1. **your name and student ID**

TALHA Husnain 24100245

1. **How to compile and run your code**

To compile and run the provided C code, follow these steps:

* + Save the code in a file with a ".c" extension, for example, "multitasking.c".
  + Open a terminal or command prompt.
  + Navigate to the directory where the code file is saved using the cd command.
  + Compile the code using the following command:
    - gcc multitasking.c -o multitasking
  + If the compilation is successful, an executable file named "multitasking" will be created in the same directory.
  + Run the program using the following command:
    - ./multitasking file1.txt file2.txt

1. **A simple description of your solution logic or pseudo code**

The provided solution involves creating three child processes in a parent process and letting them run concurrently. Each child process performs a different task.

Child One:

* Takes input for the ages of 6 people from the user.
* Calculates the average age and finds the oldest age among the input.
* Displays the average and oldest age.

Child Two:

* Loads the existing Unix program called "sort".
* Retrieves the name of the file to be sorted from the command line argument.
* Executes the "sort" program with the "-r" option to sort the lines of the file in reverse alphabetic order.
* Displays the sorted lines on the standard output.

Child Three:

* Retrieves the name of the file to be modified from the command line argument.
* Opens the file in read and update mode.
* Inserts the string "This is the updated version." at the beginning of the file.
* Replaces all occurrences of the string "table" with "desk" in the file.
* Writes the modified content back to the file.
* Displays a success message.

The main function of the program:

* Checks if the program is executed with the correct number of command-line arguments (two filenames).
* Creates three child processes using the fork system call.
* In each child process, calls the corresponding child function.
* Waits for each child process to finish and prints its exit status.
* The parent process terminates after all child processes have finished.

1. **your test run and output.**

Child 2283 here. Will calculate average age.

Please enter 6 ages:

(1): Child 2285 here. Will update file.

Child 2284 here. Will execute command sort.

Your time is limited, don't waste it living someone else's life. -Steve Jobs

You miss 100% of the shots you don't take. -Wayne Gretzky

The road to success and the road to failure are almost exactly the same. -Colin R. Davis

The only way to do great work is to love what you do. -Steve Jobs

Success is not final, failure is not fatal: It is the courage to continue that counts. -Winston S. Churchill

It does not matter how slowly you go as long as you do not stop. -Confucius

If you want to lift yourself up, lift up someone else. -Booker T. Washington

Don't watch the clock; do what it does. Keep going. -Sam Levenson

Believe you can and you're halfway there. -Theodore Roosevelt

6

(2): 7

(3): 4

(4): 3

(5): 7

(6): 4

Average age: 5.17, the highest age: 7.00

child 2283 done, the exit status is 0

child 2284 done, the exit status is 0

child 2285 done, the exit status is 0

Parent done.

1. **the limitations if your program does not output the expected result.**

Limitations if the program does not output the expected result:

If the program does not output the expected result, there could be several limitations or issues:

* Incorrect command-line arguments:

Make sure you provide the correct number of command-line arguments (two filenames) when running the program. Otherwise, it will display an error message.

* File-related issues:

Ensure that the specified input files (file1.txt and file2.txt) exist and are accessible in the current directory or provide the correct paths.

Check if the files have the necessary read and write permissions.

If the files cannot be opened, modified, or read successfully, the corresponding child processes may fail or produce unexpected results.

* Input validation:

The code assumes that the user will enter valid integer values for ages. If invalid input is provided, such as non-numeric values, the program will prompt the user to re-enter the age. Ensure valid input is provided to avoid unexpected behavior.

* Program dependencies:

The code relies on existing Unix programs, such as "sort," to perform certain tasks. If these programs are not available or accessible in the system's environment, the corresponding child processes may fail or produce unexpected results.

* Error handling:

The provided code includes basic error handling using error messages and exits in case of failures (e.g., file opening errors). However, it may not cover all possible error scenarios. Additional error handling mechanisms and checks can be implemented to improve the robustness of the program.