CSE 344 System Programming Homework #2
Project Report

Overview

The system designed for this project consists of a parent process and two child processes, utilizing two FIFOs (named pipes) to facilitate inter-process communication (IPC). This structure is designed to model a practical environment where processes perform separate tasks that are coordinated through IPC without shared memory.

Component Descriptions

Parent Process:

• **Initial Setup:** Initializes the system by creating two FIFOs, FIFO1 and FIFO2, using the mkfifo system call. These FIFOs serve as channels for sending and receiving data between the child processes.

• **Process Management:** The parent process uses the fork() system call twice to spawn two child processes. It manages these processes, handling their creation, execution, and termination.

• **Signal Handling:** Implements signal handling for SIGCHLD to manage child process termination. This handler reaps zombie processes using waitpid() and checks if all child processes have completed to clean up and exit the parent process

• **Data Handling:** Writes initial data (an array of integers) into FIFO1 and sends control commands through FIFO2 to synchronize operations between the child processes.

Child Process 1:

• **Read Operation:** Opens FIFO1 in read mode at startup and reads an array of integers. This process simulates data processing by calculating the sum of these integers.

• **Computation:** After computing the sum, this process writes the result to FIFO2, thereby passing computed results to Child Process 2.

• **Exit:** Once the data writing is complete, this process terminates, signaling the parent via SIGCHLD.

Child Process 2

• **Read Operation:** Opens FIFO2 to read the computation results sent by Child Process 1. This process waits until it receives the sum calculation before proceeding.

• **Display:** Upon receiving the sum, it performs additional checks or computations as required (e.g., confirming the operation type sent through FIFO2) and then displays the result. I could not manage synchronization so I gave command manually as “multiply” here to show that program works except that part.

• **Exit:** Similar to Child Process 1, after completing its task, it terminates and sends a signal to the parent.

Process Flow and Communication

1. Initialization: The parent process sets up the environment, including signal handling and FIFO creation.

2. Data Writing: The parent writes an array of integers to FIFO1, FIFO2 and a control command to FIFO2.

3. Data Reading and Computation by Child Process 1: Child Process 1 reads from FIFO1, calculates the sum, and writes this sum to FIFO2.

4. Command and Result Handling by Child Process 2: Child Process 2 reads from FIFO2, retrieves the sum, verifies the command, performs any additional required computations, and displays the final result.

5. Cleanup and Termination: As each child process completes its task and exits, the parent process, notified via SIGCHLD, handles cleanup (e.g., unlinking FIFOs) and eventually exits once all child processes have terminated.

Synchronization and Error Handling

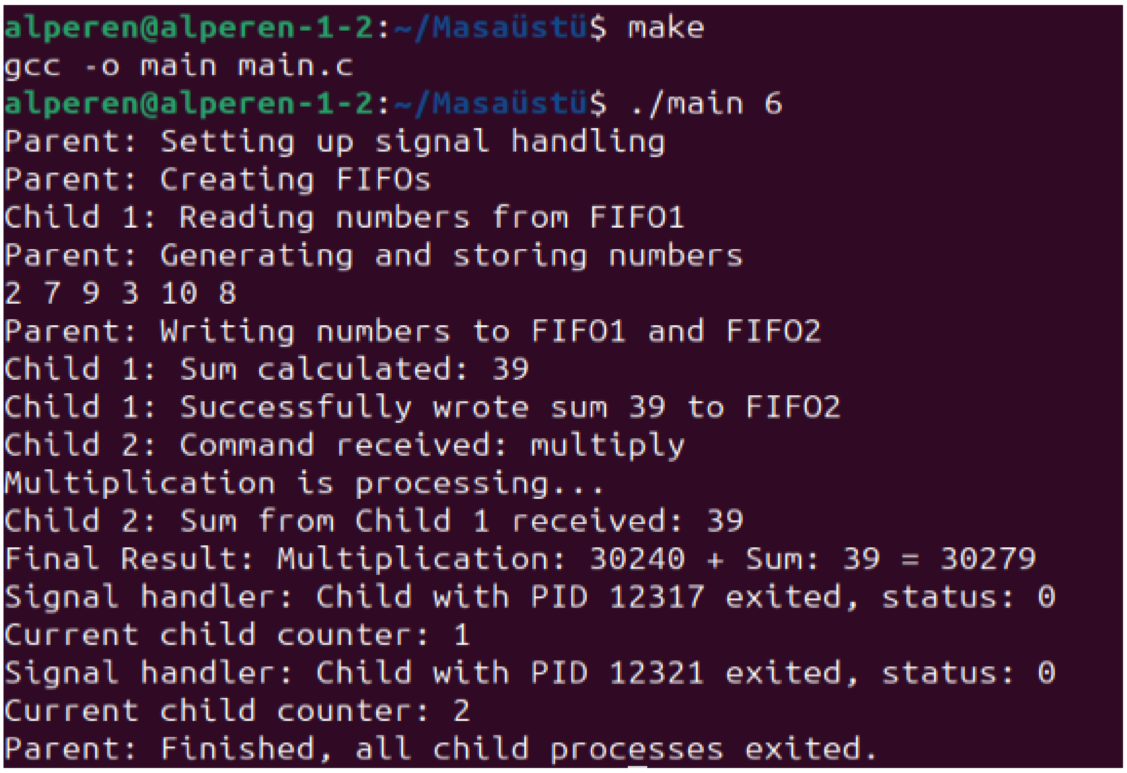
• **Synchronization:** Ensuring that Child Process 2 does not attempt to read from FIFO2 before Child Process 1 has written to it is crucial. This is managed by controlling the sequence of operations and using sleep calls to simulate process delays in this project.

• **Error Handling:** Throughout the process flow, error checks are in place to handle failures in FIFO creation, open, read/write operations, and process spawning. Errors are reported to the user via perror().

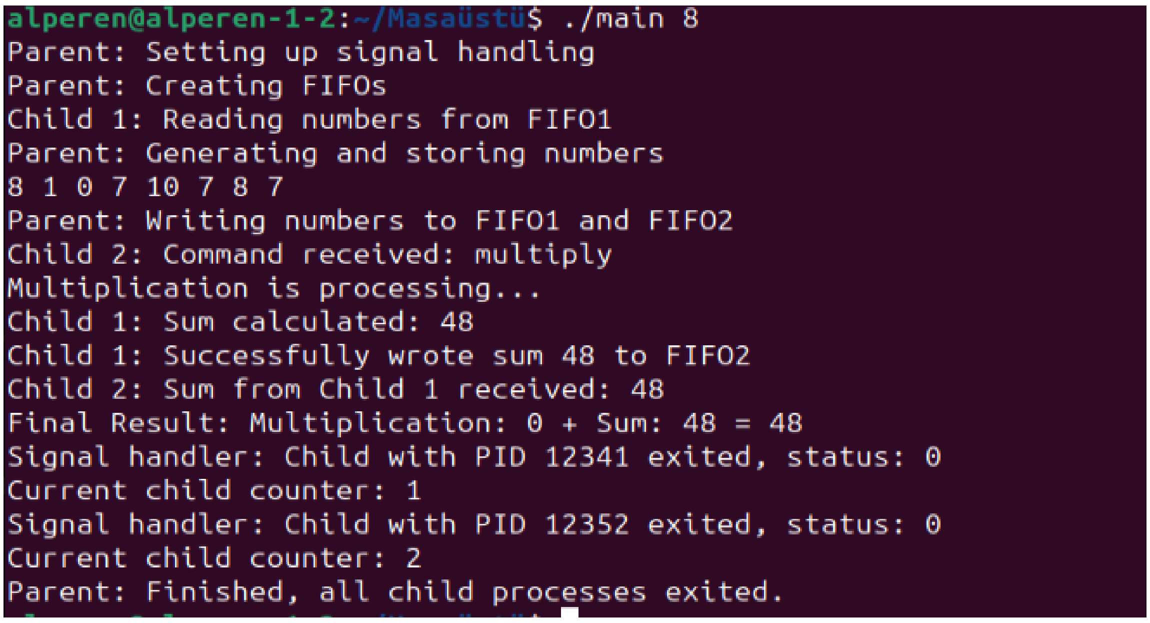
Example Test Results

To run the program make command should entered.

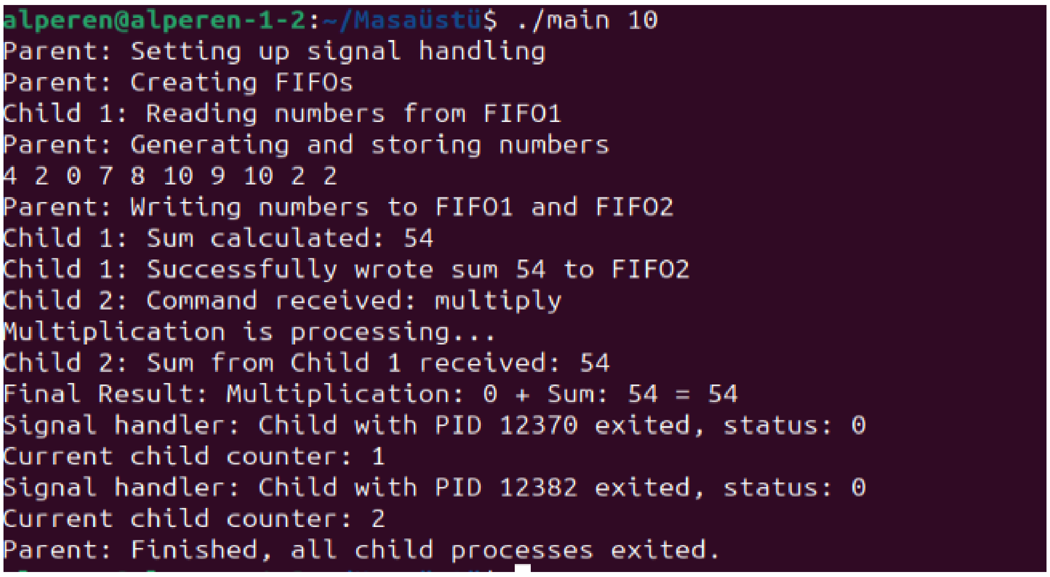
Example usage is ./main <integer>.



```
alperen@alperen-1-2:~/Masaüstü$ make
gcc -o main main.c
alperen@alperen-1-2:~/Masaüstü$ ./main 6
Parent: Setting up signal handling
Parent: Creating FIFOs
Child 1: Reading numbers from FIFO1
Parent: Generating and storing numbers
2 7 9 3 10 8
Parent: Writing numbers to FIFO1 and FIFO2
Child 1: Sum calculated: 39
Child 1: Successfully wrote sum 39 to FIFO2
Child 2: Command received: multiply
Multiplication is processing...
Child 2: Sum from Child 1 received: 39
Final Result: Multiplication: 30240 + Sum: 39 = 30279
Signal handler: Child with PID 12317 exited, status: 0
Current child counter: 1
Signal handler: Child with PID 12321 exited, status: 0
Current child counter: 2
Parent: Finished, all child processes exited.
```

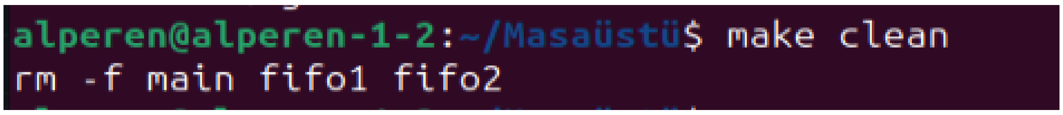


```
alperen@alperen-1-2:~/Masaüstü$ ./main 8
Parent: Setting up signal handling
Parent: Creating FIFOs
Child 1: Reading numbers from FIFO1
Parent: Generating and storing numbers
8 1 0 7 10 7 8 7
Parent: Writing numbers to FIFO1 and FIFO2
Child 2: Command received: multiply
Multiplication is processing...
Child 1: Sum calculated: 48
Child 1: Successfully wrote sum 48 to FIFO2
Child 2: Sum from Child 1 received: 48
Final Result: Multiplication: 0 + Sum: 48 = 48
Signal handler: Child with PID 12341 exited, status: 0
Current child counter: 1
Signal handler: Child with PID 12352 exited, status: 0
Current child counter: 2
Parent: Finished, all child processes exited.
```



```
alperen@alperen-1-2:~/Masaüstü$ ./main 10
Parent: Setting up signal handling
Parent: Creating FIFOs
Child 1: Reading numbers from FIFO1
Parent: Generating and storing numbers
4 2 0 7 8 10 9 10 2 2
Parent: Writing numbers to FIFO1 and FIFO2
Child 1: Sum calculated: 54
Child 1: Successfully wrote sum 54 to FIFO2
Child 2: Command received: multiply
Multiplication is processing...
Child 2: Sum from Child 1 received: 54
Final Result: Multiplication: 0 + Sum: 54 = 54
Signal handler: Child with PID 12370 exited, status: 0
Current child counter: 1
Signal handler: Child with PID 12382 exited, status: 0
Current child counter: 2
Parent: Finished, all child processes exited.
```

To remove executables and other files “make clean” should entered.



```
alperen@alperen-1-2:~/Masaüstü$ make clean
rm -f main fifo1 fifo2
```

Alperen Duran

200104004024