Custom Encryption Tool for Securing Messages

Project Overview

Title: Custom Encryption Tool

Description: The Custom Encryption Tool is a simple yet effective application developed in C for encrypting text messages. This tool is designed to demonstrate basic encryption techniques by applying a shift-based encryption algorithm to text data. Users can input their messages and an encryption key, which the tool uses to alter the ASCII values of the characters in the message. The encrypted message is then saved to a file for secure storage.

Features

- **Encryption Algorithm:** Implements a shift cipher where each character's ASCII value is shifted by a specified integer key. This provides basic encryption capabilities suitable for educational purposes.
- **User Interaction:** Prompts the user to enter a message and an integer key, enhancing the hands-on experience with encryption concepts.
- **File Handling:** Writes the encrypted message to a text file, `Encrypted_message.txt`, allowing for persistent storage of the encrypted data.

Technical Details

Header Files:

- `stdio.h`: For standard input/output operations.
- `string.h`: For string manipulation functions.

Macro Definitions:

- `#define MAX_MESSAGE_LENGTH 1024`: Defines the maximum length of the message that can be encrypted.

Functions:

1. **`char encryptChar(char c, int key)`**
   - **Purpose:** Encrypts a single character by shifting its ASCII value based on the given key.
   - **Parameters:**
     - `c` (char): The character to be encrypted.
     - `key` (int): The encryption key (integer) that determines the shift amount.
   - **Returns:** The encrypted character as a `char`.

   Logic:

   ```
   c
   Copy code
   return (c + key);
   ```

   This function simply adds the key value to the ASCII value of the character.

2. **`void encryptMessage(char *message, int key)`**
   - **Purpose:** Encrypts an entire string message using the provided key.
   - **Parameters:**
     - `message` (char*): The string message to be encrypted.

- `key` (int): The encryption key (integer) that applies to all characters in the message.

Logic:

```
c
Copy code
for (int i = 0; i < strlen(message); i++) {
    message[i] = encryptChar(message[i], key);
}
```

This function iterates through each character in the message and applies the `encryptChar` function.

3. **`int main()`**
   - **Purpose:** The main driver function that handles user input, performs encryption, and writes the result to a file.
   - **Process:**
     - Prompts the user to enter a message and an integer key.
     - Calls the `encryptMessage` function to encrypt the user-provided message.
     - Saves the encrypted message to a file named `Encrypted_message.txt`.
   - **Error Handling:** Checks if the file was successfully opened for writing and provides appropriate error messages.

How It Works

1. **Input Collection:**
   - The user is prompted to enter a message which will be encrypted.
   - The user is then prompted to enter an integer key for encryption.
2. **Encryption Process:**
   - The message is encrypted character by character using the provided key. Each character's ASCII value is incremented by the key value.
3. **File Output:**
   - The encrypted message is saved to a file, ensuring that the original message is securely stored in its encrypted form.

Compilation and Execution

1. **Compile the Program:**

```
bash
Copy code
gcc -o encryption_tool encryption_tool.c
```

2. **Run the Program:**

```
bash
Copy code
./encryption_tool
```

3. **User Input:**
   - Enter the message to be encrypted.
   - Enter the encryption key (an integer).
4. **Output:**
   - The encrypted message is saved to `Encrypted_message.txt`.

Example

Input:

```
makefile
Copy code
Message: Hello World!
Key: 3
```

Output in `Encrypted_message.txt`:

```
Copy code
Khoor Zruog!
```

Explanation: Each character in the message "Hello World!" is shifted by 3 positions in the ASCII table, resulting in the encrypted message "Khoor Zruog!".