라즈베리파이에 MYSQL설치

OneKick

MySQL 설치

OneKick

1. MySQL 설치
   - sudo apt-get install mysql-server mysql-client

2. root 계정 비밀번호 변경하기(초기 설치시에 비번없음)
   - sudo mysql –uroot
   - use mysql
   - select user, host, password from user; //패스워드가 없는것 확인됨
   - //패스워드 설정
   - >>update user set password=password('1234') where user='root';
   - >>flush privileges;
   - >>quit;

```
MariaDB [(none)]> use mysql
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [mysql]> select user, host, password from user;
+------+-----------+-------------------------------------------+
| user | host      | password                                  |
+------+-----------+-------------------------------------------+
| root | localhost | *A4B6157319038724E3560894F7F932C8886EBFCF |
+------+-----------+-------------------------------------------+
1 row in set (0.00 sec)

MariaDB [mysql]> UPDATE user SET password=PASSWORD('1234') where user='root';
Query OK, 0 rows affected (0.00 sec)
Rows matched: 1  Changed: 0  Warnings: 0

MariaDB [mysql]> FLUSH PRI
PRIMARY     PRIVILEGES
MariaDB [mysql]> FLUSH PRI
PRIMARY     PRIVILEGES
MariaDB [mysql]> FLUSH PRIVILEGES
    -> ;
Query OK, 0 rows affected (0.00 sec)

MariaDB [mysql]> quit
Bye
pi@raspberrypi:~ $
```

외부접속 허용하기

OneKick

- MySQL은 기본적으로 외부 접속이 비허용
- 설정변경할 디렉토리로 이동
  - ~$ cd /etc/mysql/mariadb.conf.d/
  - ls
- 변경파일 열기
  - sudo nano 50-server.cnf
- 주석처리
  - #bind-address = 127.0.0.1
- 변경 설정 적용하기 위한 서버 재시작
  - sudo service mysql restart

3. 테이블 생성

```
>>show databases;
>>create database your_DB;
>>use your_DB
>>show tables;
>>create table prod(
        pdate char(10), pch char(10),
        pcnt int, primary key(pdate)
        );
>>insert into prod value ('2019-1', 'yoo', 1);
>> select * from prod;
```

4. 테이블 생성 확인

>> select * from prod;

5. 생성된 데이터베이스에 접근권한 부여

>>grant all privileges on *.* to root@localhost identified by '1234';

>>flush privileges;

C언어에서 Mysql에 읽기 쓰기 가능

OneKick

- C API 라이브러리 설치하면 C언어에서 Mysql에 읽기 쓰기 가능
  - 라이브러리 설치
    - sudo apt-get install default-libmysqlclient-dev
  - 소스코드 mysql경로 변경
    - #include<mysql/mysql.h>
  - 컴파일 명령
    - gcc mysql.c –l /usr/include/mysql/mysql –Wall –g –lm mysqlclient –g

MariaDB 명령어

OneKick

- **접속하기/끝내기**
  - $ mysql -u 유저명 -p --> 접속하기
  - MariaDB[(none)] quit --> 끝내기
- **Database**
  - MariaDB[(none)] show databases ; --> DB목록보기
  - MariaDB[(none)] create database DB명 ; --> DB생성
  - MariaDB[(none)] drop database DB명 ; --> DB삭제
  - MariaDB[(none)] use DB명; --> 사용할 DB선택하기
- **Table**
  - MariaDB[DB명] show tables; --> 테이블목록보기
  - MariaDB[DB명] create table 테이블명 (컬럼명1 데이터타입1, 컬럼명2 데이터타입2, ...); --> 테이블생성
  - MariaDB[DB명] drop table 테이블명; --> 테이블삭제

에러들

OneKick

- 비번 불일치
  - ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: NO)
  - 해결책 : mysql –uroot –p

- 비번 바꾸기
  - https://freestrokes.tistory.com/43

MariaDB 명령어

OneKick

- **Column**
  - MariaDB[DB명] show columns from 테이블명; --> 컬럼목록보기1
  - MariaDB[*] show columns from 테이블명 from DB명; --> 컬럼목록보기2
  - MariaDB[*] show columns from DB명.테이블명; --> 컬럼목록보기3
  - MariaDB[DB명] show columns from 테이블명 where type like '타입명'; --> 컬럼목록보기 (특정타입)
  - ex) show columns from 테이블명 where type like 'varchar%';
  - MariaDB[DB명] alter table 테이블명 add 컬럼명 데이터타입 옵션; --> 컬럼 추가
  - MariaDB[DB명] alter table 테이블명 drop 컬럼명; --> 컬럼 삭제
  - MariaDB[DB명] alter table 테이블명 change 컬럼명 변경할컬럼명 varchar(12); --> 컬럼명 변경/타입변경
  - MariaDB[DB명] alter table 테이블명 modify 컬럼명 변경할타입명; --> 컬럼 타입변경 (ex. varchar(10)
- **Count**
  - MariaDB[DB명] select count(*) from 테이블명; --> 전체 레코드 수 (row count) 구하기
  - MariaDB[DB명] SELECT TABLE_NAME AS "Tables", round(((data_length + index_length) / 1024 / 1024), 2) "Size in MB" FROM information_schema.TABLES WHERE table_schema = "DB_NAME" ORDER BY (data_length + index_length) DESC; --> DB사이즈 구하기 (DB_NAME항목에 알고자 하는 DB명 입력)

MariaDB 명령어

- 추가하기

  insert into 테이블이름 values('2017', 1);

- **Etc**
  - MariaDB[DB명] show warnings; → 경고창 확인
  - MariaDB[DB명] flush privileges ; → 저장하기
  - MariaDB[DB명] repair table user user_frm; → 오류 시 실행
  - MariaDB[*] update user set password = password(' 암호 ') ; → 비밀번호 변경
  - MariaDB[*] show processlist ; → 현재 작업 중인 프로세스들의 진행정도 확인
  - MariaDB[*] kill " id번호 " ; → 현재 작업 중인 프로세스들 중 특정 프로세스를 종료시키기. id번호는 show processlist에서 확인할 수 있음.

ControlDB.c

OneKick

```
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <wiringPi.h>
#include <wiringSerial.h>
#include <stdlib.h>
#include "mysql.h"
#include <time.h>


// MySQL DB를 초기화 하고 사용할 수 있도록 연결하는 함수
int initDB(MYSQL * mysql, const char * host, const char * id, const char * pw, const char *
db)
{
 printf("(i) initDB called, host=%s, id=%s, pw=%s, db=%s₩n", host, id, pw, db);
 mysql_init(mysql); // DB 초기화
 if(mysql_real_connect(mysql, host, id, pw, db,0,NULL,0)) // DB 접속
 {
  printf("(i) mysql_real_connect success₩n");
  return 0; // 성공
 }
 printf("(!) mysql_real_connect failed₩n");
 return -1; // 실패
}
```

```
// DB에 쓰는 함수 - 정수형 인자 5개는 DB Table의 각 field라고 가정
int writeDB(MYSQL * mysql, int door, int gas, int flame, int fan, int pin)
{
 char strQuery[255]=""; // 쿼리 작성에 사용할 버퍼
 // 삽입 쿼리 작성, time 필드는 DATE 타입의 현재 시각
 sprintf(strQuery, "INSERT INTO twoteam(id, door, gas, flame, fan, pin, time)
values(null, %d, %d, %d, %d, %d, now())", door, gas, flame, fan, pin);
 int res = mysql_query(mysql, strQuery); // 삽입 쿼리의 실행

if (!res) // 성공
 {
        printf("(i) inserted %lu rows.₩n", (unsigned long)mysql_affected_rows(mysql));
 }
 else // 실패
 {
  fprintf(stderr, "(!) insert error %d : %s₩n", mysql_errno(mysql), mysql_error(mysql));
  return -1;
 }
 return 0;
}
```

```
// DB에서 읽어오기, id 가 primary key이고 읽은 결과는 buf에 구분자|를 이용해 반환한다.
int readDB(MYSQL * mysql, char * buf, int size, int id)
{
 char strQuery[256]=""; // select query를 작성할 버퍼
 buf[0]=0; // 반환할 값은 strcat() 함수를 이용할 수 있도록 첫 바이트에 null을 넣는다.


 // select query 작성
 sprintf(strQuery, "SELECT door, gas, flame, fan FROM twoteam WHERE id=%d;",id);
 int res = mysql_query(mysql, strQuery); // query의 실행
 if(res!=0) // 실패
 {
  return -1;
 }
 else // 성공
 {
  // select 문의 처리 결과는 커서 형태로 시스템에 의해 관리된다.
  MYSQL_RES * res_ptr = mysql_use_result(mysql); // 시스템에 결과셋을 맡긴다.
  MYSQL_ROW sqlrow = mysql_fetch_row(res_ptr);  // 결과셋에서 한줄만 받아온다.
  // 위에서 한줄만 가져오는 이유는 결과가 유일하다는 전제에서 이다.
  // 만약 결과가 여러개일 경우 반복문을 돌면서 모든 row에 대해 처리해 주어야 한다.
```

```
// make a string for return
 unsigned int field_count=0;
 while(field_count<mysql_field_count(mysql)) // 가져온 한줄의 모든 필드 값을 꺼내온다.
 {
  char buf_field[256]=""; // 각 필드에 보관된 개별 값을 저장할 임시 버퍼
  if(sqlrow[field_count])
   sprintf(buf_field,"|%s", sqlrow[field_count]); // 각 필드의 값을 문자열 형태로 가져온다.
  else sprintf(buf_field,"|0"); // 값이 null 인 경우 0으로 표시한다.
  strcat(buf,buf_field); // 가져온 필드의 값을 반환버퍼에 붙인다.(string append)
  field_count++; // 다음 필드로 계속 진행
 }


 if(mysql_errno(mysql)) // 만약 에러가 있었으면
 {
  fprintf(stderr, "(!) error: %s₩n", mysql_error(mysql)); // 원인 표시
  return -1; // 에러값 반환
 }
 mysql_free_result(res_ptr); // 시스템에 맡겨놓았던 결과셋을 이제 버리라고 한다.
}
return 0; // 정상종료 반환
}
```

```
// DB 접속 종료하기
int closeDB(MYSQL * mysql)
{
 mysql_close(mysql); // db 연결 해제
 return 1;
}
```

IoTestApp.c

OneKick

```
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <wiringPi.h>
#include <wiringSerial.h>
#include <stdlib.h>
#include "mysql.h"
#include <time.h>


// extern이란 여기는 없지만 다른파일 어딘가에는 있다는 뜻입니다.
// 아래 4개 함수는 사실 ControlDB.c 파일에 구현되어 있습니다.
extern int initDB(MYSQL*, const char * host, const char * id, const char * pw, const char * db);
extern int writeDB(MYSQL*, int door, int gas, int flame, int fan, int pin);
extern int readDB(MYSQL*, char * buff, int size, int idRow);
extern int closeDB(MYSQL *);

int main()
{
 printf("(i) This is a test program for the ControlDB.₩n"); // 인삿말
 // DB에 삽입할 4가지 정수형 정보들
 int randomDoor = 0;
 int randomGas = 0;
 int randomFlame = 0;
 int randomFan = 0;


 // 시드 랜덤값을 위해 시간으로 초기화
 srand(time(NULL));
```

```
// 4가지 정보는 테스트를 위해 랜덤으로 만들어 둔다.
randomDoor = rand()%256;
randomGas = rand()%256;
randomFlame = rand()%256;
randomFan = rand()%256;


// 잘 만들어 졌는지 한번 확인
printf("(i) random: door=%d, gas=%d, flame=%d, fan=%d₩n",
  randomDoor, randomGas, randomFlame, randomFan);


MYSQL mysql; // mysql DB의 연결 상태를 관리할 구조체 변수 생성


// DB에 연결해보자. 로컬에 접속하고, id는 root, 비번은 12341, DB명은 ssw 이다.
if(initDB(&mysql, "localhost","root","12341","ssw")<0)
{
 printf("(!) initDB failed₩n"); // 실패
 return -1;
}
else  printf("(i) initDB successd!₩n"); // 성공
```

```
int pin = 6; // 추가적으로 필요한 필드(별거 아님;)
 // DB에 쓰기 - 성공 시 0, 실패 시 -1 반환
 int res = writeDB(&mysql, randomDoor, randomGas, randomFlame, randomFan,pin);

 if(res<0) {
  printf("(!) writeDB failed₩n");
  return -1;
 }
 else printf("(i) writeDB success!, ₩n");


 // 이제 삽입한 row의 값을 다시 읽어와 보자.
 char buf[256]=""; // 결과를 받아올 문자열 버퍼
 int rowId = 1; // 테스트로 id가 1인 row의 값을 읽어온다.
 res = readDB(&mysql, buf, 256, rowId); // 읽기 실행

 if(res<0) // 실패
 {
  printf("(!) readDB failed₩n");
  return -1;
 }
 else printf("(i) readDB success!, buf=%s₩n", buf); // 성공
```

```
 if(closeDB(&mysql)<0) // DB 연결 끊기
 {
  printf("(!) clseDB failed₩n"); 실패
  return -1;
 }
 else printf("(i) closeDB success₩n"); 성공


 // 종료 알림
 printf("(i) This program will be closed.₩n");
 return 0;
}
```

Makefile 만들기

OneKick

```
CC=gcc
SOURCES = IoTestApp.c  ControlDB.c
OBJECTS = ${SOURCES:.c=.o}
CFLAGS=-I /usr/include/mysql -Wall -g
LDFLAGS=-lmysqlclient -g
RM = rm -f
OUT = IoTestApp
$(OUT): $(OBJECTS)
	$(CC) -o $(OUT) $(OBJECTS) $(LIBS) $(CFLAGS) $(LDFLAGS)
clean:
	$(RM) $(OUT) *.o
```

////////tab키로 이동

테이블 생성

OneKick

```
>>mysql –u root

>>show databases;
>>create database ssw;
>>create table twoteam(
    id int primary key auto_increment, door int, gas int,
   flame int, fan int, pin int, time date);
```