondo 0. È un errore se il processo che invoca la primitiva non è un monitor.
- `bool ma_map(vaddr src)` (da realizzare): Crea la traduzione tra la pagina che contiene l'indirizzo che ha causato il fault e il frame che corrisponde a `src`, quindi risveglia il processo monitorato. Restituisce `false` se non è stato possibile creare la traduzione. È un errore se il processo che invoca la primitiva non è un monitor, oppure se il processo monitorato non aveva causato un fault (e dunque non era sospeso in attesa della `ma_map`). È un errore se `src` non appartiene allo spazio utente condiviso o non è accessibile in scrittura.

Modificare il file `sistema.cpp` in modo da specificare la parti mancanti.