Prova pratica di Calcolatori Elettronici

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

7 febbraio 2017

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

```
struct st1 { char vi[4]; };
struct st2 { long vd[4]; };
class cl {
        char v1[4]; int v3[4]; long v2[4];
public:
        cl(st1 ss);
        cl(st1& s1, int ar2[]);
        cl elab1(char ar1[], const st2& s2);
        void stampa() {
                for (int i = 0; i < 4; i++) cout << (int)v1[i] << ' '; cout << endl;
                for (int i = 0; i < 4; i++) cout << (int)v2[i] << ', '; cout << endl;
                for (int i = 0; i < 4; i++) cout << v3[i] << ' '; cout <math><< endl << endl;
        }
};
Realizzare in Assembler GCC le funzioni membro seguenti.
cl::cl(st1 ss)
        for (int i = 0; i < 4; i++) {
                v1[i] = ss.vi[i]; v2[i] = ss.vi[i];
                v3[i] = 4 * ss.vi[i];
}
cl::cl(st1& s1, int ar2[])
        for (int i = 0; i < 4; i++) {
                v1[i] = s1.vi[i]; v2[i] = -s1.vi[i];
                v3[i] = ar2[i];
}
cl cl::elab1(char ar1[], const st2& s2)
{
        st1 s1;
        for (int i = 0; i < 4; i++) s1.vi[i] = ar1[i] - i;
        cl cla(s1);
        for (int i = 0; i < 4; i++) cla.v3[i] = s2.vd[i];
        return cla;
}
```

- 2. Vogliamo estendere il nucleo in modo da permettere ai processi di rendere temporaneamente residenti alcune delle loro pagine virtuali private (nella zona utente/privata, contenente la pila di livello utente).
 - A tale scopo aggiungiamo al nucleo le seguenti primitive:
 - natl resident(addr base, natq size): rende residenti le pagine virtuali che contengono gli indirizzi da base (incluso) a base+size (escluso). Restituisce un identificatore che può poi essere passato a nonresident() per disfare l'operazione, o 0xffffffff in caso di fallimento; È un errore se gli indirizzi da base (incluso) a base+size (escluso) non sono all'interno della zona utente/privata;
 - void nonresident(natl id): disfa l'operazione di una precedente chiamata a resident(); è un errore se id non corrisponde ad una precedente operazione resident().

Se la primitiva resident() ha successo, non devono essere più possibili page fault nelle pagine interessate fino alla corrispondente nonresident(). Questo vuol dire che la primitiva deve anche caricare le necessarie pagine o tabelle assenti, e marcarle tutte come residenti in modo che non possano essere rimpiazzate. La primitiva resident() può fallire se non riesce a caricare una pagina o tabella (ad. es., perchè tutta la memoria è occupata da pagine residenti).

Le stesse pagine o tabelle possono essere coinvolte in più operazioni resident() distinte. Per permettere ciò trasformiamo il campo residente dei descrittori di pagina fisica in un contatore (conta il numero di operazioni resident() non ancora disfatte sulla tabella o pagina corrispondente). Se la primitiva resident() fallisce, deve riportare i contatori residente al valore che avevano prima dell'inizio della primitiva.

Le primitive abortiscono il processo chiamante in caso di errore.

Modificare i file sistema.cpp e sistema.s in modo da realizzare le primitive mancanti.

SUGGERIMENTI:

- si osservi con attenzione la funzione undo_res(), già presente nel codice, e come è usata in c_nonresident();
- è possibile usare la funzione des_pf* swap2(natl proc, int liv, addr ind_virt) che carica nello spazio del processo proc la tabella o pagina di livello liv relativa all'indirizzo ind_virt e restituisce il puntatore al descrittore di pagina fisica della pagina in cui l'entità è stata caricata; restituisce 0 se non è stato possibile caricare la pagina (memoria piena e impossibile rimpiazzare).
- La parte utente privata va da ini_utn_p (incluso) a fin_unt_p (escluso).

Sistema a 32bit: usare natl al posto di natq; la funzione swap2 usa tt tipo come secondo argomento; la parte utente privata va da inizio_utente_privato (incluso) a fine_utente_privato (escluso).