Reti di Calcolatori T
Reti di Calcolatori L-A
Appello del 10/09/2012

Cognome: .....
Nome: .....
Matricola: .....

Tempo a disposizione: 2h

È obbligatorio inserire Cognome Nome Matricola e Numero Compito 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 **tutti i file sorgente e tutti gli eseguibili prodotti singolarmente** (*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 sui parametri della richiesta e si gestiscano le eccezioni, 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 sul file system della macchina server** (direttorio remoto), da parte di più clienti.
Ogni comando considera come direttorio di partenza il direttorio corrente, dove sono stati lanciati rispettivamente il client e il server. Se per esempio il client richiede la cancellazione di un file (usando una notazione relativa) nel direttorio corrente, il server cerca di cancellare il file nel direttorio dove è stato lanciato.
Si considerino solo file di testo ed, in particolare, si realizzino i seguenti servizi:

1. **conteggio delle occorrenze dei seguenti tre segni di interpunzione: ".", "!" e ";" all'interno di un file di testo:** questa operazione richiede all'utente il nome di un file, conta tutte le occorrenze dei tre segni di interpunzione all'interno del file, e visualizza a video l'esito dell'operazione;
2. **lista dei file i cui nomi contengono almeno uno di due caratteri**: questa operazione richiede all'utente i carattere, quindi visualizza la lista dei file il cui nome contiene almeno una occorrenza di ciascuno dei caratteri;
3. **sostituzione di tutte le occorrenze di un carattere con un altro carattere all'interno di un file di testo:** questa operazione richiede all'utente il nome di un file e due carattere, sostituisce tutte le occorrenze del primo carattere col secondo carattere, e visualizza a video il numero di sostituzioni effettuate.
4. **trasferimento di tutti file di un direttorio i cui nomi relativi iniziano con un certo prefisso dal server al client**: questa operazione richiede all'utente il nome del direttorio e il prefisso (cioè i primi caratteri del nome file) e trasferisce (get) i file di quel direttorio il cui nome inizia col prefisso richiesto, salvandoli sul direttorio del client.

Si richiede inoltre di **non** usare nella soluzione comandi Unix (es. il comando **ls**), ma solo primitive di sistema (es. **opendir**() in C) o funzioni di libreria (es. metodi della **classe File** in Java)

Parte Java

Utilizzando java RMI sviluppare un'applicazione C/S che consenta di effettuare le operazioni remote per:
- contare il numero di occorrenze dei seguenti tre segni di interpunzione: ".", "!" e ";" all'interno di un file di testo;
- ricevere la lista dei file i cui nomi contengono almeno uno di due carattere specificati;

Il progetto RMI si basa su un'interfaccia (contenuta nel file *RMI_interfaceFile.java*) in cui vengono definiti i metodi invocabili in remoto dal client:
- Il metodo **conta_occorrenze_interpunzione** accetta come parametro d'ingresso **il nome del file**, e deve contare tutte le **occorrenze dei seguenti tre segni di interpunzione: ".", "!" e ";" nel file**, per restituire un intero positivo con il numero di occorrenze trovate (>=0) in caso di successo, oppure -1 in caso di errori o file inesistente.
- Il metodo **lista_file_caratteri** accetta come parametri d'ingresso **i caratteri di cui si vuole effettuare la ricerca**, e restituisce la lista dei nomi dei file trovati. In caso di nessun file presente, prevedere una segnalazione di errore.

Si progettino inoltre le classi:
- ***RMI_Server*** (contenuta nel file *RMI_Server.java*), che implementa i metodi del server invocabili in remoto;
- ***RMI_Client*** (contenuta nel file *RMI_Client.java*), il processo che realizza l'interazione con l'utente **proponendo ciclicamente i servizi** che utilizzano i due metodi remoti, e stampa a video i risultati, fino alla fine del file di input da tastiera.

Parte C

Progettare un servitore multiservizio (usando obbligatoriamente una select) che consenta di effettuare le operazioni remote per:
- **sostituire tutte le occorrenze di un carattere con un altro carattere all'interno di un file**,
- **trasferimento di tutti i file di un direttorio i cui nomi relativi iniziano con un certo prefisso dal server al client,**

utilizzando un'**unica connessione per ogni sessione cliente**.

Più in dettaglio:
- Il **client_stream** è organizzato come un **processo filtro ciclico fino alla fine del file di input** e realizza la funzionalità di **trasferimento di tutti i file di un direttorio i cui nomi iniziano con un certo prefisso dal server al client** utilizzando **socket stream e un'unica socket**.
  Per ogni richiesta, il client chiede all'utente e invia al server il nome del direttorio e il prefisso, quindi riceve i file e li salva sul file system locale.
- Il **client_datagram** è organizzato come un **processo filtro ciclico fino alla fine del file di input** e realizza la funzionalità di **sostituzione di tutte le occorrenze di un carattere con un altro carattere all'interno di un file remoto** utilizzando **socket datagram**.
  Per ogni richiesta, il client invia il nome del file e i due caratteri e riceve l'esito del numero di sostituzioni stampandolo a video.
- Il **server** principale unico discrimina le richieste utilizzando la primitiva select: il **server gestisce in modo parallelo** la funzionalità di trasferimento di tutti i file di un direttorio i cui nomi iniziano con un certo prefisso dal server al client; mentre la funzionalità di sostituzione di tutte le occorrenze di un carattere con un altro carattere all'interno di un file remoto **può essere realizzata in modo o seriale o parallelo**: si motivi la scelta fatta.
  Per ogni richiesta di **trasferimento di tutti i file di un direttorio i cui nomi relativi iniziano con un certo prefisso dal server al client**, il figlio riceve il nome del direttorio e il prefisso, quindi invia nome e contenuto di tutti i file di quel direttorio che corrispondono al prefisso cercato al client.
  Per ogni richiesta di **sostituzione di tutte le occorrenze di un carattere con un altro carattere all'interno di un file remoto** il figlio riceve il nome del file e i caratteri, quindi effettua l'operazione di sostituzione sul file locale e invia la risposta al client: un intero positivo che indica il numero di sostituzioni effettuate (>=0) in caso di successo, -1 in caso di problemi.