GEBZE TECHNİCAL UNIVERSITY

CSE344

System Programming

Homework 2 Report

1. How to Run?

Open the terminal and navigate to the source directory. Then, compile the program by typing "**make**" and execute it using "**./ipc <number>**". Once executed, the program will produce output as specified by its functionality. To clean up generated files, use "**make clean**", which will remove the executable and any temporary files. Make clean is necessary to run again.

2. Parent Process

- The parent process first creates two FIFOs using the **mkfifo** function.
- It generates an array of random numbers and sends it along with a command to the child processes through the FIFOs.
- Two child processes are created using the **fork** system call, each assigned to one FIFO.
- A signal handler for **SIGCHLD** is set up to handle child process termination using the **sigaction** function.
- Upon receiving a **SIGCHLD** signal, the signal handler reaps the terminated child process using **waitpid** and increments a counter.
- Once all child processes have exited, the parent process terminates.

3. Child Process 1

- Reads random numbers from the first FIFO and calculates their sum.
- Writes the sum to the second FIFO.

4. Child Process 2

- Reads a command from the second FIFO to perform a multiplication operation.
- Performs the multiplication operation if the command is "**multiply**".

5. Error Handling

- Error handling is implemented throughout the code using functions like **perror** and checking for return values of system calls.
- Error messages are printed to **stderr** using perror in case of failures, providing clear information about the cause of the error.

6. Zombie Protection Method (Bonus)

- To prevent zombie processes, we reap terminated child processes using **waitpid** in the signal handler for **SIGCHLD**.
- This ensures that terminated child processes are properly cleaned up, preventing them from becoming zombies.

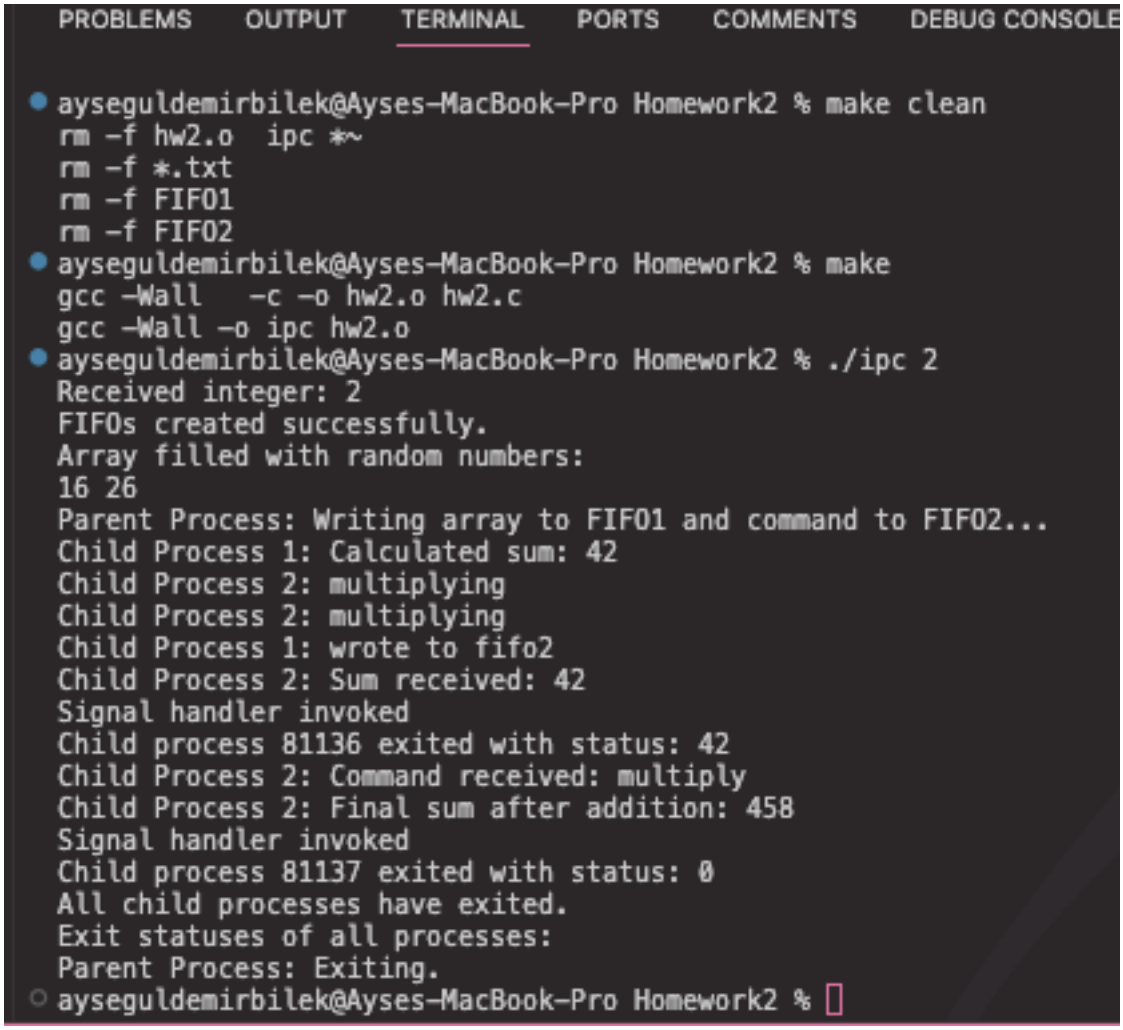
7. Printing Exit Status (Bonus)

- Exit statuses of all processes are printed at the end of the program using **wait** in a loop.
- If a child process exits normally, its exit status is printed.
- If **ECHILD** error is encountered, indicating no more child processes to wait for, an appropriate message is printed.
- Any other errors during waiting are also handled and appropriate error messages are printed.

8. Missing Task

- The parent process doesn't enters a loop, printing the message "**Pending**..." every two seconds.

9. Output


PROBLEMS
OUTPUT
TERMINAL
PORTS
COMMENTS
DEBUG CONSOLE
ayseguldemirbilek@Ayses-MacBook-Pro Homework2 % make clean
rm -f hw2.o ipc *~
rm -f *.txt
rm -f FIFO1
rm -f FIFO2
ayseguldemirbilek@Ayses-MacBook-Pro Homework2 % make
gcc -Wall -c -o hw2.o hw2.c
gcc -Wall -o ipc hw2.o
ayseguldemirbilek@Ayses-MacBook-Pro Homework2 % ./ipc 2
Received integer: 2
FIFOs created successfully.
Array filled with random numbers:
16 26
Parent Process: Writing array to FIFO1 and command to FIFO2...
Child Process 1: Calculated sum: 42
Child Process 2: multiplying
Child Process 2: multiplying
Child Process 1: wrote to fifo2
Child Process 2: Sum received: 42
Signal handler invoked
Child process 81136 exited with status: 42
Child Process 2: Command received: multiply
Child Process 2: Final sum after addition: 458
Signal handler invoked
Child process 81137 exited with status: 0
All child processes have exited.
Exit statuses of all processes:
Parent Process: Exiting.
ayseguldemirbilek@Ayses-MacBook-Pro Homework2 %