COL226: Programming Languages
Assignment 1
Converting Files Between Different Data Formats

Submission Deadline: Sunday, 25 Feb 2021 23:59
Submission Deadline with Late Penalty Friday, 2 Mar 2021 23:59

Preface

A character-separated (Libre-office terminology) or comma-seperated (MS-Excel terminology) values (CSV) file is a delimited text file that uses a special character, usually a comma to separate values. Each line of the file is a data record. Each record consists of one or more fields, separated by character(s). The use of the character(s) as a field separator is the source of the name for this file format. A CSV file typically stores tabular data (numbers and text) in plain text, in which case each line will have the same number of fields.

Sometimes the term "CSV" may also denotes several closely related delimiter-separated formats that use other field delimiters, for example, semicolons, tabs or spaces. We prefer a delimiter such as tab that is not present in the field since that simplifies format parsing. When a single character usually non-alphanumerical is used as the delimiter it is often referred to as character separated values

The one constant in "CSV"s is that each record is seperated by a newline. Newline (a.k.a line ending, end of line (EOL), or line break) is a control character or sequence of control characters that is used to signify the end of a line of text and the start of a new one. This special character is often output by text editors when pressing the Enter key. Unix based systems use '\n' or LF(line feed) as the newline character while Windows uses '\r\n' or CRLF(carriage return line feed)

Aim

The Aim of this assignment is to increase familiarity with characters and strings in SML. You will also learn about how to handle escape characters in a text file. You will learn about distinguishing between characters and meta characters.

Problem Statement

Data Format Conversion is one of the most common problems in Computer Science. A large number of tools exist in order to convert one format to another. There are numerous online tools which convert CSV (Comma Seperated Files) to TSV (Tab Seperated) Files. Similarly command lines tools like "unix2dos" and "dos2unix" are used to convert files between different newline conventions. This assignment involves the implementation of such tools in SML. You will write programs to implement the following tools

Converting Files Between Different Delimiter-Spaced-Formats

You have to write a program in SML/NJ that will read a file in a (given) delimited-space-format and convert into another delimited-space-format. The delimiter will be a single character. You will also have to account for delimiters present inside the fields. CSV format has been specified in IETF's RFC4180. We shall mostly follow the RFC4180 conventions.

- Each record is located on a separate line, delimited by a newline (LF).
- The last record is also terminated by a newline (LF).
- Within each record, there may be one or more fields, separated by delimiter (comma, semicolons etc). The last field in the record must not be followed by a delimiter.
- Each line should contain the same number of fields throughout the file.
- Spaces are considered part of a field and should not be ignored.
- Each field may or may not be enclosed in double quotes.
- Fields containing line breaks (LF), double quotes, and commas should be enclosed in double-quotes.
- If double-quotes are used to enclose fields, then a double-quote appearing inside a field must be escaped by preceding it with another double quote.

The following is a sample CSV file following the aforementioned conventions where LF indicates the newline.

```
"aaa"|"b""bb"|"ccc" LF
"ppp"|"q LF
qq"|"rrr" LF
zzz|yyy|xxx LF
```

Since some OS don't support certain characters like (—, ¡, ¿ etc), the file extension need not be renamed. The input file will be given by the string 'infilename' and output file by the string 'outfilename'.

Converting Files Between Different Newline Conventions

You have to write a program in Small Talk that will read a file which uses a (given) character(s) as EOL and convert it into file using another character(s) as EOL.

Function Specifications

You will need to create the following functions:

- A function to convert a file which uses a delimiter #DELIM1 and convert it to another file which uses a delimiter #DELIM2.

  ```
  fun convertDelimiters(infilename, delim1, outfilename, delim2)
  ```

- A function which uses the above function to convert between CSV and TSV.

  ```
  fun csv2tsv(infilename, outfilename)
  fun tsv2csv(incilename, outfilename)
  ```

- A function to convert a file which uses a newline #NEWLINE1 and convert it to another file which uses newline #NEWLINE2.

  ```
  fun convertNewlines(infilename, newline1, outfilename, newline2)
  ```

- A function which uses above function to convert between Unix and DOS files.

```
fun unix2dos(infilename, outfilename)
fun dos2unix(incilename, outfilename)
```

Exceptions

Your code also needs to throw the following exceptions at appropriate places:

- Input file could be empty in which case raise the exception

```
exception emptyInputFile
```

- You will have to check that the number of fields in each line is the same and otherwise raise an exception giving the line number with the first deviation. Line numbers in text files start from 1 and assuming that line number 1 has the 'expected" number of fields, any case of deviation will necessarily have line number greater than 1.

  You will have to check that the number of fields in each line is the same and otherwise raise an exception giving the line number with the first deviation.

```
(* Exception Raised when number of fields in a record are uneven.
Return line number, expected and actual number of fields in the record *)
(* String should be ``Expected: 10 fields, Present: 9 fields on Line 2"*)
exception UnevenFields of string
```

Note

- You are not allowed to change any of the names or types given in the specification/signature. You are not even allowed to change upper-case letters to lower-case letters or vice-versa.
- The evaluator may use automatic scripts to evaluate the assignments, especially when the number of submissions is large.
- You may define any new auxiliary functions you like in your code besides those mentioned in the specification.
- Your program should implement the given specifications/signature.
- You need to think of the most efficient way of implementing the various functions given in the specification /signature so that the function results satisfy their definitions and properties.
- The evaluator may look at your source code before evaluating it, you must explain your algorithms in the form of comments, so that the evaluator can understand what you have implemented.
- Do not add any more decorations or functions or user-interfaces in order to impress the evaluator of the program. Nobody is going to be impressed by it.
- There is a serious penalty for code similarity (similarity goes much deeper than variable names, indentation and line numbering). If it is felt that there is too much similarity in the code between any two persons, then both are going to be penalized equally. So please set permissions on your directories, so that others have no access to your programs.
- Submissions are on Gradescope.