Proyecto final

Carrera:
Ingeniería en sistemas computacionales

Materia:
Sistemas operativos

Profesor:
Gustavo Moisés Romero González

Equipo:
Cabañas Santamaria Anel Athziri
Miranda Martínez Alejandro
Roldan Velazquez Ian Jurguen

Grupo:
5S21

Índice

Introducción

En este proyecto haremos la inserción de datos desde la consola a través de Mosquitto(MQTT) a una base de datos llamada MariaDB, desde la cual podrán consultarse los datos ingresados anteriormente, pertenecientes a estudiantes del TESOEM.

Desarrollo.

Productor.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <mosquitto.h>
```

- **Bibliotecas Incluidas**: Se incluyen las bibliotecas estándar de C (stdio.h, stdlib.h, string.h) y la biblioteca de Mosquitto (mosquitto.h), que es necesaria para trabajar con el protocolo MQTT.

```
int main() {
  struct mosquitto *mosq;
  int rc;
```

- **Función Principal y Variables**: Se declara la función principal y se definen dos variables: mosq para la estructura de Mosquitto y rc para el código de retorno.

```
    mosquitto_lib_init();
  mosq = mosquitto_new("producer", true, NULL);
```

- **Inicialización de Mosquitto**: Se inicializa la biblioteca de Mosquitto y se crea una nueva instancia de cliente MQTT con el identificador "producer".

```
  rc = mosquitto_connect(mosq, "localhost", 1883, 60); if
  (rc != MOSQ_ERR_SUCCESS) {
    fprintf(stderr, "Error al conectar: %s\n", mosquitto_strerror(rc)); return
    1;
  }
  printf("Conectado al broker MQTT.\n");
```

- **Conexión al Broker MQTT**: Se intenta conectar al broker MQTT en localhost en el puerto 1883. Si la conexión falla, se imprime un mensaje de error y el programa termina.

```
  const char *datos[] = {
    "{\"MATRICULA\": 226020039, \"NOMBRE\": \"Alejandro\", \"Pap\": \"Miranda\", \"Sap\":
\"Martinez\", \"MATERIA\": \"Sistemas Operativos\", \"CALIFICACION\": \"95\", \"CARRERA\":
\"Ingenieria en Sistemas\"}",
    "{\"MATRICULA\": 226020042, \"NOMBRE\": \"Anel Athziri\",
\"Pap\":\"Cabañas\",\"Sap\":\"Santamaria\",\"MATERIA\":\"Sistemas
Operativos\",\"CALIFICACION\":\"80\",\"CARRERA\":\"Ingenieria en Sistemas\"}"
  };
```

- **Datos a Publicar**: Se definen los datos a publicar en formato JSON como un arreglo de cadenas.

```
  for (int i = 0; i < 2; i++) {
    printf("Publicando datos: %s\n", datos[i]);
    rc = mosquitto_publish(mosq, NULL, "datos/informacion", strlen(datos[i]), datos[i], 0, false); if
    (rc != MOSQ_ERR_SUCCESS) {
      fprintf(stderr, "Error al publicar: %s\n", mosquitto_strerror(rc));
      return 1;
    }
    printf("Datos publicados en el tópico 'datos/informacion'.\n");
  }
```

- **Publicación de Datos**: Se publica cada dato en el tópico datos/informacion usando un bucle for. Si hay algún error durante la publicación, se imprime un mensaje de error y el programa termina.

```
  mosquitto_disconnect(mosq);
  mosquitto_destroy(mosq);
  mosquitto_lib_cleanup();
  printf("Desconectado del broker MQTT.\n");

  return 0;
}
```

- **Desconexión y Limpieza**: Se desconecta del broker MQTT, se destruye la instancia del cliente y se limpia la biblioteca de Mosquitto.

Consumidor.c

```
#include   <stdio.h>
#include  <stdlib.h>
#include <string.h>
#include <mariadb/mysql.h>
#include <mosquitto.h>
#include <json-c/json.h>
```

- **Bibliotecas Incluidas**: Se incluyen las bibliotecas estándar de C, las bibliotecas de MariaDB/MySQL (mysql.h), Mosquitto (mosquitto.h) y JSON-C (json.h), que se usan para manipular JSON.

```
void on_connect(struct mosquitto *mosq, void *userdata, int rc) {
   if (rc == 0) {
      printf("Conectado al broker MQTT.\n");
      mosquitto_subscribe(mosq, NULL, "datos/informacion", 0);
   } else {
      fprintf(stderr, "Error al conectar: %s\n", mosquitto_strerror(rc));
   }
}
```

- **Callback de Conexión**: Función de callback que se llama cuando el cliente se conecta al broker MQTT. Si la conexión es exitosa, se suscribe al tópico datos/informacion.

```
void on_message(struct mosquitto *mosq, void *userdata, const struct mosquitto_message *message)
{
   printf("Mensaje recibido en el tópico 'datos/informacion'. Payload: %s\n", (char *)message-
>payload);
   struct json_object *parsed_json;
   struct json_object *matricula, *nombre, *pap, *sap, *materia, *calificacion, *carrera; parsed_json

   = json_tokener_parse(message->payload);

   json_object_object_get_ex(parsed_json, "MATRICULA", &matricula);
   json_object_object_get_ex(parsed_json, "NOMBRE", &nombre);
   json_object_object_get_ex(parsed_json, "Pap", &pap);
   json_object_object_get_ex(parsed_json, "Sap", &sap);
   json_object_object_get_ex(parsed_json, "MATERIA", &materia);
   json_object_object_get_ex(parsed_json, "CALIFICACION", &calificacion);
   json_object_object_get_ex(parsed_json, "CARRERA", &carrera);
```

- **Callback de Mensaje**: Función de callback que se llama cuando se recibe un mensaje en el tópico datos/informacion. Se imprime el payload del mensaje y se parsea el JSON recibido.

```
MYSQL *conn;
MYSQL_RES *res;
MYSQL_ROW row;

char *server = "localhost";
char *user = "isc";
char *password = "tesoem";
char *database = "datos";

conn = mysql_init(NULL);
```

- **Conexión a MySQL**: Se definen las variables para la conexión a la base de datos y se inicializa la conexión a MySQL.

Conclusiones.

Todos los procesos deben hacerse dentro de la máquina virtual para que la capacidad de los programas usados se adapte a la misma.
El proceso de instalación del programa en el que visualizaríamos la base de datos es algo que desconocíamos porque no estamos acostumbrados a ese sistema operativo.
Las conexiones realizadas pueden ser tediosas porque se requiere de activación y desactivación para la realización de ciertos procesos.
Verificar que los datos hayan sido ingresados de manera correcta requiere de muchos pasos a seguir antes de visualizar la tabla con los datos.

Link GitHub

https://github.com/Xolotl5/productor-consumidor