PROGETTO DI RETI INFORMATICHE

A cura di Tommaso Falleni (Mat. 564369) - 2022/2023

Descrizione generale

L'applicazione per la “prenotazione tavoli e gestione delle comande di un ristorante” implementa il paradigma client-server ed utilizza IO multiplexing con socket di tipo TCP bloccanti.

Le componenti dell’applicazione: Client, Kitchen Device, Table Device, comunicano esclusivamente con il server scambiando messaggi attraverso socket bloccanti.

Al fine di garantire una corretta gestione della lettura da socket, è stato impiegato il meccanismo della primitiva Select(), sia per le componenti che per il Server.

La primitiva select() è utilizzata per monitorare l'attività su uno o più socket e restituire il numero di socket pronti per la lettura, al fine di eliminare il rischio di blocco del processo in caso di socket non pronti. Inserendo i socket in un set, è possibile selezionare facilmente quelli pronti per la lettura e procedere con la lettura solo in tale circostanza.

Infine, è stato scelto un’implementazione con socket di tipo bloccanti con protocollo di trasporto TCP, considerando il requisito principale dell’applicazione: l’affidabilità della comunicazione e l’integrità dei messaggi, prioritaria rispetto alla velocità.

E’ stata scelta un'implementazione basata su un protocollo text, che risulta particolarmente comodo per lo scambio di informazioni prevalentemente testuali.

Come fase iniziale il Client, Kitchen Device e il Table Device dovranno instaurare una connessione TCP con il Server, seguito dall’invio di un codice identificativo (C, K o T) per identificarsi. Questo, servirà al Server per sapere quanti e quali dispositivi sono collegati con la possibilità di limitarne la connessione.

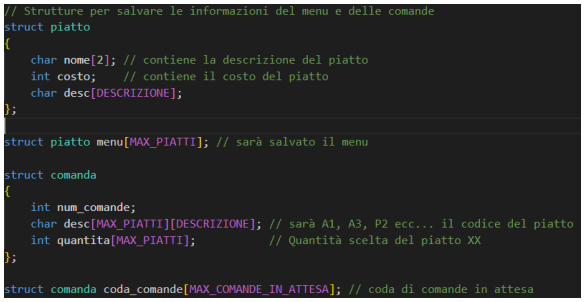
Client

Quando l'utente avvia l'esecuzione del comando “find cognome persone data ora”, il Client esegue un doppio controllo sulla data. Verifica che la data inserita sia valida e che sia futura rispetto al momento della richiesta; Se il controllo ha avuto successo viene inoltrato al server, il quale restituirà una lista dei tavoli selezionabili in base alla disponibilità ed alla quantità di persone indicate.

Per terminare la prenotazione l’utente dovrà selezionare il tavolo che preferisce con il comando “book N°”. Il server controllerà se nel frattempo è stato occupato il tavolo scelto. Se la prenotazione sarà andata a buon fine, il Client riceverà il codice da inserire nel Table Device.

Table Device

Avviato il Table Device, prima di poterlo utilizzare è richiesto una sorta di Login con codice identificativo. Superato il controllo, si richiede il Menù che verrà salvata in un array di Struct Menu.



```
// Strutture per salvare le informazioni del menu e delle comande
struct piatto
{
    char nome[2]; // contiene la descrizione del piatto
    int costo;    // contiene il costo del piatto
    char desc[DESCRIZIONE];
};

struct piatto menu[MAX_PIATTI]; // sarà salvato il menu

struct comanda
{
    int num_comande;
    char desc[MAX_PIATTI][DESCRIZIONE]; // sarà A1, A3, P2 ecc... il codice del piatto
    int quantita[MAX_PIATTI];           // Quantità scelta del piatto XX
};

struct comanda coda_comande[MAX_COMANDE_IN_ATTESA]; // coda di comande in attesa
```

Questa scelta è stata fatta per permettere di avere memorizzato il menu senza dover interpellare il Server.

Stesso meccanismo per le comande, che saranno utili per il conto finale.

Prima di poter eseguire una comanda è necessario richiedere il Menu così da averlo salvato e utilizzarlo per il conto finale.

Kitchen Device

Il kitchen Device con il comando “take” ottiene la comanda nello stato “In attesa” da più tempo ricevuta e la salva nella struttura coda_comande.

Server

Le informazioni relative al menu e ai tavoli presenti nel ristorante vengono recuperate da file per semplificare le modifiche.

Quando viene effetuata una richiesta “find”, il server controlla nella cartella Prenotazioni/ se è presente il file con il nome della data selezionata. Questo consente un’implementazione più realistica dell’applicazione, in cui possono esserci prenotazioni per giorni diversi con la possibilità di aggiungere funzionalità di “ordini storici”. Dopo aver inviato i tavoli disponibili ed averli salvati in un Array con indice *i*, il socket del client che ha fatto richiesta, il server controlla se il tavolo scelto con la “book” sia stato prenotato nel frattempo.

Per rendere il server più veloce per la richiesta del “conto”, il server controlla solo se il tavolo richiedente ha piatti in preparazione o in attesa. Se questo controllo ha esito negativo, il Table Device può calcolare il conto. Pertanto, con il comando “menu”, il menu viene salvato nella memoria del Table Device.

Il codice formato da “tavolo-data-ora” ha un doppio scopo: viene utilizzato per ricercare il nome del file nella cartella Prenotazioni e per avere il numero del tavolo, tutto in un unico codice.

Il codice nel file Prenotazioni/giorno viene modificato quando un Table Device si identifica cambiando la prima lettera da T ad X. In questo modo se un altro utente prova a connettersi al tavolo T con lo stesso codice, viene rifiutato.

```
struct comanda
{
    char tav_num[5]; // numero del tavolo da cui proviene la comanda
    int num_comande;
    char desc[MAX_PIATTI][DESCRIZIONE]; // sarà A1, A3, P2 ecc... il codice del piatto
    int quantita[MAX_PIATTI];           // quantita
    char stato;                         // stato della comanda: a (attesa), p (preparazione) o s (servizio)
    uint16_t id;
    int td_assegnato; // socket di comunicazione connesso a quello del td a cui la comanda è stata assegnata
    int kd_assegnato; // socket di comunicazione connesso a quello del kd a cui la comanda è stata assegnata
};

struct comanda coda_comande[MAX_COMANDE_IN_ATTESA];    // coda di comande in attesa/ preparazione
struct comanda comande_servite[MAX_COMANDE_IN_ATTESA]; // coda di comande in servizio
```

La lista di comande è stata suddivisa in due Array di struct comanda.

Uno contiene le comande servite e l’altro contiene le comande in attesa e in preparazione.

La lista delle comande è stata divisa in due Array per semplificare la ricerca.

Se si verificano errori di comunicazione con un Table/Kitchen Device o di arresto forzato dello stesso, il server chiude il socket di comunicazione interessato ed adotta procedure differenti a seconda del tipo di dispositivo coinvolto:

- Se si tratta di un Table Device, ripercorre la lista dell’array coda_comande. Se le comande del T.D. erano state assegnate a qualche K.D. il server manda un messaggio di avviso e rimuove tutte le comande in attesa/ preparazione relative al tavolo;
- Se si tratta Kitchen Device e ha delle comande da svolgere, avvisa il T.D. che le comande sono bloccate e gli ordini che aveva preso verranno messi nello stato “in attesa” affinchè un altro K.D. possa prenderle per finire il servizio.