Università degli Studi di Bologna
Scuola di Ingegneria

Corso di

Reti di Calcolatori T

Esercitazione 5 (proposta)

Server Multiservizio: Socket C con select

Luca Foschini

Michele Solimando

Anno accademico 2016/2017

PROGETTO DI UN SERVER MULTISERVIZIO

Sviluppare un'applicazione C/S che fornisca due servizi. Il primo servizio **conta le occorrenze di un carattere in un file** presente sul server remoto, il secondo servizio, **raddoppia le occorrenze di un carattere in un file**.

Il primo servizio viene realizzato utilizzando **socket senza connessione (datagram)**, mentre il secondo utilizzando **socket con connessione (stream)**. In particolare, si dovranno realizzare due client, e un server unico multiservizio (uso di select).

Dettagli Client

Il **primo Client**, chiede ciclicamente all'utente il nome del file e il carattere, invia al server la richiesta, e attende il pacchetto con l'esito dell'operazione che stampa a video.

Il **secondo Client**, chiede ciclicamente all'utente il nome del file e il carattere, invia al server la richiesta e riceve il file con il raddoppio delle occorrenze del carattere e lo stampa a video.

Server Multiservizio: parte datagram

Il **Server** discrimina i due tipi di richiesta utilizzando la primitiva **select**.

Le richieste di **contare le occorrenze di un carattere** vengono gestite in maniera sequenziale diretta dal server senza introdurre una connessione e usando una **socket datagram**.

Il server riceve il datagramma con il nome del file e il carattere: se il file esiste, conta le occorrenze del carattere all'interno del file e invia al client l'intero positivo corrispondente; altrimenti, in caso di errore, invia un intero negativo, ad esempio se il file non esiste.

Dettagli Server: parte stream

Le richieste per ottenere **il raddoppio di un carattere in un file**, vengono gestite in maniera concorrente multiprocesso con un processo per ogni richiesta. Il server riceve il nome del file e il carattere quindi effettua le modifiche e rispedisce il file modificato al client.

Il server **NON salva il file in locale**, ma agisce direttamente sul flusso di input, ri-direzionando la scrittura in output del file opportunamente modificato.

Proposta di estensione: Realizzazione Shell Remoto

Si vuole sviluppare un semplice **shell remoto** per l'esecuzioni di comandi sul nodo server.

In particolare, si vuole realizzare un'applicazione C/S che fornisca **due servizi**, il **primo** realizzato utilizzando **socket senza connessione** (**datagram**), mentre il **secondo** utilizzando **socket con connessione** (**stream**).

Si dovranno realizzare **due client**, e un **server unico multiservizio** (uso di select).

Primo servizio: socket datagram

Il primo servizio realizza l'**esecuzione remota di un comando** e il **recupero del valore di ritorno**. Si prevede un ciclo di:

- Invio da parte del client del **nome del comando** da eseguire e della **stringa degli argomenti** del comando;
- **Esecuzione remota del comando** sul server;
- stampa del **valore di ritorno** del comando sul client.

Il secondo servizio realizza l'**esecuzione di un comando remoto** e il **recupero dell'output del comando** e prevede:

- Invio del client del comando da eseguire e della stringa con **gli argomenti del comando**;
- **Esecuzione remota del comando** sul server, con **ridirezione dello standard output** del comando sulla socket aperta con il client;
- **Stampa** a video dell'**output del comando** sul client.

Note realizzative: gestione figli

Suggerimento: Il **primo servizio** può essere gestito da **un processo figlio** con **attesa sincrona di terminazione**. Per ogni richiesta ricevuta, il server (padre) genera un **processo figlio** per l'esecuzione del comando richiesto (`fork`), recupera il process id (pid) del figlio generato e si mette in attesa del risultato di tale figlio effettuando una `waitpid`. Alla **terminazione del figlio**, il padre recupera il **valore di ritorno del comando** e lo spedisce al client; e poi si mette in attesa di una nuova richiesta da servire in ordine.

Consiglio: si consulti il manuale *(`waitpid`)* per verificare a quale valore viene impostato il parametro di output *(`int *status`)* a fronte della terminazione di un figlio (in questo caso alla terminazione del comando eseguito dal figlio tramite l'invocazione dell'`exec`).

In particolare, esistono alcune macro per verificare le condizioni di terminazione del processo, vedi `WIFEXITED` e `WEXITSTATUS`.

Note realizzative: gestione figli (ancora)

Il **secondo servizio** viene gestito da un **processo figlio senza attesa di terminazione del padre** (**gestione multiprocesso**): per ogni richiesta ricevuta il server genera un processo figlio. Il padre, dopo aver lanciato il figlio, si mette in attesa di nuove richieste.

Il figlio si occupa della ridirezione del **canale di output** sulla socket aperta con il client e dell'**esecuzione del comando richiesto** (exec).

Consegna

Chi vuole può inviare lo svolgimento ai docenti, con lo strumento specificato sul sito del corso.