Caterin Duarte Reynosa
cduarter@ucsc.edu

CSE13S Fall 2020
Assignment 7
Design Document

## Goal

The goal for this assignment is to learn about Lempel-ziv compression and decompression. In this assignment, we will be using a Tries ADT in order to store words we can compress. We will also be using a Word ADT to decompress 'codes'.

## File I have to implement + their goal

- Encode: compresses files, texts, and binaries
- Decode: decompresses files, texts, and binaries
- Trie: the ADT for TrieNode
- Word: the ADT for Word
- Io.c: source file for i/o implementation

### Trie

- TrieNode *trie_node_create(uint16_t code)
    - Allocate memory for TrieNode* using malloc(sizeof(TrieNode)
    - Code = code
    - If != null
        - For i in range (1, alphabet (or 256):
            - trienode->children[i] = NULL
    - Return trie node
- void trie_node_delete(TrieNode *n)
    - free(trie node)
- TrieNode *trie_create(void)
    - Return trienode->code[empty_code]
- void trie_reset(TrieNode *root)
    - For i in children
        - Delete i
- void trie_delete(TrieNode *n)
    - Delete children and root recurvesily
    - For i in children
        - trie_delete(i)
    - trie_node_delete(n)
- TrieNode *trie_step(TrieNode *n, uint8_t sym)
    - If trienode->code == null
        - Return null
    - Else return trienode->code

### Word

- Word *word_create(uint8_t *syms, uint32_t len)
    - Word *sysm = (*sysm)malloc(sizeof(syms)

    - If sysm
        - sysm->len = len
        - Return syms
    - Else
        - Return NULL
- Word *word_append_sym(Word *w, uint8_t sym)
    - We shall come back to her later
- void word_delete(Word *w)
    - free(w)
- WordTable *wt_create(void)
    - Return WordTbale[ EMPTY_CODE]
- void wt_reset(WordTable *wt)
    - For i in range (1, len :
        - wt->sysms[i] = NULL

## IO

- int read_bytes(int infile, uint8_t *buf, int to_read)
    - Loop until there are no bytes left || we have read all the bytes in to_read
        - read(to_read)
        - Counter that keeps track of bytes read += 1
        - Return num of bytes read aka counter
- int write_bytes(int outfile, uint8_t *buf, int to_write)
    - Loop until we have written all the bytes|| we didn't write any bytes
        - read(to_read)
        - Counter that keeps track of bytes written += 1
        - Return num of bytes written aka counter
- void read_header(int infile, FileHeader *header)
    - read_byte(infile, header, sizeof(FileHeader)
    - If !small_endian
        - Swap endianess
- void write_header(int outfile, FileHeader *header)
    - write_byte(infile, header, sizeof(FileHeader)
    - If !small_endian
        - Swap endianess
- bool read_sym(int infile, uint8_t *sym)
    - If read_bytes(infile, magicnum?) != magic number
        - Bytes += 1
        - ?
        - Shall come back later
- void write_pair(int outfile, uint16_t code, uint8_t sym, int bitlen)
    - Will come back and do later…...
- void flush_pairs(int outfile)
    - Will come back and do later…...
- bool read_pair(int infile, uint16_t *code, uint8_t *sym, int bitlen)
    - Will come back and do later…...

- void write_word(int outfile, Word *w)
    - Will come back and do later…...
- void flush_words(int outfile)
    - Will come back and do later…...

**Compression**

- Take command line options: v, i, o
- Open file
- If i wasn't specified, i = stdin
- Use fstat
- Open outfile
- If o wasn't specified, o = stdout
- Write out the filled file header
- Create a tie
    - See the pseudo for the specifics things each thing ==
- Init a counter to keep track of next available word
- Create counters prev node and prev sysm
- Use read_sysm in a loop until it returns false
    - See pseudo for specifics things needed to do inside the loop
- Check of current node == root trie node
    - Increment counter
- Write the pair
- Flush pair
- Close the files

**Decompression**

- Take command line options: v, i, o
- Open file
- If i wasn't specified, i = stdin
- Use fstat
- Open outfile
- If o wasn't specified, o = stdout
- Write out the filled file header
- Create a word table
    - See the pseudo for the specifics things each thing ==
- Init a counter to keep track of current and next code
- Use read_apir in a loop until it returns false
    - See pseudo for specifics things needed to do inside the loop
- Check of code == stop_code
    - Increment counter
- Flush pair
- Close the files