# EC504 2021F Project: Dropbox-like Storage

Zhiyuan Liu

Jiawei Zhao

Dec 09, 2021

# Contents

1. Problem Proposal
2. Implementation
3. Code Intro
4. Features Implemented
5. Further Work

# Project Proposal

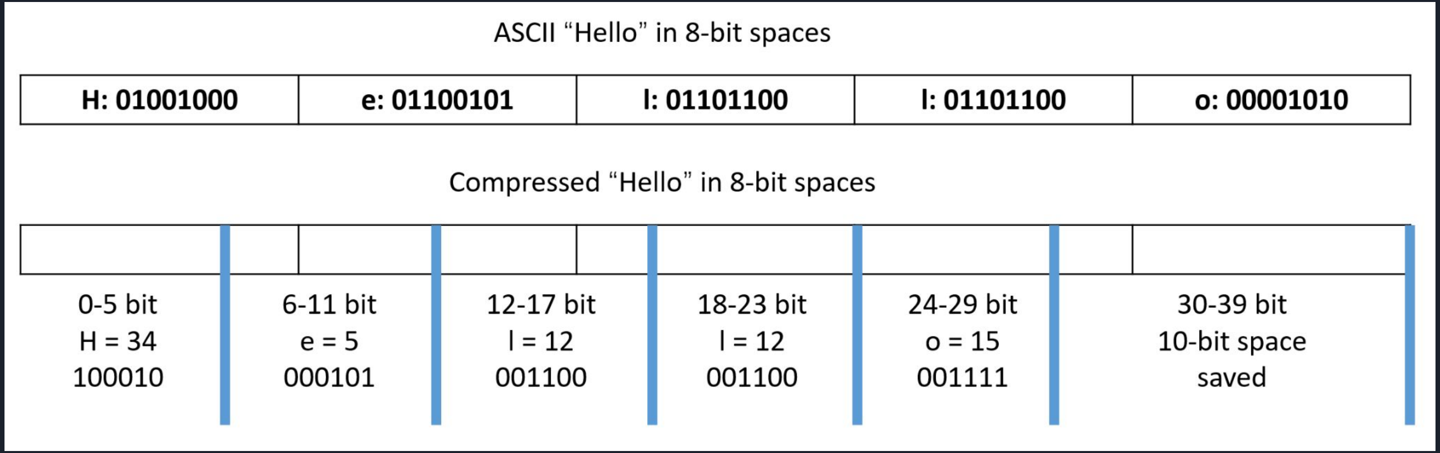
Create a dropbox-like data storage system, which could:

1. Creating a local file system
2. Loading and encoding of text files
3. retrieving back and decoding of text files
4. listing of loaded files
5. deleting of text files

# Implementation - Data Compression

Fixed-length Huffman code: convert 8-bit ASCII code into 6-bit representations.

Compression ratio: 6 / 8 = 0.75



ASCII “Hello” in 8-bit spaces

| | | | | |
|---|---|---|---|---|
| **H: 01001000** | **e: 01100101** | **l: 01101100** | **l: 01101100** | **o: 00001010** |

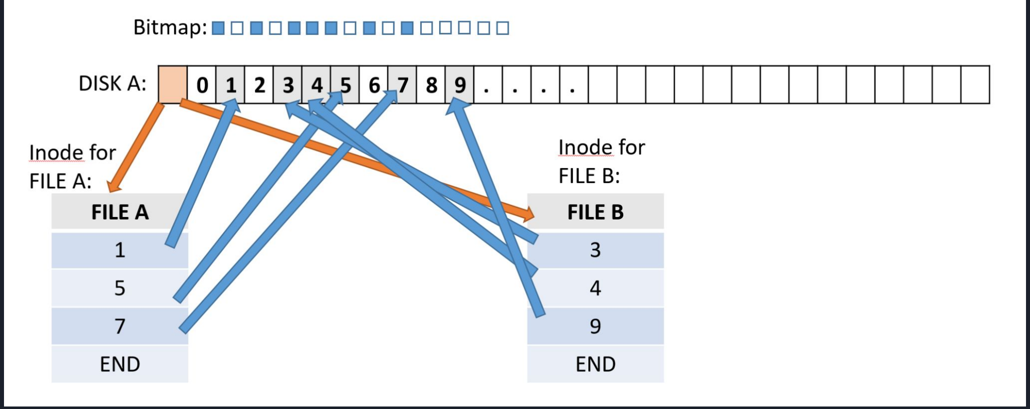
Compressed “Hello” in 8-bit spaces

| | | | | | |
|---|---|---|---|---|---|
| 0-5 bit<br>H = 34<br>100010 | 6-11 bit<br>e = 5<br>000101 | 12-17 bit<br>l = 12<br>001100 | 18-23 bit<br>l = 12<br>001100 | 24-29 bit<br>o = 15<br>001111 | 30-39 bit<br>10-bit space<br>saved |

# Implementation - Virtual file system

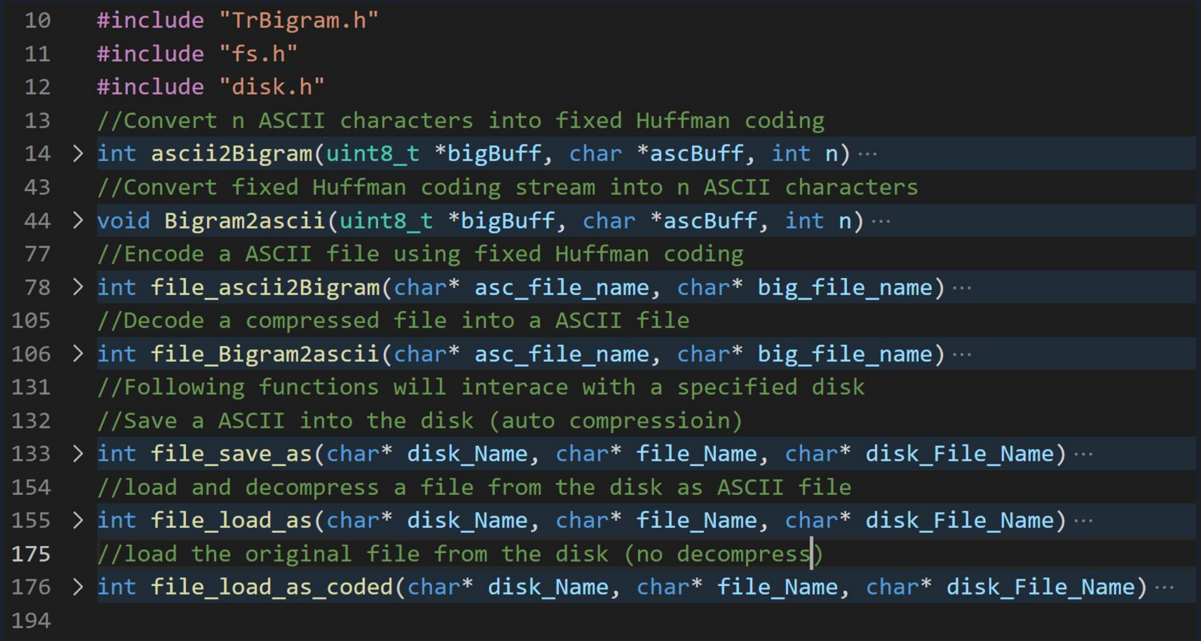
1. Meta Data
   a. Use a bitmap to manage all 1-Kb chunks
   b. Store the files’ directories and inodes  in separate chunk

2. Data Compression
   a. Huffman Coding
   b. Compare and  reuse chunks for similar files



# Code Intro

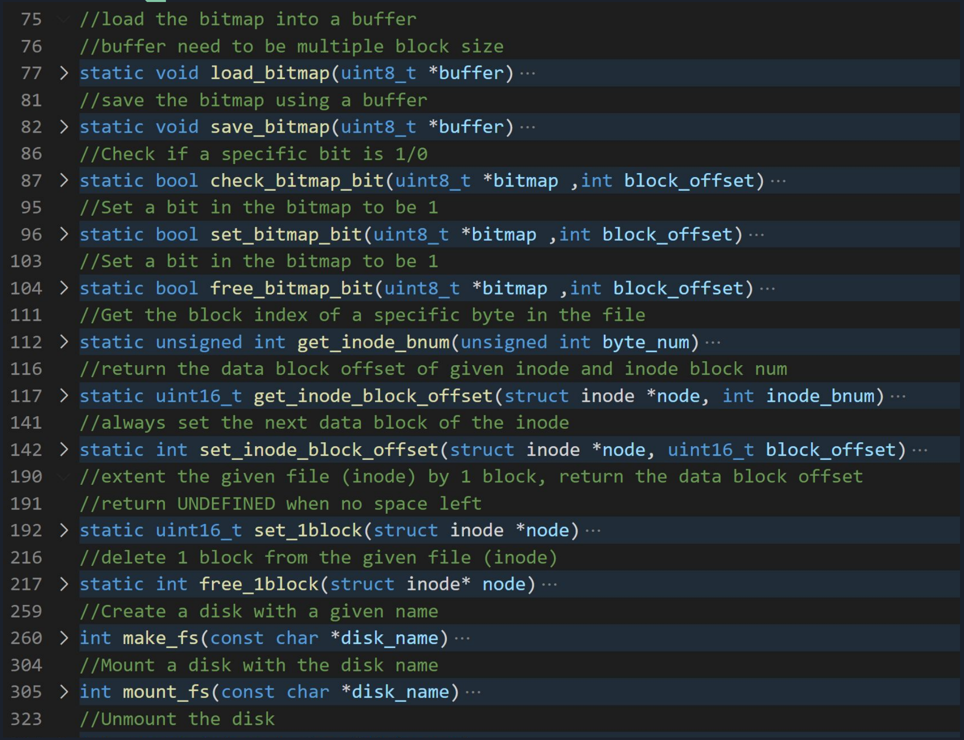
## Functions for Data Compression



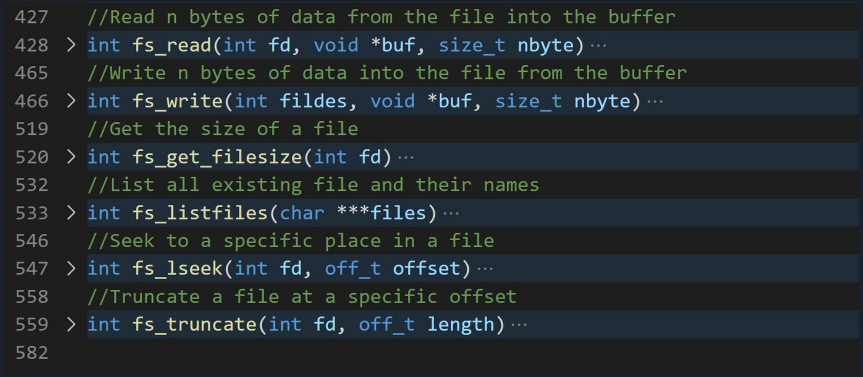
```c
#include "TrBigram.h"
#include "fs.h"
#include "disk.h"
//Convert n ASCII characters into fixed Huffman coding
int ascii2Bigram(uint8_t *bigBuff, char *ascBuff, int n)
//Convert fixed Huffman coding stream into n ASCII characters
void Bigram2ascii(uint8_t *bigBuff, char *ascBuff, int n)
//Encode a ASCII file using fixed Huffman coding
int file_ascii2Bigram(char* asc_file_name, char* big_file_name)
//Decode a compressed file into a ASCII file
int file_Bigram2ascii(char* asc_file_name, char* big_file_name)
//Following functions will interace with a specified disk
//Save a ASCII into the disk (auto compressioin)
int file_save_as(char* disk_Name, char* file_Name, char* disk_File_Name)
//load and decompress a file from the disk as ASCII file
int file_load_as(char* disk_Name, char* file_Name, char* disk_File_Name)
//load the original file from the disk (no decompress)
int file_load_as_coded(char* disk_Name, char* file_Name, char* disk_File_Name)
```

# Code Intro

## Functions for Virtual File System



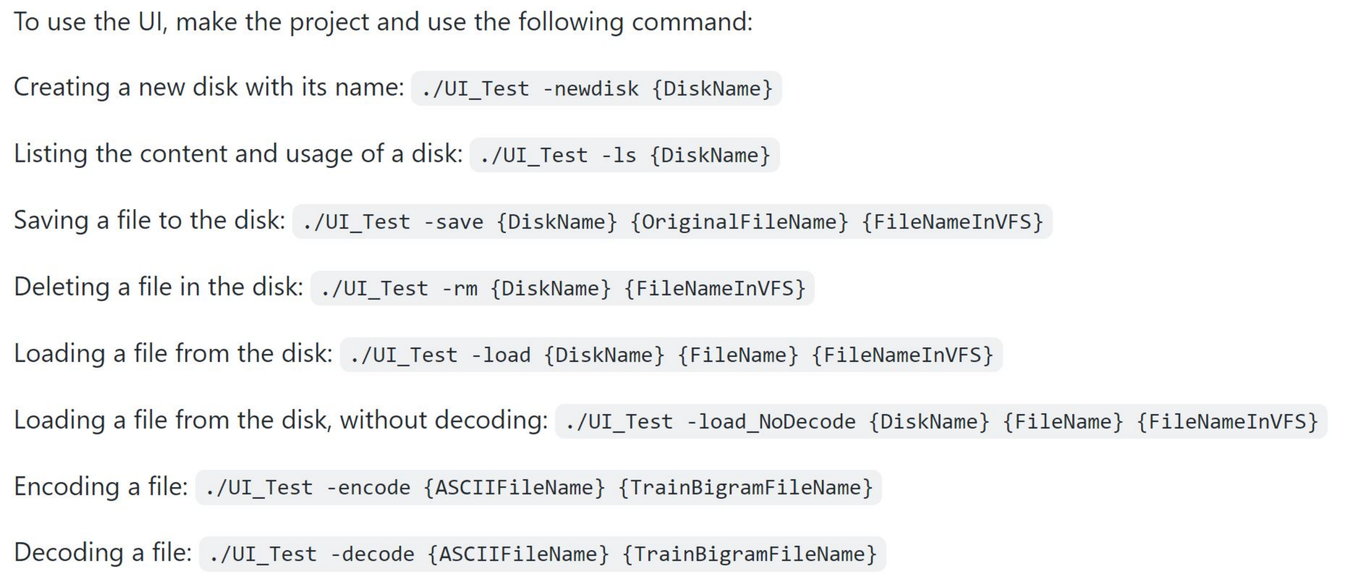
```
//load the bitmap into a buffer
//buffer need to be multiple block size
static void load_bitmap(uint8_t *buffer) ···
//save the bitmap using a buffer
static void save_bitmap(uint8_t *buffer) ···
//Check if a specific bit is 1/0
static bool check_bitmap_bit(uint8_t *bitmap ,int block_offset) ···
//Set a bit in the bitmap to be 1
static bool set_bitmap_bit(uint8_t *bitmap ,int block_offset) ···
//Set a bit in the bitmap to be 1
static bool free_bitmap_bit(uint8_t *bitmap ,int block_offset) ···
//Get the block index of a specific byte in the file
static unsigned int get_inode_bnum(unsigned int byte_num) ···
//return the data block offset of given inode and inode block num
static uint16_t get_inode_block_offset(struct inode *node, int inode_bnum) ···
//always set the next data block of the inode
static int set_inode_block_offset(struct inode *node, uint16_t block_offset) ···
//extent the given file (inode) by 1 block, return the data block offset
//return UNDEFINED when no space left
static uint16_t set_1block(struct inode *node) ···
//delete 1 block from the given file (inode)
static int free_1block(struct inode* node) ···
//Create a disk with a given name
int make_fs(const char *disk_name) ···
//Mount a disk with the disk name
int mount_fs(const char *disk_name) ···
//Unmount the disk
```



```
//Read n bytes of data from the file into the buffer
int fs_read(int fd, void *buf, size_t nbyte) ···
//Write n bytes of data into the file from the buffer
int fs_write(int fildes, void *buf, size_t nbyte) ···
//Get the size of a file
int fs_get_filesize(int fd) ···
//List all existing file and their names
int fs_listfiles(char ***files) ···
//Seek to a specific place in a file
int fs_lseek(int fd, off_t offset) ···
//Truncate a file at a specific offset
int fs_truncate(int fd, off_t length) ···
```

# Features Implemented

## User-interface Commands



To use the UI, make the project and use the following command:

Creating a new disk with its name: `./UI_Test -newdisk {DiskName}`

Listing the content and usage of a disk: `./UI_Test -ls {DiskName}`

Saving a file to the disk: `./UI_Test -save {DiskName} {OriginalFileName} {FileNameInVFS}`

Deleting a file in the disk: `./UI_Test -rm {DiskName} {FileNameInVFS}`

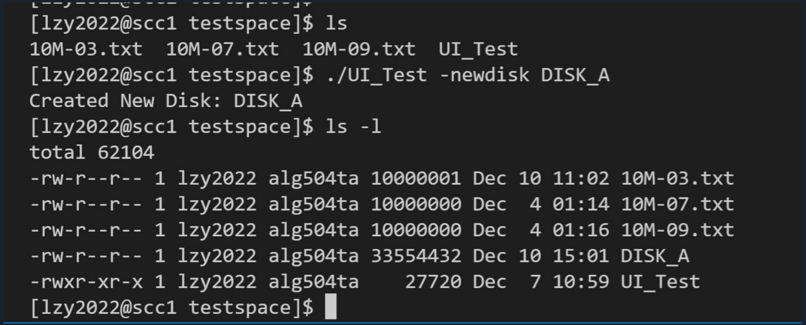
Loading a file from the disk: `./UI_Test -load {DiskName} {FileName} {FileNameInVFS}`

Loading a file from the disk, without decoding: `./UI_Test -load_NoDecode {DiskName} {FileName} {FileNameInVFS}`

Encoding a file: `./UI_Test -encode {ASCIIFileName} {TrainBigramFileName}`

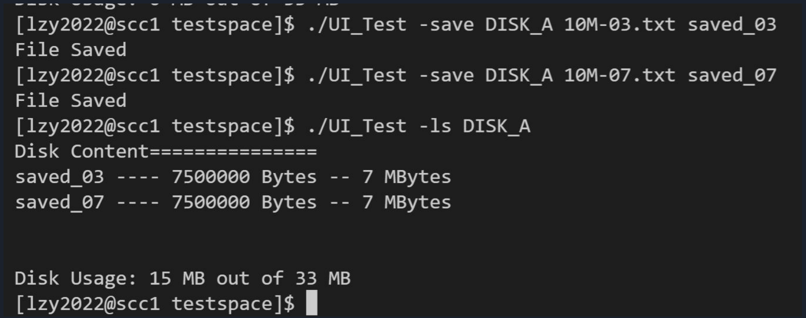
Decoding a file: `./UI_Test -decode {ASCIIFileName} {TrainBigramFileName}`

Create a disk



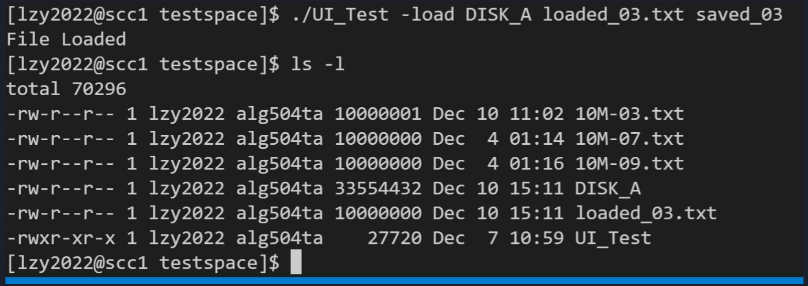
```
[lzy2022@scc1 testspace]$ ls
10M-03.txt  10M-07.txt  10M-09.txt  UI_Test
[lzy2022@scc1 testspace]$ ./UI_Test -newdisk DISK_A
Created New Disk: DISK_A
[lzy2022@scc1 testspace]$ ls -l
total 62104
-rw-r--r-- 1 lzy2022 alg504ta 10000001 Dec 10 11:02 10M-03.txt
-rw-r--r-- 1 lzy2022 alg504ta 10000000 Dec  4 01:14 10M-07.txt
-rw-r--r-- 1 lzy2022 alg504ta 10000000 Dec  4 01:16 10M-09.txt
-rw-r--r-- 1 lzy2022 alg504ta 33554432 Dec 10 15:01 DISK_A
-rwxr-xr-x 1 lzy2022 alg504ta    27720 Dec  7 10:59 UI_Test
[lzy2022@scc1 testspace]$ 
```

Save files



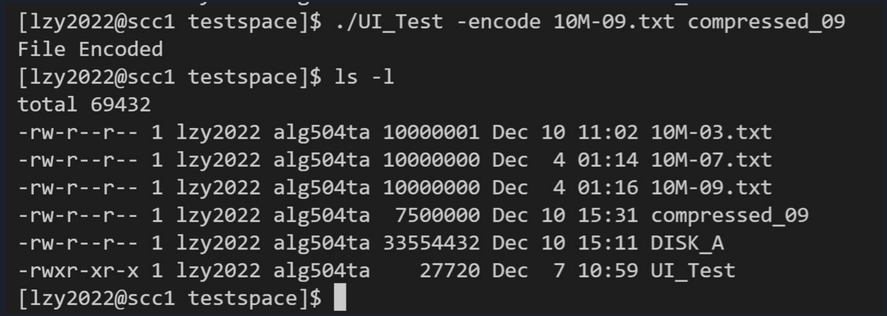
```
[lzy2022@scc1 testspace]$ ./UI_Test -save DISK_A 10M-03.txt saved_03
File Saved
[lzy2022@scc1 testspace]$ ./UI_Test -save DISK_A 10M-07.txt saved_07
File Saved
[lzy2022@scc1 testspace]$ ./UI_Test -ls DISK_A
Disk Content===============
saved_03 ---- 7500000 Bytes -- 7 MBytes
saved_07 ---- 7500000 Bytes -- 7 MBytes


Disk Usage: 15 MB out of 33 MB
[lzy2022@scc1 testspace]$ 
```

Load files



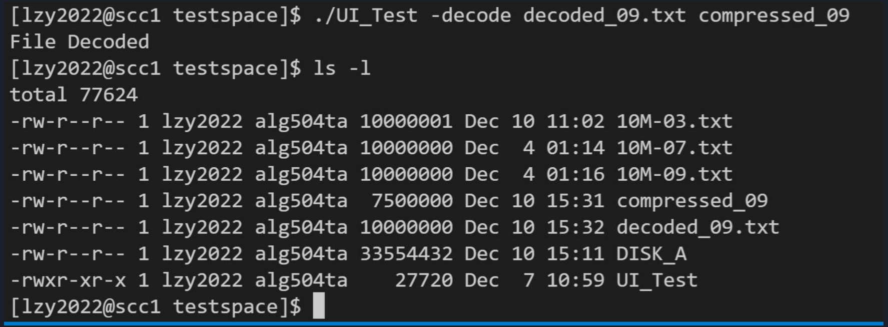
```
[lzy2022@scc1 testspace]$ ./UI_Test -load DISK_A loaded_03.txt saved_03
File Loaded
[lzy2022@scc1 testspace]$ ls -l
total 70296
-rw-r--r-- 1 lzy2022 alg504ta 10000001 Dec 10 11:02 10M-03.txt
-rw-r--r-- 1 lzy2022 alg504ta 10000000 Dec  4 01:14 10M-07.txt
-rw-r--r-- 1 lzy2022 alg504ta 10000000 Dec  4 01:16 10M-09.txt
-rw-r--r-- 1 lzy2022 alg504ta 33554432 Dec 10 15:11 DISK_A
-rw-r--r-- 1 lzy2022 alg504ta 10000000 Dec 10 15:11 loaded_03.txt
-rwxr-xr-x 1 lzy2022 alg504ta    27720 Dec  7 10:59 UI_Test
[lzy2022@scc1 testspace]$ 
```

Encode



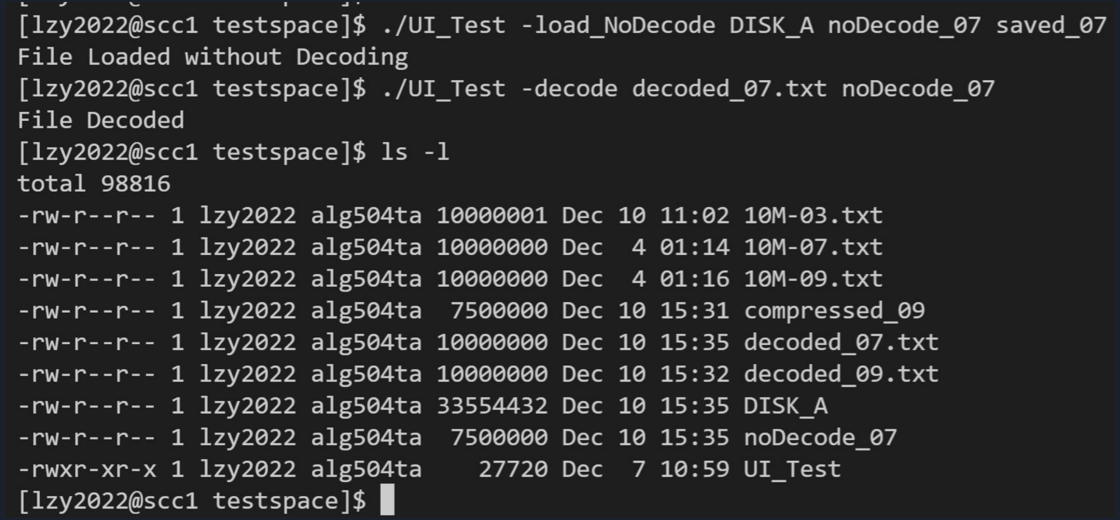
```
[lzy2022@scc1 testspace]$ ./UI_Test -encode 10M-09.txt compressed_09
File Encoded
[lzy2022@scc1 testspace]$ ls -l
total 69432
-rw-r--r-- 1 lzy2022 alg504ta 10000001 Dec 10 11:02 10M-03.txt
-rw-r--r-- 1 lzy2022 alg504ta 10000000 Dec  4 01:14 10M-07.txt
-rw-r--r-- 1 lzy2022 alg504ta 10000000 Dec  4 01:16 10M-09.txt
-rw-r--r-- 1 lzy2022 alg504ta  7500000 Dec 10 15:31 compressed_09
-rw-r--r-- 1 lzy2022 alg504ta 33554432 Dec 10 15:11 DISK_A
-rwxr-xr-x 1 lzy2022 alg504ta    27720 Dec  7 10:59 UI_Test
[lzy2022@scc1 testspace]$ 
```

Decode



```
[lzy2022@scc1 testspace]$ ./UI_Test -decode decoded_09.txt compressed_09
File Decoded
[lzy2022@scc1 testspace]$ ls -l
total 77624
-rw-r--r-- 1 lzy2022 alg504ta 10000001 Dec 10 11:02 10M-03.txt
-rw-r--r-- 1 lzy2022 alg504ta 10000000 Dec  4 01:14 10M-07.txt
-rw-r--r-- 1 lzy2022 alg504ta 10000000 Dec  4 01:16 10M-09.txt
-rw-r--r-- 1 lzy2022 alg504ta  7500000 Dec 10 15:31 compressed_09
-rw-r--r-- 1 lzy2022 alg504ta 10000000 Dec 10 15:32 decoded_09.txt
-rw-r--r-- 1 lzy2022 alg504ta 33554432 Dec 10 15:11 DISK_A
-rwxr-xr-x 1 lzy2022 alg504ta    27720 Dec  7 10:59 UI_Test
[lzy2022@scc1 testspace]$ 
```

Load files without decoding



```
[lzy2022@scc1 testspace]$ ./UI_Test -load_NoDecode DISK_A noDecode_07 saved_07
File Loaded without Decoding
[lzy2022@scc1 testspace]$ ./UI_Test -decode decoded_07.txt noDecode_07
File Decoded
[lzy2022@scc1 testspace]$ ls -l
total 98816
-rw-r--r-- 1 lzy2022 alg504ta 10000001 Dec 10 11:02 10M-03.txt
-rw-r--r-- 1 lzy2022 alg504ta 10000000 Dec  4 01:14 10M-07.txt
-rw-r--r-- 1 lzy2022 alg504ta 10000000 Dec  4 01:16 10M-09.txt
-rw-r--r-- 1 lzy2022 alg504ta  7500000 Dec 10 15:31 compressed_09
-rw-r--r-- 1 lzy2022 alg504ta 10000000 Dec 10 15:35 decoded_07.txt
-rw-r--r-- 1 lzy2022 alg504ta 10000000 Dec 10 15:32 decoded_09.txt
-rw-r--r-- 1 lzy2022 alg504ta 33554432 Dec 10 15:35 DISK_A
-rw-r--r-- 1 lzy2022 alg504ta  7500000 Dec 10 15:35 noDecode_07
-rwxr-xr-x 1 lzy2022 alg504ta    27720 Dec  7 10:59 UI_Test
[lzy2022@scc1 testspace]$
```

# Further Work

1. More file types should be considered.
2. Encode data using variable-length Huffman coding algorithms.

# THANK YOU