DOCUMENTAZIONE DEL PROGETTO DI RETI INFORMATICHE 2022/2023

A cura di Lorenzo Menchini, corso di laurea in Ingegneria Informatica

VISIONE GENERALE DEL PROGETTO:

il progetto è stato affrontato applicando le conoscenze maturate durante il corso di reti informatiche dell'anno accademico 2022/23, sfruttando anche le conoscenze di programmazione pregresse.

Tramite l'I/O multiplexing è stato possibile implementare la programmazione distribuita necessaria per realizzare il progetto, in cui è stato scelto di usare principalmente il protocollo text e di appoggiarsi ai file, coadiuvati da alcune strutture dati, delle quali è stato fatto un utilizzo minore.

SCELTE E DECISIONI TECNICHE:

come scritto sopra, le strutture dati sono state pensate con l'obiettivo di essere soltanto degli appoggi secondari per il server, garantendo ai file una funzione centrale.

Le motivazioni che hanno condotto all'utilizzo dei file sono principalmente 2:

la semplicità con cui è possibile visualizzarne il contenuto, cosa viene salvato e in che modo (assieme alla possibilità di testare casi particolari, riempendo come meglio crediamo i file prima dell'effettivo utilizzo da parte del programma)

la possibilità di utilizzare un oggetto (se pur con accessi in memoria secondaria, quindi costosi) in cui è disponibile uno spazio "dinamico". Certamente questo concetto di dinamicità poteva esser reso utilizzando una struttura dati a lista, ma sarebbe venuta meno la semplicità di stampa e di inizializzazione descritta sopra, per tanto la conclusione è stata l'utilizzo del file.

Venendo ora ai dispositivi:

Per quanto riguarda i client device questi sono stati concepiti nell'ottica del maggior rispetto possibile dell'architettura client server: i client altro non fanno che inviare i dati che ricevono da terminale, raramente applicando modifiche ai valori che questi raccolgono.

Nel caso della find ciò che si fa è andare a controllare nel relativo file (prenotazioni.txt) se la prenotazione per quel giorno, data e orario esiste già: in tal caso allora sarà premura del server allestire un buffer contenente tutti i tavoli disponibili escluso quello che già compare nel file di prenotazione.

La funzione book, quindi, non fa altro che flaggare (dato che la maggior parte dei controlli sono stati attuati nella Find) un particolare indice di questa lista, sul quale il server altro non farà che riportare l'interesse nella prenotazione sia su file (prenotazioni e prenotazioniCodici.txt) che su struttura dati. Il controllo che ha maggiore importanza nella book ha l'obiettivo di verificare che non siano avvenute delle "prenotazioni parallele", ossia situazioni in cui è connesso più di un client device: in questo caso la lista di prenotazioni fornita dalla find (scegliendo chiaramente stesso giorno, numero di persone e orario) sarà la stessa, ed è qui che la book deve garantire la consistenza andando a vedere che questo tavolo non compaia già nel file delle prenotazioni effettuate.

Per quanto riguarda i table device ed i kitchen device l'intera complessità è gestita lato server, infatti, anche il file menu è contenuto nel server e viene inviato al table device richiedente. Per quanto riguarda il comando help, non essendo notevole dal punto di vista della dimensione e dell'utilità contenutistica, ho preferito utilizzare delle stampe lato client, senza fare un file apposito.

La comanda viene gestita lato server andando a modificare il file "Comande.txt", in cui viene riportata la comanda, la prenotazione che la richiede (il tavolo relativo alla prenotazione viene ricavato mediante funzione getTable() ed è necessario per le stampe successive) il codice comanda assieme alla quantità ed il socket relativo al kitchen device che gestisce questa comanda, assieme al suo stato, espresso con un intero da 0 a 2 (attesa, preparazione, servizio).

Questo file viene poi controllato ed eventualmente modificato dai kitchen device, che a seguito di notifica prendono il controllo di una comanda, la preparano e la servono (andando dunque a modificare il valore di stato, dentro Comande.txt)

GESTIONE GENERALE E PROTOCOLLI USATI

Come accennato sopra, il modo in cui il progetto è stato realizzato cercava il più possibile di rispettare l'architettura client-server,

cercando sempre di spostare la complessità all'interno del server, riducendo quindi il processo client ad un file dove si inviano dati presi da terminale e si ricevono dati elaborati dal server.

Guardando il codice del table device o del kitchen device questo fatto risulta evidente, le funzioni "show" e "ready" inviano e ricevono dati, di maggior costo computazionale è invece il controllo del codice prenotazione, dato che in caso di errata immissione, doveva esser chiesto nuovamente.

Il modo con cui ho gestito la mutua esclusione è stato tramite le send e le receive, bloccanti; mentre il protocollo usato nella gran parte delle aree del progetto è stato il text protocol, che ho trovato di maggior aiuto in quanto ho concepito un server in modalità verbose, che quindi risponde alle richieste dei device stampando a terminale cosa sta facendo, i vantaggi sono molteplici:

- Prima su tutti la maggior facilità nel debug e nella comprensione del codice, con il text protocol è stato infatti facile mandare intere stringhe di testo dal server al client e viceversa, una volta ricevute è stato particolarmente facile stamparle e adoperarle nel migliore dei modi
- Per come è stato concepito il dispositivo client generico, la sua funzione principale è mandare e ricevere stringhe di testo. Di fronte ad una condizione di questo genere la scelta del text protocol si rivela la migliore.