## nitic_ctf{xor+substitution+block-cipher}

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

#define BLOCK_LEN 256

unsigned char idx_table[BLOCK_LEN] = {
    #include "./idx_table.csv"
};

const unsigned char* substitution_table =
    #include "./substitution_table.txt"
;

void encrypt(unsigned char input_buf[BLOCK_LEN], unsigned char
out_buf[BLOCK_LEN]) {
    unsigned char xor_table[BLOCK_LEN] = {
        #include "./xor_table.csv"
    };
    for (int i = 0; i < BLOCK_LEN; ++i) {
        out_buf[i] = substitution_table[input_buf[idx_table[i]]] ^ xor_table[i];
    }
}

int main(int argc, unsigned char const *argv[]) {
    if (argc < 2) {
        printf("Usage: %s ./file\n", argv[0]);
        return 0;
    }

    FILE* src_fp = fopen(argv[1], "rb");
    if (src_fp == NULL) {
        printf("Could not open %s\n", argv[1]);
        return 1;
    }

    unsigned char dst_filename[256+4];
    strncpy(dst_filename, argv[1], 256);
    strcat(dst_filename, ".enc");

    FILE* dst_fp = fopen(dst_filename, "wb");
    if (dst_fp == NULL) {
        printf("Could not open %s\n", dst_filename);
        return 1;
    }
```

```
    unsigned char input_buf[BLOCK_LEN], encrypted_buf[BLOCK_LEN];
    int read_len;
    do {
        read_len = fread(input_buf, sizeof(unsigned char), BLOCK_LEN, src_fp);
        if (read_len < BLOCK_LEN) {
            if (read_len == 0) break;
            for (int i = read_len; i < BLOCK_LEN; ++i) {
                input_buf[i] = '\x00';
            }
        }
        encrypt(input_buf, encrypted_buf);
        size_t wrriten_len = fwrite(encrypted_buf, sizeof(unsigned char),
BLOCK_LEN, dst_fp);
        if (wrriten_len < BLOCK_LEN) {
            printf("Failed to write to %s\n", dst_filename);
            return 1;
        }
    } while (read_len == BLOCK_LEN);

    fclose(src_fp);
    fclose(dst_fp);

    return 0;
}
```