Università degli Studi di Bologna
Facoltà di Ingegneria

Corso di
Reti di Calcolatori T

Esercitazione 4 (proposta)
Server Multiservizio: Socket C con select

Luca Foschini

Anno accademico 2013/2014

PROGETTO DI UN SERVER MULTISERVIZIO

Sviluppare un'applicazione C/S che fornisca due servizi. Il primo servizio **conta** i **caratteri**, le **parole** (si usi la definizione di parola data nell'esercitazione 1) e le **linee** di un **file di testo** presente sul server remoto. Il secondo servizio, ricevuti un **nome di direttorio** e un suffisso (ad esempio ".txt"), **restituisce la lista con i nomi di file** (presenti nel direttorio remoto) **il cui nome finisce col suffisso richiesto**.

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 di file, invia al server la richiesta, e attende il pacchetto con l'esito dell'operazione (i tre contatori interi) che stampa a video.

Il **secondo Client**, chiede ciclicamente all'utente il nome della directory remota e il suffisso, invia al server la richiesta, riceve la lista dei nomi di file che terminano col suffisso richiesto e la stampa a video.

Server Multiservizio: parte datagram

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

Le richieste di **contare** i **caratteri**, le **parole** e le **linee** di un **file di testo** 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**: se il file esiste, conta i caratteri, le parole e le linee all'interno del file e invia al client i tre interi corrispondenti; altrimenti, in caso di problemi, ad esempio se il file non esiste, invia un segnale d'errore.

Dettagli Server: parte stream

Le richieste per ottenere la **lista dei file di un direttorio remoto i cui nomi terminano per un suffisso**, vengono gestite in maniera concorrente multiprocesso con un processo per ogni richiesta. Il server riceve il nome del direttorio e il suffisso: se il direttorio richiesto esiste, lo apre e restituisce al client la **lista dei file** il cui nome contiene il suffisso richiesto; altrimenti, in caso di errore, restituisce un carattere speciale per notificare tale situazione al client, ad esempio se il direttorio non esiste.

Per **l'apertura** e **la lettura di un direttorio** in C, si utilizzino le funzioni **`opendir`** e **`readdir`**; per maggiori informazioni sull'uso di tali funzioni si veda il manuale in linea: **`man opendir`** e **`man readdir`**.

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 via email lo svolgimento dell'estensione ai docenti, in modo da discutere la soluzione ed eventualmente pubblicarla sul sito del corso