CSE 344
SYSTEM PROGRAMMING

HW 2 Report

Emre Güven

1901042621

INTRODUCTION

In this homework I have implemented a communication protocol between parent process and two child processes using FIFOs(named pipes) in C programming language. I used 2 FIFOs to transfer required data, and used 1 FIFO (inspired by web server fifo structure) to handle synchronization.
To compile program in shell: $> make
To run program correctly: $> ./program <number>
To clean unnecessary files: $> make clean

METHODS AND THEIR TASKS

main()

- Validates command-line arguments to ensure an array size is provided.
- Creates 3 FIFOs for inter-process communication. FIFO1 and FIFO2 handles necessary data transfers. SERVER_FIFO is used to send/receive request to make synchronization while data transfer.
- Generates random numbers and displays them.
- Sets up signal handlers for SIGCHLD and SIGUSR1. SIGCHLD for controlling child process terminations. SIGUSR1 for notifying parent process about sleep(10) is finished and child processes are ready to run. Until SIGUSR1 is caught, parent process displays proceeding message for each 2 seconds.
- Spawns two child processes using fork().
- Reads and writes data to/from FIFOs based on the child processes' requirements. Waits a request from child process 1. Because one end of SERVER_FIFO might be closed firstly, the process with closed end is blocked until both ends are opened. Send array to child process 1 using FIFO1. Then send command and array to child process 2 using FIFO2.
- Waits for child processes to finish their tasks.
- Cleans up resources before exiting.

handle_child1(int arr_size)

- Sends a request to the parent via SERVER_FIFO to get numbers array first.
- Reads numbers array from FIFO1.
- Calculates the sum of the numbers.
- Waits a request from child process 2 to write sum of numbers to FIFO2 in correct time.
- Sends the sum of numbers to the child process 2 via FIFO2.

handle_child2(int arr_size)

- Reads the 'multiply' command from FIFO2, compare read text with 'multiply' to ensure reading is done correctly.
- Reads the numbers from FIFO2.
- Calculates the multiplication of the numbers.

- Sends a request to the child process 1 via SERVER_FIFO to get sum of numbers.
- Reads the sum of numbers from FIFO2.

sigchld_handler(int signo)

- Handles the SIGCHLD signal, which is sent to the parent when a child process terminates.
- Prints the exit status or signal that terminated the child process.

sigusr1_handler(int signo)

- Handles the SIGUSR1 signal, which is used to notify the parent that child processes have finished their tasks.
- Sets the atomic child_finished flag to 1.

create_fifo(const char *fifo_path)

- Creates a FIFO (named pipe) if it doesn't already exist.
- Uses mkfifo() system call to create the FIFO with appropriate permissions.

open_fifo_with_retry(const char *fifo_path, int flags)

- Uses open() system call with a retry loop to ensure successful opening by handling interruptions.

read_fifo_with_retry(int fd, void *buf, size_t count)

- Uses read() system call with a retry loop to ensure successful reading by handling interruptions.

write_fifo_with_retry(int fd, const void *buf, size_t count)

- Uses write() system call with a retry loop to ensure successful writing by handling interruptions.

remove_fifo(const char *fifo_path)

- Uses unlink() system call to delete the FIFO with error checks.

close_fd(int fd)

- Uses close() system call to close the file descriptor with error checks .

print(const char *message)

- Uses write() system call to print messages to stdout with a retry loop to handle interruptions.

ERROR HANDLING

System Call Error Handling:

After all system calls er a system call, the program checks the return value. If it's -1, an error occurred, an exit(EXIT_FAILURE) is called and program terminated. The program uses perror() to print a descriptive error message to stderr, which provides information about the type of error encountered.

Retry Mechanism:

For operations that involve file or FIFO operations (reading, writing, opening), the program implements a retry mechanism using a do-while loop. This retry mechanism helps to handle errors or interruptions (like signals) that might occur during these operations. The loop will keep retrying the failed operation until it succeeds or encounters a non-recoverable error.

Signal Handling:

The program sets up signal handlers for SIGCHLD and SIGUSR1.

SIGCHLD is used to handle child process terminations, ensuring that the parent is informed when a child process exits and can handle any cleanup or reporting.

SIGUSR1 is used to notify the parent when child processes have finished sleeping and are ready to run. This helps in synchronizing the parent and child processes' activities after sleep(10).

Resource Cleanup:

Before exiting, the program cleans up by freeing allocated memory and removing FIFOs. Any failure during cleanup is also handled, ensuring that resources are properly released, and the environment is left in a consistent state.

BONUS PARTS

For zombie protection, sigchld_handler() in my code calls waitpid() with WNOHANG to reap terminated child processes without blocking the parent process.

Also the exit statuses of all processes are printed in sigchld_handler().

TESTS

I give 4 different size to program and get the results.

```
mrguven@mrguven:~/Desktop/cse344/hw2$ make
gcc -Wall -std=gnu99 program.c -o program
mrguven@mrguven:~/Desktop/cse344/hw2$ ./program 15
Parent process: SERVER_FIFO in '/tmp/server_fifo' is created!
Parent process: FIFO1 in '/tmp/fifo1' is created!
Parent process: FIFO2 in '/tmp/fifo2' is created!
Numbers: 3 13 7 1 1 2 4 9 13 12 7 13 5 8 10
Parent process: Proceeding...
Parent process: Proceeding...
Parent process: Proceeding...
Parent process: Proceeding...
Parent process: Proceeding...
Parent process: numbers array are written to FIFO1
Parent process: 'multiply' command is written to FIFO2
Parent process: numbers array are written to FIFO2
Child process 1: numbers are read from FIFO1
Child process 2: command 'multiply' is read from FIFO2
Child process 2: numbers are read from FIFO2
Child process 2: Multiplication result = 111614630400
Child process 1: Summation result = 108
Child process 1: sum = 108 is written to FIFO2
Child process 2: sum = 108 is read from FIFO2
Result: 111614630508
Child process pid: 14551 exited with status: 0
Child process pid: 14552 exited with status: 0
Parent process: Terminating...
mrguven@mrguven:~/Desktop/cse344/hw2$
```

```
mrguven@mrguven:~/Desktop/cse344/hw2$ ./program 6
Parent process: SERVER_FIFO in '/tmp/server_fifo' is created!
Parent process: FIFO1 in '/tmp/fifo1' is created!
Parent process: FIFO2 in '/tmp/fifo2' is created!
Numbers: 5 0 5 0 4 5
Parent process: Proceeding...
Parent process: Proceeding...
Parent process: Proceeding...
Parent process: Proceeding...
Parent process: Proceeding...
Parent process: numbers array are written to FIFO1
Child process 1: numbers are read from FIFO1
Parent process: 'multiply' command is written to FIFO2
Parent process: numbers array are written to FIFO2
Child process 1: Summation result = 19
Child process 2: command 'multiply' is read from FIFO2
Child process 2: numbers are read from FIFO2
Child process 2: Multiplication result = 0
Child process 1: sum = 19 is written to FIFO2
Child process 2: sum = 19 is read from FIFO2
Result: 19
Child process pid: 14599 exited with status: 0
Child process pid: 14600 exited with status: 0
Parent process: Terminating...
mrguven@mrguven:~/Desktop/cse344/hw2$
```

```
mrguven@mrguven:~/Desktop/cse344/hw2$ ./program 25
Parent process: SERVER_FIFO in '/tmp/server_fifo' is created!
Parent process: FIFO1 in '/tmp/fifo1' is created!
Parent process: FIFO2 in '/tmp/fifo2' is created!
Numbers: 14 2 2 5 4 19 5 2 19 19 23 13 13 15 14 21 15 11 4 11 5 0 2 1 20
Parent process: Proceeding...
Parent process: Proceeding...
Parent process: Proceeding...
Parent process: Proceeding...
Parent process: Proceeding...
Parent process: numbers array are written to FIFO1
Parent process: 'multiply' command is written to FIFO2
Parent process: numbers array are written to FIFO2
Child process 2: command 'multiply' is read from FIFO2
Child process 1: numbers are read from FIFO1
Child process 2: numbers are read from FIFO2
Child process 2: Multiplication result = 0
Child process 1: Summation result = 259
Child process 2: sum = 259 is read from FIFO2
Child process 1: sum = 259 is written to FIFO2
Result: 259
Child process pid: 14607 exited with status: 0
Child process pid: 14608 exited with status: 0
Parent process: Terminating...
mrguven@mrguven:~/Desktop/cse344/hw2$
```

```
mrguven@mrguven:~/Desktop/cse344/hw2$ ./program 4
Parent process: SERVER_FIFO in '/tmp/server_fifo' is created!
Parent process: FIFO1 in '/tmp/fifo1' is created!
Parent process: FIFO2 in '/tmp/fifo2' is created!
Numbers: 3 3 2 3
Parent process: Proceeding...
Parent process: Proceeding...
Parent process: Proceeding...
Parent process: Proceeding...
Parent process: Proceeding...
Parent process: numbers array are written to FIFO1
Child process 1: numbers are read from FIFO1
Parent process: 'multiply' command is written to FIFO2
Parent process: numbers array are written to FIFO2
Child process 1: Summation result = 11
Child process 2: command 'multiply' is read from FIFO2
Child process 2: numbers are read from FIFO2
Child process 2: Multiplication result = 54
Child process 1: sum = 11 is written to FIFO2
Child process 2: sum = 11 is read from FIFO2
Result: 65
Child process pid: 14618 exited with status: 0
Child process pid: 14617 exited with status: 0
Parent process: Terminating...
mrguven@mrguven:~/Desktop/cse344/hw2$
```