Università degli Studi di Bologna
Scuola di Ingegneria

Corso di
Reti di Calcolatori T

Esercitazione 8 (svolta)

RPC: Inizializzazione Strutture Dati sul Server

Antonio Corradi, Luca Foschini
Michele Solimando, Giuseppe Martuscelli, Marco Torello
Anno accademico 2022/2023

SPECIFICA

Si richiede la progettazione e la realizzazione del servizio di **prenotazione delle postazioni di una sala lettura**

Le postazioni sono organizzate in **file** e **colonne**, e si prevedono tre tipi di prenotazioni:

Bibliotecari (B), Professori (P), e Dottorandi (D)

Si realizzino le operazioni per:

- **Prenotare una postazione**: questa operazione richiede all'utente il **tipo di prenotazione**, la **fila** e la **colonna**, quindi aggiorna la struttura dati contenente le prenotazioni
- **Visualizzare lo stato della sala**: questa operazione visualizza l'attuale **stato di occupazione della sala** indicando, per ogni postazione, il tipo di prenotazione

STRUTTURA DATI

La **struttura dati** con lo stato della sala potrebbe essere visualizzata con la seguente tabella; ogni postazione libera viene etichettata con il carattere ‘L’, mentre le postazioni occupate sono opportunamente etichettate con ‘B, ‘P’, o ‘D’ a secondo del tipo di prenotazione

	Col. 1	Col. 2	Col. 3	Col. 4	Col. 5	Col. 6	Col. 7
Fila 1	**L**	**L**	**L**	**L**	**L**	**L**	**L**
Fila 2	**D**	**L**	**L**	**L**	**D**	**D**	**B**
Fila 3	**L**	**P**	**L**	**L**	**D**	**D**	**L**
Fila 4	**L**	**L**	**L**	**L**	**D**	**D**	**L**
Fila 5	**L**	**L**	**L**	**L**	**L**	**L**	**L**
Fila 6	**L**	**L**	**P**	**L**	**P**	**P**	**L**
Fila 7	**L**	**L**	**L**	**L**	**P**	**P**	**L**

SPECIFICA: ULTERIORI DETTAGLI

Il **Client** realizza l'interazione con l'utente proponendo ciclicamente i servizi che utilizzano le due procedure remote e stampa a video gli esiti delle chiamate, fino alla fine del file di input da tastiera

Il **Server** implementa le procedure invocabili:

- La procedura `prenota_postazione` accetta come parametro d'ingresso la struttura dati Input contenente il **tipo di prenotazione ('B', 'P', o 'D'), la fila e la colonna che si vogliono prenotare,** e restituisce **l'esito dell'operazione - 0 in caso di successo, -1 altrimenti**: ad esempio, se il posto che si vuole occupare è già prenotato
- La procedura `visualizza_stato` non ha alcun parametro di ingresso (ha come parametro di ingresso un puntatore a void), e restituisce la struttura **dati Sala** che rappresenta lo stato di occupazione attuale della sala

INIZIALIZZAZIONE STRUTTURA DATI

C'è un problema di **inizializzazione della struttura dati** che mantiene lo stato di occupazione della sala sul server:

la struttura deve essere inizializzata prima della partenza del server impostando tutte le postazioni allo stato di default libero (stato 'L')

Nella generazione automatica del codice, RPCGEN produce lo stub del programma server (che contiene il main)

Bisogna considerare soluzioni per **attuare l'operazione di inizializzazione**, **eventualmente** anche **modificando i file generati in modo automatico**

INIZIALIZZAZIONE DATI: ALCUNE POSSIBILITÀ

1. Inserire una **procedura locale di inizializzazione nel server che** specifica le procedure remote (nel server). L'invocazione di una qualsiasi procedura remota invocata dal client, provoca l'invocazione di tale procedura se e solo se l'inizializzazione non è già stata invocata (introduzione di una variabile di controllo **`static`** nel file utente)
2. Aggiungere all'interfaccia del server una **procedura remota di inizializzazione**. Questa procedura remota deve essere invocata dal **client prima di qualsiasi altra procedura**, e che deve essere eseguita dal server al più una volta
3. Inserire il **codice di inizializzazione direttamente all'interno del main (stub del server)**
4. Inserire una **procedura locale di inizializzazione nel server che** specifica le procedure remote (nel server). Tale procedura deve essere invocata all'interno del main **prima** dell'invocazione della procedura **`svc_run()`** che mette il server in ascolto di nuove richieste. Per poter fare questo, è necessario dichiarare la procedura di inizializzazione come **`extern`** all'interno del file dello stub contenente stub e main (lato server)

FILE SALA.X

```
const LUNGHFILA=7;
const NUMFILE=10;

struct Input{ char tipo; int fila; int colonna;
    };
struct Fila { char posto[LUNGHFILA]; };
struct Sala { Fila fila[NUMFILE]; };

program SALA {
  version SALAVERS{
    int  PRENOTA_POSTAZIONE(Input) = 1;
    Sala VISUALIZZA_STATO(void) = 2;
  } = 1;
} = 0x20000013;
```

Compilazione per generare il file header, le conversioni xdr e gli stub:

```
rpcgen sala.x
```

FILE SALA_S.C: INIZIALIZZA()

```
#include ...
/* STATO SERVER */
static Sala sala;   /* variabile per la sala */
static int inizializzato=0;  /* inizializzazione attuata */

void inizializza()    // Possibilità 1
{ int i, j;
  if( inizializzato == 1 ) return;
  // inizializzazione struttura dati
  for (i = 0; i < NUMFILE; i++)
    for (j = 0; j < LUNGHFILA; j++)
      sala.fila[i].posto[j] = 'L';

  // Eventuale riempimento altri posti
  sala.fila[1].posto[0] = 'D';
  sala.fila[2].posto[1] = 'P';
  sala.fila[5].posto[2] = 'P';
  ...
  inizializzato = 1;
  printf("Terminata init struttura dati!\n");
}
```

FILE SALA_S.C: IMPLEMENTAZIONE PROCEDURE

```
int * prenota_postazione_1_svc
          (Input *input, struct svc_req *rqstp)
{static int result = -1;
 result=-1;
 inizializza();  // Invoco l'inizializzazione

if(sala.fila[input>fila].posto[input>colonna]!='L')
       return (&result);
else
 {sala.fila[input->fila].posto[input->colonna]=
          input->tipo;
  result=0; return (&result);
 }
} // prenota_postazione


Sala* visualizza_stato_1_svc
             (void *in, struct svc_req *rqstp)
{ inizializza();    // Invoco l'inizializzazione
  return (&sala);
} // visualizza_stato
```

FILE SALA_C.C: IMPLEMENTAZIONE DEL CLIENT 1/4

```
#include ...
main (int argc, char *argv[])
{char *host; CLIENT *cl;
 int *ris, *start_ok;  void *in;
 Sala *sala; Input input;
 char str[5]; char c, ok[256];
 int  i, j, fila, col;

  if (argc != 2)
  { printf ("usage: %s server_host\n", argv[0]);
    exit (1);
  }
  host = argv[1];

  cl = clnt_create (host, SALA, SALAVERS, "udp");
  if (cl == NULL)
  { clnt_pcreateerror (host); exit (1); }
```

FILE SALA_C.C: IMPLEMENTAZIONE DEL CLIENT 2/4

```
printf("Inserire:\nV) per vedere la sala\tP) per prenotare
   la postazione \t^D per terminare: ");
while (gets (ok))
{ if( strcmp(ok,"P")==0 )
   {gets(ok);          // Leggo e controllo il tipo
    while( (strcmp(ok, "P")!=0) && (strcmp(ok, "D")!=0) &&
             (strcmp(ok, "B")!=0) )
    { printf("Lettera sbagliata! Inserisci P, D o B: \n");
   gets(ok);}
   input.tipo = ok[0];

   fila = -1;          // Leggo la fila
   while( fila < 0 || fila > (NUMFILE-1) )
   { printf("Inserisci la fila (da 0 a %i): \n", (NUMFILE-
   1));
     while (scanf("%i", &fila) != 1)
     { do { c = getchar(); printf("%c ", c);}
       while (c!= '\n'); printf("Fila: ");
     }
   }
   gets(ok); //Consumo fine linea
  input.fila = fila;
```

FILE SALA_C.C: IMPLEMENTAZIONE DEL CLIENT 3/4

```
col=-1; // Leggo la colonna
while( col<0 || col>(LUNGHFILA-1) )
{ printf("Inserisci la colonna (0 - %i):\n", (LUNGHFILA-1));
    while (scanf("%i", &col) != 1)
    { do{c=getchar(); printf("%c ", c);}
      while (c!= '\n');
      printf("Colonna: ");
    }
  }
  gets(ok); // Consumo fine linea
  input.colonna = col;
   ris = prenota_postazione_1(&input, cl);
// Invocazione remota
if(ris ==NULL){clnt_perror(cl, host); exit(1); }
if(*ris<0) printf("Problemi ...\n");
else printf("Prenotazione effettuata con successo\n");
} // if P
/* ad ogni invocazione controlliamo sempre
   che non ci sia stato un errore di RPC (risultato NULL) e
   poi di logica (a secondo del valore atteso e dalla logica del programma */
```

FILE SALA_C.C: IMPLEMENTAZIONE DEL CLIENT 4/4

```
 else if( strcmp(ok,"V")==0 )
 {// Invocazione remota
  sala = visualizza_stato_1(in,cl);
  if(sala == NULL) { clnt_perror(cl, host);
                     exit(1); }
   printf("Stato di occupazione della sala:\n");
    for(i=0; i<NUMFILE; i++)
    { for(j=0; j<LUNGHFILA; j++)
          printf("%c\t", sala->fila[i].posto[j]);
       printf("\n");
    }
  } // if V

  else printf("Argomento di ingresso errato!!\n");
  printf("Inserire:\nV) per vedere la sala\tP) ... ");
 } // while

 // Libero le risorse, distruggendo il gestore di trasporto
 clnt_destroy (cl);
 exit (0);
}   // main
```

COMPILAZIONE ED ESECUZIONE

Compilazione per generare l'eseguibile del **client**:

```
gcc -o c sala_c.c sala_clnt.c sala_xdr.c
```

→ produce il comando `c`

Compilazione per generare l'eseguibile del **server**:

```
gcc -o s sala_s.c sala_svc.c sala_xdr.c
```

→ produce il comando `s`

Esecuzione

1. Mandare in esecuzione il server con il comando: `s`
2. Mandare in esecuzione il client con il comando:

```
c serverhost
```