Documentazione Progetto Reti Informatiche 2022/2023

Vito Gabriele Marino – Mat. 564394

Il paradigma scelto per l'applicazione è ovviamente quello client server, dove il server, interagendo con tre tipi differenti di client, gestisce il complesso delle strutture dati utili a realizzare l'astrazione di un ristorante.
Tali strutture dati sono inizializzate in fase d'avvio dal server, recuperando i dati dagli appositi file, contenuti nella directory "data".
Tra questi, solo il file "*reservations.txt*" verrà aggiornato in fase d'arresto, eliminando le prenotazioni scadute ed aggiungendo eventualmente le nuove.
Si presuppone per semplicità che i dati contenuti in questi file vengano mantenuti ed eventualmente manipolati in maniera consistente, mantenendo l'apposito formato.
In particolare, per motivi implementativi, è essenziale che gli ID dei tavoli all'interno del file "*tables.txt*" vengano assegnati in maniera crescente a partire da 1. Un approccio simile è stato utilizzato nella gestione del menù.
Sia i file contenenti dati, sia i vari messaggi scambiati tra client e server sono gestiti in modalità testo.
Questa scelta è stata dettata dal fatto che la maggior parte dei dati in transito sono appunto delle stringhe di testo (piatti del menù, dati relativi alle prenotazioni, informazioni sui tavoli ecc..), dunque ha consentito di semplificare il lavoro a patto di trasferire a volte quantità maggiori di dati (ad esempio nel caso in cui a transitare sono degli interi su più cifre).

Per quanto riguarda il server, al fine di servire in maniera concorrente più client ed ascoltare lo "stdin" è stata utilizzata la tecnica dell'I/O multiplexing.
A livello applicativo sono stati implementati diversi protocolli per gestire la comunicazione.
Di base, l'invio e la ricezione dei messaggi vengono effettuati tramite le funzioni ***send_msg*** e ***recv_msg***, definite all'interno del file "i*ncludes/utility.c*", che si occupano rispettivamente di inviare e ricevere la lunghezza effettiva del messaggio prima di inviare/ricevere il messaggio stesso.
Questo ha semplificato notevolmente il lavoro.

Al fine di implementare in maniera efficace la chiusura del server, avvisando come richiesto i table e kitchen device connessi, è stato implementato un piccolo protocollo di autenticazione, utile per distinguere il tipo di client connesso e salvare l'identificatore del socket associato in un'apposita struttura dati, in modo da poterlo appunto recuperare per notificare la chiusura imminente del server.
Il protocollo è molto semplice, una volta connesso il client manderà al server una stringa contenente l'identificatore "cl, td o kd". Il server effettua le operazioni necessarie e risponde con "OK" o, nel caso del table device, con "OK_TID", dove TID è l'id del tavolo assegnato al table device.
In mancanza di specifiche e per semplicità, ho infatti supposto che ad ogni table device venga assegnato uno specifico tavolo, tra quelli liberi, dal server stesso al momento della connessione. Inoltre, sempre per semplicità, non è stata implementata alcuna forma di verifica o autenticazione dell'utente basata sulle prenotazioni. Si può supporre che il table device svolga semplicemente il suo compito, ossia essere associato ad un tavolo e gestire le

ordinazioni giornaliere provenienti da esso.

Per quanto riguarda l'implementazione dei vari comandi, quando è necessario inviare una serie di dati (es: lista dei tavoli liberi), il server invia una serie di messaggi, uno per ogni elemento della lista, segnalando la fine inviando la stringa "END\0".
In alcuni casi, come nel comando "comanda", la gestione di alcuni possibili errori è gestita lato server (es: se viene richiesto un piatto errato all'interno di una comanda, questo viene scartato).

Per quanto riguarda le strutture dati, c'è un array principale di strutture "Table", all'interno dei quali sono presenti le liste relative alle prenotazioni e alle comande in servizio o in preparazione. Quest'ultima scelta consente di effettuare più rapidamente il calcolo del conto.
Riguardo le comande, esistono altre due liste, una per le comande in attesa ed una per le comande archiviate (vengono mostrate dal comando stat e sono sostanzialmente tutte le comande consumate in giornata).
C'è poi un array contenente i piatti del menù ed uno che mantiene la lista dei KD connessi.
I TD sono associati ai tavoli, tramite un campo "td" all'interno della struttura tavolo dove è salvato l'identificatore del socket associato.

La seguente tabella riassume invece i formati dei messaggi di rete:

Comando	Messaggio di rete	Commenti
"find name num dd-mm-YYYY hh"	"find name num timestamp"	-
"book i"	"book TID name num timestamp"	Utile nel caso in cui un altro client prenoti il tavolo per quell'orario.
"menu"	"menu"	-
"comanda {P1-i} {P2-i}...	"comanda TID {P1-i} {P2-i}..."	Il server scarta i piatti non in menù.
"conto"	"conto TID"	Il server scarta le comande ancora in attesa, calcola il conto e resetta la lista delle comande del tavolo.
"take"	"take"	-
"show"	"show"	All'interno della struttura "comanda" c'è un apposito campo "kd" per identificare il kd a cui è associata.
"ready comi-Tj"	"ready i j"	-
"stop"	"CLOSE\0"	Quando il server sta per arrestarsi, invia il messaggio "CLOSE\0" a tutti i td ed i kd connessi.