MCS - Lab 1 - Drum challenge in Erlang

Team members

- Valentin Minder
- Mélanie Huck
- James Nolan

Introduction

In this second exercise of lab 1, we will try to solve the drum challenge, originally written for Go developers. The challenge requires a good handling of binary files, which makes Erlang an interesting alternative language to solve the challenge.

API methods

Our code being heavily commented, we chose to reproduce our functions' documentation in the report.

Public methods

decode_file/1

Decode a file.

- **Input**: file name of .splice file
- **Output**: {ok, VersionName, Tempo, Tracks} with Tracks as {TrackNo, Instrument, MeasureStruct} with MeasureStruct as an array of array of bytes, grouped 4 by 4

render_file/1

Decode and renders a file.

- **Input**: file name of .splice file
- **Output**: {ok, Render} where Render is a List (string) representing the file.

Internal / Parsing methods

parse_header/1

Parse a binary file and extract headers, leaving the content as binary. Returns error parse_header if magic is incorrect or not enough data provided. Any subsequent data is unused.

- **Input**: a binary of .slice file
- **Ouput**: a tuple {ok, VersionName, Tempo, TracksBinary}

parse_tracks/1

Recursively parse tracks binary and extract their headers and their content, the measure.

- **Input**: a binary of tracks
- **Output**: an array of tracks, each containing {TrackNo, Instrument, MeasureStruct} MeasureStruct as defined in parse_measure(Bin).

parse_measure/1

Parse a binary Measure and returns its structure as an array of array Byte can only have values 0 or 1, otherwise an error is returned.

- **Input**: binary representation of a Measure Example: <<0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1>>
- **Output**: an array of array of bytes, grouped 4 by 4 Example: [[0, 0, 0, 1], [0, 0, 0, 0], [0, 0, 0, 1], [0, 0, 0, 1]]

parse_measure_no_validation/1

Parse without validation of bytes

validate_measure/1

Validation of bytes of the Measure. If failing the whole group is returned.

validate_group_within_measure/1

Validation of group with measure : each byte is either 0 or 1.

Internal / rendering methods

render/3

Renders the header of the structure and calls the rendering of tracks

- **Input**: a Tuple of Version, Tempo, and array of Tracks Example: {"foo bar 42", 120.0, [Tracks]}
- **Output**: a List, rendering of headers (each on its line), rendering of all tracks
- **Example**: "Saved with HW Version: foo bar 42\nTempo: 120\nTracksRendering"

float_format/2

Takes a Float and returns a List, formatted as follow:

- if the Float with Precision decimals evaluates to an Int, the Int representation (eg: Float 12.0 -> "12")
- otherwise, the Float representation with Precision digits (eg: Float 12.432, Precision 1 -> "12.4")

render_tracks/2

Recursively pretty render of tracks, for each calls the rendering of its Measure

- **Input**: An array of Tracks, A integer Example: [{0, "kick", Measure}]
- **Ouput**: a List (string) rendering all Tracks, each Track rendered on its own line. Example: "(0) kick\tMeasureRender\n"

render_measure/1

Pretty render of a measure

- **Input**: an array of arrays of byte, called a Measure Example: [[0, 0, 1, 0], [0, 0, 1, 0], [1, 0, 1, 0], [0, 0, 1, 0]]
- **Output**: a List (string), where global array elements separated by |, 0 is -, 1 is x Example: |--x-|--x-|x-x-|--x-|

render_group/1

Pretty render of a group within a measure

- **Input**: an array of byte, called a Group Example: [1, 0, 1, 0]
- **Output**: a List (string), where 0 is -, 1 is x Example: x-x-

binary_to_string/1

Takes a Bin and returns the list represented, without trailing zeroes.

remove_trailing_zeroes/1

Takes a list, return the list limited up to the first 0 found, not included.

Conclusion

Both exercises in this first lab showed how useful were Erlang's built-in mecanisms to decode binary files based on a specified protocole.

Ressources

- A copy of the Drum challenge
- A given list of tests to pass