Compressing Files

CS3411 Spring 2013
Program Three
Due: April 2, 2013, class time

In this project, we will develop a program called *czy* which compresses the data stream given to its standard input and write the compressed stream to its standard output and another one called *dzy* which decompresses a compressed stream directed at its standard input and writes decompressed data to its standard output.

The compression algorith we'll employ is a variation of a technique called *run-length encoding*. A run-length encoding compression scheme counts repeated sequences of symbols and stores the count of symbols followed by the symbol. In this project, our symbols are characters (8-bit sequences of data). There are two data formats used for encoding/decoding:

Infrequent symbol is a 9 bit quantity such that the leading bit is one and the next 8 bits indicate the character (i.e., 1-<symbol>).

Frequent symbol is a 9 bit quantity such that the leading bit is zero, followed by a 4-bit repeat count, followed by a 4-bit frequent character code. A repeat-count of zero means the character appears once.

The compression algorithm is outlined below:

1. Count the frequency of symbols and sort them from highest frequency down to lowest. Top 16 characters make-up the frequent characters, and they are assigned codes 0-15. This is the *dictionary*.
2. Output the 16-byte dictionary.
3. Seek the input file to the beginning (if the input is not seekable, give an error message and exit).
4. Read the next character.
5. If the current character is a frequent character, calculate the run-length (i.e., find out how many times it repeats) for a maximum of 16. Consume that many characters from the input. Output the current frequent character encoding using 9 bits.
6. If the current character is not a frequent character, form an infrequent character encoding by prefixing the current character with a binary one and output 9 bits of encoded data.
7. repeat the above steps until the end of file.
8. pad with zero bits to a byte boundary as needed.

The decoding algorithm is outlined below:

1. Load 16 bytes from the file into the *Dictionary*.
2. read 1 bit from the file. If it is a zero, read 4 more bits and store these into repeat-count. Read the next 4 bits from the file. This is the character code **code**. Dictionary[code] gives the symbol. Output this symbol as many times as indicated by the repeat-count.
3. if the bit is one, output the next 8 bits from the input as the decoded symbol.

In the following example, the diff should not report any differences:

```
czy < input-file  | dzy > new.input-file
diff input-file new.input-file.
```

Once you make sure your program is working correctly, use your program to compress its own source and its own object, i.e., czy.c and czy. Compress the same programs with gzip as well. Write a README file and indicate the compression ratio's you obtained in this README file.

Submission Requirements

Your submission must be written in C.

Use Canvas to submit a tar file named `prog3.tgz` that contains:

- A copy of the source **with comments.**
- A makefile with *all*, *czy*, *dzy* and *clean.*
- A README file showing statistics.
- A file named TESTS in the main project directory that contains a description of the test cases you executed against the code to verify correct operation.

When I execute the commands: `gtar zxf prog3.tgz; make` against your submission, two executables named `czy` and `dzy` should be created in the current directory. No hardcopy submission is required.