Mechanics CSV

CSCI-243 Homework

5/14/2018

1 Introduction

Two of the most fundamental programming tasks are to read and write formatted data. The degree of complexity of the data formats can vary wildly. The commandine program 'cp' simply moves data byte-by-byte with no real formatting. The commandline program 'grep' on the other hand operates line-by-line. The commandline program 'cut' operates field-by-field.

Note that these are not C functions, but rather are standalone tools which can be executed in terminal sessions. You will write a program to process a particularly common type of structured data: comma separated values (CSV).

2 Reading Text Data

One of the purposes of this assignment is to give you experience working with character arrays in C. It is recommended to process the input using `fgets` or `getchar`. The `scanf` function (and all its variants) and the `strtok` function are all explicitly forbidden for this assignment. You will need the function `strtol` to convert from a string representation of a number to a C integer type. See `man strtol` for more information, or tutorials point for a friendlier presentation.

You may read the entire file into a large multidimensional array, but that is not necessary. It is possible to do this in a single while loop with `fgets`. It is possible to solve this in a nested while loop with `getchar`. You will need to understand character arrays and array indexing.

3 Parsing the Data

You should consider how to break up the data by searching across character arrays for the ',' character. You should focus on the low-level aspects of your program, such as how you transform a string into a number, and how to avoid going out of bounds.

4 The Problem

We are interested in a form of summary statistics for the dataset. Your objective is to write a program which accepts data in the form supplied above from standard in, and produces a report containing the number of zip codes, the total population, and information about the zipcodes with the fewest and the most people. The output for the sample csv file should be:

```
Total population across 50 zipcodes is 1750367.
The fewest people live in 14619, population 1234.
The most people live in 90210, population 56789.
```

Please take a look at the following pairs of sample input and output. You may be prompted for a program to read the files – we recommend viewing them as plain text files using a text editor such as `vi, emacs, gedit, or notepad`. If you notice any discrepancies, please report them to your instructor ASAP.

INPUT	OUTPUT
sample0.csv	output0.txt
sample1.csv	output1.txt
sample2.csv	output2.txt

These files can be obtained from the Homework section on myCourses in Homework 2.

4.1 Get the Files

Go to myCourses Homework section and download the 6 sample files from the Homework 2 section. Copy them to the working directory for you homework assignment.

You may make the following assumptions:

- No line will be longer than 256 characters.
- There will be at most 1024 lines of input.
- The input is correctly formatted (e.g., number and type of columns).
- The input begins with a header line, which you may wish to ignore.
- No two zipcodes will have the same population.
- No zipcode will be listed more than once in the data.

5 Program Structure

Because this assignment is relatively short, there is no need to use multiple files. You should put all your code into a file named `readcsv.c`.

5.1 Compilation

After the first homework, and our discussion of compiling and linking, you are probably ready to compile programs without help. Just in case you would like a refresher, one way to compile your program is to run the command:

```
gcc -Wall -Wextra -std=c99 -pedantic readcsv.c -o readcsv
```

WARNING: If you mistype that line you can overwrite and permanently delete your source file! Make backups or use version control.

5.2 Execution

Once you have compiled your program into an executable file named `readcsv`, you may run it using either of the following methods. Both rely on reading a file by redirection through standard input (stdin)

- `./readcsv < sample.csv`
- `cat sample.csv | ./readcsv`

6 Submission Instructions

Put a zip file, `hw2.zip`, containing the file(s) for this assignment into the correct MyCourses drop box.

7 Grading

- 20%: You submitted a `non-trivial` program that compiles without error.
- 20%: Your source file has your real name, user name / email address, and a high-level description of your program. In addition, you must document each of the functions you have written. This is an easy 20 points – don't miss out!
- 10%: Your program's output is correctly formatted. (I.e., exactly three lines, no spaces.)
- 50%: Your program's numerical output is correct.