Prova pratica di Calcolatori Elettronici

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

7 giugno 2024

1. Siano date le seguenti dichiarazioni, contenute nel file `cc.h`:

```
struct st1 { char vc[4]; };
class cl {
        st1 s; long v[4];
public:
        cl(char c, st1& s2);
        void elab1(st1 s1);
        void stampa() {
                for (int i = 0; i < 4 ;i++) cout << s.vc[i] << ' '; cout << endl;
                for (int i = 0; i < 4; i++) cout << v[i] << ' '; cout << endl << endl; }
};
```

   Realizzare in Assembler GCC le funzioni membro seguenti.

```
void cl::elab1(st1 s1)
{
        cl cla('k', s1);
        for (int i = 0; i < 4; i++) {
                if (s.vc[i] <= s1.vc[i]) {
                        s.vc[i] = cla.s.vc[i]; v[i] = i - cla.v[i];
                }
        }
}
```

2. Aggiungiamo al sistema il meccanismo dei *watch*, tramite il quale un processo può monitorare tutti i cambiamenti del valore di una variabile causati dagli altri processi. Un processo installa un watch specificando l'indirizzo di una variable e la sua dimensione, diventando quindi *watcher* e proprietario del watch. Il watcher si può poi sospendere in attesa che un altro processo modifichi la variabile, e a quel punto si risveglia ricevendo il nuovo valore.

   Per realizzare il meccanismo operiamo nel modo seguente:

   1. il watch può essere installato solo su variabili che si trovano nello spazio utente condiviso;
   2. quando il watch viene installato, il sistema disabilita le scritture sulla pagina che contiene la variabile da osservare;
   3. questo comporta che tutti i processi che tentano di scrivere in quella pagina ricevono una eccezione di page fault;
   4. il sistema intercetta questo page fault, riabilita le scritture sulla pagina, abilita il Single Step per il processo che stava tentando di scrivere e lo rimette in esecuzione: diciamo che il processo è sotto osservazione (*watched*);

5. quando il processo sotto osservazione riceve l'eccezione di debug (subito dopo aver eseguito l' operazione di scrittura nella pagina), il sistema confronta il nuovo valore della variabile con quello precedente e, se differiscono, lo passa al watcher; disabilita nuovamente le scritture nella pagina e disattiva il Single Step sul processo watched (che smette così di essere sotto osservazione).
6. il passaggio del nuovo valore al watcher è sincrono: il processo resta sospeso fino a quando il watcher non ha ricevuto il valore.

Per assicurarci che, nei punti 3 e 4, solo il processo sotto osservazione possa scrivere nella pagina, il sistema lo fa girare con le interruzioni esterne disabilitate (ovviamente solo per il tempo in cui il processo è sotto osservazione). Per semplicità prevediamo che nel sistema si possa installare un solo watch alla volta. Quando il watcher termina il watch corrente viene disattivato, in modo che un altro processo ne possa installare uno nuovo.

Attenzione: il watcher deve osservare i cambiamenti della variabile osservata nell'ordine in cui si sono verificate, indipendentemente dalla priorità dei processi che le hanno causate. Casi particolari: le scritture del watcher stesso vanno permesse, ma non vanno notificate; i processi che causano page fault o eccezioni di debug non correlate con il meccanismo del watch devono essere abortiti normalmente.

Aggiungiamo i seguenti campi al descrittore di processo:

```
    bool being_watched;
    natq new_watch_value;
```

Il campo `being_watched` vale `true` se il processo è sotto osservazione; il campo `new_watch_value` contiene il nuovo valore della variabile scritto dal processo mentre era sotto osservazione.

Aggiungiamo la seguente struttura dati:

```
struct watch_des {
    vaddr v;
    natq size;
    natl watcher_id;
    natq old_value;
    des_proc *watcher_waiting;
    des_proc *watched_waiting;
} watch_state;
```

Il campo `v` contiene l'indirizzo virtuale della variabile osservata (se 0, non ci sono watch installati); il campo `size` contiene la dimensione (in byte) della variabile osservata; il campo `watcher_id` contiene l'id del processo watcher; il campo `old_value` contiene l'ultimo valore scritto nella variable osservata; il campo `watcher_waiting` punta alla testa della lista su cui si sospende il watcher in attesa dei processi osservati; il campo `watched_waiting` punta alla testa della lista su cui si sospendono i processi osservati in attesa di passare il valore al watcher (questa lista non è ordinata per priorità, ma in base all'ordine delle scritture operate dai processi).

Infine aggiungiamo le seguenti primitive (abortiscono il processo in caso di errore):

- `bool setwatch(void *ptr, size_t size)` (già realizzata): installa un watch sulla variabile di indirizzo `ptr` e dimensione `size`; è un errore se la variabile non si trova nello spazio utente condiviso o non è accessibile in lettura, o se non ha una dimensione di 1, 2, 4, o 8, o non è allineata naturalmente; restituisce `false` se c'è già un watch installato, `true` altrimenti;
- `natq watch()`: restituisce il prossimo valore della variabile osservata (eventualmente esteso senza segno a 8 byte); è un errore se non ci sono watch installati o se il processo non è il watcher corrente.

Modificare il file `sistema.cpp` per completare le parti mancanti.