CS 2301 Operating Systems 1: Programming Assignment Unit 2

Anonymous Student

University of the People

CS 2301 Operating Systems 1: Programming Assignment Unit 2

This lab report documents my completing all 4 tasks for the Unit 2 programming assignment to demonstrate my understanding of the parent/child relationship.

**Write Out the Code to Your Program**

After reviewing our textbook, I created a program called ProgrammingAssignmentUnit2.c to initialize a variable, x, and then call fork(). The program also changes the value of x depending on which path is followed. The parent path multiplies x by 4 and the child path divides x by 2. Here is the code I wrote based on the examples from our reading assignment:

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/wait.h>

int main(int argc, char \*argv[])

{

int x=100;

printf("I am the main process (pid:%d) and I have set x equal to %d\n", (int) getpid(), (int) x);

int rc = fork();

if (rc < 0) {

// fork failed; exit

fprintf(stderr, "fork failed\n");

exit(1);

}

else if (rc == 0) {

// child (new process)

printf("In the child path (pid:%d), x starts at %d.\n", (int) getpid(), (int) x);

x=x/2;

printf("In the child path (pid:%d), x has been changed to %d.\n", (int) getpid(), (int) x);

}

else {

// parent goes down this path (original process)

wait(NULL);

printf("In the parent path (pid:%d), x starts at %d.\n", (int) getpid(), (int) x);

x=x\*4;

printf("In the parent path (pid:%d), x has been changed to %d.\n", (int) getpid(), (int) x);

}

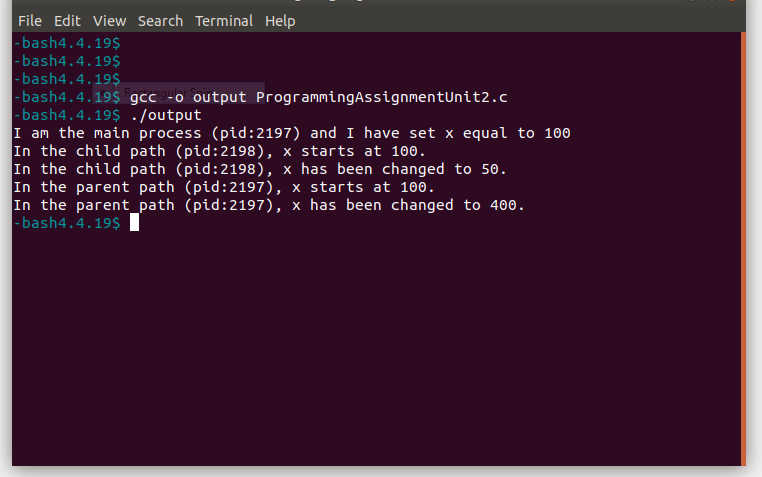
return 0;

}

(Adapted from Arpaci-Dusseau & Arpaci-Dusseau, 2012)

**Run the Program and Show Your Result**

Here is a screenshot of the command I used to run ProgrammingAssignmentUnit2.c, along with the contents of the output file that was created, when I ran the program in Terminal.



**What Value is the Variable in the Child Process?**

The initial value of x = 100 in the main process (pid:2197). When the child process is created, x still has a value 100. This is demonstrated in line 2 of the output, which states “In the child path (pid:2198), x starts at 100.” This shows that the child is a copy of the parent.

**What Happens to the Variable When Both the Child and Parent Change the Value of x?**

As stated before, the initial value of x = 100 in the main process (pid:2197) and both the child and parent each start with x = 100 after fork() is called. However, the child and parent each go on to change the value of x independently. The child changes the value of x to 50 as demonstrated in line 3 of the output, which states “In the child path (pid:2198), x has been changed to 50.” On the other hand, the parent changes the value of x to 400, which is demonstrated in line 5 of the output, which states “In the parent path (pid:2197), x has been changed to 400.” So, even though the child was a copy of the parent process, it has its own identity independent of the parent.

**Conclusion**

This assignment shows how the fork() system call works. When the main process (pid:2197) was forked, a child process (pid:2198) was created. Aside from the pid numbers (and therefore the address spaces), the child is an exact copy of the parent, with the same initial value of x, namely 100. However, now that there are two processes (the parent and child), they are now independent, allowing a change in one to not affect the other. Specifically, the value of x could be changed in the parent or child, and this will not affect the value of x in the other process. By doing this programming assignment, I was able to both demonstrate the relationship between parent and child processes as well as further my understanding virtualization.

References

Arpaci-Dusseau, R. & Arpaci-Dusseau, A. (2012). Operating Systems: Three Easy Pieces. Madison, WI: University of Wisconsin-Madison. Retrieved from <http://pages.cs.wisc.edu/~remzi/OSTEP/>/