CSE344 Homework2 Report

Akif Safa Angi-200104004079

1. Problem Definition and My Solution Approach

The task required the implementation of an inter-process communication (IPC) system using FIFOs (First-In-First-Out) in C. The objective was to create a parent process that spawns two child processes and facilitates communication between them through named pipes (FIFOs). Each child process was assigned specific tasks, including reading random numbers from the FIFOs, performing operations (such as summation and multiplication), and printing the sum of results on the screen. Signal handling for SIGCHLD and SIGINT, error handling, and bonus features like zombie process protection and printing exit statuses were also part of the requirements.

FIFO Creation

Two FIFOs, named FIFO1 and FIFO2, were created using `mkfifo` function. FIFO1 was utilized to transmit random numbers from the parent process to Child Process 1, while FIFO2 was used for communication between Parent Process - Child Process 2 and Child Process 1 - Child Process 2.

Parent Process

The parent process first takes an argument from the command line and performs its validation. Then, it creates FIFOs. Subsequently, it uses `fork()` to create child processes. After establishing connections for writing and reading using FIFOs with child processes, it generates a random number array (for easier understanding, random numbers were modulated by 10) based on the argument number received from the command line and sends it to Child 2 and Child 1. It also sends a command ("multiply") to FIFO2 for Child Process 2. After the writing operations, it prints "proceeding" to the screen every 2 seconds within a while loop and waits for the child processes to finish. The while loop maintains a counter for the child processes, incrementing it by one each time a child process finishes. Once all child processes have finished, it exits the loop, deletes the FIFOs, and terminates the program.

Child Process 1

After establishing a connection with the parent process using FIFO1, it sleeps for 10 seconds. This process reads the random numbers from FIFO1, calculates their sum, and writes the result to FIFO2.

Child Process 2

After establishing a connection with the parent process using FIFO2, it sleeps for 10 seconds. Then, after reading the random array and the command from FIFO2, it performs the necessary command operation with the random array numbers. Next, it reads the result from Child 1 through FIFO2, adds this result to the result obtained from the command, and prints it to the screen.

Signal Handling

Signal handlers were implemented for SIGCHLD and SIGINT signals. The SIGCHLD handler reaped terminated child processes, printed their exit statuses, and updated a counter to track the number of finished child processes. The SIGINT handler set a flag to indicate interruption, which was checked at various points in the program to ensure clean termination.

Error Handling

Error handling mechanisms were implemented to detect and handle errors related to FIFO creation, data/command transmission, and child process execution. Error messages were provided to indicate the nature of the error encountered.

Bonus Features

Additional features such as zombie process protection and printing exit statuses were implemented to enhance the robustness and usability of the system.

2-) Detailed Implementation

FIFO Usage

- FIFO1: Used to transmit random numbers from the parent process to Child Process 1.

- FIFO2: Facilitates simplified communication between the parent process, Child Process 2, and Child Process 1. It involves sending the random array to Child Process 2, sending the multiplication command to Child Process 2, and transmitting the summation result from Child Process 1 to Child Process 2.

Parent Process Actions:

- Spawned child processes and set up signal handlers.

- Initialized the random number array.

- Transmitted random numbers and command to respective FIFOs.

- Monitored child process termination and handled clean-up.

Child Process Actions:

- Child Process 1:

  - Read random numbers from FIFO1.

  - Calculated their sum.

  - Wrote the result to FIFO2.

Child Process 2 Actions:

  - Read the random numbers from FIFO2.

  - Read the command from FIFO2.

  - Performed multiplication operation if the command was "multiply".

  - Printed the sum of results obtained from both child processes.

3-) System Requirements

1-) The program just requires two commands from the command line. The first is the name of the program, and the second is the number indicating how many elements the array should have.

2-) After typing 'make', it will delete files other than main.c and makefile (to ensure no leftover files from the old program) and compile the program.

You can run the program with “./main <number>” command after compiling it using the makefile. The makefile also includes an run command. It can be used by using “make run <number>” command.

Also “make clean” remove the all files except main.c and makefile.

4-) The Parts I Have Completed

I have completed all the requirements of the homework and tested.

5-) Testing and Validation

The implemented solution underwent rigorous testing to ensure functionality and correctness. Test scenarios included successful creation and usage of FIFOs, accurate calculation of results by child processes, proper handling of signal interruptions, and error-free execution of the program.

For successful run

```
Number of arrays:
2 2 2 2 3
proceeding
proceeding
proceeding
proceeding
Child process with PID 106 terminated with status: 0
Sum of the two results: 59
proceeding
proceeding
Child process with PID 107 terminated with status: 0
```

For interrupt

```
Number of arrays:
2 4 4 2 1
^Cproceeding
SIGINT caught by: 108
SIGINT caught by: 109
Child process with PID 109 terminated with status: 255
```