CP367: Assignment 2 – Winter 2024

Due on Feb 12, 2024 (Before 11:59 PM)

This is an **individual** assignment, and we will try to practice creating Unix shell and wc command using C programming.

General Instructions:

- 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.
- **Gradescope** platform would be used to upload the assignments for grading. The link to the Gradescope assignment is available on Myls course page. Drag and drop your code file(s) for submission into Gradescope. **Make sure your file name is as suggested in the assignment; using a different name may score zero.**
- Please note that the submitted code will be checked for plagiarism. Submitting these .c files would confirm that you have not received unauthorized assistance in preparing the assignment. You also confirm that you are aware of course policies for submitted work.
- Marks will be deducted from any questions where these requirements are not met.
- Multiple attempts will be allowed, but your last submission will be graded before the deadline. Instructors reserve the right to take off points for not following directions.

Warning:

Follow the assignment instructions to the letter in terms of the file names and function names, as this assignment will be auto graded. If anything is not as described, the auto-grading fails, and your assignment will be given a **Zero** mark.

Question 1

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 `wc` function. Do not use the function `wc` in your program.

The program should provide a main menu asking for the user command: 'f' for enter 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 print (User's input is in bold):

```
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 to Gradescope, make sure to name it like:

- `word_count.c`

Question 2

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 "`myshell>`" 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 following functions to implement the program:

- `while(1)` loop creates an infinite loop for the shell to continuously accept commands.
- `fgets()` is used to read a command from the standard input.
- `strtok()` 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 programme 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 to Gradescope, make sure to name it like:

- `shell.c`