UNIVERSITÀ DI PISA

CORSO DI LAUREA IN INGEGNERIA INFORMATICA
SPECIFICHE PROGETTO A.A. 2022/2023

Reti Informatiche, cod. 545II, 9 CFU

Prof. Giuseppe Anastasi, Ing. Francesco Pistolesi

Si richiede di progettare un'applicazione distribuita basata sul paradigma client-server che implementi un sistema di prenotazione tavoli e gestione delle comande di un ristorante.

1. VISIONE D'INSIEME

Considerare uno scenario in cui un ristorante è dotato di sale che ospitano tavoli. I tavoli possono essere prenotati accedendo a un ***server*** che implementa un *servizio di prenotazione*.

Ogni tavolo è dotato di un ***table device*** tramite il quale i clienti possono inviare comande alla cucina e possono verificarne lo stato di avanzamento.

In cucina, sono posizionati vari ***kitchen device*** tramite i quali i cuochi possono accettare le comande, leggerne il contenuto e impostarne lo stato di avanzamento.

Il *servizio di gestione delle comande* è svolto dallo stesso server che implementa il servizio di prenotazione: il server è quindi multiservizio.

Un cliente può prenotare un tavolo usando un ***client*** che si collega al servizio di prenotazione offerto dal server, e invia una richiesta di prenotazione composta da cognome, numero di persone, e un time slot (data e ora in cui si recherà al ristorante). Quando il server riceve una richiesta di prenotazione, seleziona le sale in cui c'è disponibilità, cioè le sale in cui c'è almeno un tavolo libero per il time slot scelto dal cliente, il cui numero di posti sia almeno uguale al numero di commensali. Il server invia poi al client le informazioni sui tavoli disponibili in ogni sala. Queste informazioni possono contenere, opzionalmente, dettagli sull'ubicazione dei tavoli liberi, per esempio per i tavoli con affaccio verso l'esterno.

Il cliente sceglierà quindi un tavolo e l'applicazione, lato cliente, costruirà un messaggio di prenotazione, ricevuto il quale il server finalizzerà la prenotazione. Il server salverà la prenotazione e il timestamp in cui è stata finalizzata[1]. Se la prenotazione va a buon fine, il server invia un codice di prenotazione, unitamente al numero di tavolo e di sala associati alla prenotazione.

[1] I dati possono essere salvati su file, oppure mantenuti in memoria.

Quando un gruppo di clienti arriva al ristorante, si siede al tavolo prenotato e inizia a interagire col relativo table device che, per prima cosa, richiede il codice di prenotazione. Il table device è inizialmente bloccato e si sblocca solo se si inserisce il codice di prenotazione corretto. Per validare il codice di prenotazione, il table device comunica col server. Se il codice di prenotazione è valido, il table device mostra il menu, altrimenti chiede nuovamente un codice di prenotazione.

Quando il table device riconosce il codice di prenotazione valido, mostra il menu. Nel menu, ogni piatto ha un nome e un codice. Per esempio, “A1”, “A2”, eccetera possono essere i codici di due antipasti; P1, P2, eccetera i codici dei primi piatti, e così via. Il menu è fissato giornalmente (prima dell'apertura del ristorante) e non cambia per tutto il giorno.

Dopo aver visualizzato il menu, il cliente può inviare una comanda al server, contattando il servizio dedicato. La comanda è costituita da un insieme di piatti e dalle corrispondenti quantità. È possibile inviare più comande all'interno dello stesso pasto. Con una comanda, si può per esempio decidere di ordinare gli antipasti per tutti, oppure gli antipasti per metà dei commensali e per gli altri far partire subito i primi piatti. A seguire, ci saranno altre comande (per lo stesso tavolo) fino alla chiusura del pasto, segnalata dal cliente con un apposito messaggio di richiesta del conto, inviato al server. Le comande inviate restano visibili sul table device in stato "*in attesa*" finché un cuoco non se ne prende carico.

Quando il server riceve una comanda, viene inviata una notifica[2] a tutti kitchen device. Ogni kitchen device è gestito da un cuoco della cucina. Il primo cuoco che accetta la comanda dovrà cucinare tutti i piatti che la compongono. Quando una comanda è accettata da un kitchen device non può più essere accettata da altri kitchen device. Un kitchen device può accettare comande a partire da quella che si trova nello stato "*in attesa*" da più tempo rispetto alle altre. Dopo l'accettazione, il kitchen device mostra a video i piatti che compongono la comanda, e invia un messaggio al server, il quale invia un messaggio al relativo table device che visualizza lo stato della comanda come "*in preparazione*".

Quando tutti i piatti della comanda sono pronti, il cuoco lo segnala tramite il kitchen device il quale invia un messaggio al server che lo inoltra al table device: la comanda passa nello stato "*in servizio*".

Questo meccanismo si ripete fino al termine del pasto (richiesta del conto). Le informazioni relative alle comande dei vari pasti possono essere mantenute in memoria, o scritte su file.

Il progetto può essere sviluppato su un'unica macchina virtuale, utilizzando tanti terminali quanti sono i device inseriti nello scenario applicativo. Per esempio, si può immaginare di avere un minimo di due table device, due kitchen device, un server e un client. I relativi processi, per semplicità, possono essere mandati tutti in esecuzione su localhost.

Il formato dei messaggi, la loro codifica, e i protocolli di scambio dei messaggi fra i vari dispositivi dovranno essere scelti da ogni studente, coerentemente con lo scenario applicativo descritto, e costituiranno la “firma” di ogni implementazione dell'applicazione distribuita da progettare. Il server può essere implementato come si desidera, utilizzando una delle tecniche viste a lezione.

Tutte le scelte implementative metteranno in luce l'unicità di ogni progetto e saranno oggetto di discussione in sede di orale. Spiegare nella documentazione le motivazioni che hanno condotto alle scelte effettuate, discutendone sinteticamente i pregi e i difetti, in maniera critica e sistematica.

[2] La notifica può semplicemente essere un '*' mostrato a video. Si può pensare di stampare a video tanti asterischi quante sono le comande ricevute. Per esempio, se lo stato è '****', ci sono quattro comande in attesa di accettazione.

2. DETTAGLI IMPLEMENTATIVI

L'applicazione distribuita da sviluppare deve implementare quanto descritto nel Paragrafo 1. Le scelte progettuali devono essere spiegate in una **relazione di al più 2 pagine**, usando anche figure e schemi intuitivi (anche tracciati a mano, fotografati e inglobati nella documentazione). Nella relazione, devono essere messi in luce pregi e difetti delle scelte fatte sotto i punti di vista tipici delle applicazioni distribuite descritti durante il corso: quantità di traffico generato, possibili bottleneck, criticità, scalabilità, e così via.

2.1 SERVER

Il **server** è mandato in esecuzione come segue:

```
./server <porta>
```

dove **<porta>** è la porta associata al server.

Appena mandato in esecuzione, il processo **server** mostra a video una guida dei comandi. Dopo la digitazione di un comando da standard input, oppure dopo la ricezione di un messaggio dalla rete, il server mostra a video cosa sta facendo, con un formato a piacere[3].

I comandi accettati dal server sono:

stat ***table|status***
senza parametri, restituisce lo stato di tutte le comande giornaliere; se si fornisce il numero di tavolo *table* restituisce tutte le comande relative al tavolo *table* relative al pasto in corso; se si fornisce uno stato *status* fra 'a', 'p', ed 's' (associati nell'ordine a "in attesa", "in preparazione", e "in servizio") il comando restituisce tutte le comande di pasti in corso nello stato *status*, per ogni tavolo.

Esempio di esecuzione che richiede le comande al tavolo T1[4]:

```
stat T1⏎
com1 <in servizio>
A1 2
A2 1
P1 1
com2 <in preparazione>
S1 4
```

Esempio di esecuzione che richiede le comande in preparazione al tavolo T1:

```
stat p⏎
com2 T1
S1 3
S2 1
com1 T2
A1 2
com3 T3
D3 2
```

L'uscita indica tre comande ai tavoli T1, T2, e T3. La prima comanda, com2 al tavolo T2, è composta da tre unità del secondo piatto S1 e un'unità del secondo piatto S2.

[3] Si può pensare che il server lavori in modalità *verbose*, e informi l'utente continuativamente su ciò che sta facendo.
[4] Se non ci sono comande per il tavolo, il comando non restituisce niente.

stop
il server si arresta. Questa operazione è possibile solo se non ci sono comande attualmente nello stato “in preparazione” o “in attesa”. Prima di arrestarsi, il server invia un messaggio a tutti i table device e a tutti i kitchen device, i quali, a loro volta, avviano la procedura di disconnessione e arresto.

2.2 TABLE DEVICE

Il table device si avvia col seguente comando

```
./td <porta>
```

dove **<porta>** è la porta associata.

All’avvio, i comandi disponibili sono:

- **menu**
- **comanda {<piatto_1-quantità_1>...<piatto_n-quantità_n>}**
- **conto**

La schermata di avvio deve essere come segue

```
****************************** BENVENUTO ******************************
Digita un comando:

1) help          --> mostra i dettagli dei comandi
2) menu          --> mostra il menu dei piatti
3) comanda       --> invia una comanda
4) conto         --> chiede il conto
```

I comandi accettati da tastiera dal table device sono i seguenti:

menu
mostra il menu, cioè l’abbinamento fra codici, nomi dei piatti e relativi prezzi[5].
Esempio di visualizzazione del menu:

```
menu⏎

A1 - Antipasto di terra          7
A2 - Antipasto di mare           8
P1 - Spaghetti alle vongole      10
P2 - Rigatoni all’amatriciana    6
S1 - Frittura di calamari        20
S2 - Arrosto misto               15
D1 - Crostata di mele            5
D2 - Zuppa inglese               5
```

comanda {<piatto_1-quantità_1>...<piatto_n-quantità_n>}
invia una comanda alla cucina.

[5] Considerare i prezzi come numeri interi.

Esempio di invio di una comanda composta da due antipasti A1, un antipasto A2 e un primo piatto P1:

```
comanda A1-2 A2-1 P1-1⏎
COMANDA RICEVUTA
```

conto
invia al server la richiesta di conto. Il server calcola il conto e lo invia al table device, che lo mostra a video.

Esempio di richiesta del conto dove, in ogni riga, è presente il codice del piatto, la quantità, e il subtotale:

```
conto⏎
P1 1 10
P2 1 6
S2 2 30
Totale: 46
```

2.3 KITCHEN DEVICE

Il kitchen device si avvia col seguente comando

```
./kd <porta>
```

dove **<porta>** è la porta associata.

All'avvio, i comandi disponibili sono:

- **take --> accetta una comanda**
- **show --> mostra le comande accettate (in preparazione)**
- **set --> imposta lo stato della comanda**

take
accetta la comanda nello stato di attesa da più tempo.

Esempio di esecuzione del comando che mostra che è stata accettata la comanda com1 del tavolo T1, composta da 2 antipasti A1 e 1 antipasto A2.

```
take⏎
com1 T1
A1 2
A2 1
```

show
visualizza l'elenco delle comande accettate dal kitchen device, e attualmente nello stato "in preparazione".

Esempio di esecuzione del comando che mostra che è stata accettata la comanda com1 del tavolo T1, composta da 2 antipasti A1, 1 antipasto A2 e la comanda com3 del tavolo T2 composta da 2 dessert D2.

```
show⏎
com1 T1
A1 2
A2 1
P1 1
com3 T2
D2 2
```

ready *com*
imposta allo stato “in servizio” la comanda *com*.

Esempio di esecuzione del comando che imposta a “in servizio” la comanda com3 del tavolo T2:

```
ready com3-T2⏎
COMANDA IN SERVIZIO
```

2.4 CLIENT

Il client serve solo a inviare le prenotazioni. Si avvia col seguente comando

```
./cli <porta>
```

dove **<porta>** è la porta associata.

All’avvio, i comandi disponibili sono:

- **find --> ricerca la disponibilità per una prenotazione**
- **book --> invia una prenotazione**
- **esc --> termina il client**

find *cognome persone data ora*
invia una richiesta di disponibilità dove la data è espressa in formato GG-MM-AA e l’ora come HH.

Esempio di esecuzione del comando:

```
find rossi 3 25-05-22 12⏎
```

In risposta, il server potrebbe dare queste tre opzioni:

```
1) T1  SALA1 FINESTRA
2) T4  SALA1 CAMINO
3) T23 SALA2 PORTA_INGRESSO
```

book *opz*
invia una richiesta di prenotazione per l’opzione *opz*

Esempio di esecuzione del comando dove si sceglie la seconda opzione

```
book 2⏎
PRENOTAZIONE EFFETTUATA
```

REQUISITI

- I dati sono scambiati tramite **socket**. Se non si definisce un preciso formato di messaggio, prima di ogni scambio, il ricevente deve essere informato su **quanti byte** leggere dal socket.

- Per gestire le varie informazioni, è possibile usare **strutture dati a piacere**. Gli aspetti che non sono specificati in questo documento possono essere implementati liberamente o ignorati.

- Usare gli **autotools** (comando **make**) per la compilazione del progetto. Usare la breve guida disponibile sul sito Elearn del corso.

- Il codice **deve essere indentato e commentato** in ogni sua parte: significato delle variabili, processazioni, ecc. I commenti possono essere evitati nelle parti banali del codice.

- Va prodotta una **documentazione di NON PIÙ DI 2 PAGINE** in cui spiegare, anche con l'aiuto di figure o grafici, le scelte fatte, i formati di strutture dati e messaggi di rete (campi, numero di byte, ecc.).

- Scrivere uno **script che compili il progetto, mandi in esecuzione il server e faccia il boot di 3 client**, su finestre diverse del terminale. Fare in modo che in ogni peer ci siano alcune informazioni nei file delle comande, in modo da poter testare le varie funzionalità.

CONSEGNA

Il progetto deve essere caricato sul sistema elearn.ing.unipi.it (sulla pagina del corso) **non oltre le 72 ore che precedono il giorno dell'esame**, usando le credenziali di Ateneo per l'accesso. Per ogni appello, sarà creata una nuova sezione per le consegne. In caso di problemi con il portale, si può sottomettere il progetto per email a francesco.pistolesi@unipi.it, **sempre rispettando il requisito delle 72 ore di anticipo**.

VALUTAZIONE

Il progetto è visionato e testato prima del giorno dell'esame. Prima di effettuare la consegna, **testare il codice sulla VM Debian 8 usata durante il corso**. Durante l'esame, sarà può essere richiesta l'esecuzione del programma e saranno fatte domande sia sul codice che sulle scelte fatte. Questa parte compone il giudizio assegnato al progetto.

La valutazione del progetto prevede le seguenti fasi:

1. **Compilazione del codice**
   I device e il server vanno compilati con opzione **`-Wall`** che mostra i vari warning. Non vi dovranno essere warning o errori. L'opzione **`-Wall`** va abilitata anche durante lo sviluppo del progetto, interpretando i messaggi forniti dal compilatore. **Se il progetto non compila, è insufficiente**;

2. **Esecuzione dell'applicazione**
   In questa fase si verifica il funzionamento dell'applicazione e il rispetto delle specifiche;

3. **Analisi del codice sorgente**
   Può essere richiesto di spiegare parti del codice e apportarvi semplici modifiche.

FUNZIONI DI UTILITÀ

Una documentazione di alto livello per le funzioni necessarie per leggere e scrivere file a blocchi possono essere visualizzate al seguente indirizzo:

https://www.programiz.com/c-programming/c-file-input-output