Assignments

Assignment 3 - Returned

Honor Pledge Accepted
Draft - In progress
Submitted
Returned

Assignment Details

Title	Assignment 3
Student	Ramtin Sirjani
Submitted Date	Oct 24, 2022 4:54 PM
Grade	**100.00 (max 100.00)**
History	Oct 24, 2022 4:54 PM EDT Ramtin Sirjani (rsirjani) saved draft Oct 24, 2022 4:54 PM EDT Ramtin Sirjani (rsirjani) submitted

Instructions

Assignment overview

This assignment consists of two parts.
In part one, you are required to write a C program that works with arrays.
In part two, you are to answer questions using short-answer

Part one: 70%

- Write a program to accept integers separated by spaces, place the integers into an array, and work on the integers found in the array
  - Your program will behave as follows.
    - Ask the user for how many integers they will provide
      - The user must provide a minimum of 5 integers and a maximum of 12.
    - Print the number the user gave you and size of the array in bytes (using `sizeof`)
    - Prompt the user for the list of integers
      - Each integer must be separated by spaces. ~~If the user does not provide a valid integer, or if the user does not provide enough integers, prompt the user for the entire list of integers again~~ (This is going to be optional instead. If you can accommodate an error like this, then great, otherwise, you can assume the user will not lie.)
    - Part 1
      - Print out:
        - The elements of the array and which position they are in
    - Part 2
      - Print out:
        - The largest value in the array
    - Part 3
      - Print out:
        - The elements of the array in reverse order
    - Part 4

- Print out:
  - The sum of all values in the array
- Part 5
  - Print out:
    - The elements of the array from largest to smallest (descending order)
- Part 6
  - Print out:
    - The elements of the array except the first and last element are swapped

- Each part should be implemented as a function. Therefore, your program should have a main function and at least 6 functions
- Each part works independently of the rest. In other words, no part should manipulate the original array itself.
- Your program should follow good programming styles, i.e. write clear code, choose good variable names, use appropriate functions, make proper comments, etc.
- Here is some sample output:

```
Please enter the number of integers to process: 5
    There is enough room in your array for 5 integers (20 bytes)

Please enter your integers separated by spaces: 10 99 1 31423 55

Part 1:
    Your array is: [0] = 10,  [1] = 99, [2] = 1, [3] = 31423, [4] = 55

Part 2:
    The largest value in your array is: 31423

Part 3:
    Your array in reverse is: 55 31423 1 99 10

Part 4:
    The sum of all values in your array is: 31588

Part 5:
    Your array in sorted order is: 31423 99 55 10 1

Part 6:
    Your array with first and last element switched is: 55 99 1 31423 10
```

Part two: 30%

Provide short answers to the following questions. If you aren't sure of the answer yourself, you can use Google, man pages, or just play around with commands on Gaul.

1. Files
   1. Suppose the command `ls` were unavailable. What command could you use to view files in the current directory instead? Explain why the command works?
   2. What option do I need to supply to the `rm` command to ensure I am prompted before a file or directory is removed? What option do I need to supply to the rm command to ensure I am **not** prompted before a

   file or directory is removed? What happens if both options are used simultaneously - which one has precedence?
   3. What are some differences between the `rename` command and the `mv` command?
   4. The `file` command, when run against an executable program shows `ELF 64-bit LSB executable`. What does `ELF` mean in this context?
   5. Consider the empty file `test.txt`. Why does `ls -l test.txt` say the filesize is `0` but `du test.txt` says the file size is `1`?
2. Directories
   1. Find a better way to do the following

      -
        ```
        mkdir grandparent
        mkdir grandparent/parent
        mkdir grandparent/parent/child
        ```

   2. How do you create a directory with a space in it? How do you remove a directory with a space in it?
   3. How do I create a directory and set it's permissions to `dr-x------` using only the `mkdir` command?
   4. Suppose I was working in `/dir1/dir2/dir3` and I `cd` to `/dir4/dir5/dir6`. What is an easy way to get back to `/dir1/dir2/dir3` without using the explicit command `cd /dir1/dir2/dir3` (perhaps I forgot what the full path name was).
   5. Suppose I used `pushd` to add 10 folders to my stack. How do I quickly determine which folder is the 6th folder in my stack and how do I quickly cd to it?
3. Permissions
   1. What option(s) do I need to add to `chmod` to print out what modifications are being made?
   2. What option(s) do I need to add to `chown` to get `file2` ownership to simply match `file1`?
   3. What option(s) do I need to add to the `stat` command to display file permission, user ownership, and group ownership only?
   4. How do you grant a group permission to write data to a file, but prevent the users in that group from moving, removing, or renaming the file?
   5. When using `chmod` in symbolic mode, what happens if u/g/o/a are omitted. Eg. `chmod +x <filename>`

Testing and submitting

You should test your program from part 1 by compiling and running it on Gaul before submitting (This is how the TA will be testing your program). Capture the screen of your testing using screenshots. Demonstrate your program works with at least two runs using different input.

For part two, write your answers in a text file, upload it to Gaul.

Add your part 1 and part 2 files to a `251xxxxxx-Assignment3` folder.

Create a single tarball called `251xxxxxx-Assignment3.tar` containing a directory structure that looks like this:

```
251xxxxxx-Assignment3
   |----- Part1
   |-------- part1.png
   |-------- part1.c
   |----- Part2
   |-------- part2.txt
```

Download your tarball and upload it to Assignment 3 in OWL.

I need help

Check to see if your question has already been posted in the forums. If not, post your question.

https://owl.uwo.ca/x/yvBGmc

Submitted Attachments

- 250680632-Assignment3.tar (120 KB; Oct 24, 2022 4:54 pm)

Assignment 3 - C Programming (data types and I/O), Unix (Files, Directories, Permissions)

Program 1 - Organization and Style

Instructions followed. Properly documented and commented. Useful naming conventions. Appropriate spacing.

Inadequate or Incomplete

Did not follow instructions at all. Commenting is inadequate or missing. Naming is consistently inadequate. Spacing and program layout is difficult to follow.

0 Points

Poor

Did not follow many key instructions. Commenting is generally inadequate. Naming is generally inadequate. Spacing and program layout is difficult to follow.

3.5 Points

Fair

Followed most instructions. Commenting is generally okay. Naming is generally okay. Spacing and program layout is generally okay.

5 Points

Good

Followed most instructions. Commenting is generally good. Naming is generally good. Spacing and program layout is generally good.

7.5 Points

Exceptional

Followed instructions. Commenting is exceptional. Naming is exceptional. Spacing and program layout is exceptional.

10 Points

Comment for Program 1 - Organization and Style

Done

0 0

Program 1 - Functionality

Program compiles. Program does what is asked. Program is not buggy.

Inadequate or Incomplete

Program does not compile or compiles with many adjustments. Program is very buggy and/or rarely produces correct results. Or, no submission received.

0 Points

Poor

Program does compile or compiles with minimal adjustments. Program is very buggy and/or rarely produces correct results.

15 Points

Fair

Program does compile. Program has only few bugs and usually produces correct results.

30 Points

Good

Program does compile. Program has no obvious bugs and usually produces correct results.

45 Points

Exceptional

Program does compile. Program has no bugs and consistently produces correct results.

60 Points

💬

Comment for Program 1 - Functionality

Done

0 0

Question 1

0

0 Points

2

2 Points

4

4 Points

6

6 Points

8

8 Points

10

10 Points

Comment for Question 1

Done

0 0

Question 2

0

0 Points

2

2 Points

4

4 Points

6

6 Points

8

8 Points

10

10 Points

Comment for Question 2

Done

0 0

Question 3

0

0 Points

2

2 Points

4

4 Points

6

6 Points

8

8 Points

10

10 Points

Comment for Question 3

Done

0 0

Total: **0**

Back to list