Project 1: Unix File Operations

Posted: Tuesday February 13, 2024.
Due: 11:59PM, Monday, February 26/March 4/March 11, 2024.

Project Objective: Review programming skills in C under GLUE UNIX that includes:

1. file operations in C
2. command-line arguments in C
3. arrays, conditional statements, and loops
4. basic data processing techniques and problem solving skills

Project Description

Through the course of this assignment, you will develop a program that performs some basic file operations similar to UNIX commands. Your program will first identify the operation to perform from the command line and then perform the operation on the data from input file(s). The project is split into three stages with separate deadlines so you can start early and get the project done.

General Requirements:

1. Your program will begin with one of the following commands from the usage menu:

```
a.out 0 file1.txt file2.txt
a.out 1 file1.txt file2.txt file3.txt
a.out 2 m-n file1.txt file2.txt
a.out 3 file1.txt
a.out 4 file1.txt
a.out 5 file1.txt
```

0	copy
1	interleave
2	cut
3	more
4	grep
5	Word count

`a.out` is the executable file; 0-4 are the operation codes shown in the above table; the rest arguments are file names for the input/output files pertain to the corresponding operation.

2. Your program should initially verify that an appropriate option has been selected.
   a. If the first argument after the `a.out` is not between 0 and 5, or the number of files after this value is incorrect (for example, two file names for option 3), then your program should print out the above usage menu and **terminate**.
   b. If the command is correct, but one or more of the files cannot be open, then your program should **terminate** with an error message:

```
file file1.txt cannot open.
```

   where `file1.txt` is the file that cannot be open. If there are more than one file that cannot be open, printing out any one of them will be ok.

3. When user enters a valid operation, your program should proceed to run the user-specified operation as described below.

Option 0: Copy

a. Command `a.out 0 file1.txt file2.txt`
b. Requirements

This is exactly the same as UNIX cp command. It makes an identical copy of `file1.txt` with file name `file2.txt` and the program terminates. The implementation of this option will be provided to you in the project template.

Option 1: Interleave

a. Command `a.out 1 file1.txt file2.txt file3.txt`
b. Requirements

Your program should open files `file1.txt` and `file2.txt` as input and create output file `file3.txt` as follows:
The first line of file3.txt is the first line of file1.txt
The second line of file3.txt is the first line of file2.txt
The third line of file3.txt is the second line of file1.txt
The fourth line of file3.txt is the second line of file2.txt
…

When one input file reaches the EOF, the remaining lines in the other file should be copied to the output file and the program terminates.

Option 2: Cut

a. Command `a.out 2 m-n file1.txt file2.txt`
b. Requirements

This operation simulates the UNIX `cut` command, which writes the m-th character to the n-th character from each line in `file1.txt` to `file2.txt`. For the following file

```
This is ENEE 150.
123456789
Project 1 is on UNIX files.
Short
It seems to be challenging and fun.
a.out 2 2-6 file1.txt file2.txt
```

on command `a.out 2 2-6 file1.txt file2.txt`
the output file should be

```
his i
23456
rojec
hort
t see
.out
```

Option 3: More

c. Command `a.out 3 file1.txt`
d. Requirements

This operation simulates the UNIX `more` command. Your program should first print out the first 10 lines of the input file `file1.txt` on the screen and then wait for user input

- If the user inputs 'n', your program should print out the next line of the file
- If the user inputs 'p', your program should print out the next ten lines of the file
- If the user inputs 'q', your program should terminate without any more print out

When you don't have 10 lines from the input file to print, for example the file has only 5 lines or there are only 5 lines left, then just print out these remaining lines and terminate the program.

Option 4: Grep

a. Command `a.out 4 file1.txt`

b. Requirements

This operation simulates the basic feature of the UNIX `grep` command. Your program should prompt the user by printing out

```
Input search string (10 chars max):
```

Then (1) read in user input, which can contain any characters, (2) search in `file1.txt` and (3) print out every line that contains at least one instance of the input character string.

Option 5: Word Count

a. Command `a.out 5 file1.txt`

b. Requirements

This operation implements the UNIX `wc` command. Your program should print out the following information about the input file:

- The number of lines
- The number of words
- The number of characters
- The shortest line in terms of characters
- The shortest line in terms of words
- The top three longest words

Each line has no more than 100 characters. A line ends with the '\n' character except that the last line of the file ends with EOF.

A word is defined as a string of letters (upper and lower case) and digits (0-9), starting with a letter or digit and ending before the first character that is not a letter or digit. A word has no more than 30 characters. For example,

`J.K. Rowling's Harry Potter` will be considered as 6 words: `J, K, Rowling, s, Harry, Potter`

Every character that can be scanned in with %c option will be counted as a character.

When there is a tie in the last three items (shortest line and longest words), printing out any of the answers will be ok (see the following example).

A sample output (when input file is `gringotts.txt`)

```
The number of characters is: 302
The number of words is: 58
The number of lines is: 9
The shortest line (28 characters): J.K. Rowling's Harry Potter
The shortest line (5 words): Enter, stranger, but take heed
The top three longest words are: stranger, treasure, treasure
```

Special Notice

1. A Question and Answer file **Q_and_A.txt** will be posted under GLUE **public/proj1/** directory and updated whenever we get common questions. Please check the file frequently. Email notice will be sent to you only when important update is added.
2. A sample executable file and a couple of input files will be posted on GLUE **public/proj1/** directory. Please play with them and use the UNIX diff command to ensure that your program generates the same output as the sample executable.
3. A sample program template attached at the end of this document. You do not have to use it for your project, but it may give you some hint on how to complete the project. Note that the cp option is implemented in this template and you can use it in your program.

Project Requirements:

1. You must program using C under GLUE UNIX system and name your program **p1.c**.
2. Your program must be properly documented (header, comments, variable naming, etc.)
3. submit the project (**p1.c** only) the same way you submit your homework or quiz. There will be three stages of submission, the requirements and due date for each stage are listed below. Use the same file name (**p1.c**) for submission and document your code.

a) Feb. 26: first submission, your code should be able to check command line arguments and file safety, as well as implement options 0-2. (35%)
b) Mar. 4: second submission, options 3 and 4 should be implemented. (30%)
c) Mar. 11: final submission, option 5 should be implemented. (35%)

Grading Criteria:

Correctness:	85%
Stage 1: 30%	
Stage 2: 25%	
Stage 3: 30%	
Good coding style and documentation:	15%
5% for each stage	
Penalty: program that does not compile under GLUE UNIX:	-100%
Late submission within 24 hours after the deadline	-40%
Late submission 24 hours or more after the deadline	-100%

A programming template/outline just for your reference. It is by no means a complete program or defines all the necessary variables. You do not have to follow it.

```
/* project header: document your code here with information such as (1) the
developer of the code (you), (2) a short description of the code (so people
not in our class will be able to understand), (3) revision history of the
code, (4) anything that will be useful for people (including you after a
couple of years) to understand the code, and (5) anything you want your TA
know before he grades your code (perhaps this should be put at the very
beginning of the code to get the TA's attention.
*/

#include <stdio.h>
#include <stdlib.h>

int main(int argc, char * argv[])
{ declare variables that include at least the followings:
      input/output files;
      variables for keyword for search, line number, word number, character
            number, shortest line in words, shortest line in characters,
            shortest line in words, top three longest word, and others;

  check whether the command line argument is valid and terminate the program
      if the command line is incorrect;
  open all files with safety check and terminate the program if any file
      cannot be open properly;
  // these safety checks can also be done inside the "switch()" statement

  switch (argv[1][0])
  { case '0':
      while( fscanf(in, "%c", &ch) != EOF )
        fprintf(out, "%c", ch);
      fclose(out);
      fclose(in);
      break;

    case '1':

    case '2':

    case '3':

    case '4':

    case '5':

    default:
  }

  return 0;
}
```