

Reti di Calcolatori T

Appello del 14/01/2022

Compito 1

Cognome:
Nome:
Matricola:

Tempo a disposizione: 3h

È obbligatorio inserire Cognome, Nome, e numero di Matricola all'inizio di ogni file sorgente, pena la non valutazione del compito, che viene stampato in modo automatico solo in caso siano presenti gli elementi detti sopra.

Si devono consegnare **singularmente tutti i file sorgente e tutti gli eseguibili prodotti** (per favore, solo quelli relativi ai file sorgente consegnati!!!).

La prova intende valutare le capacità progettuali e di programmazione sia in **ambiente Java** che in **ambiente C**, pertanto è consigliato sviluppare **entrambe** le soluzioni richieste al meglio.

In entrambi gli esercizi, sia in Java che in C, si effettuino gli opportuni controlli sugli argomenti della richiesta e si gestiscano le eccezioni verso l'utente, tenendo presente i criteri secondo cui si possa ripristinare il funzionamento del programma oppure si debba forzarne la terminazione.

Leggete con attenzione le specifiche del problema prima di impegnarvi "a testa bassa" nello sviluppo delle singole parti. Naturalmente, ci aspettiamo che i componenti da consegnare siano stati provati e siano funzionanti.

Si richiede la progettazione e la realizzazione di **servizi invocabili** da parte di più clienti **sul file system di una macchina server** (direttorio remoto). I file binari possono avere qualsiasi estensione (non '.txt') e sono composti da dati binari. I file di testo sono i file con nome con suffisso '.txt' e composti solo da linee di caratteri ASCII.

In particolare, si realizzino i seguenti servizi:

1. **Conteggio delle linee che iniziano con una lettera maiuscola e contengono almeno uno specificato numero di occorrenze di un carattere indicato dall'utente all'interno di tutti i file di testo presenti sul direttorio remoto:** questa operazione richiede all'utente *un carattere maiuscolo* e il *numero di occorrenze*, quindi scandisce ciascun file di testo riga per riga contando il numero di righe che iniziano con una lettera maiuscola e contengono (in qualsiasi posizione della riga) un numero di occorrenze del carattere uguale o superiore a quelle indicate dall'utente; infine, restituisce a console l'esito dell'operazione.
2. **Eliminazione di tutte le occorrenze di caratteri numerici all'interno di un file di testo:** questa operazione richiede all'utente il *nome di un file*, elimina tutte le occorrenze di *caratteri numerici* (da '0' a '9') dal file stesso, e visualizza sul cliente a video il numero di eliminazioni effettuate dal server o un'indicazione di errore.
3. **Lista dei file di testo di un direttorio specificato dall'utente:** questa operazione richiede all'utente un *nome di direttorio*, quindi visualizza la lista dei *file di testo* (con suffisso '.txt') contenuti nel direttorio richiesto.
4. **Trasferimento dal server al client di tutti i file binari di un direttorio:** questa operazione richiede all'utente il *nome del direttorio* e trasferisce tutti i file binari in esso contenuti (i cui nomi non abbiano suffisso '.txt'), salvandone il contenuto sul direttorio corrente del client.

Ogni comando considera come direttorio di partenza il *direttorio corrente del client e del server*. Se per esempio il client richiede la lista dei sottodirettori (usando una notazione relativa) nel direttorio corrente, il server deve listare i sottodirettori del direttorio a partire dal corrente da cui è stato lanciato. Si richiede inoltre di **non** usare nella soluzione comandi Unix (es. il comando **ls**), ma solo primitive di sistema (es. **opendir()** in C) e funzioni di libreria (es. metodi della **classe File** in Java).

Parte C

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

- eliminare **tutte le occorrenze di caratteri numerici** all'interno di un file di testo,
- ricevere **la lista dei file di testo di un direttorio specificato dall'utente**.

Il progetto prevede che il server metta a disposizione due procedure (parte di una interfaccia specificata in XDR e contenuta nel file *RPC_xFile.x*) invocabili in remoto dal client:

- La procedura **elimina_occorrenze** accetta come parametro d'ingresso **il nome del file**, elimina tutte le occorrenze di caratteri numerici (da '0' a '9') all'interno di quello stesso file, e restituisce **un intero positivo** con il numero di eliminazioni effettuate (≥ 0) in caso di successo, **oppure -1** in caso di insuccesso, ad esempio errori o altro.
- La procedura **lista_filetesto** accetta come parametro d'ingresso **il nome del direttorio**, e restituisce la lista **dei primi N ($N \leq 6$) nomi dei file di testo trovati, se ci sono**. In caso di direttorio inesistente, prevedere una segnalazione di errore.

Si progettino inoltre i sorgenti:

- **RPC_Server** (contenuta nel file *RPC_Server.c*), che implementa le procedure del server invocabili in remoto;
- **RPC_Client** (contenuta nel file *RPC_Client.c*), il processo filtro che realizza l'interazione con l'utente, **propone ciclicamente i servizi** che utilizzano le due procedure remote, e stampa a video i risultati, fino alla fine dello stream di input.

Parte Java

Sviluppare un'applicazione C/S basata su **socket stream** che realizzi le operazioni remote per:

- **contare il numero delle linee che iniziano con una lettera maiuscola e contengono almeno uno specificato numero di occorrenze di un carattere indicato** all'interno di **tutti i file di testo** presenti sul direttorio remoto,
- **trasferire dal server al client tutti i file binari di un direttorio**,

utilizzando un'**unica connessione per ogni sessione cliente e un unico processo figlio sul servitore**: questo vincolo è da **intendersi come prioritario e fondamentale**.

Più in dettaglio:

- Il **cliente** è organizzato come un **processo filtro ciclico che consuma l'input fino a fine file** e, per ogni iterazione del ciclo, chiede all'utente quale tipo di operazione vuole effettuare e realizza le interazioni col server utilizzando **una sola connessione per la intera sessione**; alla ricezione di fine file, libera opportunamente le risorse e termina.
Per ogni richiesta ricevuta dall'utente, il client prima invia il tipo di servizio al server, poi gestisce gli invii e le ricezioni necessarie alla realizzazione dello specifico servizio richiesto.
Nel caso della funzionalità di **conteggio del numero di linee** all'interno di **tutti i file di testo** presenti sul direttorio (corrente) remoto, il client chiede all'utente e invia al server il carattere e il numero di occorrenze, quindi riceve l'esito e lo stampa a video.
Nel caso della funzionalità di **trasferimento dal server al client di tutti i file binari di un direttorio**, per ogni richiesta, il client chiede all'utente il nome del direttorio, quindi invia al server il nome del direttorio e riceve i file salvandone il contenuto sul direttorio del client.
- Il **server** è organizzato come un **unico processo che gestisce in modo parallelo** l'interazione coi clienti, generando **un figlio per tutta la sessione di richieste da quel client**. Per ogni richiesta, il processo figlio che serve la sessione con una prima lettura discrimina il tipo di funzionalità richiesto, poi gestisce opportunamente l'operazione e si pone in attesa di nuove richieste dallo stesso client; alla lettura della fine sessione, il figlio termina.
Per ogni richiesta di **conteggio del numero di linee** all'interno di **tutti i file di testo** presenti sul direttorio (corrente) remoto, il figlio legge il carattere e il numero di occorrenze, quindi esegue l'operazione di conteggio, scandendo ciascun file di testo riga per riga e contando il numero di righe che iniziando con una lettera maiuscola e contengono (in qualsiasi posizione della riga) almeno un numero di occorrenze del carattere pari (cioè maggiore o uguale) a quelle indicate dall'utente, su ciascun file di testo trovato; quindi invia la risposta al client, rappresentata da un intero positivo che indica il numero totale di occorrenze contate (≥ 0) in caso di successo, -1 in caso di problemi.
Per ogni richiesta di **trasferimento dal server al client di tutti i file binari di un direttorio**, il figlio legge il nome del direttorio, quindi attua un protocollo per inviare tutti i file binari trovati al client.