STONE ANALYSIS

DOES IT REALLY GO BACK TO THE JURASSIC ERA?

STONE ANALYSIS

binary name: stone_analysis
language: everything working on “the dump”
compilation: when necessary, via Makefile, including re, clean and fclean rules

- ✓ The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.
- ✓ All the bonus files (including a potential specific Makefile) should be in a directory named bonus.
- ✓ Error messages have to be written on the error output, and the program should then exit with the 84 error code (0 if there is no error).

This is it! After many conspiracies and orbits simulations, the stone is finally in your hands. However, you still need to analyze it to explore its full potential.
It seems that the stone has its own way to communicate, and that goes by producing weird noises. No pattern seems to emerge from it, however further investigation needs to be made, as there may be some message hidden inside the sound waves themselves!

The project

You need to create a program that reads an audio signal from a wav file, and either hides a secret message inside, decodes the message hidden inside it or find the *N* top frequencies in it.

The wav file that contains the captured signal needs to be decomposed into discrete frequencies using Discrete Fourrier Transform (DFT), and eventually recomposed using the Inverse Discrete Fourrier Transform (IDFT). The secret message may be hidden by mapping specific characters to specific sound wave frequencies, or you could implement an algorithm of your own (see next section: *Steganography*).

When analyzing the signal, the top *N* frequencies must be displayed in the decreasing order of their magnitude. If two frequencies have the same magnitude, they can be displayed interchangeably.

The audio files will always be encoded in *16bits WAVE PCM monocanal* at a sampling rate of *48kHz*.

Coupling a regular naive DFT/IDFT implementation with an algorithmic complexity of $O(n^2)$, with a high-level language with high execution time (like Python) is probably a bad idea...

Steganography

You must hide (and retrieve) a message inside the audio file. You need to handle at least every digit and letter (without the case) from the latin alphabet, as well as the *SPACE* character.

You are free to use any method to cypher/decypher the message as long as you follow the next few rules:

- ✓ The output wav file **MUST** have strictly the same header as the original file.
- ✓ The output file **MUST NOT** be perceptibly different from the original file.
- ✓ Your encoding algorithm **SHOULD NOT** be too trivial.

Any library handling **audio editing**, **Fourier transforms** (such as *numpy.fft* in python) or **steganography** is *de facto* forbidden!

Examples

```
~/G-CNA-400> ./stone_analysis --help
USAGE
./stone_analysis [--analyze IN_FILE N | --cypher IN_FILE OUT_FILE MESSAGE | --decypher
IN_FILE]

IN_FILE      An audio file to be analyzed
OUT_FILE     Output audio file of the cypher mode
MESSAGE      The message to hide in the audio file
N          Number of top frequencies to display
```

```
~/G-CNA-400> ls ; ./stone_analysis --cypher input.wav output.wav ‘‘Miaou’’ && du -b *.wav
input.wav stone_analysis
368492   input.wav
368492   output.wav
```

```
~/G-CNA-400> ./stone_analysis --decypher output.wav | cat -e
MIAOU$
```

```
~/G-CNA-400> ./stone_analysis --analyze input.wav 3 | cat -e
Top 3 frequencies:$
494.0 Hz$
330.0 Hz$
415.0 Hz$
```

Bonus

- ✓ Encode the whole ASCII table (be careful about **Nyquist frequency**!)
- ✓ Spectral visualization
- ✓ Add advanced sound editing
- ✓ Encode message using Discrete Wavelet Transform (DWT)

v 1.1

{EPITECH}