



ESERCITAZIONE 9:

Remote Procedure Call (RPC)

- Gestione di una struttura dati memorizzata in remoto
- Operazioni di lettura/scrittura sulla struttura
- Richiedere le operazioni mediante una chiamata ad operazione remota (RPC)





RPC



FILE FATTORE.X

```
struct input{string nome<32>; string operazione<16>;};
struct judge{char nome[32]; int voti;};
struct output{judge giudice[4];};

program FATTORE{
  version FACTVERS{
    output CLASSIFICA_GIUDICI(void)=1;
    int ESPRIMI_VOTO(input)=2;
  }=1;
}=0x20000013;
```

rpcgen fattore.x

```
C fattore_clnt.c
C fattore_svc.c
C fattore_xdr.c
h fattore.h
```

fattore.h

```
struct input {
  char *nome;
  char *operazione;
};
typedef struct input input;

struct judge {
  char nome[32];
  int voti;
};
typedef struct judge judge;

struct output {
  judge giudice[4];
};
typedef struct output output;
```

client
server

*gcc -o server fattore_impl.c
fattore_svc.c fattore_xdr.c*

*gcc -o client client.c fattore_clnt.c
fattore_xdr.c*



IMPL.
PROCEDURE
1



INIZIALIZZAZIONE

```
#define N 8
#define Ngiudici 4
typedef struct candidato
{
    char nome[32];
    char giudice[32];
    char categoria;
    char nomeFile[64];
    char fase;
    int voti;
} candidato;

static candidato candidati[N];
static int inizializzato = 0;
```

Struttura dichiarata in
locale perché utilizzata
solo server-side

```
void inizializza()
{
    int i;
    for (i = 0; i < N; i++)
    {
        sprintf(candidati[i].nome, "%d" , i);
        sprintf(candidati[i].giudice, "%d", i % Ngiudici );
        candidati[i].categoria = 'U';
        sprintf(candidati[i].nomeFile, "%d.txt", i );
        candidati[i].fase = 'S';
        candidati[i].voti = 0;

        printf("candidato: %s\tgiudice: %s\n",candidati[i].nome,candidati[i].giudice);
    }
    inizializzato = 1;
    printf("Inizializzazione eseguita\n");
}
```



IMPL.
PROCEDURE
2



ESPRIMI VOTO

- Logica gestita tramite controllo di stringhe
- Operazioni permesse: **add** e **sub**
- **Sub** non scende mai sotto 0

```
int *esprimi_voto_1_svc(input * in, struct svc_req *rp)
{
    static int flag;
    flag=-1;
    int i;
    if(!inizializzato)inizializza();
    for (i = 0; i < N; i++)
    {
        if (strcmp(candidati[i].nome, in->nome) == 0)
        {
            if (strcmp("add", in->operazione) == 0)
            {
                candidati[i].voti++;
                printf("Voto aggiunto a candidato %s", candidati[i].nome);
                flag=1;
                return &flag;
            }
            if (strcmp("sub", in->operazione) == 0)
            {
                if(candidati[i].voti==0){
                    return &flag;
                }
                candidati[i].voti--;
                printf("Voto sottratto a candidato %s", candidati[i].nome);
                flag=1;
                return &flag;
            }
        }
    }
    return &flag;
}
```

CLASSIFICA GIUDICI

- La struttura res è locale perché non necessita di visibilità globale
- Res viene re-inizializzata ad ogni chiamata per evitare l'accumulo di voti

```
output *classifica_giudici_1_svc(void * nientedinentepropriouna, struct svc_req *rp)
{
    static output res;
    printf("qua almeno dai");
    int i, j;
    for (i = 0; i < Ngiudici; i++) // Inizializzazione giudici...
    if(!inizializzato)inizializza();
    for (i = 0; i < N; i++)
    {
        for (j = 0; j < Ngiudici; j++)
        {
            if (strcmp(candidati[i].giudice, res.giudice[j].nome)==0)
            {
                res.giudice[j].voti += candidati[i].voti;
                break;
            }
        }
    }
    judge temp;
    for (j = 0; j < Ngiudici; j++)
    {
        for (i = 0; i < Ngiudici - j - 1; i++)
        {
            if (res.giudice[i].voti > res.giudice[i + 1].voti)
            {
                temp = res.giudice[i];
                res.giudice[i] = res.giudice[i + 1];
                res.giudice[i + 1] = temp;
            }
        }
    }
    return (&res);
}
```



CLIENT 1



```
// creazione gestore di trasporto
if ((cl = clnt_create(server, FATTORE, FACTVERS, "udp")) == NULL)
{
    clnt_pcreateerror(server);
    exit(1);
}
printf("Inserisci operazione desiderata:\n>vota\n>classifica\n");
while (gets(procedure))
{
```

Il Client interagisce con l'utente proponendogli ciclicamente i servizi che utilizzano le due procedure remote. Richiede gli input necessari, invoca il servizio specificato e stampa a video gli esiti delle chiamate, fino alla fine del file di input da tastiera.

```
if (strcmp(procedure, "vota") == 0)
{
    printf("Inserisci il nome del partecipante: ");
    gets(nome);
    printf("Inserisci la procedura richiesta: ");
    gets(operazione);
    if (strcmp(operazione, "add") != 0 && strcmp(operazione, "sub") != 0)
    {
        printf("Operazione non esistente, seleziona \"add\" o \"sub\"\n");
        continue;
    }
    in.nome = nome;
    in.operazione = operazione;
    res = esprimi_voto_1(&in, cl);
    switch (*res)
    {
        case 0:
            printf("Il concorrente ha già zero voti, diminuire è follia!!!!!!\n");
            break;
        case -1:
            printf("Il concorrente non esiste, non lo ha certamente preso il governo cinese :):)\n");
            break;
        default:
            printf("Hai votato con successo, bravo campione!");
            break;
    }
}
```

```
else if (strcmp(procedure, "classifica") == 0)
{
    out = classifica_giudici_1( no, cl);
    if (out == NULL)
    {
        clnt_perror(cl, server);
        exit(1);
    }
    for (i = 0; i < Ngiudici; i++)
    {
        printf("Giudice: %s \t Voti: %d\n", out->giudice[i].nome, out->giudice[i].voti);
    }
}
```



FINE



CONCLUSIONI

- Creazione e gestione delle socket UDP o TCP automaticamente generata da rpc
- Implementazione delle procedure più semplice, in quanto è necessario «dichiarare» i metodi nel file .x e implementarli quindi in un file .c
- Utilizzo delle procedure più immediato, dato che il client deve solo includere il file .h generato da rpcgen per richiamare i metodi
- Inizializzazione realizzata nell'implementazione per non modificare i file auto-generati da rpc