GTU Department of Computer Engineering

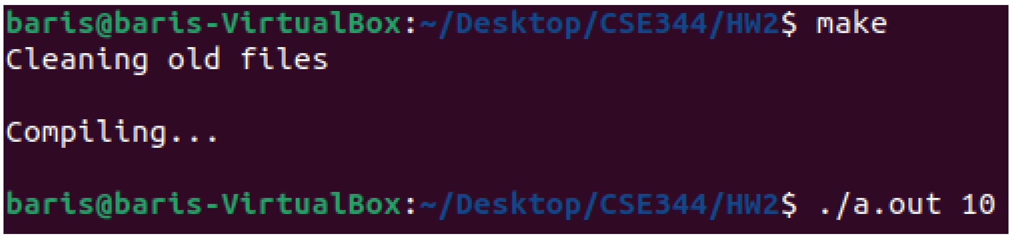
CSE 344 – Spring 2024

Homework 2 Report

Barış Batuhan Bolat

210104004029

Usage

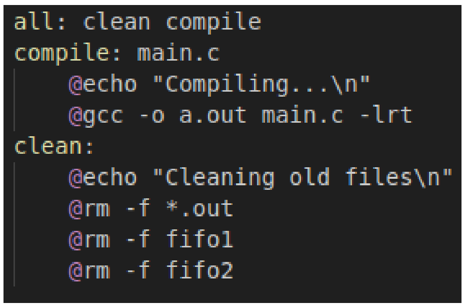


```
baris@baris-VirtualBox:~/Desktop/CSE344/HW2$ make
Cleaning old files

Compiling...

baris@baris-VirtualBox:~/Desktop/CSE344/HW2$ ./a.out 10
```

- I prepared a **makefile** for easy compile and clean unnecessary files.
- **make** command has 2 steps:
    - **make clean :** Deletes old fifo and out files
    - **make compile :** Compiles main.c by using **-lrt** option for fifo



```
all: clean compile
compile: main.c
    @echo "Compiling...\n"
    @gcc -o a.out main.c -lrt
clean:
    @echo "Cleaning old files\n"
    @rm -f *.out
    @rm -f fifo1
    @rm -f fifo2
```

Algorithm and Code

- The program expects a single integer argument.
- It validates the argument to ensure it contains only digits and is within the range **0 to 100**.

1. Parent Process

- The parent process opens **FIFO2** for writing.
- It writes the list of random numbers to **FIFO1** for Child 1.
- It writes the command string ("**multiply**") to **FIFO2** for Child 2.
- The parent enters a loop that continues until child_count reaches 2 (indicating both children have terminated).
- Inside the loop:
    - It prints "**Proceeding...**" messages while waiting child processes finish to standard output, simulating work.
    - It uses sleep to introduce a delay between checks.
- Once both children are finished:
    - The parent closes the file descriptors for **FIFO1** and **FIFO2**.
    - It removes the named pipes using unlink.
    - The parent exits with success (**EXIT_SUCCESS**).

2. Child Process 1

- The parent process forks, creating child 1.
- Inside child 1 (**pid == 0**):
    - It prints “**Proceeding…**” messages ro **STDOUT** five times for simulating work.

- Child 1 opens **FIFO1** for reading using **open(FIFO1, O_RDONLY)**.
- It reads the entire list of numbers from the parent process using **read(fd, nums2, sizeof(nums2))**.
- fd is the file descriptor for **FIFO1**.
- It calculates the sum of the numbers in the list.
- Child 1 opens **FIFO2** for writing using **open(FIFO2, O_WRONLY).**
- It writes the calculated sum to **FIFO2** using **write(fd2, &sum, sizeof(sum)).**
- fd2 is the file descriptor for **FIFO2.**
- Both file descriptors are closed.
- Child 1 exits with success (**EXIT_SUCCESS**).

3. Child Process 2

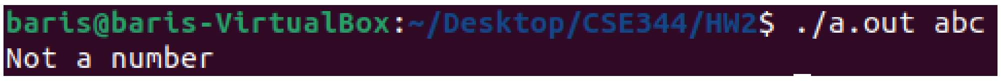
- The parent process forks again, creating Child 2.
- Inside Child 2 **(pid2 == 0)**:
    - o It prints "**Proceeding...**" messages to **STDOUT** five times, simulating work.
- Child 2 opens **FIFO2** for reading using **open(FIFO2, O_RDONLY).**
- It reads the list of numbers from the parent process using **read(fd, numbers2, sizeof(numbers2))**.
- It reads the command **string ("multiply")** from the parent process using **read(fd, command, sizeof(command)).**
- It validates the received command to be "**multiply**".
- If the command is valid, Child 2 calculates the multiplication of the numbers in the list.
- It prints the calculated sum and multiplication results to standard output using formatted strings and write.
- It calculates the total result **(sum + multiplication)**.
- It prints the total result to standard output using formatted strings and write.
- Child 2 closes the file descriptor.
- It exits with success (**EXIT_SUCCESS**).

4. Signal Handler

- This function takes three arguments:
    - o **sig:** The signal number (in this case, **SIGCHLD**).
    - o **info:** A pointer to a **siginfo_t** structure containing information about the child process termination.
    - o **ucontext:** A pointer to a **ucontext_t** structure holding context information (not used here).
- The function uses a loop with waitpid and the **WNOHANG** flag to check for any terminated child processes.
- Inside the loop:
    - o **waitpid** returns the process ID (PID) of the terminated child.
    - o It checks if the child exited normally using **WIFEXITED(status).**
    - o If yes, it extracts the exit status using **WEXITSTATUS(status).**
    - o A buffer is created to format a message including the child PID and exit status.
    - o The counter **child_count** is incremented to track terminated child processes.

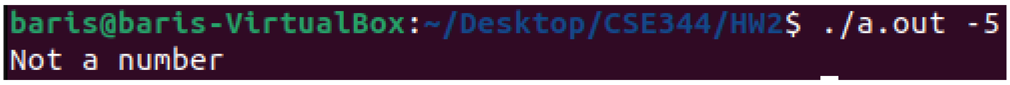
Test Cases

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



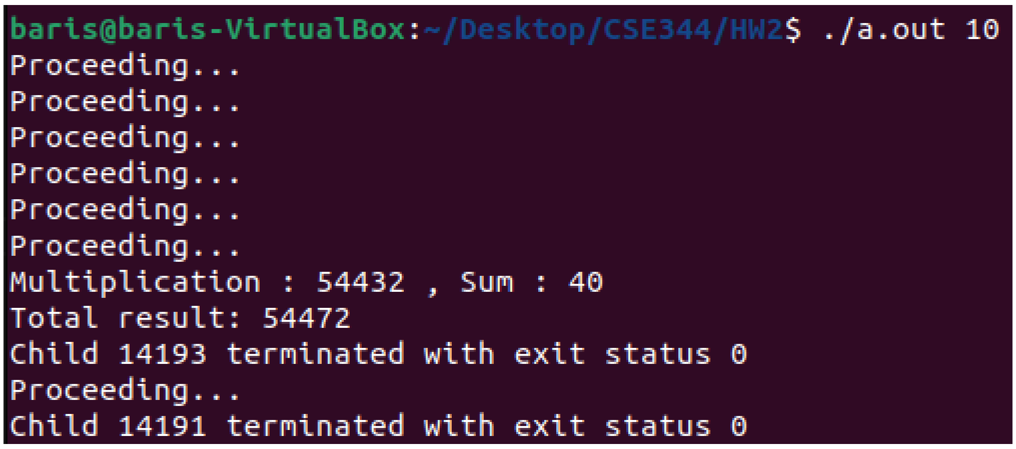
```
baris@baris-VirtualBox:~/Desktop/CSE344/HW2$ ./a.out abc
Not a number
```

2. **Test Case :** Invalid Input (Negative number)
   Input : -5
   Results :



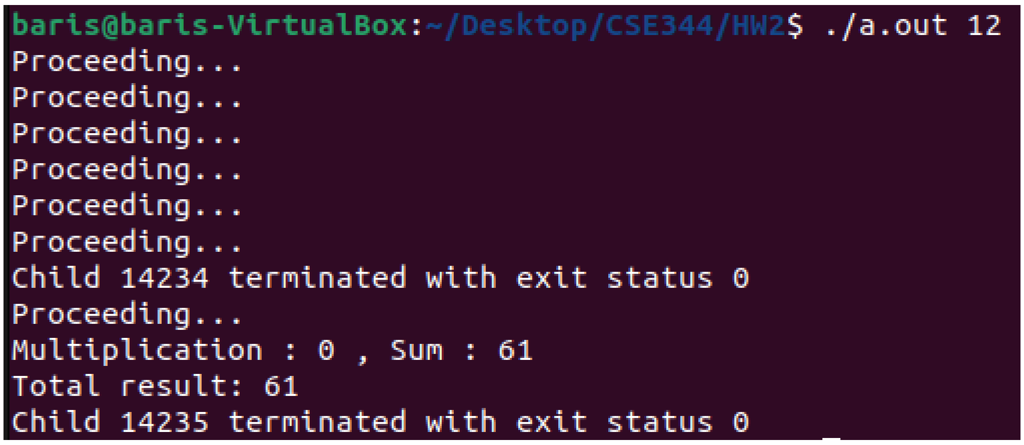
```
baris@baris-VirtualBox:~/Desktop/CSE344/HW2$ ./a.out -5
Not a number
```

3. **Test Case :** Valid Input with non zero random number
   Input : 10
   Results :



```
baris@baris-VirtualBox:~/Desktop/CSE344/HW2$ ./a.out 10
Proceeding...
Proceeding...
Proceeding...
Proceeding...
Proceeding...
Proceeding...
Multiplication : 54432 , Sum : 40
Total result: 54472
Child 14193 terminated with exit status 0
Proceeding...
Child 14191 terminated with exit status 0
```

4. **Test Case :** Valid Input with at least one zero random number
   Input : 12
   Results :



```
baris@baris-VirtualBox:~/Desktop/CSE344/HW2$ ./a.out 12
Proceeding...
Proceeding...
Proceeding...
Proceeding...
Proceeding...
Proceeding...
Child 14234 terminated with exit status 0
Proceeding...
Multiplication : 0 , Sum : 61
Total result: 61
Child 14235 terminated with exit status 0
```