CP-367 A (Winter 2026) Assignment 2 due date: Feb. 15, 2026

Description: a take-home , individual assignment.

Objectives: demonstrating your understanding and application of Linx and C system calls.

Task: Create Unix shell and wc command using C programming, compile and run each then show the outputs.

Submission: zip all the files and a single Word document that contain screenshots of all the outputs. Make sure you include your name and student ID# as an author in each program as well as the Word document. Then submit the zip file to the Assignment 2 drop bob in Dropbox section on MyLS.

Note(1): For this assignment, you must use C99 language syntax. Your code must compile using make without errors.

Test your program thoroughly with the GCC compiler in a Linux environment.

If your code does not compile, then you will score zero. Therefore, ensure you have removed all syntax errors from your code.

Note (1): The use of generative AI is not permitted in this course in terms of generating partial or complete solutions – it is a violation of the Academic Code of Conduct. However, the students can search and learn about the assignment topics using AI tools.

Note (2): It is the student's responsibility to read, understand and follow the Laurie's Academic Code of Conduct.

Note (3): students with approved accommodation by WLU must indicate in the comment box in the drop box that they have an approved accommodation with details of what they are entitled to, so it matches my record.

General rubrics:

- Functionality: 50% (Program meets all the specified requirements)

- Code Quality: 30% (Readability, style, comments, proper variable naming)

- Error Handling: 10% (Implementation of basic error handling if required)

- Testing: 10% (Evidence of thorough testing)

Question 1 [10 marks]

Write a C program (name it word_count.c) that asks the user to enter the name of a text input file in the current directory. The program reads the text file and produces an output stream that echoes the file on the stdout and counts the number of lines, words, and average number of words per line. Treat any characters between the last '\n' and the EOF marker as a final line of text. This program is a variation of the Unix we function. Do not use the function xc in your program.

The program should provide a main menu asking or the user command: 'f' for entering the input file name, and 'q' for quitting the program. If the user enters a wrong command, or if the file cannot be opened or read, the program will tell the user so. After the command is executed (or not), the main menu is reproduced again, unless the command is 'q' in which case the program terminates.

Use the following command to compile the code.

```
$ gcc -Werror -Wall -g -std=gnu99 word_ count.c -o word_ count
```

The above code will create the ./word_ count executable file

The expected output for executing:

When running on Linux Terminal the command . /word_count. It will prin:

This program counts the number of lines and words of a file Enter f for entering file name, q to quit: f

Enter file name: input2.txt

Content of input2.txt:

We will buy very pretty things

A-walking through the faubourgs.

Violets are blue, roses are red,

Violets are blue, I love my loves.

Number of lines of file1: 4

Number of words of file1: 23

Average number of words per line of file1: 5.75

Enter f for entering file name, q to quit: f

Enter file name: notknown.txt

notknown.txt cannot be opened

Enter f for entering file name, q to quit: q

Good bye

Important Note: When submitting a source code file, make sure to name it like:

word_count.c

Question 2 [10 marks]

Write a C program (name it shell. c) to develop a basic Unix shell interface to execute basic Unix commands input by the user. This program will use parent-child process to run the commands that are entered by the user. The parent process will print the shell prompt "myshel1>" where the user will input the Unix commands. The parent process will parse the input and identify the command along with arguments and child will execute using execvp() function. You can call the following functions to implement the program:

• while(1) loop creates an infinite loop for the shell to continuously accept commands.

• fqets() is used to read a command from the standard input.

• strok() is used to split the command into arguments (for use with execvp()).

• fork() creates a new process, and execvp() replaces the child process with a new program.

• wait() is used in the parent process to wait for the completion of the child process.

• After the command is executed (or not), the main menu is reproduced again, unless the command is'q'in which case the program terminates (use strcmp() to compare the command with'q'to exit the program).

• If the user press enter without entering command, then program should show "myshell>"prompt again and be ready to accept new command. The other implementation details are at your discretion, and you are free to explore.

Use the following command to compile the code:

$ gcc -Werror -Wall -g -std=gnu99 shell.c -o shell

The above code will create the ./shell executable file

The expected output for executing:

When running on Linux Terminal the command ./shell. It will print the prompt, and you can enter any Unix command:

myshell> echo hello

hello myshell>

uname Linux

myshell> touch file.txt

myshell> q

Important Note: When submitting a source code file, make sure to name it like:

shell.c