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

Submission Deadline: Sunday, 21 Feb 2021 23:59
Submission Deadline with Late Penalty Friday, 26 Feb 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 denote several closely related delimiter-separated formats that use other field delimiters, for example, semicolons, tabs or vertical-bars (usually a non-alphanumeric printable visible special character). 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-alphanumeric 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). The following special cases may also occur.

1. The input file may have only a single EOL character at the end. Then this is regarded as a file representing a table with only one row.
2. The input file may have one or more EOL characters and there may be no occurrence of the delimiter. Such a file represents a table with as many rows as there are EOLs but only one column.

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-Sepatated-Formats

You have to write a program in SML/NJ that will read a file in a (given) delimiter-separated-format and convert into another delimiter-separated-format. The delimiters will be single characters. You will also have to account for delimiters present inside the fields. We shall follow the following conventions for delimiters.

- The backspace character (\) is usually used as an "escape character" and hence should not be used as delimiter, though it could occur within a field.
- If `c` is the delimiter then `\c` is used to escape it if it occurs within a field. If the string `\c` occurs as a substring of a field then it occurs as `\\c`
- When converting file `input.csv` (delimited by `c`) to `output.dsv` (delimited by `d`) you will have to ensure that
  - All occurrences of `\c` in `input.csv` become `c` in `output.dsv`,
  - All occurrences of `d` in `input.csv` become `\d` in `output.dsv`,
  - If `\d` occurs as a substring of a field in `input.csv` then it occurs as `\\d` in `output.dsv`.

Converting Files Between Different Newline Conventions

You have to submit a SML/NJ program in a file called `csv2dsv.sml` that will read a file which uses a (given) character(sequence) as EOL and convert it into a file using another character(equence) as EOL.

Function Specifications

You will need to create the following functions:

- A function to convert a file which uses a character delimiter `delim1` and convert it to another file which uses a character 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 string `newline1` and convert it to another file which uses newline string `newline2`.

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

- A function which uses the 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
  ```

  If the input file is not a text file or not present, then SML/NJ should automatically raise an uncaught exception.

- 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.

  ```
  (* Exception Raised when number of fields is uneven in the input file.
     Return line number, expected and actual number of fields *)
  exception UnevenFields of string
  (* "Expected: 10 fields, Present: 9 fields on Line 2"*)
  ```

- There could be other exceptions that have not been detailed here. Raise them as appropriate and give them suitable names.

Note

1. 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.
2. The evaluator may use automatic scripts to evaluate the assignments, especially when the number of submissions is large.
3. You may define any new auxiliary functions you like in your code besides those mentioned in the specification.
4. Your program should implement the given specifications/signature.
5. 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.
6. 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.
7. 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.
8. 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.