GTU Department of Computer Engineering

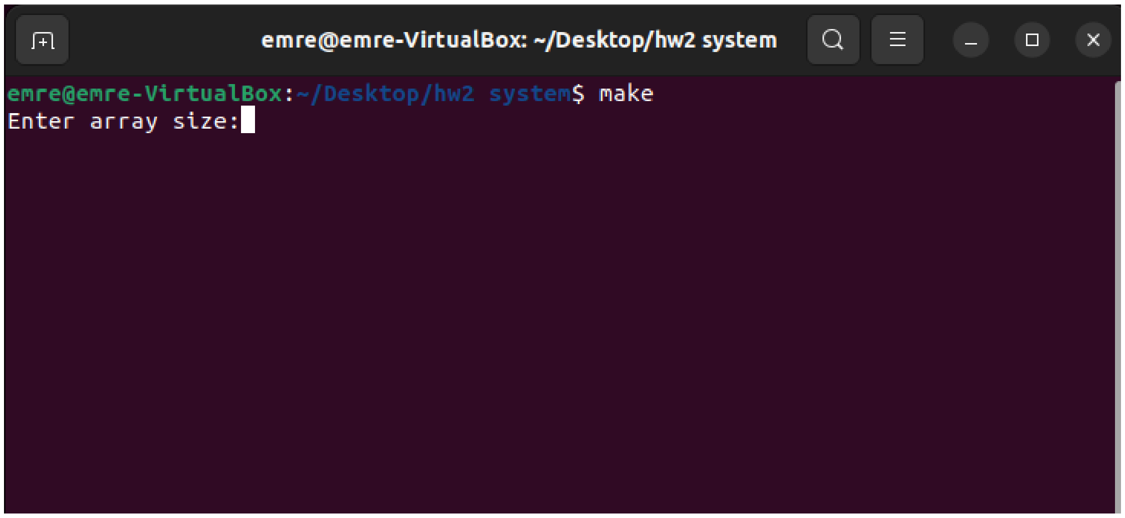
CSE 344 – Spring 2024

Homework 2 Report

EMRE YAVUZ

200104004003

Usage



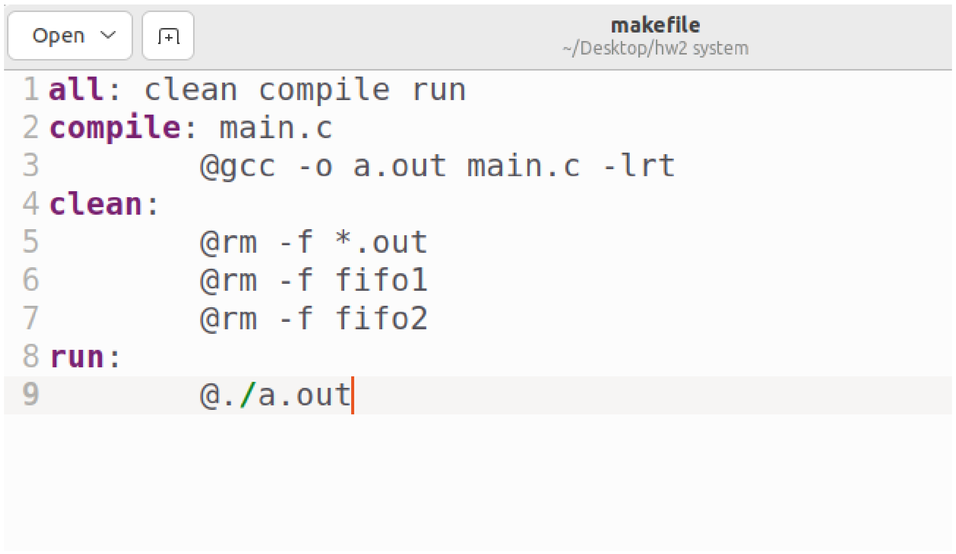
```
emre@emre-VirtualBox:~/Desktop/hw2 system$ make
Enter array size:
```

I preapared a makefile for easy compile and clean unnecessary files,fifos.

Make command has 3 steps:

make clean :Deletes old fifo and out files

make compile : Compiles main.c by using -lrt option for fifo



```
all: clean compile run
compile: main.c
	@gcc -o a.out main.c -lrt
clean:
	@rm -f *.out
	@rm -f fifo1
	@rm -f fifo2
run:
	@./a.out
```

Algorithm and Code

1. Main Function

The main function is responsible for orchestrating the inter-process communication (IPC) using named pipes (FIFO1 and FIFO2) and managing the child processes' operations for calculation.

Steps:

- Create FIFOs:
- The function starts by deleting any existing named pipes (FIFO1 and FIFO2) using unlink.
- It then prompts the user to input the array size, validating the input to ensure it is an integer.
- After validating the array size, it creates new named pipes (FIFO1 and FIFO2) using mkfifo.
- Set Up Signal Handler:
- The function initializes a sigaction struct and configures it to handle the SIGCHLD signal, which is triggered when child processes terminate.
- The handler function handle_child_termination is set for SIGCHLD with flags SA_RESTART and SA_NOCLDSTOP.
- Generate Array of Random Numbers:
- The function seeds the random number generator with the current time using srand.
- It then generates an array of random integers between 0 and 9 with the size specified by the user.

Fork Child Processes:

- The function forks two child processes (pid1 and pid2), each responsible for specific tasks:

Child Process 1:

- Waits for a short period while printing "proceeding" intermittently.
- Opens FIFO1 for reading and reads the random array from the parent process.
- Calculates the sum of the array and writes it to FIFO2.
- Closes FIFO1 and FIFO2, then exits successfully.

Child Process 2:

- Opens FIFO2 for reading and reads the array and command from the parent process.
- Reads the sum from FIFO2.
- If the command is "multiply," it calculates the product of the array and adds it to the sum to compute the total result.
- Prints the total result to stdout and exits successfully.

Parent Process:

- Writes the random array and the "multiply" command to FIFO2.
- Writes the random array to FIFO1.
- Waits for child processes to terminate, printing "proceeding" during waiting periods.
- Closes and removes the named pipes (FIFO1 and FIFO2).
- Exits successfully.

2. handle_child_termination(int signum)

- This function handles the SIGCHLD signal, which is triggered when child processes terminate.
- Reap Terminated Children:
- Uses waitpid with WNOHANG to reap all child processes that have exited.
- Print Termination Details:
- For each terminated child process, the function prints the child process's ID and its exit status.
- Increment child_count:
- Increments the global child_count variable for each terminated child process.

3. Child Process 1

- The first child process performs the following tasks:
- Wait for a Short Period:
- Waits for a short period while printing "proceeding" intermittently.
- Read from FIFO1:
- Opens FIFO1 for reading and reads the random array sent by the parent process.
- Calculate Sum:
- Calculates the sum of all elements in the array.
- Write Sum to FIFO2:
- Opens FIFO2 for writing and writes the calculated sum to FIFO2.
- Clean Up:

- Closes FIFO1 and FIFO2 and exits with success.

4. Child Process 2

- The second child process performs the following tasks:
- Read from FIFO2:
- Opens FIFO2 for reading and reads the random array and command from the parent process.
- Read Sum from FIFO2:
- Reads the sum from FIFO2.
- Check Command and Calculate Total:
- If the command is "multiply," it calculates the product of the array and adds it to the sum to compute the total result.
- Print Total Result:
- Prints the total result to stdout.
- Clean Up:
- Closes FIFO2 and exits successfully.

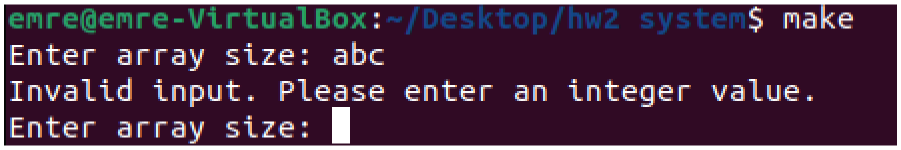
Overall, the program uses named pipes for IPC between the parent process and two child processes. The child processes perform different calculations on the array, with one computing the sum and the other computing the product and sum combination based on the provided command.

Test Cases

1.Test Case : Invalid Input (Non-numeric characters)

Input : abc

Result:

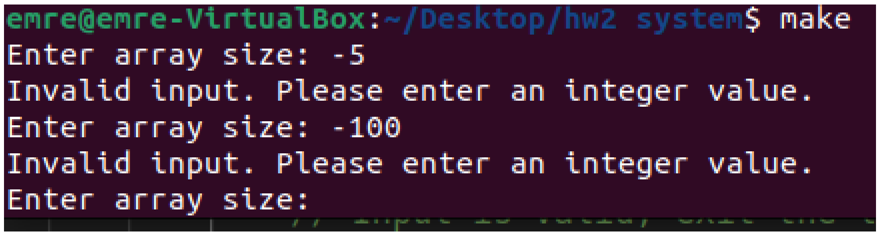


```
emre@emre-VirtualBox:~/Desktop/hw2 system$ make
Enter array size: abc
Invalid input. Please enter an integer value.
Enter array size: 
```

2.Test Case : Invalid Input (Non-numeric characters)

Input : -5

Results :

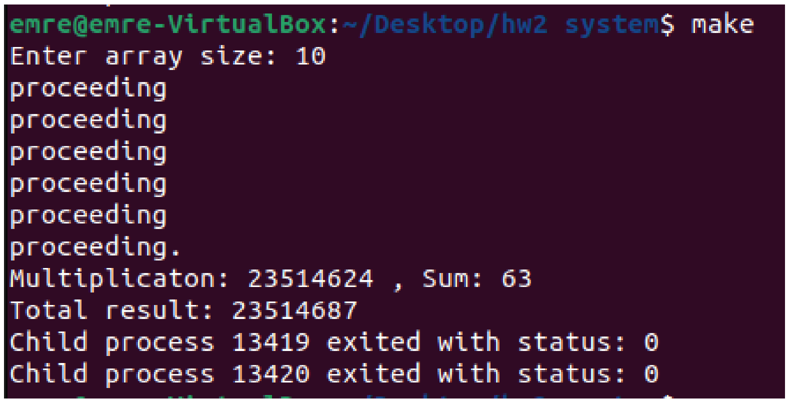


```
emre@emre-VirtualBox:~/Desktop/hw2 system$ make
Enter array size: -5
Invalid input. Please enter an integer value.
Enter array size: -100
Invalid input. Please enter an integer value.
Enter array size:
```

3.Test Case : Valid Input with non zero random number

Input : 10

Results :

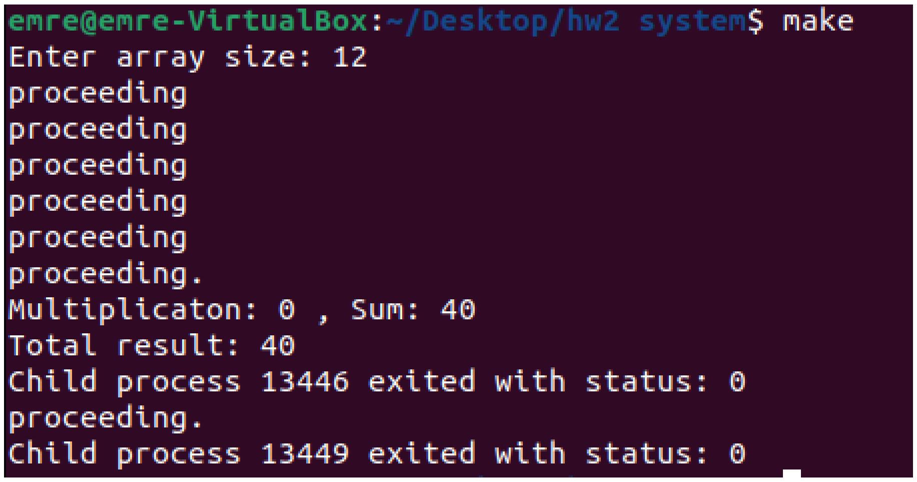


```
emre@emre-VirtualBox:~/Desktop/hw2 system$ make
Enter array size: 10
proceeding
proceeding
proceeding
proceeding
proceeding
proceeding.
Multiplicaton: 23514624 , Sum: 63
Total result: 23514687
Child process 13419 exited with status: 0
Child process 13420 exited with status: 0
```

4.Test Case : Valid Input with at least one zero random number

Input : 12

Results :



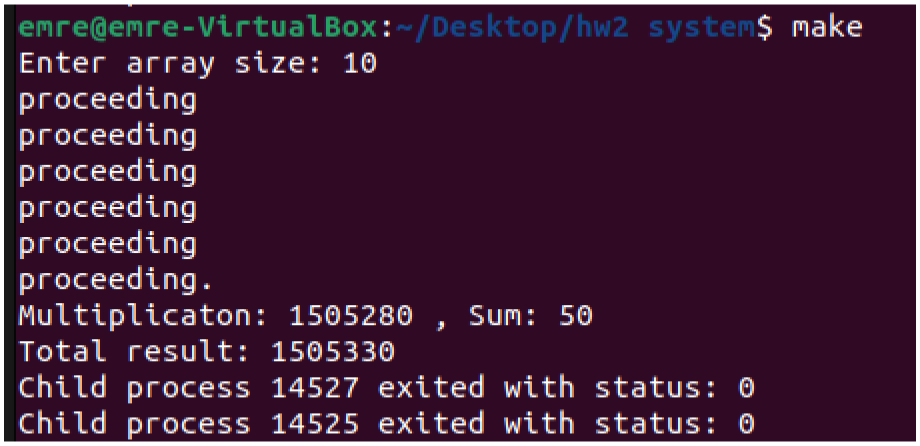
```
emre@emre-VirtualBox:~/Desktop/hw2 system$ make
Enter array size: 12
proceeding
proceeding
proceeding
proceeding
proceeding
proceeding.
Multiplicaton: 0 , Sum: 40
Total result: 40
Child process 13446 exited with status: 0
proceeding.
Child process 13449 exited with status: 0
```

5. Test Case : Valid input with broken synchronization

Input : 3

Results :

- Sometimes, the program fails to send the sum value to FIFO2, resulting in the sum remaining as 0. However, I managed to fix this issue by adding a sleep(10) statement.



```
emre@emre-VirtualBox:~/Desktop/hw2 system$ make
Enter array size: 10
proceeding
proceeding
proceeding
proceeding
proceeding
proceeding.
Multiplicaton: 1505280 , Sum: 50
Total result: 1505330
Child process 14527 exited with status: 0
Child process 14525 exited with status: 0
```