Il progetto: il server

Volendo implementare, per motivi prestazionali, il server di ricezione dei dati in linguaggio C, è possibile accedere a un DBMS MySQL o MariaDB usando la API nativa resa disponibile in forma di libreria C sia per sistema operativo Windows sia per sistema operativo Linux e di cui si riportano nella seguente tabella le principali funzioni:

Funzione	Descrizione
`mysql_library_init`	Inizializzazione delle funzioni di libreria.
`mysql_init`	Creazione della struttura che rappresenta la connessione al server DBMS.
`mysql_real_connect`	Connessione a un database (devono essere specificati: l'indirizzo IP del server DBMS, il nome dello specifico database, lo *username* e la *password*).
`mysql_query`	Esecuzione di un comando in linguaggio SQL.
`mysql_store_result`	Salvataggio del risultato di un'interrogazione.
`mysql_fetch_row`	Acquisizione di una singola riga del risultato di un'interrogazione.
`mysql_free_result`	Rilascio dell'area di memorizzazione del risultato di un'interrogazione.
`mysql_close`	Chiusura della connessione a un database.
`mysql_library_end`	Rilascio delle risorse per le funzioni di libreria.

ESEMPIO

Il seguente programma in linguaggio C visualizza in sequenza tutte le posizioni relative a uno specifico dispositivo di un concorrente della gara, il cui codice viene fornito come argomento della riga di comando, registrate nel database *Tracking* del servizio ospitato in un DBMS MySQL in esecuzione sullo stesso computer con le credenziali di accesso dell'utente *root* privo di *password*:

```
#include <stdio.h>
#include <string.h>
#include <windows.h>
#include <mysql.h>

MYSQL mysql;
MYSQL* database;
MYSQL_RES* result;
MYSQL_ROW row;

int main(int argc, char* argv[])
{
  char query[1024] = "SELECT latitudine, longitudine FROM
                      Posizioni WHERE dispositivo = '";
  if (argc != 2)
    return 0;
  mysql_library_init(0, null, null);
  mysql_init(&mysql);
  database = mysql_real_connect(&mysql, "127.0.0.1", "root",
                                "", "Tracking", 0, 0, 0);
  if (database == null)
    {
      mysql_library_end();
      return 0;
    }
```

```
    strcat(query, argv[1]);
    strcat(query, "';");
    if (mysql_query(database, query) != 0)
      {
        mysql_close(database);
        mysql_library_end();
        return 0;
      }
    result = mysql_store_result(database);
    if (result == null)
      {
        mysql_close(database);
        mysql_library_end();
        return 0;
      }
    while ((row = mysql_fetch_row(result)) != null)
          printf("(%s;%s)\r\n", row[0], row[1]);
    mysql_close(database);
    mysql_library_end();
    return 1;
  }
```

Utilizzando la libreria C per socket UDP introdotta nel capitolo A2, il codice del server che riceve i dati di posizione dei dispositivi mobili e li memorizza nel database è il seguente[2]:

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <mysql.h>
#include "UDP.H"

#define UDP_PORT 12345

struct DATAGRAM
{
  char access_code[32];
  int32_t timestamp;
  float latitude;
  float longitude;
} packet;

MYSQL mysql;
MYSQL* database;
MYSQL_RES* result;
MYSQL_ROW row;

int main(int argc, char* argv[])
{
  unsigned char *buffer = (unsigned char *)&packet;
  unsigned long ip_add;
  unsigned short udp_port;
  char command[1024];
  char code[33];

  mysql_library_init(0, null, null);
  mysql_init(&mysql);
```

2. Tutte le differenze tra la versione per sistema operativo Windows e sistema operativo Linux sono confinate all'interno di questa libreria: il codice presentato non deve essere in alcun modo modificato.

```
  database = mysql_real_connect(&mysql, "127.0.0.1", "root", "",
                                "Tracking", 0, 0, 0);
  if (database == null)
    return 0;
  UDP_open(UDP_PORT);

  printf("Servizio attivo...\r\n");

  for (;;)
    {
      memcpy(code, packet.access_code, 32);
      code[32] = '\0';
      if (UDP_receive(&ip_add, &udp_port, buffer,
                      sizeof(struct DATAGRAM)) > 0)
        {
          sprintf(command, "INSERT INTO posizioni(dispositivo, timestamp,
                            latitudine, longitudine)
                            VALUES ('%s',FROM_UNIXTIME(%i),%f,%f);",
                            code, (int)(packet.timestamp),
                            packet.latitude, packet.longitude);
          printf("%s @ %i (%f,%f): ",code, (int)(packet.timestamp),
                 packet.latitude, packet.longitude);
          if (mysql_query(database, command) != 0)
            printf("ERROR\r\n");
          else
            printf("OK\r\n");
          mysql_query(database, command);
        }
    }
  mysql_close(database);
  mysql_library_end();
  UDP_close();
  return 1;
}
```

OSSERVAZIONE Il ciclo infinito del server obbliga a una terminazione forzata del server in esecuzione. Una soluzione maggiormente professionale avrebbe previsto l'esecuzione del ciclo di ricezione e memorizzazione dei dati in un *thread* separato condizionato dal valore di una variabile booleana e il *main* sospeso in attesa di un input al seguito del quale modificare la variabile booleana per causare la terminazione del *thread* e del programma.