CSC9V4 Practical 8: In or out, not both!

Introduction

Input and output – Time to get to it… This practical contains advanced exercises. It is important to build solutions modularly, adding pieces to a working program and not moving forward until the program with the new addition works well. You also probably will need to closely reference slides in order to have a clear understanding of the function you will be using. Use redundant printing functions, e.g. printf(), putchar(), … to monitor the functioning of your program during development/debugging, as appropriate.

The Other wc

Write a program called **my_wc.c** that does each of the following calculations, each one implemented by a different function. Be sure that one works before moving onto the next one.

a) Count the number of characters in a text file.
b) Count the number of words in a text file. A 'word' is any sequence of non-white-space characters. For our purposes, it is sufficient to assume that only spaces separate words.
c) Count the number of lines in a text file.

The name of the file should be passed in as a parameter. One test file will be provided in the practical. Test your program on that file: let the program print the result of a), b) and c) above.

Your program should be robust, e.g. properly check that the expected parameters are passed to the program, and that I/O possible errors are properly dealt with.

Check Point 1

Hex without the Witchcraft -- ADVANCED

Write a program that displays the contents of a file as bytes *and* characters. Specify a text file as a command line argument ('Properties→Run). Recall the first C program we introduced, pun.c; if provided as a parameter, the output would appear as follows*:

```
 Offset              Bytes              Characters
 ------  -----------------------------  ----------
  0  23    69 6E 63 6C 75 64 65 20 3C   #include <
 10  73    74 64 69 6F 2E 68 3E 0D 0A   stdio.h>..
 20  0D    0A 69 6E 74 20 6D 61 69 6E   ..int main
 30  28    69 6E 74 20 61 72 67 63 2C   (int argc,
 40  20    63 68 61 72 20 2A 61 72 67    char *arg
 50  76    5B 5D 29 20 7B 0D 0A 0D 0A   v[]) {....
 60  20    20 70 72 69 6E 74 66 28 22     printf("
 70  54    6F 20 43 20 6F 72 20 6E 6F   To C or no
 80  74    20 74 6F 20 43 5C 6E 22 29   t to C\n")
 90  3B    0D 0A 0D 0A 20 20 72 65 74   ;....  ret
100  75    72 6E 20 30 3B 0D 0A 7D 0D   urn 0;..}.
110  0A                                 .
```

Each line shows 10 bytes from the file, first as hex and then as characters. The number in the `Offset` column indicates the position in the file of the first byte on the line.

In the Characters column, characters are printed if deemed printable by the `isprint` function (in <`ctype.h`>. Otherwise, a period appears for non-printable characters.

* NOTES: (i) The example assumes a Windows text file, your results can be slightly different if you are using a different OS (or require different assumptions on the file and its structure); (ii) be sure to open the file in "`rb`" mode.

Check Point 2