/\*

Name :- Sk Sahil.

Date :- 14/8/2022

Description :- Project **Steganography (deode.c)**

**\*/**

#include <stdio.h>

#include "decode.h"

#include "types.h"

#include <string.h>

#include "common.h"

#include <stdlib.h>

// Function definition for read and validate decode args

Status\_d read\_and\_validate\_decode\_args(char \*argv[], DecodeInfo \*decInfo)

{

if (strcmp(strstr(argv[2], "."), ".bmp") == 0)

{

decInfo->d\_src\_image\_fname = argv[2];

}

else

return d\_failure;

if (argv[3] != NULL)

decInfo->d\_secret\_fname = argv[3];

else

decInfo->d\_secret\_fname = "decode.txt";

return d\_success;

}

// Function definition for open files for decoding

Status\_d open\_files\_dec(DecodeInfo \*decInfo)

{

//Stego Image file

decInfo->fptr\_d\_src\_image = fopen(decInfo->d\_src\_image\_fname, "r");

//Do Error handling

if (decInfo->fptr\_d\_src\_image == NULL)

{

perror("fopen");

fprintf(stderr, "ERROR: Unable to open file %s\n", decInfo->d\_src\_image\_fname);

return d\_failure;

}

//Dest file

decInfo->fptr\_d\_secret = fopen(decInfo->d\_secret\_fname, "w");

//Do Error handling

if (decInfo->fptr\_d\_secret == NULL)

{

perror("fopen");

fprintf(stderr, "ERROR: Unable to open file %s\n", decInfo->d\_secret\_fname);

return d\_failure;

}

// If no failure then return e\_success

return d\_success;

}

// Function definition for decode magic string

Status\_d decode\_magic\_string(DecodeInfo \*decInfo)

{

fseek(decInfo->fptr\_d\_src\_image, 54, SEEK\_SET);

int i = strlen(MAGIC\_STRING);

decInfo->magic\_data = malloc(strlen(MAGIC\_STRING) + 1);

decode\_data\_from\_image(strlen(MAGIC\_STRING), decInfo->fptr\_d\_src\_image, decInfo);

decInfo->magic\_data[i] = '\0';

if (strcmp(decInfo->magic\_data, MAGIC\_STRING) == 0)

return d\_success;

else

return d\_failure;

}

// Function definition for decoding data fom image

Status\_d decode\_data\_from\_image(int size, FILE \*fptr\_d\_src\_image, DecodeInfo \*decInfo)

{

int i;

char str[8];

for (i = 0; i < size; i++)

{

fread(str, 8, sizeof(char), fptr\_d\_src\_image);

decode\_byte\_from\_lsb(&decInfo->magic\_data[i], str);

}

return d\_success;

}

// Function definition for decode byte from lsb

Status\_d decode\_byte\_from\_lsb(char \*data, char \*image\_buffer)

{

int bit = 7;

unsigned char ch = 0x00;

for (int i = 0; i < 8; i++)

{

ch = ((image\_buffer[i] & 0x01) << bit--) | ch;

}

\*data = ch;

return d\_success;

}

// Function definition for decode file extn size

Status\_d decode\_file\_extn\_size(int size, FILE \*fptr\_d\_src\_image)

{

char str[32];

int length;

fread(str, 32, sizeof(char), fptr\_d\_src\_image);

decode\_size\_from\_lsb(str, &length);

if (length == size)

return d\_success;

else

return d\_failure;

}

// Function definition decode size from lsb

Status\_d decode\_size\_from\_lsb(char \*buffer, int \*size)

{

int j = 31;

int num = 0x00;

for (int i = 0; i < 32; i++)

{

num = ((buffer[i] & 0x01) << j--) | num;

}

\*size = num;

}

// Function definition for decode secret file extn

Status\_d decode\_secret\_file\_extn(char \*file\_ext, DecodeInfo \*decInfo)

{

file\_ext = ".txt";

int i = strlen(file\_ext);

decInfo->d\_extn\_secret\_file = malloc(i + 1);

decode\_extension\_data\_from\_image(strlen(file\_ext), decInfo->fptr\_d\_src\_image, decInfo);

decInfo->d\_extn\_secret\_file[i] = '\0';

if (strcmp(decInfo->d\_extn\_secret\_file, file\_ext) == 0)

return d\_success;

else

return d\_failure;

}

// Function definition decode extension data from image

Status\_d decode\_extension\_data\_from\_image(int size, FILE \*fptr\_d\_src\_image, DecodeInfo \*decInfo)

{

for (int i = 0; i < size; i++)

{

fread(decInfo->d\_src\_image\_fname, 8, 1, fptr\_d\_src\_image);

decode\_byte\_from\_lsb(&decInfo->d\_extn\_secret\_file[i], decInfo->d\_src\_image\_fname);

}

return d\_success;

}

// Function definition for decode secret file size

Status\_d decode\_secret\_file\_size(int file\_size, DecodeInfo \*decInfo)

{

char str[32];

fread(str, 32, sizeof(char), decInfo->fptr\_d\_src\_image);

decode\_size\_from\_lsb(str, &file\_size);

decInfo->size\_secret\_file = file\_size;

//printf("%d\n", file\_size);

return d\_success;

}

// Function definition for decode secret file data

Status\_d decode\_secret\_file\_data(DecodeInfo \*decInfo)

{

char ch;

for (int i = 0; i < decInfo->size\_secret\_file; i++)

{

fread (decInfo->d\_src\_image\_fname, 8, sizeof(char), decInfo->fptr\_d\_src\_image);

decode\_byte\_from\_lsb(&ch, decInfo->d\_src\_image\_fname);

fputc(ch, decInfo->fptr\_d\_secret);

}

return d\_success;

}

// Function definition for do decoding

Status\_d do\_decoding(DecodeInfo \*decInfo)

{

if (open\_files\_dec(decInfo) == d\_success)

{

printf("Open files is a success\n");

if (decode\_magic\_string(decInfo) == d\_success)

{

printf("Decoded magic string Successfully\n");

if (decode\_file\_extn\_size(strlen(".txt"), decInfo->fptr\_d\_src\_image) == d\_success)

{

printf("Decoded file extension size Succesfully\n");

if (decode\_secret\_file\_extn(decInfo->d\_extn\_secret\_file, decInfo) == d\_success)

{

printf("Decoded Secret File Extension Succesfully\n");

if (decode\_secret\_file\_size(decInfo->size\_secret\_file, decInfo) == d\_success)

{

printf("Decoded secret file size Successfully\n");

if (decode\_secret\_file\_data(decInfo) == d\_success)

{

printf("Decoded secret file data Succuessfully\n");

}

else

{

printf("Decoding of secret file data is a failure\n");

}

}

else

{

printf("Decode of secret file size is a failure\n");

return d\_failure;

}

}

else

{

printf("Decode of Secret file extension is a failure\n");

return d\_failure;

}

}

else

{

printf("Decoded of file extension size is a failure\n");

return d\_failure;

}

}

else

{

printf("Decoding of magic string is a failure\n");

return d\_failure;

}

}

else

{

printf("Open files is a failure\n");

return d\_failure;

}

return d\_success;

}