Vivian Do
CSC 4320
Jan 22 2023

Spring 2023 – 4320/6320 Section 006 Operating Systems

Homework 1: Due 01/22/2023 at 11:59 PM

Your program must compile with gcc and execute on snowball.cs.gsu.edu! Please see https://cscit.cs.gsu.edu/sp/guide/snowball for more details. You may use whatever IDEs / text editors you like, but you must submit your responses on iCollege.

1. Briefly describe the semantics of the following Unix system calls (12 Points):
   a. fork();
   b. exit();
   c. getpid();
   d. getppid();
   e. waitpid();
   f. execl();

a) *fork()* is used to create new processes.
b) *exit()* is used to stop a calling process immediately.
c) *getpid()* is used to return the process ID of the calling process.
d) *getppid()* is used to return the parent process ID of the calling process.
e) *waitpid()* is used to suspend the calling process until a child process has ended or stopped.
f) *execl()* is used to replace the current process image with a new process image.

2. Write a C program that uses each of the above system calls at least once. Submit your .c file and screenshots of the corresponding output. Make sure that your program compiles and executes without error. (23 Points)

My Code:

vdo10@gsuad.gsu.edu@snowball:~/OS

```
[vdo10@gsuad.gsu.edu@snowball OS]$ cat hw1.c
/*
Vivian Do
CSC 4320
Jan 22 2023
*/

// adding libraries to use in program
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/wait.h>
#include <stdlib.h>

// main method
int main() {
    // creating variable for fork() function
    int id = fork();
    // checking if child id is 0
    if (id == 0) {
        // child id waits for 1 second
        sleep(1);
    }

    // printing ids from getpid() & getppid() functions
    printf("Child ID: %d | Parent ID: %d\n", getpid(), getppid());

    // creating variable for waitpid() function
    int waiting = waitpid(-1, NULL, 0);
    // checking for processing results of current id
    if (waiting == -1) {
        // printing results
        printf("%d has no children to wait for.\n\n", getppid());
    }
    else if (waiting != -1) {
        // printing results
        printf("%d has finished processing.\n\n", waiting);
    }
    else {
        exit(0);
    }

    // specifying path name & creating arguments
    char *path = "/bin/ls";
    char *arg1 = "ls";
    // pathing to my directory on snowball
    char *arg2 = "/home/vdo10/OS";

    // calling execl() function
    execl(path, path, arg1, arg2, NULL);
    // terminating program
    exit(0);
}
[vdo10@gsuad.gsu.edu@snowball OS]$
```

Screenshot:

```
[vdo10@gsuad.gsu.edu@snowball OS]$ cc hw1.c
[vdo10@gsuad.gsu.edu@snowball OS]$ ./a.out
Child ID: 28068 | Parent ID: 27982
Child ID: 28069 | Parent ID: 28068
28068 has no children to wait for.

/bin/ls: cannot access ls: No such file or directory
/home/vdo10/OS:
a.out  hw1.c
28069 has finished processing.

/bin/ls: cannot access ls: No such file or directory
/home/vdo10/OS:
a.out  hw1.c
[vdo10@gsuad.gsu.edu@snowball OS]$
```