A0:

Start Assignment

Due Tuesday by 11:59pm **Points** 100 **Submitting** a file upload **File Types** c
Available Feb 9 at 4:30pm - Mar 1 at 6am

A0: Word count

Abstract

We've been working with some of the basic elements of C and practical aspects of running code on the iLabs. In this assignment you'll extend that work and practice using some of the things we've been talking about in order to implement some common Linux built-in functions.

Introduction

You'll implement the defaul version of the word count function, or "wc". When you invoke wc on a file it will print the number of lines, words and characters in the file. Your code should accept a single filename as an argument. Be sure your code is robust! Be sure to check that you get exactly one argument. When you try to open the file, check to see if it is opened with no error. If in any case your code does run in to an unrecoverable error, it should print an informative error message (e.g. "Error: Expected one argument, found two: "./file" "name") and return with -1. Be sure to handle cases where the filename given does not exist, you can not open it in read mode, or the file is empty. Check everything. It should be impossible for your code to fail unexpectedly.

Methodology

Your code will be invoked with the name of the file to count as a commandline parameter (e.g.: ./wordcount "./file.txt").

Once you have a filename, open it with open() and read from it using read(). Do not use the f-functions (e.g. fopen, fread).

Once you have the file open, you can read it in one character at time. Keep track of the number of lines you find in the file, the number of words in the file (whitespace characters separated by non-whitespace characters) and total characters. If you read in characters one at a time, you can easily check to see what kind of character it is. In all cases, increment your number of characters read. If it is a whitespace character and the previous character was not a whitespace character, your word count

should increase. If it is a newline character, the number of lines should increase and if the previous character was not a newline or whitespace, your word count should increase.

When you hit the end of the file, close the file and print the line count, word count, character count and the name of the file you inspected, separated by spaces.

e.g.:
./wordcount "testfile.txt"
11 36 231 testfile.txt

.. for the file "stuff.txt" containing:
hello out
there all yo
2134 42 u

.. your code should be run as:
./wordcount "./stuff.txt"

.. and should output:
3 8 35 stuff.txt

You should check out and make use of the "ctype" library. If you check the manual for "isspace", you'll get the page for the entire ctype library. It should prove quite useful.

Do not use strtok.
Do not read in more than one character at once.
Do not use any f-type file command (e.g. fopen, fscanf)

Submission:
You should submit your source code, named wordcount.c, through Canvas.

Note: presume we will compile your code in the following manner:
gcc wordcount.c -o wordcount
.. do not rely on fsanitize, C99 or C89 mode, or other compilation flags or user libraries.
.. do not supply a Makefile or anything other than your source file.