COMP1511 18s2

Week-06 Laboratory Exercises

Objectives

- reading and writing files
- line-based processing of input
- using command line arguments

Topics

Preparation

Before the lab you should re-read the relevant lecture slides and their accompanying examples. **We will only assess lab Exercises 01, 02 and 03 to derive marks for this lab.** However, you will learn a lot by attempting and solving the challenge exercises.

Getting Started

Create a new directory for this lab called `lab06` by typing:

```
$ mkdir lab06
```

Change to this directory by typing:

```
$ cd lab06
```

Exercise-00: Print Characters One per Line (pair)(warmup)

This is a pair exercise to complete with your lab partner.

Write a C program, `one_per_line.c`, which reads in a line from standard input and writes out the characters one per line. Do not write a line for the new line character.

The output from your program should look **exactly** like this:

```
$ dcc one_per_line.c -o one_per_line
$ ./one_per_line
Enter a string: Hello
H
e
l
l
o
$ ./one_per_line
Enter a string: I'll never.
I
'
l
l

n
e
v
e
r
.
$
```

Hint: don't use **scanf**, use **fgets** to read the string.
Note, your program needs to read only 1 one_per_line string - it doesn't have to read until the end of input.

You can assume lines contain at most 4096 characters.

When you think your program is working you can use `autotest` to run some simple automated tests:

```
$ 1511 autotest one_per_line
```

When you are finished on this exercise you and your lab partner must both submit your work by running **give**:

```
$ give cs1511 wk06_one_per_line one_per_line.c
```

Exercise-01: Is it a Palindrome? (pair)

This is a pair exercise to complete with your lab partner.

A palindrome is a sequence which is the same forwards as backwards.
Write a program, `palindrome.c`, which reads a string and tests if it is a palindrome.

For example:

```
$ ./palindrome
Enter a string: kayak
String is a palindrome
$ ./palindrome
Enter a string: canoe
String is not a palindrome
$ ./palindrome
Enter a string: if if fi fi
String is a palindrome
$ ./palindrome
Enter a string: if if if fi
String is not a palindrome
```

Hint: don't use **scanf**, use **fgets** to read the string.
Note, your program needs to read only one string - it doesn't have to read until the end of input.

You can assume lines contain at most 4096 characters.

When you think your program is working you can use `autotest` to run some simple automated tests:

```
$ 1511 autotest palindrome
```

When you are finished on this exercise you and your lab partner must both submit your work by running **give**:

```
$ give cs1511 wk06_palindrome palindrome.c
```

Note, even though this is a pair exercise, you both must run **give** from your own account before **Sunday 02 September 23:59:59** to obtain the marks for this lab exercise.

Exercise-02: Create a File of Integers (pair)

This is a pair exercise to complete with your lab partner.

Write a C program, `create_integers_file.c` which takes 3 arguments.
The first & second arguments will specify a range of integers.

The third argument will specify a filename.

Your program should create a file of this name containing the specified integers.

For example:

```
$ ./create_integers_file 40 42 fortytwo.txt
$ cat fortytwo.txt
40
41
42
$ ./create_integers_file 1 5 a.txt
$ cat a.txt
1
2
3
4
5
$ ./create_integers_file 1 1000  1000.txt
$ wc 1000.txt
1000 1000 3893 1000.txt
```

Your program should print a suitable error message if given the wrong number of arguments or if the file can not be created. **Hint**: use `fopen` to open the file and `fprintf` to output to the file. If you need some help starting off, read this example program to see how to use these functions to create and write to a file.

When you think your program is working you can use `autotest` to run some simple automated tests:

```
$ 1511 autotest create_integers_file
```

When you are finished on this exercise you and your lab partner must both submit your work by running **give**:

```
$ give cs1511 wk06_create_integers_file create_integers_file.c
```

Note, even though this is a pair exercise, you both must run **give** from your own account before **Sunday 02 September 23:59:59** to obtain the marks for this lab exercise.

Exercise-03: Print The First Lines of a File (pair)

This is a pair exercise to complete with your lab partner.

Write a program, `first_lines.c`, which given a filename as argument prints the first 10 lines of the file. If the file has less than 10 lines the entire file should be printed.
Note this program does not create any files. Its just prints part of the contents of a file to standard output.

For example, supposed we have a file `1000.txt` the first 1000 integers one per line (from a previous exercise).

```
$ ./first_lines 1000.txt
1
2
3
4
5
6
7
8
9
10
$
```

But `first_lines.c` should work for files containing any sort of text, e.g.:

```
$ ./first_lines /usr/include/stdio.h
/* Define ISO C stdio on top of C++ iostreams.
   Copyright (C) 1991-2016 Free Software Foundation, Inc.
   This file is part of the GNU C Library.

   The GNU C Library is free software; you can redistribute it and/or
   modify it under the terms of the GNU Lesser General Public
   License as published by the Free Software Foundation; either
   version 2.1 of the License, or (at your option) any later version.

   The GNU C Library is distributed in the hope that it will be useful,
```

It should also be possible to specify that a different number of lines be printed.
This will be done by passing the string "-n" as the first argument to the program

The second argument will be the the number of lines to be printed .

The file will now be the third argument to the program.

For example:

```
$ ./first_lines -n 3 1000.txt
1
2
3
$
```

You should print a suitable message if incorrect arguments are supplied or the file can not be read.
You can assume lines have at most 1024 characters, but if possible avoid this assumption.

When you think your program is working you can use `autotest` to run some simple automated tests:

```
$ 1511 autotest first_lines
```

When you are finished on this exercise you and your lab partner must both submit your work by running **give**:

```
$ give cs1511 wk06_first_lines first_lines.c
```

Note, even though this is a pair exercise, you both must run **give** from your own account before **Sunday 02 September 23:59:59** to obtain the marks for this lab exercise.

Exercise-04: Is it a Palindrome - the Sequel (individual) [Challenge]

This is an individual exercise to complete by yourself.

Write a program, `punctuated_palindrome.c`, which reads a string and tests if it is a palindrome.
Characters which are not letters should be ignored .

Differences between upper case and lower case are ignored. For example:

```
$ ./punctuated_palindrome
Enter a string: Do geese see God?
String is a palindrome
$ ./punctuated_palindrome
Enter a string: Do ducks see God?
String is not a palindrome
$ ./punctuated_palindrome
Enter a string: Madam, I'm Adam
String is a palindrome
$ ./punctuated_palindrome
Enter a string: Madam, I'm Andrew
String is not a palindrome
```

Hint: you might find C library functions in `#include <ctype.h>` useful.
You can assume lines contain at most 4096 characters.

When you think your program is working you can use `autotest` to run some simple automated tests:

```
$ 1511 autotest punctuated_palindrome
```

When you are finished working on this exercise you must submit your work by running **give**:

```
$ give cs1511 wk06_punctuated_palindrome punctuated_palindrome.c
```

Exercise-05: Print The Last Lines of a File (individual)[Challenge]

This is an individual exercise to complete by yourself.

Write a program, `last_lines.c`, which given a filename as argument prints the last 10 lines of the file. If the file has less than 10 lines the entire file should be printed.
You can assume lines have at most 1024 characters, but you can not make any assumption about how many lines are in the file.

You can not read the entire file into an array.

For example:

```
$ ./last_lines 1000.txt
991
992
993
994
995
996
997
998
999
1000
$ ./create_numbers_file 1 100000 100000.txt
$ wc  100000.txt
100000 100000 588895 100000.txt
$ ./last_lines 100000.txt
99991
99992
99993
99994
99995
99996
99997
99998
99999
100000
$
```

When you think your program is working you can use `autotest` to run some simple automated tests:

```
$ 1511 autotest last_lines
```

When you are finished working on this exercise you must submit your work by running **give**:

```
$ give cs1511 wk06_last_lines last_lines.c
```

Submission

When you are finished each exercises make sure you submit your work by running **give**.
You can run **give** multiple times. Only your last submission will be marked.

Don't submit any exercises you haven't attempted.

If you are working at home, you may find it more convenient to upload your work via give's web interface.

Remember you have until **Sunday 02 September 23:59:59** to submit your work.

Automarking will be run several days after the submission deadline for the test. When complete you can view automarking here and you can view the the resulting mark via give's web interface

You can read more about lab assessment here