Università degli Studi di Napoli

Scuola Politecnica e delle Scienze di Base

Area Didattica di Scienze Matematiche Fisiche e Naturali

Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione

Progetto sistemi operativi

Traccia A

Professore:
Alberto Finzi

Candidati:
Mario Turco
Matr. $N860002503$
Francesco Longobardi
Matr. $N860002468$

Anno Accademico 2019/2020

Indice

1 Istruzioni preliminari

1.1 Modalità di compilazione

Il progetto è provvisto di un file makefile il quale è in grado di compilare autonomamente l'intero progetto. Per utilizzare il makefile aprire la cartella del progetto tramite la console di sistema e digitare "make".
In alternativa è possibile compilare manualemente il client ed il server con i seguenti comandi:

```
gcc -o server server.c boardUtility.c parser.c list.c -lpthread
gcc -o client client.c boardUtility.c parser.c list.c -lpthread
```

2 Guida all'uso

2.1 Server

Una volta compilato il progetto è possibile avviare il server digitando da console il seguente comando

```
./server users
```

L'identificativo *users* ri riferisce al nome del file sul quale sarà salvata la lista degli utenti e delle loro credenziali.
È possibile scegliere un nome a piacimento per il file purchè esso sia diverso da *log*.

2.2 Client

Una volta compilato il progetto è possibile avviare il cient digitando da console il seguente comando:

```
./client ip porta
```

Dove *ip* andrà sostituito con l'ip o l'indirizzo URL del server e *porta* andrà sostituito con la porta del server.
Una volta avviato il client comprarià il menu con le scelte 3 possibili: accedi, registrati ed esci.
Una volta effettuata la registrazione dell'utente è possibile effettuare l'accesso al programma al seguito del quale verranno mostrate sia la mappa dei gioco si le istruzioni di gioco.

3 Comunicazione tra client e server

Di seguito verranno illustrate le modalità di comunicazione tra client e server.

3.1 Informazioni preliminari sul server

Il socket del server viene configurato con famiglia di protocoli PF_NET, con tipo ci trasmissione dati SOCK_STREAM e con protocollo TCP. Mostriamo di seguito il codice sorgente:

```
1  void configuraSocket(struct sockaddr_in mio_indirizzo) {
2    if ((socketDesc = socket(PF_INET, SOCK_STREAM, 0)) < 0) {
3      perror("Impossibile creare socket");
4      exit(-1);
5    }
6    if (setsockopt(socketDesc, SOL_SOCKET, SO_REUSEADDR, &(int){1}, sizeof(int)) <
         0)
7      perror("Impossibile impostare il riutilizzo dell'indirizzo ip e della "
8             "porta\n");
9    if ((bind(socketDesc, (struct sockaddr *)&mio_indirizzo,
10             sizeof(mio_indirizzo))) < 0) {
11     perror("Impossibile effettuare bind");
12     exit(-1);
13   }
14 }
```

È importante notare anche come il server riesca a gestire in modo concorrente più client tramite l'uso di un thread dedicato ad ogni client. Una volta aver configurato il socket, infatti, il server si mette in ascolto per nuove connessioni in entrata ed ogni volta che viene stabilita una nuova connessione viene avviato un thread per gestire tale connessione. Di seguito il relativo codice:

```
1  void startListening()
2  {
3      pthread_t tid;
4      int clientDesc;
5      int *puntClientDesc;
6      while (1)
7      {
8        if (listen(socketDesc, 10) < 0)
9          perror("Impossibile mettersi in ascolto"), exit(-1);
10       printf("In ascolto..\n");
11       if ((clientDesc = accept(socketDesc, NULL, NULL)) < 0)
12       {
13         perror("Impossibile effettuare connessione\n");
14         exit(-1);
15       }
16       printf("Nuovo client connesso\n");
17       puntClientDesc = (int *)malloc(sizeof(int));
18       *puntClientDesc = clientDesc;
19       pthread_create(&tid, NULL, gestisci, (void *)puntClientDesc);
20     }
21     close(clientDesc);
22     quitServer();
23 }
```

In particolare al rigo 19 notiamo la creazione di un nuovo thread per gestire la connessione in entrata.