HW4 Report

110511010 楊育陞

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

- The issue of the original hw4_1 program is that the parent process will not wait for the child process to finish. So I added the wait() function at line 18 to make the parent process wait for the child process to finish.
- Without the code I added at line 18, the parent process will not wait for the child process to finish. Parent process will sleep for 1 second and child process will sleep for 3 seconds. So the parent process will wake up first and print the value of x. Then the child process will wake up and print the value of x. Since the execution time is small, the child process will print the result after the parent process exits. So the execution result will be as follows:

  ```
  project3 $ ./hw4_1
  Parent has x = -4
  project3 $ Child has x = 11
  ^C
  project3 $
  ```

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

- When calling fork(), system will create a new process control block for the child process. The child process will have its own copy of the parent process’s stack, heap, and data segment.
- As above, the child process will have its own copy of the parent process’s data segment. So after modification of the x by each processes. The value of variable “x” in the parent process and the child process can be different.

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

- The major difference between setjmp & longjmp and goto is that goto can only jump locally and unconditionally.
- **goto** can only jump to the code locally. It can not jump to the code in other functions. So it is not suitable for error handling.
- **setjmp & longjmp** can jump to the code non-locally. It can jump to the code in other functions such as previous function call. So it is suitable for error handling.