Dal Setup alla Compilazione di Applicazioni C con MSYS2

📋 Indice

🎯 Panoramica del Progetto

Questa guida documenta il percorso completo per sviluppare applicazioni C che si interfacciano con PostgreSQL su Windows. Utilizzeremo:

- **PostgreSQL 17**: Database server
- **pgAdmin 4**: Interfaccia grafica per gestione database
- **MSYS2**: Ambiente Unix-like con MinGW per compilazione
- **libpq**: Libreria C per connessione PostgreSQL

Obiettivo Finale

Compilare ed eseguire un'applicazione C (`File.c`) che si connette a un database PostgreSQL (`DataBase`) e esegue query interattive.

💿 Installazione PostgreSQL su Windows

Passo 1: Download

1. Vai su **https://www.postgresql.org/download/windows/**
2. Clicca su "Download the installer"

3. Scarica la versione più recente (PostgreSQL 17.x)

Passo 2: Installazione

1. **Esegui l'installer** come amministratore
2. **Seleziona componenti**:
   - ✅ **PostgreSQL Server** (obbligatorio)
   - ✅ **pgAdmin 4** (interfaccia grafica)
   - ✅ **Stack Builder** (per estensioni)
   - ✅ **Command Line Tools** (psql, pg_dump, etc.)
3. **Configurazione**:
   - **Directory installazione**: `C:\Program Files\PostgreSQL\17\`
   - **Data Directory**: `C:\Program Files\PostgreSQL\17\data\`
   - **Password superuser**: Imposta una password forte (es. `root`)
   - **Porta**: `5432` (default)
   - **Locale**: `Italian, Italy` (opzionale)

Passo 3: Verifica Installazione

1. **Verifica servizio Windows**:
   - Premi `Win + R` → digita `services.msc`
   - Cerca `postgresql-x64-17`
   - Stato deve essere **"In esecuzione"**
2. **Test da Command Line**:

```
C:\Program Files\PostgreSQL\17\bin\psql -U postgres
# Inserisci password quando richiesta
```

Struttura Directory PostgreSQL

```
C:\Program Files\PostgreSQL\17\
├── bin\                    # Eseguibili (psql, pg_dump, postgres.exe)
├── include\                # Header files per development
│   └── postgresql\
│       └── libpq-fe.h     # Header principale per libpq C
├── lib\                    # Librerie per linking
│   ├── libpq.lib          # Libreria statica per MSVC
│   └── libpq.dll          # Libreria dinamica
├── share\                  # Documentazione e configurazioni
└── data\                  # File del database
```

🖥️ Configurazione pgAdmin 4

Primo Avvio

1. **Apri pgAdmin 4**:
   - Start → PostgreSQL 17 → pgAdmin 4
   - Si aprirà nel browser web predefinito
2. **Imposta Master Password**:
   - Al primo avvio, pgAdmin chiede una password master
   - Questa protegge le password salvate dei server

Configurazione Server PostgreSQL

1. **Aggiungi server**:
   - Pannello sinistro → Click destro su "Servers"
   - Create → Server...
2. **Tab "General"**:
   - **Name**: `Local PostgreSQL 17`
   - **Server group**: Servers (default)
3. **Tab "Connection"**:
   - **Host name/address**: `localhost`
   - **Port**: `5432`
   - **Maintenance database**: `postgres`
   - **Username**: `postgres`
   - **Password**: [password impostata durante installazione]
   - ✅ **Save password**: attivato
4. **Clicca "Save"**

Test Connessione

Nel **Query Tool** (icona fulmine), esegui:

```
SELECT version();
SELECT current_database(), current_user;
```

🔧 Setup MSYS2 per Compilazione C

Perché MSYS2?

PostgreSQL su Windows viene compilato con MSVC, ma le sue librerie di sviluppo per MinGW sono distribuite tramite MSYS2. Questo ci permette di usare GCC per compilare applicazioni C.

Installazione MSYS2

1. **Download**:
   - Vai su https://www.msys2.org/
   - Scarica `msys2-x86_64-YYYYMMDD.exe`
2. **Installazione**:
   - Esegui installer
   - Directory: `C:\msys64\` (default)
   - Al termine si aprirà automaticamente MSYS2

Configurazione Iniziale

⚠️ **IMPORTANTE**: Usa sempre **MSYS2 MinGW 64-bit**, non MSYS2 MSYS!

1. **Apri MSYS2 MinGW 64-bit**:
   - Start → MSYS2 → MSYS2 MinGW 64-bit
2. **Aggiorna sistema** (obbligatorio):

   ```
   pacman -Syu
   ```

   Se richiesto, chiudi MSYS2 e riaprilo, poi:

   ```
   pacman -Syu
   ```

3. **Installa toolchain di sviluppo**:

   ```
   pacman -S mingw-w64-x86_64-toolchain
   ```

   Conferma installazione di tutti i pacchetti (circa 200MB)
4. **Installa PostgreSQL development files**:

   ```
   pacman -S mingw-w64-x86_64-postgresql
   ```

Verifica Setup

```
# Verifica GCC
gcc --version
# Output atteso: gcc.exe (Rev1, Built by MSYS2 project) 13.x.x
```

```
# Verifica libpq header
ls /mingw64/include/libpq-fe.h
# Output atteso: /mingw64/include/libpq-fe.h

# Verifica librerie
ls /mingw64/lib/libpq*
# Output atteso: libpq.dll.a, libpq.a
```

🗄️ Creazione e Gestione Database

Creazione Database in pgAdmin

1. **Naviga nel pannello**:
   - Servers → Local PostgreSQL 17 → Databases
2. **Crea nuovo database**:
   - Click destro su "Databases" → Create → Database...
   - **Database**: `DataBase`
   - **Owner**: `postgres`
   - **Encoding**: `UTF8`
   - **Template**: `template1`
   - Clicca "Save"

Esecuzione Script SQL

Questo è il punto critico dove spesso si incontrano errori. Segui attentamente:

1. **Apri Query Tool**:
   - Seleziona database `LotoBianco`
   - Clicca icona "Query Tool" (⚡)
2. **Carica script**:
   - File → Open
   - Seleziona `db2versione.sql`
3. **Problema comune: Vincoli di Integrità**

   ❌ **ERRORE TIPICO:**

```
ERROR: la INSERT o l'UPDATE sulla tabella "cliente" viola il vincolo di
chiave esterna
La chiave (via, citta, cap)=(Via Roma 10, Milano, 20121) non è presente
nella tabella "indirizzo"
```

✅ **SOLUZIONE:** Ordine corretto di creazione

```sql
-- 1. PRIMA: Crea tabella referenziata
CREATE TABLE indirizzo (
    via VARCHAR(100) NOT NULL,
    citta VARCHAR(100) NOT NULL,
    cap VARCHAR(10) NOT NULL,
    PRIMARY KEY (via, citta, cap)
);

-- 2. POI: Inserisci dati nella tabella referenziata
INSERT INTO indirizzo (via, citta, cap) VALUES
('Via Roma 10', 'Milano', '20121'),
('Viale delle Rose 25', 'Torino', '10121'),
('Corso Italia 50', 'Firenze', '50123'),
('Via Garibaldi 3', 'Napoli', '80134'),
('Via Verdi 12', 'Bologna', '40121');

-- 3. QUINDI: Crea tabella che referenzia
CREATE TABLE cliente (
    cf VARCHAR(16) NOT NULL PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    cognome VARCHAR(100) NOT NULL,
    via VARCHAR(100) NOT NULL,
    citta VARCHAR(100) NOT NULL,
    cap VARCHAR(10) NOT NULL,
    telefono VARCHAR UNIQUE NOT NULL,
    FOREIGN KEY (via, citta, cap) REFERENCES indirizzo(via, citta, cap)
);

-- 4. INFINE: Inserisci clienti
INSERT INTO cliente (cf, nome, cognome, via, citta, cap, telefono)
VALUES
('ZGOMRA85C10F205Y', 'Maria', 'Zago', 'Via Roma 10', 'Milano', '20121',
'3201234567'),
-- ... altri clienti
```

4. **Esegui script**:
   - Clicca "Execute" (▶) o premi F5
   - Controlla messaggi nella parte inferiore

Verifica Creazione Tabelle

```sql
-- Lista tutte le tabelle
SELECT table_name
FROM information_schema.tables
WHERE table_schema = 'public'
```

```
ORDER BY table_name;

-- Test dati
SELECT COUNT(*) as totale_clienti FROM cliente;
SELECT COUNT(*) as totale_trattamenti FROM trattamento;
```

💻 Sviluppo Applicazioni C con libpq

Struttura Progetto

```
C:\Users\[TUO_NOME]\Desktop\progetto\
├── File.c       # Codice sorgente principale
├── db2versione.sql      # Script creazione database
├── build.sh            # Script compilazione (opzionale)
└── File.exe     # Eseguibile (dopo compilazione)
```

Esempio Codice C Base

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <libpq-fe.h>

// Configurazione database - ADATTA QUESTI VALORI
#define DBNAME "LotoBianco"
#define USER "postgres"
#define PASSWORD "root"        // La TUA password PostgreSQL
#define HOST "127.0.0.1"
#define PORT "5432"

void eseguiQuery(char *query, PGconn *con) {
    PGresult *rs = PQexec(con, query);

    if (PQresultStatus(rs) != PGRES_TUPLES_OK) {
        printf("Errore durante l'esecuzione della query: %s\n",
               PQerrorMessage(con));
    } else {
        int rows = PQntuples(rs);
        int cols = PQnfields(rs);

        printf("\nRisultati della query (%d righe):\n", rows);
        printf("----------------------------------------\n");

        // Stampa header
```

```
        for (int j = 0; j < cols; j++) {
            printf("%-20s", PQfname(rs, j));
        }
        printf("\n");

        // Stampa dati
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                printf("%-20s", PQgetvalue(rs, i, j));
            }
            printf("\n");
        }
        printf("----------------------------------------\n");
    }
    PQclear(rs);
}

void chiudiConnessione(PGconn *conn) {
    PQfinish(conn);
    exit(1);
}

int main() {
    PGconn *conn;
    int scelta;

    printf("=== Applicazione Database LotoBianco ===\n\n");

    // Connessione al database
    conn = PQsetdbLogin(HOST, PORT, NULL, NULL, DBNAME, USER, PASSWORD);

    // Verifica della connessione
    if (PQstatus(conn) != CONNECTION_OK) {
        printf("ERRORE: Connessione al database fallita:\n%s\n",
               PQerrorMessage(conn));
        printf("\nVerifica che:\n");
        printf("1. PostgreSQL sia in esecuzione\n");
        printf("2. Database '%s' esista\n", DBNAME);
        printf("3. Password sia corretta\n");
        chiudiConnessione(conn);
    }

    printf("✅ Connessione al database riuscita!\n\n");

    // Menu principale
    do {
        printf("\n=== MENU PRINCIPALE ===\n");
        printf("1. Mostra tutti i clienti\n");
        printf("2. Mostra tutti i trattamenti\n");
        printf("3. Query personalizzata\n");
```

```
        printf("4. Esci\n");
        printf("Scelta: ");
        scanf("%d", &scelta);

        switch(scelta) {
            case 1: {
                char *query = "SELECT cf, nome, cognome, telefono FROM
cliente ORDER BY cognome;";
                printf("\n=== CLIENTI ===");
                eseguiQuery(query, conn);
                break;
            }
            case 2: {
                char *query = "SELECT codice, data_ora, prezzo, cf_cliente
FROM trattamento ORDER BY data_ora;";
                printf("\n=== TRATTAMENTI ===");
                eseguiQuery(query, conn);
                break;
            }
            case 3: {
                char query[1000];
                printf("Inserisci query SQL: ");
                getchar(); // consume newline
                fgets(query, sizeof(query), stdin);
                printf("\n=== RISULTATO QUERY ===");
                eseguiQuery(query, conn);
                break;
            }
            case 4:
                printf("Arrivederci!\n");
                break;
            default:
                printf("Opzione non valida!\n");
        }
    } while (scelta != 4);

    // Chiusura connessione
    PQfinish(conn);
    return 0;
}
```

Compilazione Passo-Passo

1. **Apri MSYS2 MinGW 64-bit**
2. **Naviga alla directory del progetto**:

```
cd /c/Users/[TUO_NOME]/Desktop/progetto
# Oppure usa tab completion:
```

```
cd /c/Users/[TAB][TAB]
```

3. **Verifica file sorgente**:

```
ls -la File.c
```

4. **Compila l'applicazione**:

```
gcc File.c -lpq -o File.exe
```

✅ **Se successo**, vedrai:

```
# Nessun output = compilazione riuscita
ls File.exe  # File creato
```

❌ **Se ERRORE**, vedi sezione Risoluzione Errori

5. **Esegui l'applicazione**:

```
./File.exe
```

📝 Percorso Completo: Dalla Chat al Funzionamento

Questa sezione ripercorre esattamente il problema affrontato nella chat originale:

Il Problema Iniziale

```
File.c:4:10: fatal error: libpq-fe.h: No such file or directory
```

La Soluzione Step-by-Step

1. **Problema**: GCC di Windows non trova libpq-fe.h
   - **Causa**: Header non installato o path errato
   - **Tentativo fallito**: Specificare path manualmente con `-I`
2. **Soluzione**: Usare MSYS2 con PostgreSQL nativo
   - **Installazione**: `pacman -S mingw-w64-x86_64-postgresql`
   - **Risultato**: Headers e librerie disponibili automaticamente
3. **Problema**: Errore di autenticazione database

```
FATALE: autenticazione con password fallita per l'utente "postgres"
```

- **Causa**: Password nel codice C non corrispondeva a quella reale
- **Soluzione**: Modificare `#define PASSWORD "root"`

4. **Problema**: Tabelle non esistono

```
ERRORE: la relazione "trattamento" non esiste
```

- **Causa**: Script SQL non eseguito correttamente
- **Soluzione**: Rieseguire script in pgAdmin

5. **Problema**: Vincoli di chiave esterna

```
ERROR: la chiave (via, citta, cap) non è presente nella tabella
"indirizzo"
```

- **Causa**: Ordine errato di creazione tabelle
- **Soluzione**: Creare e popolare `indirizzo` prima di `cliente`

Il Successo Finale

```
$ gcc File.c -lpq -o File.exe
$ ./MainDataBase.exe

Scegli un'opzione:
1. Trattamenti che hanno utilizzato un prodotto...
2. Dipendenti specialisti che hanno eseguito...
...
Scelta: 1
Risultati della query:
1    2024-06-05 09:00:00    Bioline Jatò    Pure Gel Cleansing    2027-01-31
...
```

🚨 Risoluzione Errori Comuni

Errori di Compilazione

1. `libpq-fe.h: No such file or directory`

Possibili cause e soluzioni:

A) Non stai usando MSYS2 MinGW 64-bit

```
# Verifica che sei nel terminale giusto
echo $MSYSTEM
# Output dovrebbe essere: MINGW64
```

B) libpq non installato

```
pacman -S mingw-w64-x86_64-postgresql
```

C) Stai usando GCC di Windows (non MSYS2)

```
which gcc
# Output dovrebbe essere: /mingw64/bin/gcc
# NON: /c/mingw32/bin/gcc o /c/Program Files/...
```

2. `undefined reference to 'PQsetdbLogin'`

Causa: Flag di linking errato

```
# ❌ SBAGLIATO
gcc -lpq File.c -o File.exe

# ✅ CORRETTO
gcc File.c -lpq -o File.exe
```

3. Errori di sintassi C

Problema: Caratteri strani nel codice

```
// ❌ Caratteri Unicode invece di virgolette normali
printf("Errore");

// ✅ Virgolette ASCII corrette
printf("Errore");
```

Errori di Esecuzione

1. `The program can't start because libpq.dll is missing`

Soluzioni:

A) Copia DLL nella directory dell'eseguibile

```
cp /mingw64/bin/libpq.dll ./
```

B) Aggiungi MSYS2 al PATH di Windows

- Sistema → Proprietà → Impostazioni avanzate → Variabili d'ambiente
- Aggiungi `C:\msys64\mingw64\bin` alla variabile PATH

2. `connection to server failed: FATAL: password authentication failed`

Verifica password:

```
# Test manuale connessione
psql -h localhost -p 5432 -U postgres -d LotoBianco
# Se funziona, il problema è nel codice C
```

Aggiorna password nel codice:

```
#define PASSWORD "TUA_PASSWORD_REALE"
```

3. `connection to server failed: FATAL: database "LotoBianco" does not exist`

Verifica esistenza database:

```
-- In pgAdmin Query Tool (connesso a 'postgres')
SELECT datname FROM pg_database WHERE datname = 'LotoBianco';
```

Se non esiste, crealo in pgAdmin.

Errori Database

1. `relation "tabella" does not exist`

Diagnosi:

```
-- Verifica tabelle esistenti
\dt  -- in psql
-- O in pgAdmin:
SELECT table_name FROM information_schema.tables WHERE table_schema =
'public';
```

Se non ci sono tabelle: Riesegui script `db2versione.sql`

2. `violates foreign key constraint`

Esempio tipico:

```
ERROR: insert or update on table "cliente" violates foreign key constraint
DETAIL: Key (via, citta, cap)=(Via Roma 10, Milano, 20121) is not present in
table "indirizzo".
```

Soluzione sistematica:

1. **Identifica dipendenze:**

```sql
-- Trova tutti i vincoli FK
SELECT
    tc.table_name,
    kcu.column_name,
    ccu.table_name AS foreign_table_name
FROM information_schema.table_constraints AS tc
JOIN information_schema.key_column_usage AS kcu
    ON tc.constraint_name = kcu.constraint_name
JOIN information_schema.constraint_column_usage AS ccu
    ON ccu.constraint_name = tc.constraint_name
WHERE tc.constraint_type = 'FOREIGN KEY';
```

2. **Ordine corretto di inserimento:** Prima le tabelle "padre", poi le "figlie"
3. **Script corretto:**

```sql
-- Prima: tabelle senza dipendenze
INSERT INTO indirizzo VALUES (...);
INSERT INTO dipendente VALUES (...);

-- Poi: tabelle con FK
INSERT INTO cliente VALUES (...);
INSERT INTO trattamento VALUES (...);
```
