



Azienda Software

Applicazione in C

Università degli Studi di Napoli "Parthenope"

Dipartimento di Scienze e Tecnologie

PROGETTO CORSO DI RETI DI CALCOLATORI

A.A 2024/2025

Emanuele Pacilio Mat: 0124002756

Contents

1	Introduzione	2
1.1	Traccia del progetto	2
1.2	Architettura di comunicazione	2
1.3	Protocollo di comunicazione	3
2	Implementazione	4
2.1	Architettura del sistema	4
2.2	Dettagli implementativi del client	4
2.2.1	Client	4
2.2.2	ClientAdmin	4
2.2.3	ClientCheck	4
2.3	Dettagli implementativi del server	4
2.3.1	ServerL	4
2.3.2	ServerG	4
2.3.3	Portale	5
2.3.4	File licenze	5
3	Manuale Utente	5
3.1	Istruzioni per la compilazione	5
3.2	Istruzioni per lo start e stop dei server	5
4	Simulazione dell'applicazione	7
4.1	Caso 1) Registra Licenza	7
4.2	Caso 2) Validità licenza	7
4.3	Caso 3) Modifica validità licenza	8
4.4	BONUS) stampa file	9

1 Introduzione

1.1 Traccia del progetto

L'obiettivo del progetto è sviluppare un sistema distribuito per la gestione di licenze software. Ogni licenza è identificata da un codice univoco (ID), possiede uno stato di validità e una data di scadenza. Il sistema permette sia la modifica delle licenze da parte dell'amministratore, sia la consultazione e verifica da parte degli utenti.

L'architettura è composta da più entità server collegate tra loro:

- **Server L (Licenze)**: è il server principale del sistema. Mantiene il file delle licenze (`licenze.csv`) e ne gestisce la registrazione, aggiornamento e verifica. È l'unico server che accede direttamente al file delle licenze.
- **Portale**: funge da intermediario tra i client utente e il Server L. Si occupa di inoltrare richieste di registrazione (**REGISTRA**) e di modifica (**UPDATE**) delle licenze. Quindi è responsabile solo delle **operazioni di modifica**.
- **Server G**: è un server di sola consultazione. Riceve richieste di verifica (**CHECK**) e richieste di lista completa delle licenze (**LISTA**) e le inoltra al Server L. Non modifica le licenze, ma si occupa esclusivamente del **recupero dei record**.

Sono presenti tre applicazioni client che interagiscono con il sistema:

- **Client**: consente all'utente di registrare nuove licenze tramite il Portale.
- **ClientCheck**: consente di verificare la validità delle licenze tramite il ServerG
- **ClientAdmin**: consente all'amministratore di visualizzare tutte le licenze e verificare lo stato delle licenze tramite il Server G ed inoltre consente di modificare la validità delle licenze tramite Portale.

1.2 Architettura di comunicazione

L'architettura del sistema è suddivisa in due flussi indipendenti:

- **Flusso Client**: comunica esclusivamente con il **Portale** ed inoltra la richiesta di registrazione licenza al Portale che a sua volta la inoltra al **server L** per salvarla sul file `licenze.csv`.
- **Flusso ClientCheck**: comunica esclusivamente con il **serverG**, inoltra comandi al **Server L** per recuperare le licenze o per controllare la validità di una.
- **Flusso ClientAdmin**: comunica con entrambi i server: il **Server G** per recuperare le licenze e **Portale** per validare/invalidare una licenza.

1.3 Protocollo di comunicazione

La comunicazione avviene su **TCP/IP** tramite socket. Ogni server utilizza una porta dedicata:

- Server L: porta 9000 (gestione licenze)
- Server G: porta 9001 (consultazione licenze)
- Portale: porta 9002 (modifica licenze)

2 Implementazione

2.1 Architettura del sistema

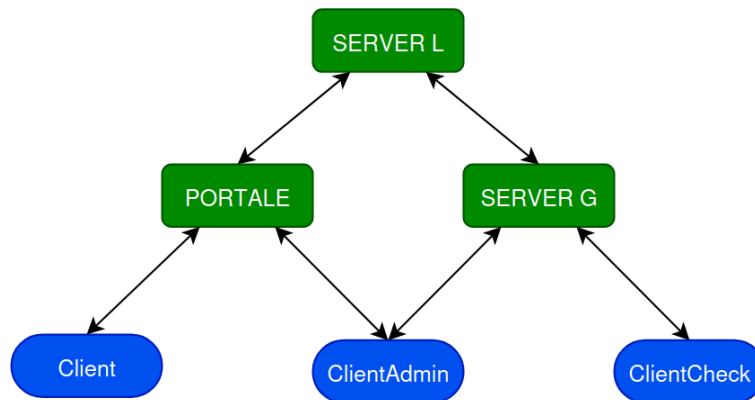


Figure 1: Architettura del sistema

Ogni server usa un modello multi-threading per gestire più connessioni contemporaneamente.

2.2 Dettagli implementativi del client

2.2.1 Client

Il clientR ha il compito di inviare richieste al portale per aggiungere nuove licenze al file.

2.2.2 ClientAdmin

Il ClientAdmin ha il compito di inviare richieste al portale per attivare/disattivare le licenze.

2.2.3 ClientCheck

Il ClientCheck ha il compito di inviare richieste al serverG per verificare la validità delle licenze.

2.3 Dettagli implementativi del server

2.3.1 ServerL

Il ServerL ha il compito di gestire un file di licenze registrate e di rispondere alle richieste dei client in modo concorrente utilizzando thread.

2.3.2 ServerG

Il ServerG ha il compito di interrogare il ServerL per rispondere alla verifica di validità da parte del ClientCheck.

2.3.3 Portale

Il Portale ha il compito di inviare comandi di insert da parte del Client e di update da parte del ClientAdmin delle le licenze verso il ServerL.

2.3.4 File licenze

Il server gestisce il file delle licenze tramite il file csv denominato *licenze.csv*, dove vengono memorizzate tutte le licenze registrate e dove verranno memorizzate tutte le licenze che verranno registrate.

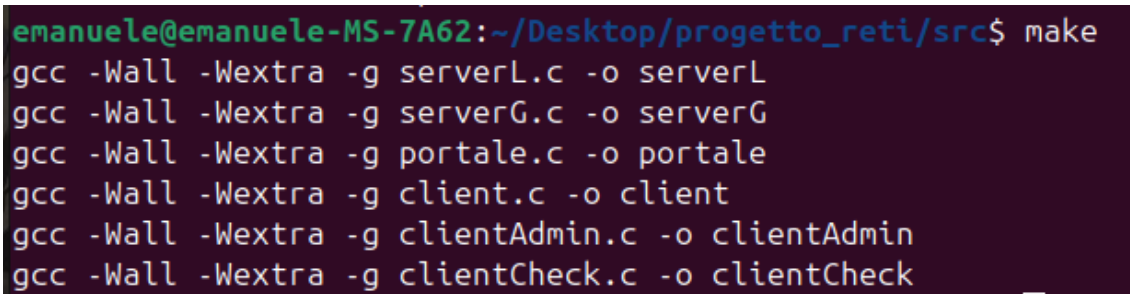
3 Manuale Utente

3.1 Istruzioni per la compilazione

Per la **compilazione** dei file è stato creato un makefile per evitare di compilare a mano ogni file, inoltre sono state programmate in **bash** le operazioni di avvio/stop server e di scelta client. Per compilare i vari progetti possiamo eseguire o l'istruzione **make** oppure se vogliamo compilare e subito far partire i server **make run-all**

make

make run-all



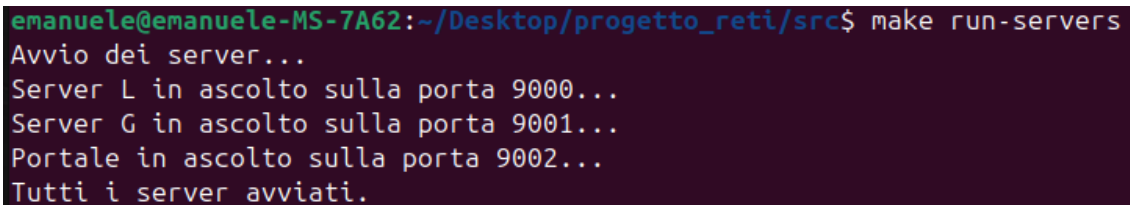
```
emanuele@emanuele-MS-7A62:~/Desktop/progetto_reti/src$ make
gcc -Wall -Wextra -g serverL.c -o serverL
gcc -Wall -Wextra -g serverG.c -o serverG
gcc -Wall -Wextra -g portale.c -o portale
gcc -Wall -Wextra -g client.c -o client
gcc -Wall -Wextra -g clientAdmin.c -o clientAdmin
gcc -Wall -Wextra -g clientCheck.c -o clientCheck
```

Figure 2: compilazione

3.2 Istruzioni per lo start e stop dei server

Compilato il codice bisogna ora far partire i server.

make run-servers



```
emanuele@emanuele-MS-7A62:~/Desktop/progetto_reti/src$ make run-servers
Avvio dei server...
Server L in ascolto sulla porta 9000...
Server G in ascolto sulla porta 9001...
Portale in ascolto sulla porta 9002...
Tutti i server avviati.
```

Figure 3: partenza-server

Qualora volessimo interrompere l'esecuzione dei server

```
make stop-servers
```

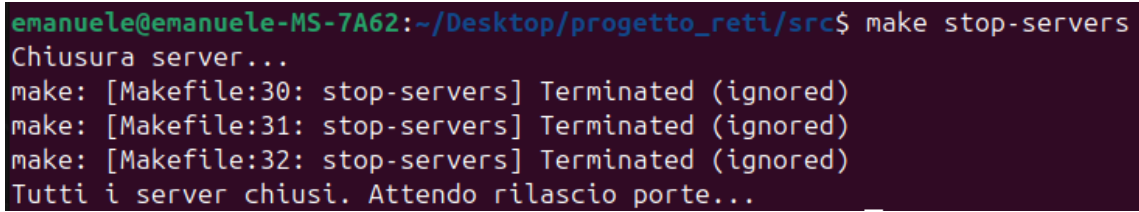
A terminal window with a dark background and light-colored text. The prompt is 'emanuele@emanuele-MS-7A62:~/Desktop/progetto_reti/src\$'. The command 'make stop-servers' has been executed. The output shows 'Chiusura server...' followed by three lines of 'make: [Makefile:30: stop-servers] Terminated (ignored)', 'make: [Makefile:31: stop-servers] Terminated (ignored)', and 'make: [Makefile:32: stop-servers] Terminated (ignored)'. The final line of output is 'Tutti i server chiusi. Attendo rilascio porte...'.

Figure 4: stop-server

Avviati i server scegliamo quale client vogliamo utilizzare

```
make menu-client
```

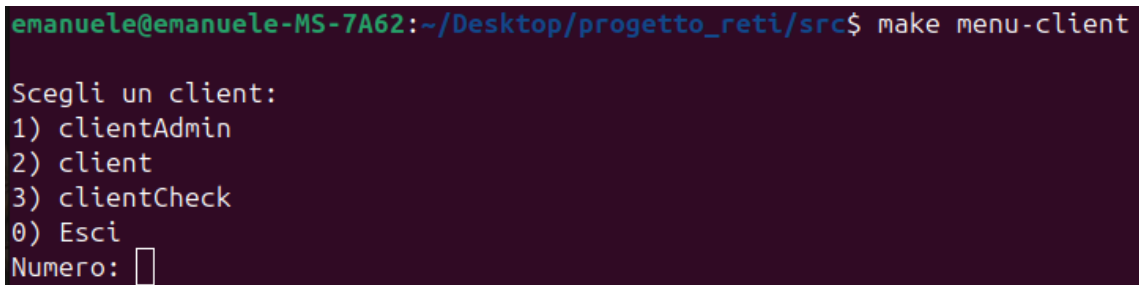
A terminal window with a dark background and light-colored text. The prompt is 'emanuele@emanuele-MS-7A62:~/Desktop/progetto_reti/src\$'. The command 'make menu-client' has been executed. The output shows 'Scegli un client:' followed by a list of options: '1) clientAdmin', '2) client', '3) clientCheck', and '0) Esci'. Below the list, the text 'Numero: ' is followed by a cursor (a small square) indicating where to enter a choice.

Figure 5: menu-client

4 Simulazione dell'applicazione

4.1 Caso 1) Registra Licenza

Scelgo il **client**, seleziono l'opzione **Registra licenza**, inserisco i dati richiesti e qualora l'operazione andasse a buon fine il portale risponde con **OK** e la licenza viene salvata nel file con stato di validità "*valida*" e con data di scadenza pari ad 1 anno dopo dal momento dell'inserimento. Qualora la licenza fosse già registrata il server me lo notifica.

```
=== Registrazione Licenza ===  
Inserisci ID licenza: 200  
Risposta portale: OK
```

Figure 6: Licenza registrata

```
=== Registrazione Licenza ===  
Inserisci ID licenza: 1  
Risposta portale: ERRORE: ID già registrato
```

Figure 7: Licenza già registrata

4.2 Caso 2) Validità licenza

Scelgo il **ClientCheck**, seleziono l'opzione **Controlla licenza**; inserisco l'id della licenza da controllare ed il clientCheck invia al serverG una richiesta di verifica della validità di una licenza. Il Server mi restituirà "*VALIDA*" qualora la licenza fosse valida, "*INVALIDA*" altrimenti.

```
=== Controllo Licenza ===  
Inserisci ID licenza da verificare: 1  
Stato licenza: VALIDA
```

Figure 8: Licenza valida


```
=== Controllo Licenza ===  
Inserisci ID licenza da verificare: 4  
Stato licenza: INVALIDA
```

Figure 9: Licenza invalida

```
=== Controllo Licenza ===  
Inserisci ID licenza da verificare: 5  
Stato licenza: ERRORE: ID non esistente
```

Figure 10: Licenza inesistente

4.3 Caso 3) Modifica validità licenza

Scelgo il **ClientAdmin**, seleziono l'opzione **Modifica licenza**; inserisco l'id della licenza da modificare e seleziono se voglio validare o invalidare la licenza, qualora provassi a validare una licenza già valida o viceversa non ci sarà nessun tipo di modifica.

```
=== Modifica Licenza Admin ===  
Inserisci ID licenza da modificare: 1  
Scegli operazione:  
1. Invalida  
2. Ripristina  
Scelta: 1  
Risposta portale: OK
```

Figure 11: Modifica validità effettuata

```
=== Modifica Licenza Admin ===
Inserisci ID licenza da modificare: 100
Scegli operazione:
1. Invalida
2. Ripristina
Scelta: 1
Risposta portale: ERRORE: ID non esistente
```

Figure 12: Licenza non trovata

4.4 BONUS) stampa file

Per fini didattici è stata introdotta una funzione **Mostra licenze** per i client, **ClientAdmin** e **ClientCheck** che stampa in formato tabellare il contenuto del file che ha come scopo quello di semplificare il test delle funzioni principali del progetto.

```
=== Menu Client Admin ===
1. Modifica licenza (invalida/ripristina)
2. Mostra licenze
0. Esci
Scelta: 2
+-----+-----+-----+
| ID          | Validità | Scadenza  |
+-----+-----+-----+
| 1           | invalida | 2026-10-21 |
| 2           | invalida | 2026-10-20 |
| 3           | valida   | 2026-10-20 |
| 20          | valida   | 2026-10-21 |
| 30          | invalida | 2026-10-21 |
+-----+-----+-----+
```

Figure 13: Stampa tabella