

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:

[illegible]

Figure 1: Part 1 Output

```
[swebster@anaconda29 0S]$ ./hw1_2 4
I am process 1 and my process ID is 15766
I am process 2 and my process ID is 15767
I am process 3 and my process ID is 15768
I am process 4 and my process ID is 15769
I am process 5 and my process ID is 15770
I am process 6 and my process ID is 15771
I am process 7 and my process ID is 15772
I am process 8 and my process ID is 15773
[swebster@anaconda29 0S]$ ./hw1_2 4 8
I am process 1 and my process ID is 15779
I am process 2 and my process ID is 15780
I am process 3 and my process ID is 15781
I am process 4 and my process ID is 15782
I am process 5 and my process ID is 15783
I am process 6 and my process ID is 15784
I am process 7 and my process ID is 15785
I am process 8 and my process ID is 15786
I am process 1 and my process ID is 15787
I am process 2 and my process ID is 15788
I am process 3 and my process ID is 15789
I am process 4 and my process ID is 15790
I am process 5 and my process ID is 15791
I am process 6 and my process ID is 15792
I am process 7 and my process ID is 15793
I am process 8 and my process ID is 15794
I am process 9 and my process ID is 15795
I am process 10 and my process ID is 15796
I am process 11 and my process ID is 15797
I am process 12 and my process ID is 15798
I am process 13 and my process ID is 15799
I am process 14 and my process ID is 15800
I am process 15 and my process ID is 15801
I am process 16 and my process ID is 15802
[swebster@anaconda29 0S]$
```

*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:**

```
1. /*****
2.  * Sean Webster
3.  * Operating Systems
4.  * Homework 1
5.  * Due 9/23
6.  *
7.  *
8.  * *****/
9.  * prog.c
10.     * creates n subprocesses using fork and displays "child running"
11.     * and the exits. also runs execv to open b.c in each child process
12.     * *****/
13.
14.     #include <sys/types.h>
15.     #include <sys/wait.h>
16.     #include <unistd.h>
17.     #include <stdio.h>
18.     #include <stdlib.h> // for strtol
19.
20.     int main(int argc, char **argv)
21.     {
22.         // Allow for arg stuff
23.         int j;
24.         for(j = 0; j <= argc; j++)
25.         {
26.             int res;
27.             int i = 0;
28.             /* duplicate the current process */
29.             for(i = 0; i < strtol(argv[j], NULL, 10); i++)
30.             {
31.                 res = fork();
32.                 /* *****
33.                 * If fork was successful there are now two processes at
34.                 * this point
35.                 * **** */
36.                 /* check whether fork is successful */
37.                 if (res < 0)
38.                 {
39.                     perror("fork");
40.                     exit(-1);
41.                 }
42.                 /* Check whether this is the father or the child */
43.                 /* the child got 0 from fork. */
44.                 if (res == 0)
45.                 {
46.                     char *args[] = {"", NULL};
47.                     printf("Child Running\n");
48.                     res = execv("b", args);
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.     exit(0);
60. }

```

## Part 2:

```

1. /*****
2.  * Sean Webster
3.  * Operating Systems
4.  * Homework 1
5.  * Due 9/23
6.  *
7.  *
8.  * *****/
9.  * hw1_2.c
10.  * takes an integer input n and outputs 2*n processes
11.  * displaying the phrase "I am process i and my process id
12.  * is PID"
13.  * *****/
14.
15.
16. #include <sys/types.h>
17. #include <sys/wait.h>
18. #include <unistd.h>
19. #include <stdio.h>
20. #include <stdlib.h> // for strtol
21.
22. int main(int argc, char **argv)
23. {
24.     // Allow for arg stuff
25.     int j;
26.     for(j = 0; j <= argc; j++)
27.     {
28.         int res;
29.         int i = 0;
30.         /* duplicate the current process */
31.         for(i = 0; i < 2 * strtol(argv[j], NULL, 10); i++)
32.         {
33.             res = fork();
34.             /* *****
35.              * If fork was successful there are now two processes at
36.              * this point
37.              * **** */
38.             /* check whether fork is successful */
39.             if (res < 0)
40.             {
41.                 perror("fork");
42.                 exit(-1);
43.             }
44.             /* Check whether this is the father or the child */
45.             /* the child got 0 from fork. */
46.             if (res == 0)
47.             {
48.                 // Part A
49.                 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.     }

```