Lab 6 – Regular Expressions

1. **Introduction**: My lab uses regular expressions to examine data and check for valid phone numbers, valid addresses, and sorts the log entries from a log file. My program can be used to identify if a phone number is a valid American or international Phone number. It can also be used to identify if an address is correctly formatted. And lastly it can find and print any insufficient privilege error messages from appropriately formatted logs.
2. **Process**: The regex crate allows rust to use regular expressions on strings of data. These data sets, or haystacks, contain information that is matched to a defined regular expression. Most of my time on the lab was dedicated to creating these regular expressions. The haystacks are loaded from the provided example files. Other haystacks can easily be provided by editing the txt files in my project directory.

The rest of the program is dedicated to printing the results generated by the regular expressions. Phone numbers are compared against both a North American phone number regex and an international format regex. All phone numbers are displayed in a column and two rows display whether they match the expressions.

Addresses are similar except they have only a single regular expression to check against and therefore only a single column to display if the expression matches.

Logs are displayed only if they match the regex, unlike phones and addresses where every line from the file is printed.

Bellow are my regular expressions:

let rgx\_american\_number = r"(?x)

        ^.\* # Beginning of line

        \(??\d {3,4}\)??  # Optional area code with or without parenthesis

        [-,\s]?? \d{3}    # Three digits

        [-,\s]?? \d{4}    # Last Four digits";

    let international\_number = r"(?x)^\+

        (?:\d\s?){1,3}

        (\(\d{1,4}\)

        |

        \d{1,4})

        (?:[.\-\s]?\d{1,})+$";

    let rgx\_addresses = r"(?xs)

        ^.\*        # Beginning of address

        [,\s]      # Space or comma before state

        [A-Z]{2}   # State

        [,\s]{1,2} # Space after state

        [0-9]{5}   # Zip";

    let rgx\_log = r"(?x)

        v10\.4\.6 .\*? # Version number v10.4.6

        ERROR .\*?     # Error

        insufficient \s permissions .\*? # message

        ";

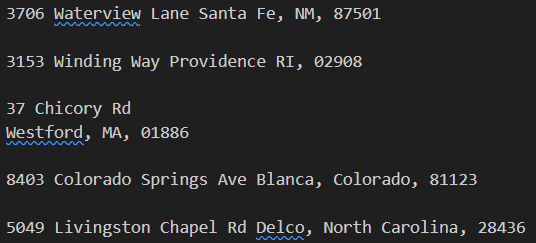
1. **Testing**: For my test I used the three files provided. Key snippets are show below.

**Phone\_numbers.txt** includes a variety of phone numbers in standard US formats with optional parenthesis around the area code and spaces or dashes separating the segments. Also European phone numbers are given with foreign area codes and strangely segmented numbers still separated by spaces or dashes.

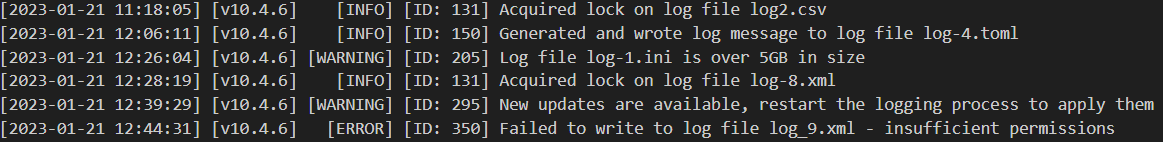
A screenshot of a computer

Description automatically generated

**Adresses.txt** includes domestic addresses one or two lines. Some addresses have the State spelled out instead of using a two-letter abbreviation.



**Log\_file.txt** includes a sample log from a computer. Each entry includes a time, log version, warning level, ID, and error message.



1. **Results**: Here is the output for the phone numbers. Not that my regex script for American numbers has some false positives, like on the second number, which starts with +44 clearly marking it as non-American to a human but being marked as American by my regex. My script does check for +1 numbers and accepts them but does not exclude +*n* numbers where *n* is not one. Valid internation number works much better. This is mainly due to the fact that all international numbers provided have a country code. Brits (and Norther Irish) calling domestic numbers do not include the +44, but the provided numbers are from an US perspective where the country codes of all international numbers are included.

Note: I haven’t shown most of the domestic phone numbers in the middle of phone\_numbers.txt because they all exclusively match American as they should.

A screenshot of a computer

Description automatically generated

A black and white screen with white x and x

Description automatically generated

**Addresses:**

The multi-line addresses match but only show that they match on the last line. See below.



Also, the addresses file contains several addresses with the state spelled out. I have created a regex script that rejects these addresses as they are not valid, as per [USPS 28.2.22.224](https://pe.usps.com/text/pub28/pub28c2_009.htm). With these considerations in mind my regex is accurate for the provided addresses.

A computer screen with white text

Description automatically generated

**Log:**

The entire log file is scanned and each line that matches the following conditions is displayed. The entry must be from version v 10.4.6, have a level of ERROR, and the message must contain “insufficient permission”. In effect this log gives all insufficient permission errors from the current version and prints the full line to the console, as seen below.

A screen shot of a computer

Description automatically generated

1. **Conclusion**: The hardest part of this lab was creating the regex scripts. Still there are small errors in my expressions. With phone numbers I don’t reject any phone numbers that do match but I do match a few numbers that should be rejected. This is because the American regex will identify the country codes in some European numbers as part of the area code, so the shorter European numbers be incorrectly identified as being American length twelve numbers.

To make the lab more rust intensive I decided to lightly format the outputs into tables. I cropped most of this out of my report as it reduces readability when in a word document but in a command prompt my output looks good and is quickly readable.

1. **References and Acknowledgements:** I used the Regex crate and used its official documentation as a reference for creating my expressions. I also worked with my classmate Emery L. and used her international phone number regex as it worked much better than mine.
2. **Extra credit**: Does this count? I used rust in lab 5 in a non-trivial way. I already got credit for 5 so I understand if it is not acceptable as EC for lab 6. My regex splits the left set from the set of tuples on the right and trims excess brackets.

A screen shot of a computer

Description automatically generated

