Prova pratica di Calcolatori Elettronici

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

7 giugno 2023

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

```
struct st1 {
        char vc[3];
};
class cl {
        st1 s;
        long v[3];
public:
        cl(char c, st1& s2);
        void elab1(st1& s1);
        void stampa()
        {
                for (int i = 0; i < 3 ;i++) cout << s.vc[i] << ' '; cout << endl;
                for (int i = 0; i < 3; i++) cout << v[i] << ' '; cout << endl << endl;
        }
};
```

   Realizzare in Assembler GCC le funzioni membro seguenti.

```
void cl::elab1(st1& s1)
{
        cl cla('q', s1);
        for (int i = 0; i < 3; i++) {
                if (s.vc[i] < s1.vc[i]) {
                        s.vc[i] = cla.s.vc[i];
                        v[i] = cla.v[i] - i;
                }
        }
}
```

2. Aggiungiamo una nuova zona "utente/cow" alla memoria virtuale dei nuovi processi utente. Il contenuto di questa zona è inizializzato all'avvio del sistema e ogni processo ne possiede una copia privata.

   Invece di copiare l'intera zona ogni volta che creiamo un nuovo processo, adottiamo la tecnica del *copy on write* (abbreviato in *cow*): tutti i processi condividono inizialmente la stessa zona, ma in sola lettura, e copiamo le singole pagine se e solo se un processo tenta di scrivervi.

   Più in dettaglio:

   - all'avvio del sistema creiamo e inizializziamo la zona cow, allocando e inizializzando i frame e creando le opportune traduzioni partendo da una tabella di livello 4 che chiamiamo `cow_root`, avendo cura di vietare le scritture in tutti gli indirizzi della zona;

- alla creazione di ogni processo utente copiamo le entrate opportune di `cow_root` nella tabella radice del nuovo processo;
- se un processo genera un page fault per accesso in scrittura su un indirizzo appartenente alla zona utente/cow, invece di terminare il processo copiamo la corrispondente pagina, abilitiamo l'accesso in scrittura e facciamo ripartire il processo.

Attenzione: l'accesso in scrittura deve essere abilitato solo per il processo che ha generato il fault, e solo sulla pagina che contiene l'indirizzo cha ha causato il fault. Inoltre, se l'accesso *non* appartiene alla zona utente/cow del processo, il processo deve essere abortito come al solito.

Per realizzare il meccanismo appena descritto sono state definite le nuove costanti `ini_utn_w` e `fin_utn_w`, che delimitano gli indirizzi riservati alla nuova zona, e la costante `DIM_USR_COW` che ne specifica la dimensione effettiva. Inoltre sono state aggiunte le seguenti funzioni, chiamate nei punti opportuni:

- `bool crea_cow_condivisa()` (chiamata durante l'inizializzazione del sistema): crea e inizializza la zona cow iniziale, con dimensione pari a `DIM_USR_COW`, visibile a partire dall'indirizzo `ini_utn_w`; la zona deve inizialmente contenere solo byte nulli; restituisce `false` se non è stato possibile creare la zona, `true` altrimenti;
- `void copia_cow_condivisa()` (chiamata durante la creazione di un processo): copia le entrate opportune di `cow_root` nella tabella radice del nuovo processo;
- `aggiorna_cow_privata(vaddr v)` (chiamata durante un page fault): se `v` cade nella zona utente/cow effettua la copia, aggiorna la traduzione come descritto e restituisce `true`; se `v` non cade nella zona utente/cow, o se l'aggiornamento fallisce per qualche motivo, restiuisce `false`;
- `void distruggi_cow_privata()` (chiamata durante la distruzione di un processo): disfa quanto eventualmente fatto nelle precedenti `copia_cow_condivisa()` e `aggiorna_cow_privata()`.

Modificare il file `sistema.cpp` scrivendo le parti mancanti tra i tag `SOLUZIONE`.