



**Università degli Studi di Bologna
Scuola di Ingegneria**

Corso di Reti di Calcolatori T

Esercitazione 8 (proposta) Remote Procedure Call (RPC)

Antonio Corradi, Luca Foschini

Michele Solimando, Giuseppe Martuscelli, Marco Torello

Anno accademico 2019/2020

SPECIFICA

Utilizzando RPC sviluppare un'applicazione C/S che consenta di effettuare le operazioni remote per:

- **contare i caratteri, le parole e le linee** di un **file di testo** presente sul server remoto;
- **contare il numero di file** (presenti nel **direttorio remoto** indicato dal client) **la cui dimensione è maggiore di un intero** indicato dal client.

PROCEDURE REMOTE

In particolare, si sviluppino le seguenti procedure:

- La procedura **file_scan** accetta come parametro d'ingresso **il nome del file** e restituisce **una struttura dati** che contiene **tre interi**, corrispondenti al **numero di caratteri**, **parole** e **linee** nel file, oppure un opportuno codice di errore in caso ad esempio il file sia vuoto oppure non sia presente sul server.
- La procedura **dir_scan** accetta come parametro d'ingresso **il nome del direttorio remoto** e **una soglia numerica**. In caso di successo, restituisce **un intero positivo** con il numero di file la cui dimensione supera la soglia inserita, -1 altrimenti.

NOTE REALIZZATIVE

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

Il **Server** implementa le procedure invocabili in remoto restituendo l'esito delle operazioni come indicato sopra

REPETITA: XDR - ALCUNI CONSIGLI SULLA DEFINIZIONE DEI TIPI DI DATI

I dati al **primo livello** (cioè quelli passati direttamente alle funzioni) possono essere passati **SOLO per valore** e **NON si possono passare** tipi di dato complessi (ad esempio gli array). Ad esempio:

```
string ECHO(string s);
```

Sì 😊

```
char[] ECHO(char arrayCaratteri[12]);
```

No 😞

I dati al **secondo livello** (cioè definiti all'interno di altre strutture dati) possono invece usare anche strutture dati complesse (ad esempio array) e puntatori.

```
struct Input{char arrayCaratteri[12];};  
... Input ECHO(Input i);
```

Sì 😊

Le **matrici** vanno però sempre definite **PER PASSI**:

```
struct Matrix{char arrayCaratteri[12][12];};
```

No 😞

```
struct Riga{char arrayCaratteri[12];};  
struct Matrix{Riga riga[12];};
```

Sì 😊

REPETITA: XDR - ALCUNI CONSIGLI SULLA DEFINIZIONE DEI TIPI DI DATI

Come definire una stringa all'interno di una struttura di un file XDR

```
struct Input
{
    string direttorio <50>;
    int x;
};
```

Una stringa va definita indicando la dimensione come massimo di default '<>' o massimo indicato esplicitamente '<50>', usando le **parentesi angolari**



PROPOSTA DI ESTENSIONE: MULTIPLE GET



Si vuole sviluppare un'applicazione C/S basata su RPC e su socket con connessione per il **trasferimento di tutti i file di un direttorio remoto dal server al client** (**multiple get**).

In particolare, si vogliono realizzare due modalità di trasferimento, la prima con **client attivo** (**il client effettua la connect**), la seconda con **server attivo** (**il server effettua la connect**). Si dovranno realizzare un **client** e un **server**; l'utente, per ogni trasferimento, decide quale delle due modalità utilizzare.

Per entrambe le modalità, si prevede **un'interazione iniziale sincrona (realizzata con una richiesta RPC sull'oggetto remoto server)** per trasferire la **lista dei file** da inviare e **l'endpoint** (host e porta) **di ascolto**; quindi, una seconda fase di **trasferimento dei file** dal server al client (**realizzata con socket connesse**).



TRASFERIMENTO PIÙ DIRETTORI: CLIENT ATTIVO



La procedura remota accetta come argomento di ingresso il **nome del direttorio** e restituisce una struttura dati con l'**endpoint di ascolto del server** e la **lista con i nomi e la lunghezza di tutti i file** da trasferire.

Il **Client** richiede ciclicamente all'utente il **nome del direttorio** da trasferire, **effettua la chiamata RPC** e **riceve l'endpoint di ascolto**, quindi **stabilisce una connessione** con il server remoto e riceve i file salvandoli sul direttorio locale.

Il **Server** implementa la **procedura RPC richiesta** ed è realizzato come **server concorrente** e **parallelo**. Per ogni nuova richiesta ricevuta il processo padre, dopo **aver accettato la richiesta RPC**, attiva un processo figlio a cui affida la **creazione della socket di ascolto** e il completamento del servizio richiesto; quindi il padre restituisce **la lista dei file e il proprio endpoint**.



TRASFERIMENTO PIÙ DIRETTORI: SERVER ATTIVO



Il metodo remoto accetta come argomento di ingresso il **nome del direttorio** e l'**endpoint di ascolto del client** e restituisce la **lista con i nomi e la lunghezza di tutti i file** da trasferire.

Il **Client** richiede ciclicamente all'utente il **nome del direttorio** da trasferire, **crea la socket di ascolto**, **effettua la chiamata RPC** e riceve la lista dei file da trasferire; infine, **effettua la accept** e i trasferimenti di file necessari.

Il **Server** implementa la procedura RPC richiesto ed è realizzato come **server concorrente e parallelo**; per ogni nuova richiesta ricevuta il processo padre, dopo aver **accettato la richiesta RPC**, attiva un processo figlio a cui affida la **creazione della socket** su cui eseguire la **connect** e il completamento del servizio richiesto; quindi **restituisce la lista dei file**.