Sean Webster

Operating Systems
Homework 1
Due 9/23/2016

Objective: The objective of this assignment was to learn how process creation works by creating parent and child processes in C.

Background: Processes are one of the main building blocks of operating systems, and allow for task delegation. By learning about them, a large part of operating systems can then be understood.

Functions Used: The fork() function was used, which creates a child process. The exit() function, which exits a process, was also used. Waitpid(), which waits for a process with a pid to die, was used as well. Execv(), which overwrites a process with a new function, was also used.

Results:

```
[swebster@anaconda29 OS]$ ./prog 2 4 6
Child Running
a.out  b.c    hw1_2.c    HW1a    prog   progb.c  test    test.o
b      hw1_2  HW1_2.png  HW1a.c  progb  prog.c   test.c
Child Running
a.out  b.c    hw1_2.c    HW1a    prog   progb.c  test    test.o
b      hw1_2  HW1_2.png  HW1a.c  progb  prog.c   test.c
Child Running
a.out  b.c    hw1_2.c    HW1a    prog   progb.c  test    test.o
b      hw1_2  HW1_2.png  HW1a.c  progb  prog.c   test.c
Child Running
a.out  b.c    hw1_2.c    HW1a    prog   progb.c  test    test.o
b      hw1_2  HW1_2.png  HW1a.c  progb  prog.c   test.c
Child Running
a.out  b.c    hw1_2.c    HW1a    prog   progb.c  test    test.o
b      hw1_2  HW1_2.png  HW1a.c  progb  prog.c   test.c
Child Running
a.out  b.c    hw1_2.c    HW1a    prog   progb.c  test    test.o
b      hw1_2  HW1_2.png  HW1a.c  progb  prog.c   test.c
Child Running
a.out  b.c    hw1_2.c    HW1a    prog   progb.c  test    test.o
b      hw1_2  HW1_2.png  HW1a.c  progb  prog.c   test.c
Child Running
a.out  b.c    hw1_2.c    HW1a    prog   progb.c  test    test.o
b      hw1_2  HW1_2.png  HW1a.c  progb  prog.c   test.c
Child Running
a.out  b.c    hw1_2.c    HW1a    prog   progb.c  test    test.o
b      hw1_2  HW1_2.png  HW1a.c  progb  prog.c   test.c
Child Running
a.out  b.c    hw1_2.c    HW1a    prog   progb.c  test    test.o
b      hw1_2  HW1_2.png  HW1a.c  progb  prog.c   test.c
Child Running
a.out  b.c    hw1_2.c    HW1a    prog   progb.c  test    test.o
b      hw1_2  HW1_2.png  HW1a.c  progb  prog.c   test.c
Child Running
a.out  b.c    hw1_2.c    HW1a    prog   progb.c  test    test.o
b      hw1_2  HW1_2.png  HW1a.c  progb  prog.c   test.c
[swebster@anaconda29 OS]$
```

Figure 1: Part 1 Output

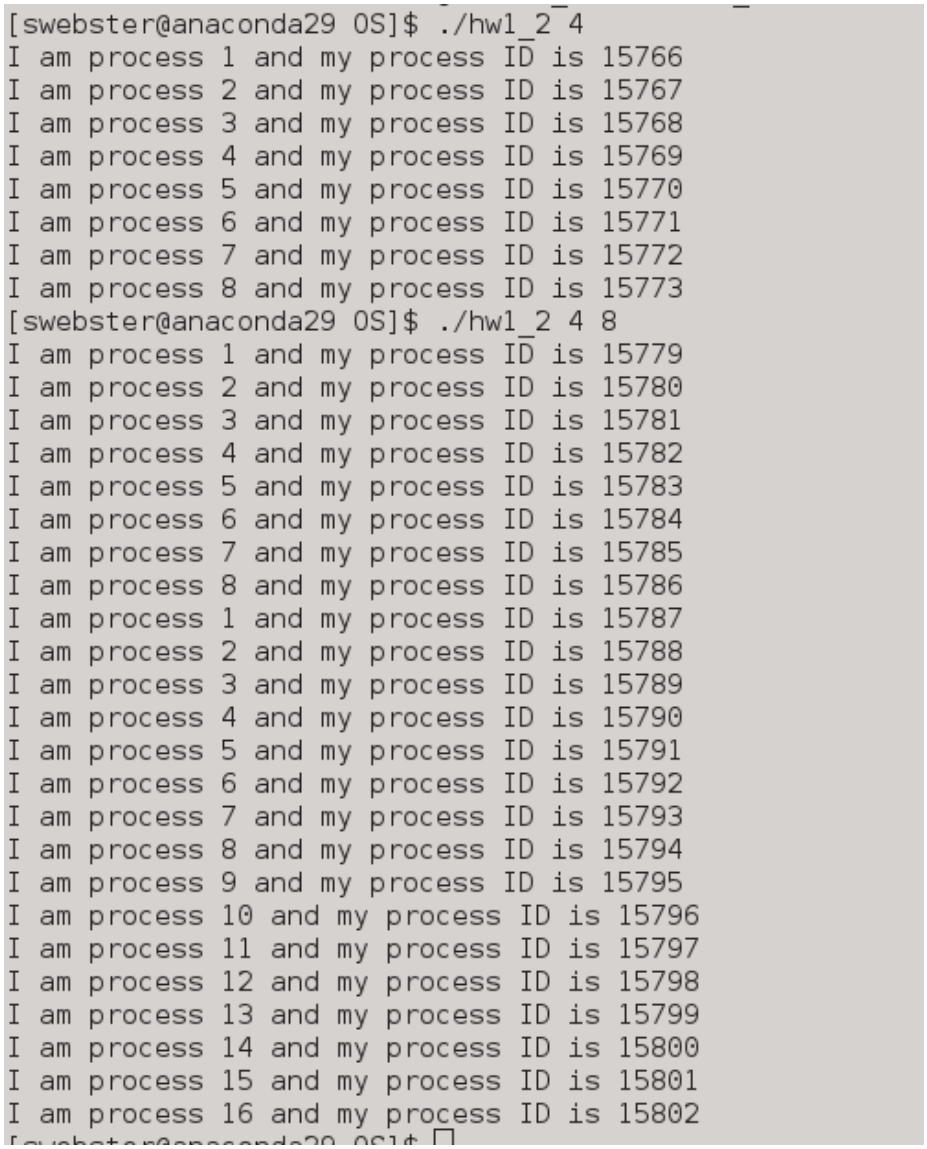

Figure 2: Part 2 Output

Conclusions and Observations: Process creation has become an easy thing. However, there is not much that I know yet about how to view and manipulate the processes I have created. I believe that with more experience, I can truly understand and view the processes I created properly. Execv was tricky to learn at first, but looking up the function helped greatly. All other functions were pretty much explained in the slides and in class, which some code in my programs even coming from the slides.

Readme:

Sean Webster

Homework 1 readme

Source Code Files:

- prog.c
- hw1_2.c
- b.c

Executables:

- prog
- hw1_2
- b

Instructions for running:

Run programs with intended number of processes in arg section

Commands:

```
./prog 8
./hw1_2 8
```

Part 1:

```
/*****************************************************
 * Sean Webster
 * Operating Systems
 * Homework 1
 * Due 9/23
 *
 *
 * ************************************************
 * prog.c
     * creates n subprocesses using fork and displays "child running"
     * and the exits. also runs execv to open b.c in each child process
     * ************************************************/

    #include <sys/types.h>
    #include <sys/wait.h>
    #include <unistd.h>
    #include <stdio.h>
    #include <stdlib.h>  // for strtol

    int main(int argc, char **argv)
    {
                // Allow for arg stuff
          int j;
          for(j = 0; j <= argc; j++)
          {
                int res;
                int i = 0;
                /* duplicate the current process */
                for(i = 0; i < strtol(argv[j], NULL, 10); i++)
                {
                     res = fork();
                     /* *******
                     * If fork was successful there are now two processes at
  this point
                     **** */
                     /* check whether fork is successful */
                     if (res < 0)
                     {
                           perror("fork");
                           exit(-1);
                           }
                     /* Check whether this is the father or the child */
                     /* the child got 0 from fork. */
                     if (res == 0)
                     {
                           char *args[] = {"", NULL};
                           printf("Child Running\n");
                           res = execv("b", args);
                           // Child Exits
                           exit(0);
                     }
                     else
                     { /* parent process */
                           int child_pid = res;
                           waitpid(child_pid, NULL, 0);
                           /* parent will wait for the child to complete */
                     }
```

```
                }
          }
          exit(0);
     }
```

Part 2:

```
/*****************************************************
 * Sean Webster
 * Operating Systems
 * Homework 1
 * Due 9/23
 *
 *
 * ***************************************************
 * hw1_2.c
     * takes an integer input n and outputs 2*n processes
     * displaying the phrase "I am process i and my process id
     * is PID"
     * ***************************************************/


    #include <sys/types.h>
    #include <sys/wait.h>
    #include <unistd.h>
    #include <stdio.h>
    #include <stdlib.h>  // for strtol

    int main(int argc, char **argv)
    {
          // Allow for arg stuff
          int j;
          for(j = 0; j <= argc; j++)
          {
                int res;
                int i = 0;
                /* duplicate the current process */
                for(i = 0; i < 2 * strtol(argv[j], NULL, 10); i++)
                {
                     res = fork();
                     /* *******
                     * If fork was successful there are now two processes at
  this point
                     **** */
                     /* check whether fork is successful */
                     if (res < 0)
                     {
                           perror("fork");
                           exit(-1);
                           }
                     /* Check whether this is the father or the child */
                     /* the child got 0 from fork. */
                     if (res == 0)
                     {
                           // Part A
                           printf("I am process %d and my process ID is %d\n",
```

```
  i + 1, getpid());
49.                               // Child Exits
50.                               exit(0);
51.                         }
52.                         else
53.                         { /* parent process */
54.                               int child_pid = res;
55.                               waitpid(child_pid, NULL, 0);
56.                               /* parent will wait for the child to complete */
57.                         }
58.                   }
59.             }
60.             exit(0);
61.       }
```

```
1./*****************************************************
2. * Sean Webster
3. * Operating Systems
4. * Homework 1
5. * Due 9/23
6. *
7. *
8. * ***************************************************
9. * b.c
10.       * Program to be run by part one of HW 1, executes system
11.       * command 'ls'
12.       * ***************************************************/
13.      #include <sys/types.h>
14.      #include <sys/wait.h>
15.      #include <unistd.h>
16.      #include <stdio.h>
17.
18.      int main(int argc, char **argv)
19.      {
20.            system("ls");
21.
22.
23.            return(0);
24.      }
```