

Gebze Technical University
Computer Engineering
Department

CSE 344 – System Programming

Homework-2 Report

Muhammet Akkurt

1901042644

Purpose and Details of the Assignment:

The purpose of this program is to demonstrate the use of FIFOs (named pipes) for inter-process communication (IPC) and the use of signal processing to manage sub-processes in an application. The program generates a sequence of random numbers, computes their sum and product in separate subprocesses, and uses signal processing to monitor the termination of these subprocesses.

The program initiates by validating the command line input to ensure it receives a single integer which determines the size of the array used for generating random numbers.

The program dynamically allocates memory for storing random numbers. It then creates two FIFOs for the IPC between parent and child processes. An array of random numbers is generated, each number ranging from 0 to 9. These numbers are displayed in the standard output.


Signal processing for “SIGCHLD” is set using the “sigaction” system call. The handler function is designed to capture exiting subprocesses, increase the number of completed subprocesses and log the exit status.


The first child forks the process. After 10 seconds of sleep, it reads the numbers from the first FIFO, calculates their sum and writes the result to the second FIFO.

A second child process forks after the first one. After 10 seconds of sleep, it reads the sum and a command from the second FIFO. After checking that it has received a "multiply" command, it calculates the product of the same randomly

generated set of numbers and prints the product result on the screen by adding the sum result from the first FIFO with the product result.

The use of `sleep(2)` in this code is primarily for synchronizing the operations between parent and child processes to prevent race conditions and ensure data is available when needed.

 **void signal_handler(int signal):** Its primary role is to ensure that when a child process terminates, its exit status is captured, and the process is properly reaped to prevent it from becoming a zombie. The function employs `waitpid()` with the `WNOHANG` option in a loop. This approach checks for any child processes that have terminated without blocking if none have. It processes multiple child terminations in a single signal reception, effectively clearing up any terminated children immediately. For each terminated child, the function logs the process ID (PID) and its exit status to the standard error.

 **void setup_signal_handling():** This function configures the handling of the `SIGCHLD` signal. It sets up the `signal_handler` function to respond to `SIGCHLD` signals. The `SA_RESTART` flag is set to ensure that system calls interrupted by the signal are automatically restarted. The `SA_NOCLDSTOP` flag is used to suppress the signal when a child process stops (rather than exits), reducing unnecessary signal handling if only termination is of concern.

Test Scenarios:

```
ubuntu@ubuntu-VirtualBox: ~/Desktop/workspace
ubuntu@ubuntu-VirtualBox:~/Desktop/workspace$ make
gcc -Wall -g -o main main.c
ubuntu@ubuntu-VirtualBox:~/Desktop/workspace$ ./main 5
Randomly generated numbers: 2 7 5 4 3
proceeding
proceeding
proceeding
Child with PID 4145 exited with status 0.
proceeding
proceeding
proceeding
Received sum: 21 and command: multiply in child process
proceeding
proceeding
proceeding
Multiplication result: 840
Total result (sum + multiplication): 861
Child with PID 4149 exited with status 0.
All child processes have completed.
ubuntu@ubuntu-VirtualBox:~/Desktop/workspace$
```

```
ubuntu@ubuntu-VirtualBox: ~/Desktop/workspace
ubuntu@ubuntu-VirtualBox:~/Desktop/workspace$ make
gcc -Wall -g -o main main.c
ubuntu@ubuntu-VirtualBox:~/Desktop/workspace$ ./main 10
Randomly generated numbers: 5 5 7 4 7 0 8 4 2 0
proceeding
proceeding
proceeding
Child with PID 4198 exited with status 0.
proceeding
proceeding
proceeding
Received sum: 42 and command: multiply in child process
proceeding
proceeding
proceeding
Multiplication result: 0
Total result (sum + multiplication): 42
Child with PID 4203 exited with status 0.
All child processes have completed.
ubuntu@ubuntu-VirtualBox:~/Desktop/workspace$
```

```
ubuntu@ubuntu-VirtualBox: ~/Desktop/workspace
ubuntu@ubuntu-VirtualBox:~/Desktop/workspace$ make
gcc -Wall -g -o main main.c
ubuntu@ubuntu-VirtualBox:~/Desktop/workspace$ ./main 50
Randomly generated numbers: 2 3 5 9 4 9 4 8 5 5 3 8 6 7 0 8 8 0 4 5 1 7 6 4 7 5 6 3 3 9 2 7 2 7 9 8 8
3 7 5 1 2 3 7 2 5 8 0 5 2
proceeding
proceeding
proceeding
Child with PID 4234 exited with status 0.
proceeding
proceeding
proceeding
Received sum: 247 and command: multiply in child process
proceeding
proceeding
proceeding
Multiplication result: 0
Total result (sum + multiplication): 247
Child with PID 4239 exited with status 0.
All child processes have completed.
ubuntu@ubuntu-VirtualBox:~/Desktop/workspace$
```

```
ubuntu@ubuntu-VirtualBox: ~/Desktop/workspace
ubuntu@ubuntu-VirtualBox:~/Desktop/workspace$ make
gcc -Wall -g -o main main.c
ubuntu@ubuntu-VirtualBox:~/Desktop/workspace$ ./main 100
Randomly generated numbers: 8 9 4 9 9 7 4 6 8 7 0 0 5 5 6 5 8 3 3 7 5 4 6 8 3 6 5 0 4 5 2 4 6 6 3 7 5
7 3 3 7 4 4 2 1 2 8 1 5 3 0 2 0 8 1 3 4 8 5 0 5 8 7 1 6 0 8 2 0 1 7 7 7 1 1 0 4 1 1 1 5 4 6 5 2 7 0
9 7 7 9 2 5 6 3 2 9 3 6 9
proceeding
proceeding
proceeding
Child with PID 4326 exited with status 0.
proceeding
proceeding
proceeding
Received sum: 447 and command: multiply in child process
proceeding
proceeding
proceeding
Multiplication result: 0
Total result (sum + multiplication): 447
Child with PID 4331 exited with status 0.
All child processes have completed.
ubuntu@ubuntu-VirtualBox:~/Desktop/workspace$
```