John Kucera

Prof. Suzanna Schmeelk

CMSC 412 Operating Systems

22 January 2020

Week 2: Homework 2 with Test Runs

**Creating Processes in C with fork()**

**Solution Description**: In my C program, I used switch statements with fork() to print the PID of a parent or child. If fork() returned 0, then the child process was being executed and the child’s unique statement would be printed (containing PID). If fork() returned a positive integer, then the parent process was being executed and the parent was to wait() until the child finishes executing. When the child finished execution, the parent could execute and print its unique statement. To handle 3 processes (G, P, and C), I nested another switch statement to be reached when P was being executed.

So, G waits for P to execute, then P waits for C to execute. C prints its statement, then P can print its statement, then G can print its statement. In the switch statements, I also account for fork() functions that return -1 and are therefore errors.

On the next page are 5 test runs with screencap evidence (this program requires no input, so an input column was unnecessary). The screencaps are taken from repl.it where I ran the program.

**Test Runs**

|  |  |  |  |
| --- | --- | --- | --- |
| **Expected Output** | **#** | **Actual Output** | **Pass?** |
| (Applies to all test runs)  When “Run” is clicked, processes G, P, and C are created. When they execute in order of C > P > G, they should print unique statements:  (XYZ represent numbers)  C should print “I am the child process C and my pid is X. My parent P has pid Y. My grandparent G has pid Z.”.  P should print “I am the parent process P and my pid is Y. My parent G has pid Z.”.  G should print “I am the grandparent process G and my pid is Z.”.  **C’s pid should be 1 greater than P’s and P’s pid should be 1 greater than G’s.** For example, if G = 1, then P = 2, and C = 3. | **1** |  | **YES** |
| **2** |  | **YES** |
| **3** |  | **YES** |
| **4** |  | **YES** |