Prova pratica di Calcolatori Elettronici

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

5 giugno 2025

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

```
struct st1 { char vc[4]; };
class cl {
        long v[4]; st1 s;
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. Modifichiamo il nucleo per aggiungere parte del supporto per lo swap-in e swap-out dei processi. In particolare, un processo P_1 può richiedere lo swap-out di un altro processo P_2 invocando la primitiva `swap_out()` a cui deve passare l'identificatore di P_2 ed un buffer in grado di contenere una pila utente (`DIM_USR_STACK` byte). La primitiva rimuove la pila utente di P_2 dalla sua memoria virtuale, deallocando anche i corrispondenti frame, ma prima ne copia il contenuto nel buffer di P_1 (l'idea è che P_1 a questo punto dovrebbe salvare il buffer su disco, ma di questo non ci preoccupiamo). Il processo P_2 è a questo punto è *rimosso dalla memoria* e non può andare in esecuzione. Se lo schedulatore trova in testa alla coda pronti un processo rimosso, lo rimuove dalla coda pronti ma non lo mette in escuzione e passa a scegliere il successivo; setta però un flag nel descrittore del processo rimosso, per segnalare che il processo era stato schedulato. Un altro processo (tipicamente lo stesso che aveva richiesto lo swap-out, ma non necessariamente) potrà in seguito richiedere lo swap-in di un processo usando la primitiva `swap_in()`, a cui passa l'identificare del processo da ricaricare in memoria e un buffer che contiene i byte con cui inizializzare la pila. Se il processo era stato schedulato mentre era rimosso, la primitiva provvede anche a reinserirlo in coda pronti, rispettando le priorità.

   Per realizzare questo meccanismo aggiungiamo i seguenti campi al descrittore di processo:

```
bool swapped_out;
bool scheduled;
```

Il campo `swapped_out` vale true se il processo è rimosso dalla memoria, e false altrimenti; il campo `scheduled` vale true se il processo è stato schedulato mentre era rimosso dalla memoria, e false altrimenti.

Aggiungiamo poi le seguenti primitiva (abortiscono il processo in caso di errore, controllano problemi di Cavallo di Troia):

- `bool swap_out(natl pid, char *buf)`: esegue lo swap-out del processo `pid`. È un errore se il processo `pid` è un processo di sistema, o è lo stesso processo che invoca la primitiva; restituisce `false` se il processo non esiste o è già rimosso.
- `bool swap_in(natl pid, const char *buf)`: esegue lo swap-in del processo `pid`. È un errore se il processo `pid` è un processo di sistema; restituisce `false` se il processo non esiste o non è rimosso, o se la creazione della pila fallisce.

Modificare il file `sistema.cpp` in modo da realizzare le parti mancanti.