Homework 1 - MPCS 51040

October 8, 2014

1 General Instructions

1.1 Compiling

For each of the following, write a single file program to carry out the specified functionality. The programs must compile with gcc −std=c11 −Wall −ansi −pedantic and should be named p1.c, p2.c . . . p10.c.

1.2 Handing in

To hand in your homework, you need to commit all requested files (with correct filenames!) to your personal subversion repository. You should have access to a repository named yourcnetid-mpcs51040-aut-14.

Make a subdirectory called 'homework/hw1' and place your files under that directory. Don't forget to commit your files! You can check on `http://phoenixforge.cs.uchicago.edu` to make sure all files were committed to the repository correctly.

1.3 Code samples

This document, and any file you might need to complete the homework can be found the subversion repository `https://wooldridge.cs.uchicago.edu/svn/mpcs51040-aut-14`.

2 Assignments

2.1 Program 1

Fix is_numeric.c (is in the repository) to handle:

- values with multiple decimal points
- values that begin with a zero that are not immediately followed by a decimal point
- integer values that exceed LONG_MAX in magnitude

2.2 Program 2

Write a program that replaces all tabs in a text file with two spaces. The name of the file will be passed on the commandline, and the output of the program should go to standard out.

Example:

```
PROMPT>> ./p2 <input_file_name>
output goes to standard out
(program ends)
```

2.3 Program 3

Write a program that takes one or more scrambled words as input and produces a list of valid unscrambled versions of them. Base your code on the short "dictionary" (list of words) provided in the repository (i.e. your code only needs to work correctly on the provided words).

```
PROMPT>> ./p3
 Enter word(s) : pta bda
  pta :  tap pat apt
  bda : bad
 Enter word(s) : ^d
 (program ends)
```

2.4 Program 4

Write a function (not main) named read_file that reads a text file into an array of strings. In C one way to implement this is as an array of array of char of fixed length. This is declared as char text[MAX_LINES][MAX_CHAR_PER_LINE].

Write a test main that tests read_file (remember to call the program file p4.c) and prints contents to standard out.

```
PROMPT>> ./p4 <input_file_name>
(input file printed here)
(program ends)
```

2.5 Program 5

Write a program that lists the number of words in a file that contain a repeated letter. Include the number of occurrences of each word.

```
PROMPT>> ./p5 <input_file>
  better : 3
  butter : 1
  tall : 1
(program ends)
```

2.6 Program 6

Write a program that prompts continuously for input (a single word) and lists all possible unique permutations of that word. If more than a single word is input, only the first word is evaluated.

```
PROMPT>> ./p6
enter word: pop
pop
opp
enter word: ^d
(program ends)
```

2.7 Program 7

Write a program that takes a single integer as input and flips its endianness. (See http://en.wikipedia.org/wiki/Endianness for more information.) You can assume integers are 32 bits long.

```
PROMPT>> ./p7 <input integer>
<output integer>
(program ends)
```

2.8 Program 8

Write a program that prints the bit pattern of a float. What are the max and min values of a float on the platform you're using?

```
PROMPT>> ./p8 <input float>
<ouptut bit pattern>
(program ends)
```

2.9 Program 9

Write a function called atoi2 that checks for malformed integers and returns an error code if one is encountered. You can decide on the appropriate error codes, but at minimum check for non-numeric values and decimal points. Write a test main that uses atoi2 to add two ints and reports an error message to the user if (and based on) the error code returned (remember to call the file p9.c).

For example:

```
PROMPT>> ./p9 3 2
sum is: 5
(program ends)
PROMPT>> ./p9 3 x23
error: integer may not begin with non-digit character
(program ends)
```

2.10 Program 10

Write a program that takes a 1d array of floats as input and returns their running average using a user-specified averaging window width and zero-padded boundaries (so that number of input values = number of output values).

```
PROMPT>> ./p10 <input file name> <filter width>
(output values)
(program ends)
```

For example if an input file (test_input.txt) reads:

```
1.23 2.34 3.45 4.56 6.67
```

And you run it as:

```
PROMT>> ./p10 test_input.txt 3
0.41 1.19 2.34 3.45 4.89
(program ends)
```