FATİH SULTAN MEHMET VAKIF ÜNİVERSİTESİ

Student:

Name: Ali

Surname: Yılmaz

ID Number: 2121221031

Department: Computer Engineering

Project:

Topic: Distributed Calculator

Course:

Name: Operating Systems

Instructor: Ali Yılmaz Çamurcu

Content

1- Project Topic

This project aims to develop a multi-operation calculator. The calculator supports four basic mathematical operations (addition, subtraction, multiplication, division) and saves the result of each operation in a file. The project is implemented in the C programming language and designed to run on a Linux environment. The project consists of the following main components:

1. **calculator.c Main Program:**
    - Accepts operation selection from the user and directs the entered two numbers to the appropriate subprocess.
    - Uses pipe and fork system calls to facilitate data communication between processes.
    - Displays the result and saves it to a file after the operations are completed.
2. **Subprograms (addition.c, subtraction.c, etc.):**
    - Each subprogram performs a specific mathematical operation (e.g., addition, subtraction).
    - Processes the numbers entered by the user and returns the result to the main program.
3. **Result Logging Mechanism:**
    - The saver.c program saves the result received from the main program into the results.txt file.
    - Results are stored sequentially and in an organized manner.

2- Tasks Completed During the Project

Coding Process:

- Writing the calculator.c main program:
  - Designed the user interface.
  - Created menus, operation selections, and data entry mechanisms.
- Developed subprograms for the following:
  - addition.c: Adds two numbers and returns the result.
  - subtraction.c: Subtracts the second number from the first.
  - multiplication.c: Multiplies two numbers.
  - division.c: Divides the first number by the second (handles division errors).
- The saver.c file was added to save the operation results in results.txt.

Use of Pipe and Fork:

- **Pipe Mechanism:** Facilitated data transfer between the main program and subprograms.
- **Fork:** Created separate processes for each subprogram.

Debugging and Testing:

- Each program component was tested independently.
- Error handling mechanisms were added for special cases (e.g., division by zero).

Result Logging:

- Successfully saved results in the results.txt file.
- **Example output:**
  - Result: 15
  - Result: 5
  - Result: 42
  - Result: 3.00

File Organization and Packaging:

- All files were compiled and made executable using a Makefile.
- Project files were compressed and prepared for delivery.

3- Additional Notes

- **Technical Details:**
  - The project utilized system-level capabilities of the C programming language.
  - Pipe, fork, and exec system calls were explored and successfully implemented.
  - Compilation was performed using the GCC compiler on a Linux operating system.
- **User Interface:**
  - After receiving the operation selection from the user, the program asked for two numbers.
  - Menus were designed to be simple and user-friendly:

  Calculator Menu:

  1. Addition
  2. Subtraction
  3. Multiplication
  4. Division
  5. Exit

  Enter your choice:
- **Special Case Handling:**
  - Appropriate error messages were displayed when the user provided incorrect input.
  - For instance, when attempting division by zero, the following message was shown:

    Error: Division by zero is not allowed.

4- References

1. Brian Kernighan, Dennis Ritchie, *The C Programming Language*, Prentice Hall, 1988. Accessed at: https://archive.org/details/cprogramminglang00bria
2. man7.org, "Linux manual pages - pipe(2)", accessed at: https://man7.org/linux/man-pages/man2/pipe.2.html.
3. man7.org, "Linux manual pages - fork(2)", accessed at: https://man7.org/linux/man-pages/man2/fork.2.html.
4. GeeksforGeeks, "fork() in C", accessed at: https://www.geeksforgeeks.org/fork-system-call/.
5. Educational materials provided by the instructor.