4. Multi-process Programming and Signal Handling

This assignment primarily aims to have everyone practice the programming of multi-process, signals, and exception handling.
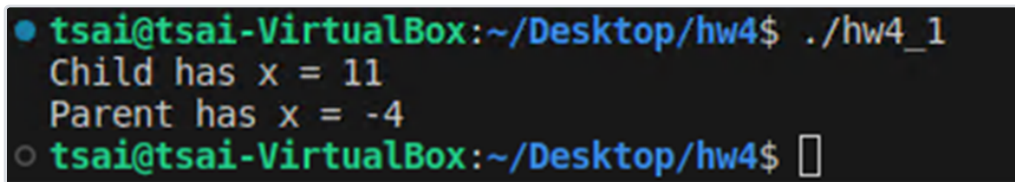
Throughout this assignment, you will engage with the following key concepts:

1. **Multi-Process Interaction:** You will employ the **fork()** system call to create multiple processes, allowing them to explore the intricacies of parallelism and concurrent execution. This experience will shed light on how processes communicate, share resources, and cooperate to achieve specific objectives.

2. **Process Synchronization:** The assignment's usage of **wait()** will let you practice the use of synchronization, enabling processes to manage the execution order. Understanding how to coordinate processes is essential in preventing race conditions and ensuring orderly execution.

3. **Signaling Mechanisms:** By utilizing **kill()** and **signal()**, you will learn how to harness the power of signals for inter-process communication and event handling. This aspect of the assignment provides insight into asynchronous communication between processes and demonstrates how signals can be customized to trigger specific actions.

4. **Exception Handling with longjmp() and setjmp():** The use of `longjmp()` and `setjmp()` will familiarize you with exception handling mechanisms that facilitate non-local jumps within the program's control flow. Through this, you will learn how to gracefully handle errors and unexpected situations, enhancing the reliability and robustness of their programs.

Assignment

In this assignment, you are required to finish the implementation of two programs `hw4_1` and `hw4_2`.

In `hw4_1,` the parent process will fork a child process. The parent and child processes will then operate on the variables within their own processes. You need to ensure that the child process completes its execution before the parent process. Please complete the missing code at line 18 so that the output will look as follows. You should not modify the rest of the code in hw4_1.

In your report, please provide answers to the following questions:

Q1: What are the issues in the original hw4_1 program? Specifically, what would have happened without the code you added at line 18?

Q2: Explain why the values of variable "x" in the parent process and the child process can be different (11 vs -4).

In `hw4_2`, this program will request the user to input a "jump case" number. Subsequently, it will fork a child process. The child process will keep waiting for the parent process to terminate it, and the parent process will wait for a signal to trigger the jump_case. If the jump case is "0/1," the parent process will terminate the child process. For the other input cases, the parent process will print "unknown error", and the child process will still be waiting to be killed. You need to press 'ctrl+z' to stop the child process and kill it by yourself. Please complete the missing code in `hw4_2` and ensure the output looks as follows.

Case1 (when the user enters '1/0'):

```
tsai@tsai-VirtualBox:~/Desktop/os_2023/4.Multi-process_Programming_and_Signal_Handling$ ./hw4_2
Input jump_case : 1
Parent is waiting for signal
Child is waiting for signal
Parent is waiting for signal
Child is waiting for signal
```

```
Parent is waiting for signal
Child is waiting for signal
^C
Child is waiting for kill
Child is waiting for kill
Child is waiting for kill

Going to kill child
Child is waiting for kill
Child is waiting for kill
Child is waiting for kill
Child has been killed.
END
tsai@tsai-VirtualBox:~/Desktop/os_2023/4.Multi-process_Programming_and_Signal_Handling$
```

Others (when the user enters the other characters):

```
tsai@tsai-VirtualBox:~/Desktop/os_2023/4.Multi-process_Programming_and_Signal_Handling$ ./hw4_2
Input jump_case : 2
Parent is waiting for signal
Child is waiting for signal
Parent is waiting for signal
Child is waiting for signal
Parent is waiting for signal
Child is waiting for signal
```

```
Parent is waiting for signal
Child is waiting for signal
^C
Child is waiting for kill

Unknown Error
Child is waiting for kill

Unknown Error
Child is waiting for kill

Unknown Error
Child is waiting for kill

Unknown Error
Child is waiting for kill
^Z
[2]+  Stopped                 ./hw4_2
tsai@tsai-VirtualBox:~/Desktop/os_2023/4.Multi-process_Programming_and_Signal_Handling$
```

```
tsai@tsai-VirtualBox:~/Desktop/os_2023/4.Multi-process_Programming_and_Signal_Handling$ jobs
[1]+  Stopped                 ./hw4_2
tsai@tsai-VirtualBox:~/Desktop/os_2023/4.Multi-process_Programming_and_Signal_Handling$ kill %1

[1]+  Stopped                 ./hw4_2
tsai@tsai-VirtualBox:~/Desktop/os_2023/4.Multi-process_Programming_and_Signal_Handling$ jobs
[1]+  Terminated              ./hw4_2
tsai@tsai-VirtualBox:~/Desktop/os_2023/4.Multi-process_Programming_and_Signal_Handling$ jobs
tsai@tsai-VirtualBox:~/Desktop/os_2023/4.Multi-process_Programming_and_Signal_Handling$
```

Please also provide answers to the following questions in your report:

Q3: Explain the difference between "setjmp & longjmp" and "goto".

Submission

Please submit a **zip** file to E3, which contains your program sources and report.

For the program source code (40%):
- The program must implemented using C.
- Make sure your code can be compiled on **Ubuntu 22.04 AMD64**.
- Make sure your outputs are correct.

For the report part, you must answer the above questions Q1, Q2 and Q3(60%):

The name of the zip file should be <student_id>.zip, and the structure of the file should be as the following:

<stduent_id>.zip
    |- <student_id>/
        |- hw4_2.c
        |- hw4_1.c
        |- hw4.pdf

For questions, please contact TA Pin-Yuan Tasi <z0922413020@gmail.com>