RPC

Servizio di Gestione delle Votazioni

ESERCITAZIONE 9 RDC

14 DICEMBRE 2021

Bonantini Alessandro
Galeone Christian
Garcia Fernandez Diego
Piras Gabriele
Predieri Nicholas

Introduzione esercizio

L'obiettivo dell'esercitazione è la **realizzazione** e la **gestione** di un **sistema di votazione di un talent show**.
Il server mantiene per ogni partecipante le seguenti informazioni: ***candidato*** (chiave della tabella che può comparire una sola volta), ***giudice***, ***categoria***, ***nome del file*** *che descrive il candidato* (nome unico all'interno di un direttorio sul server), ***fase*** *in cui si trova il candidato* e un ***intero*** *che riporta il numero totale di voti ricevuti da un candidato* (come da tabella presentata più avanti)

Obiettivo è la realizzazione di alcune funzionalità di supporto usando RPC di SUN

Specifiche

Si realizzino le operazioni per:

- **Visualizzare la classifica dei giudici**: questa operazione non richiede parametri di input e restituisce i nomi dei giudici ordinati per punteggio totale. Si ricordi che ogni giudice può avere più candidati, per i quali occorre sommare tutti i punteggi
- **Aggiungere un voto per un candidato**: questa operazione richiede all'utente il nome del candidato e aggiunge un voto alla prestazione identificata
- **Sottrarre un voto per un candidato**: questa operazione richiede all'utente il nome del candidato e sottrae un voto. Si assuma che il punteggio minino possibile sia zero

fattoreX.x

```
const N=7;
const NUMCOL=6;
const MAXS = 256;

struct Input{ string cand <MAXS>; string op <MAXS>;};
struct Nome { string c <MAXS> ;};
struct Elenco { Nome nome[N] ;};

program FATTOREX{
    version FATTOREXVERS{
    Elenco CLASSIFICA_GIUDICI(void) = 2;
    int ESPRIMI_VOTO(Input) = 1;
    } = 1;
} = 0x20000013;
```

fattoreX_s.c LATO SERVER

```
static Elenco elenco;
static Tab tab;
void inizializza() {
    int i, j;
    if(inizializzato== 1) return;
      for( i= 0; i< N; i++){//inizializzazione struttura
       strcpy(tab.show[i].candidato, "L");
       strcpy(tab.show[i].giudice, "L");
       tab.show[i].categoria= 'L';
       strcpy(tab.show[i].nomeFile,"L");
       tab.show[i].fase='L';
       tab.show[i].voto= -1;
    }
    strcpy(tab.show[2].candidato,"Brasco");
    strcpy(tab.show[2].giudice,"Bowie");
    tab.show[2].categoria='U';
    strcpy(tab.show[2].nomeFile,"BrascoProfile.txt");
    tab.show[2].fase='A';
    tab.show[2].voto=100;
    //Eventuale riempimento altri candidati

    inizializzato = 1;
    printf("Terminata init struttura dati !\n");}
```

```
typedef struct{
char candidato[MAXS]; char
giudice[MAXS];
char categoria;
char nomeFile[MAXS];
char fase; int voto; } Show;

typedef struct { Show show[N]
; } Tab ;
```

fattoreX_s.c ESPRIMI_VOTO

```
int * esprimi_voto_1_svc(Input *input, struct svc_req *rqstp)
{
    result=-1;
    inizializza();//Invoco l'inizializzazione
    int i;
    for( i= 0; i< N; i++){
    if(strcmp(tab.show[i].candidato, input->cand)==0){
        if(input->op[0] == '+'){
            input->op[0]=' ';
            tab.show[i].voto += atoi(input->op);}
        if(input->op[0] == '-'){
            input->op[0]=' ';
            if((tab.show[i].voto-atoi(input->op))>=0)
                tab.show[i].voto -= atoi(input->op);
        }
        result=0;}}

     if (result == -1) {
        perror("Il candidato indicato non esiste!");
        exit(-1);
     }
    return(&result);}
```

fattoreX_s.c classifica_giudici

```c
Elenco *classifica_giudici_1_svc(void *in, struct svc_req *rqstp)
{
    inizializza(); //Invoco l'inizializzazione
    G g[N];
    int j=0, k, l;
    int isIn = 0;


     for(k=0; k<N; k++){//inizializzo array G
               strcpy(g[k].nome, "L");
                g[k].votoTot = 0;
           }
     for(int i= 0; i< N; i++){ //costruisco G
         if(strcmp(tab.show[i].giudice, "L") != 0){//se il giudice nella main tab non è vuoto
             for( k=0; k<N; k++){
                 if(strcmp(g[k].nome, tab.show[i].giudice)==0){
                     isIn=1;
                     g[k].votoTot += tab.show[i].voto;//se già inserito, sommo il voto al totale
                 }
             }
             if(!isIn){//inizializzazioni nome giudice e voto in G
                 strcpy(g[j].nome, tab.show[i].giudice);
                  g[j].votoTot += tab.show[i].voto;
                 j++; //tengo il conto del riempimento di G
             }//altrimenti le operazioni che servivano sono già state fatte
            isIn = 0;}}
```

```c
typedef struct{
     char nome[MAXS];
     int votoTot;
} G;
```

fattoreX_s.c classifica_giudici

```
    //ordino G
     quickSortE(g, j);

     //riempo elenco giudici
    for(int i=0, k=0; i<j; i++, k++){
        strcpy(elenco.nome[i].c, g[k].nome);
    }

    return (&elenco);

} //classifica_giudici
```

fattoreX_c.c LATO CLIENT

```
while (gets(ok))
    {   if(strcmp(ok,"V")==0)
        {
            printf("\nInserisci nome candidato: ");
            scanf("%s", candidato);

            printf("\nInserisci numero specificando tipo di operazione (+ o -): ");
             scanf("%s", op); //leggo e controllo
    while(op[0] != '+' && op[0]!='-')
    {
        printf("Simbolo sbagliato! Inserisci + (aggiunta) o - (sottrazione): \n");
        scanf("%s", op);}
    input.op = op;
    input.cand = candidato;

    gets(ok);//Consumo fine linea


    ris= esprimi_voto_1(&input, cl);
    // Invocazione remota
    if(ris == NULL) {clnt_perror (cl, host); exit(1); }
    if(*ris <0) printf("Problemi... \n");
    else printf("Operazione effettuata con successo! \n");
    }   // if V
```

fattoreX_c.c LATO CLIENT

```
    else if(strcmp(ok,"G")==0)
    {// Invocazione remota
        elenco= classifica_giudici_1(in,cl);
        if(elenco == NULL) { clnt_perror(cl, host);
        exit(1);}
        printf("Stato di occupazione dell'elenco: giudici \n");

        for( i= 0; i< N; i++){
        printf("%s\n", elenco->nome[i].c);
        }

    } // if G
    else
    printf("Argomento di ingresso errato \n");
   printf("Inserire: \nG) per vedere i giudici \tV) per esprimere il voto  \t^D per
terminare:");
    } // while
    // Libero le risorse , distruggendo il gestore di trasporto
    clnt_destroy(cl);
    exit (0);

}// main
```