Name: Fabiola INGABIRE
ADDIDAS ASSIGNMENT
Robotic Team

ON 24th June 2024

Encoding and Decoding Program

Introduction

This project involves an encoding and decoding system implemented in C. The system reads characters from an input file, encodes them by adding parity bits, and writes the encoded data to a new file. It then decodes the data, removing the parity bits and reconstructing the original characters in another file.

Program Overview

Encoding Process

1. Reading Input: The program reads characters from an input file.
2. Encoding Characters: Each character is converted to its binary representation, split into nibbles, and parity bits are added.
3. Writing Encoded Data: The encoded characters are written to an encoded file.

Decoding Process

1. Reading Encoded Data : The program reads pairs of encoded bytes from the encoded file.
2. Decoding Characters: Each pair is converted back to its original binary representation, combining nibbles and removing parity bits.
3. Writing Decoded Data: The decoded characters are written to a decoded file.

Code Explanation

Encoding Code:

1. Reading the Input File:
- The program starts by calling the `finalencoded` function, which opens the input file for reading (`inputfileName`) and the encoded file for writing (`encodedfileName`).
- If either file cannot be opened, it prints an error message and exits the function.

2.Reading Characters from the Input File:
- Inside a `while` loop, the program reads one character at a time from the input file using `fgetc`. This function returns the ASCII value of the next character in the file.
- If `fgetc` returns `EOF` (end-of-file), the loop breaks, ending the reading process.

3. Encoding Each Character:
- For each character read from the input file, the `encode` function is called with the character's ASCII value and an array to store the encoded result.

4. Converting Character to Binary:
- Inside the `encode` function, the `makenible` function is called to split the character's ASCII value into two 4-bit nibbles.
- The `decimalTobinary` function converts the decimal value of the character to an 8-bit binary array, which is then split into two 4-bit arrays: `msb` (most significant bits) and `lsb` (least significant bits).

5. Adding Parity Bits:
- The `addparityTonible` function is called twice, once for each nibble (`msb` and `lsb`).
- For each nibble, the function calculates parity bits using the `findpartybit` function, which checks the evenness or oddness of specific bit groups.
- The parity bits are combined with the original bits to form an 8-bit binary number, which is then converted back to a decimal value using the `binaryTodecimal` function.

6. Storing Encoded Values:
- The resulting encoded values for `msb` and `lsb` (with parity bits added) are stored in the `encoded` array.
- The encoded values are then written to the encoded file using `fwrite`.

7.Closing Files:
- Once all characters have been read and encoded, the input and encoded files are closed using `fclose`.

- Power Function: Computes the power of two using bit shifting.
- Check Even/Odd Function: Checks if two numbers are equal.
- Find Parity Bit Function: Determines the parity bit based on the nibbles.
- Decimal to Binary Function: Converts a decimal number to its binary representation.
- Binary to Decimal Function: Converts a binary array back to its decimal representation.
- Make Nibble Function: Splits a binary number into two nibbles.
- Add Parity to Nibble Function: Adds parity bits to a nibble.

- Encode Function: Combines the above functions to encode a character.
- Final Encode Function: Manages file operations for encoding.

Decoding Code

1.Opening the Encoded File:
- The program starts by calling the `finaldecoded` function, which opens the encoded file (`encodedfileName`) for reading in binary mode and the decoded file (`decodedfileName`) for writing in text mode.
- If either file cannot be opened, an error message is printed and the function exits.

2. Reading Encoded Pairs:
- Inside a `while` loop, the program reads two characters (bytes) at a time from the encoded file using `fread`. These two characters represent the encoded `msb` (most significant byte) and `lsb` (least significant byte) of the original character.
- If `fread` reads fewer than two characters (indicating the end of the file), the loop breaks.

3.Decoding Each Pair:
- For each pair of encoded characters read from the encoded file, the `decode` function is called with the `msb` and `lsb` values and an array to store the decoded result.

4.Converting Encoded Bytes to Binary:
- Inside the `decode` function, the `decimalTobinary` function is called twice, once for each encoded byte (`msb` and `lsb`), converting them to 8-bit binary arrays.

5. Extracting Original Nibbles:
- The original 4-bit nibbles are extracted from the binary arrays, ignoring the parity bits. For `msb`, the nibbles are taken from indices 1 to 4, and for `lsb`, from indices 1 to 4 as well.

6. Combining Nibbles:
- The two 4-bit nibbles are combined into a single 8-bit binary array representing the original character before encoding.

7. Converting Binary to Decimal:
- The combined 8-bit binary array is converted back to a decimal value using the `binaryTodecimal` function, representing the original ASCII value of the character.

8. Storing Decoded Character:
- The resulting decoded character is stored in the `decoded` array and then written to the decoded file using `fputc`.

9.Closing Files:
- Once all encoded pairs have been read and decoded, the encoded and decoded files are closed using `fclose`.

- Power Function: Computes the power of two using bit shifting.
- Decimal to Binary Function: Converts a decimal number to its binary representation.
- Binary to Decimal Function: Converts a binary array back to its decimal representation.
- Decode Function: Reconstructs the original character from the encoded data.
- Final Decode Function: Manages file operations for decoding.

Usage Instructions

1. Compiling the Code:
Use a `Makefile` to compile the encoding and decoding programs.

2.Running the Programs:
- To encode a file: Execute the encoding program.
- To decode a file: Execute the decoding program.

3. Example Usage:
- Ensure you have an input file named `inputfile`.
- Run the encoding program to create `encodedfile`.
- Run the decoding program to create `decodedfile`.

Conclusion

This project demonstrates a complete encoding and decoding system in C, including parity bit generation and removal. The detailed implementation ensures data integrity and provides a robust solution for encoding and decoding operations. The provided instructions should help you understand and use the program effectively.