**CSE – 344 Homework 2 Report**

Süleyman Burak Yaşar

1901042662

**Usage Of Makefile**

**make:** Compiles the code

**run:** Run the program -> make run “number”

**run2:** Run the program with valgrind -> make run2 “number”

**clean:** Remove test program

A computer code on a black background

Description automatically generated**Taking Number from User**

From the terminal, a single number is taken as a parameter from the user. First, the number of entered parameters is checked to determine if only one parameter has been entered. Then, the entered parameter is converted to a number.

**Generate Random Array**

**A black background with white text

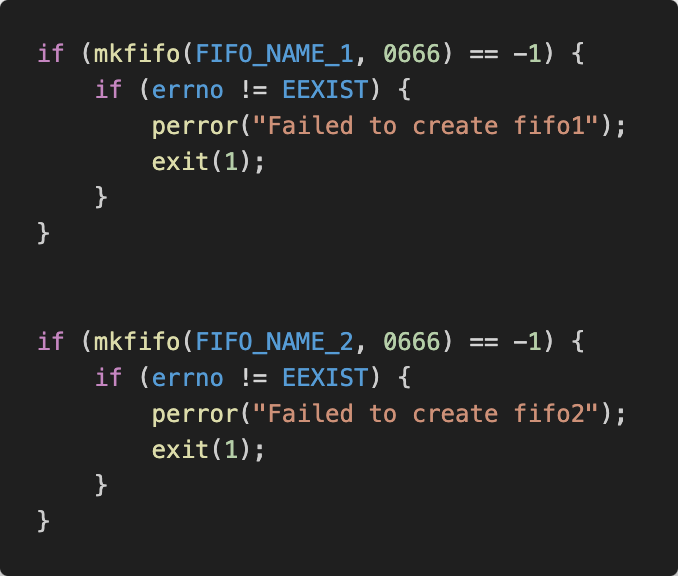
Description automatically generated**

An array consisting of random elements of a length equal to the number taken from the user is created. To prevent overflow during future multiplication, random numbers are chosen between 0 and 9.A computer code on a black background

Description automatically generated

**A black background with blue text

Description automatically generatedCreating Fifo**

****

Fifo\_1 and Fifo\_2 are created using the mkfifo() function. If there is any error during creation, the program prints the error and terminates.

**A screenshot of a computer program

Description automatically generated**

**Write Array to Fifos**

Fifo\_1 and fifo\_2 are opened for writing.

The array created is written to Fifo\_1 and Fifo\_2.

The multiply command is written to Fifo\_2.

**A screen shot of a computer program

Description automatically generated**

**Child 1**

At the start of the child process, it is first put to sleep for 10 seconds using the sleep() function. Fifo\_1 is opened for reading. Fifo\_1 is read using the read() function, and the data is saved to a new array. The sum of the elements of this array is calculated using the sum\_array() function and assigned to the sum variable. Fifo\_2 is opened to write the calculated total. The total value is written to fifo\_2. After this operation, the child process 1 completes its task and terminates.

**Child 2**

**A screen shot of a computer program

Description automatically generated**

At the start of the child process, it is first put to sleep for 10 seconds using the sleep() function. Fifo\_2 is opened for reading. First, the array sent by the parent is read using the read() function from Fifo\_1 and saved to a new array. The second read operation retrieves the command sent by the parent and stores it in a variable. It checks if the read command is "multiply." If the command is correct, the multiplication of the elements of the array is calculated and stored in a variable.

To read the total value sent by child\_1, fifo\_2 is read again. The read value is stored in a variable. At the end of the process, the total and multiplication values are printed, and the process terminates.

**Signal Handling**

**A black background with blue letters

Description automatically generated**

Initially, the total number of children to be created is defined.

**A screen shot of a computer code

Description automatically generated**

Inside the main function, the necessary arrangements were made to catch signals coming from the child process.

**A computer screen shot of text

Description automatically generated**

The sigchld\_handler() function is a signal handler that manages the status of child processes when a SIGCHLD signal is received by the parent process. This handler uses the `waitpid` function to non-blockingly query all child processes whose statuses have not yet been reported (`WNOHANG` option). If a child process has exited normally, it prints the exit status; if terminated by a signal, it prints which signal caused the termination. It also decreases the child process counter, updating the number of active processes. It is also provide zombie protection.

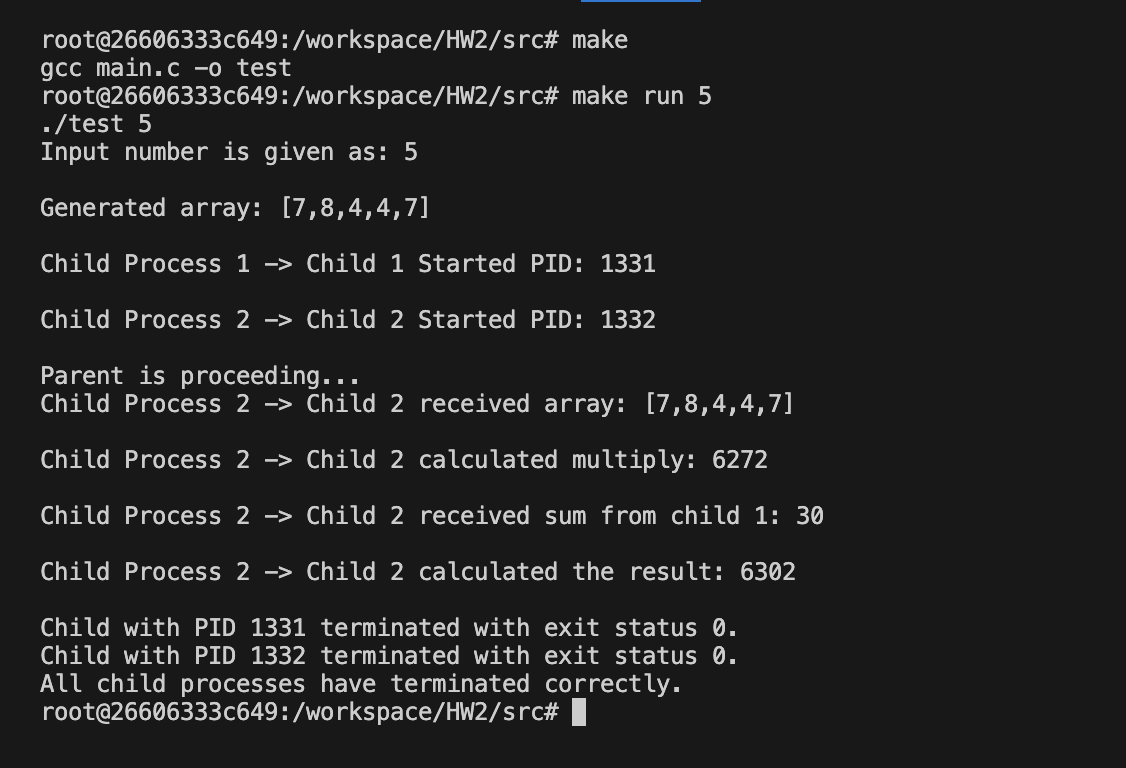
**A computer screen with text on it

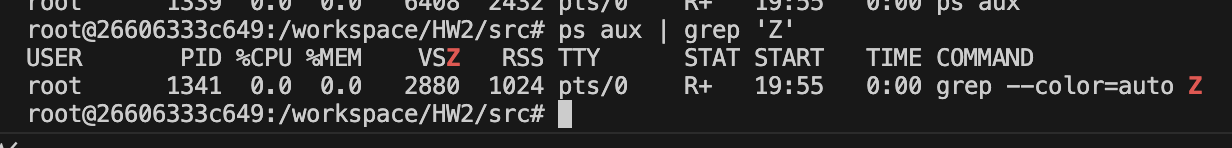
Description automatically generated**

The parent process waits in a while loop until all child processes have terminated. Once the number of active children reaches zero, indicating that all children have finished, the parent process also terminates.

**Test 1**

Output



****Zombie test

Memory Leak test

A screenshot of a computer screen

Description automatically generated

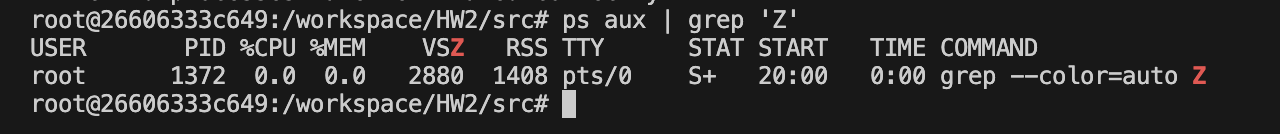
**Test2**

Output

**A screenshot of a computer program

Description automatically generated**

Zombie

****

**A screenshot of a computer error

Description automatically generated**Memory Leak

**Additional Information**

All tests were conducted using Docker. It was also re-tested on a computer with Ubuntu 22.04

installed. Just in case, I'm including the Docker file inside the document as well.