HW#1
Advanced Operating Systems, Spring 2023

M.X, Du (CBB108047)
Department of Computer Science and Information Engineering
National Pingtung University

1. Write a program that calls fork(). Before calling fork(), have the main process access a variable (e.g., x) and set its value to something (e.g., 100). What value is the variable in the child process? What happens to the variable when both the child and parent change the value of x?

   Solution: Please refer to List 1 (q1.c):

   Listing 1: q1.c

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 #include <unistd.h>
 4 
 5 int
 6 main(int argc, char *argv[])
 7 {
 8     int x=100;
 9     int rc = fork();
10     if (rc < 0) {
11         // fork failed; exit
12         fprintf(stderr, "fork␣failed\n");
13         exit(1);
14     } else if (rc == 0) {
15         // child (new process)
16         printf("hello,␣I␣am␣child␣(pid:%d)\n", (int) getpid());
17         printf("x␣was␣%d␣", x);
18         x=222;
19         printf(",but␣I␣have␣changed␣it␣to␣222...\n");
20         printf("Now,␣x␣is␣%d!\n", x);
21     } else {
22         // parent goes down this path (original process)
23         printf("hello,␣I␣am␣parent␣of␣%d␣(pid:%d)\n",
24                rc, (int) getpid());
25         printf("x␣was␣%d␣", x);
26         x=111;
27         printf(",but␣I␣have␣changed␣it␣to␣111...\n");
28         printf("Now,␣x␣is␣%d!\n", x);
29     }
30     return 0;
31 }
```

   Its execution results are as follows:

```
1 $ cc q1.c
2 $ ./a.out
3 hello, I am parent of 10963 (pid:10962)
4 x was 100 ,but I have changed it to 111...
5 Now, x is 111!
6 hello, I am child (pid:10963)
7 x was 100 ,but I have changed it to 222...
8 Now, x is 222!
9 $
```

   As we can see in Line 8 of Listing 1, we have declared a variable x with value 100. As we can see that in Lines 14-20 and 22-28 show the code for the child process and the parent process, respectively. In Lines 17 and 25, we print out the value of x for the child and the parent process, which both are 100 (as shown in lines 4 and 7 of the execution results). It is because the child process is created with the same content of the parent – including the data segment as well as the stack. Later, in Lines 18 and 26, we changed the values of x in the child and the parent to 222 and 111, respectively. We also print out the changed values in Lines 20 and 28, which are 222 and 111. This can be shown that the forked child process has the same contents of its parent at the time it was created. However, after that, the child and the parent are two independent processes.

2. Write a program that calls fork(). Before calling fork(), have the main process access a variable (e.g., x) and set its value to something (e.g., 100). What value is the variable in the child process? What happens to the variable when both the child and parent change the value of x?

   Solution:

   Consider the following program:

   Listing 2: q2.c

```c
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 #include <unistd.h>
 4 
 5 int
 6 main(int argc, char *argv[])
 7 {
 8     int x=100;
 9     int rc = fork();
10     if (rc < 0) {
11         // fork failed; exit
12         fprintf(stderr, "fork failed\n");
13         exit(1);
14     } else if (rc == 0) {
15         // child (new process)
16         printf("hello, I am child (pid:%d)\n", (int) getpid());
17         printf("x was %d ", x);
18         x=222;
19         printf(",but I have changed it to 222...\n");
20         printf("Now, x is %d!\n", x);
21     } else {
22         // parent goes down this path (original process)
23         printf("hello, I am parent of %d (pid:%d)\n",
24                rc, (int) getpid());
25         printf("x was %d ", x);
26         x=111;
27         printf(",but I have changed it to 111...\n");
28         printf("Now, x is %d!\n", x);
29     }
30     return 0;
31 }
```

   Please check the following results:

```
$ cc q2.c
$ ./a.out
hello, I am parent of 10963 (pid:10962)
x was 100 ,but I have changed it to 111...
Now, x is 111!
```

```
hello , I am child (pid:10963)
x was 100 ,but I have changed it to 222...
Now, x is 222!
$
```

As shown in Listing 2, Lines 14-21 are the code segament for the child process. ......

3. Write a program that calls fork(). Before calling fork(), have the main process access a variable (e.g., x) and set its value to something (e.g., 100). What value is the variable in the child process? What happens to the variable when both the child and parent change the value of x?

   Answer: In my opinion,